Parallel implementation and application of particle scale heat transfer in the Discrete Element Method

Amit Ravindra Amritkar

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Mechanical Engineering

Danesh K. Tafti, Chair
Kenneth S. Ball
Clinton L. Dancey
Mark R. Paul
Calvin J. Ribbens

June 20th 2013
Blacksburg, Virginia

Keywords: Computational fluid dynamics (CFD), Heat Transfer, OpenMP, MPI, Hybrid parallelization, Performance tools, Multiphase flows, Fluidized beds

Parallel implementation and application of particle scale heat transfer in the Discrete Element Method

Amit Ravindra Amritkar

ABSTRACT

Dense fluid-particulate systems are widely encountered in the pharmaceutical, energy, environmental and chemical processing industries. Prediction of the heat transfer characteristics of these systems is challenging. Use of a high fidelity Discrete Element Method (DEM) for particle scale simulations coupled to Computational Fluid Dynamics (CFD) requires large simulation times and limits application to small particulate systems. The overall goal of this research is to develop and implement parallelization techniques which can be applied to large systems with $O(10^5 - 10^6)$ particles to investigate particle scale heat transfer in rotary kiln and fluidized bed environments.

The strongly coupled CFD and DEM calculations are parallelized using the OpenMP paradigm which provides the flexibility needed for the multimodal parallelism encountered in fluid-particulate systems. The fluid calculation is parallelized using domain decomposition, whereas N-body decomposition is used for DEM. It is shown that OpenMP-CFD with the first touch policy, appropriate thread affinity and careful tuning scales as well as MPI up to 256 processors on a shared memory SGI Altix. To implement DEM in the OpenMP framework, ghost particle transfers between grid blocks, which consume a substantial amount of time in DEM, are eliminated by a suitable global mapping of the multi-block data structure. The global mapping together with enforcing perfect particle load balance across OpenMP threads results in computational times between 2-5 times faster than an equivalent MPI implementation.

Heat transfer studies are conducted in a rotary kiln as well as in a fluidized bed equipped with a single horizontal tube heat exchanger. Two cases, one with mono-disperse 2 mm particles rotating at 20 RPM and another with a poly-disperse distribution ranging from 1-2.8 mm and rotating at 1 RPM are investigated. It is shown that heat transfer to the mono-disperse 2 mm particles is dominated by convective heat transfer from the thermal

boundary layer that forms on the heated surface of the kiln. In the second case, during the first 24 seconds, the heat transfer to the particles is dominated by conduction to the larger particles that settle at the bottom of the kiln. The results compare reasonably well with experiments. In the fluidized bed, the highly energetic transitional flow and thermal field in the vicinity of the tube surface and the limits placed on the grid size by the volume-averaged nature of the governing equations result in gross under prediction of the heat transfer coefficient at the tube surface. It is shown that the inclusion of a subgrid stress model and the application of a LES wall function (WMLES) at the tube surface improves the prediction to within ± 20% of the experimental measurements.

Dedicated to my family

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Danesh Tafti, W. S. Cross Professor of Mechanical Engineering, to whom I owe my deepest gratitude. Dr. Tafti motivated and supported me in overcoming all the obstacles in the completion of this research work. I would like to thank my PhD Committee Dr. Kenneth Ball, Dr. Clinton Dancey, Dr. Mark Paul, and Dr. Calvin Ribbens for their support towards the completion of my research goals.

I would like to thank the National Science Foundation for supporting part of this work. The support of HPC resources of TeraGrid (now XSEDE) and ARC at Virginia Tech is greatly appreciated.

I would also like to thank the staff of the Mechanical Engineering Department for their help during the course of this work. I also thank my lab-mates and friends for the continuous encouragement, intriguing discussions and moral support.

Last but not the least; I would like to thank my family for offering me unconditional support in completing this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$\vec{a}$	Contravariant vector
α	Thermal diffusivity
Bi	Biot number
d_p	Particle diameter
d_t	Tube diameter
c_p	Specific heat capacity
e	Coefficient of restitution
E	Modulus of elasticity
$\vec{F}$	Force
$\sqrt{g}$	Jacobian of transformation
g^{ij}	Elements of contravariant metric tensor
$\vec{g}$	Gravitational acceleration
h	Heat transfer coefficient
κ	Thermal conductivity
K	Spring stiffness
L	Bed width
m	Reduced mass
Nu	Nusselt number = hd_p/κ
ΔP	Pressure drop
Pr	Prandtl number
q"	Heat flux
Q	Flow rate
$r_{c,ij}$	Radius of contact area between particles i and j
Re_p	Local Reynolds number based on particle diameter = $\varepsilon u_p d_p/\nu$
Re_d	Local Reynolds number based on tube diameter = $\varepsilon u_p d_t/\nu$
R	Radius
t	Time
T	Fluctuating, modified or homogenized temperature

u_i	Cartesian velocity vector
Δx_i	Grid spacing
β	Momentum exchange coefficient
ε	Void fraction
σ	Poisson's ratio
ρ	Density
$\vec{\xi}$	Computational coordinates
τ	Torque
ν	Fluid viscosity
γ	Partition coefficient of friction generated heat flow
μ	Coefficient of friction
$\vec{V}$	Velocity
y^+	Non-dimensional wall distance

Subscripts

$_{ref}$	Reference value
$_t$	Based on turbulence
$_p$	Particle property
$_f$	Fluid property
$_w$	Wall property
$_{fric}$	Friction based quantity
$_T$	Tangential component
$_N$	Normal component

Superscripts

*	Dimensional Values

1. Introduction

Motivation

Dense fluid–particulate systems are frequently encountered in a wide range of applications in the chemical, petrochemical, energy, metallurgical and pharmaceutical industries. The complexity of these multiphase flows makes it difficult to study them experimentally and requires the use of computational techniques. There are two mainstream approaches of modeling fluid-particulate multiphase flows, two fluid model and discrete element method (DEM), also called as discrete particle method (DPM). In the two fluid model, solid and fluid phases are treated as interpenetrating media interacting through interphase momentum and energy exchange terms. Only volume or ensemble average information of flow quantities is obtained and it lacks the detailed description of physics at the particle scale. For the accurate prediction of fluid particulate flows, it is essential that both the fluid–particle as well as the particle–particle interactions be accurately modeled. This is addressed in the DEM approach which solves the particulate phase in the Lagrangian frame where each particle is tracked, giving details of individual particle behavior, while the fluid is treated in an Eulerian frame. DEM is widely used in the numerical analysis of dense particulate systems in which the solid volume fractions are typically greater than 40%. The DEM includes models for particle-particle and particle-surface collisions using a soft-sphere model, particle-particle and particle-surface conduction heat transfer during each collision, and particle-gas convective heat transfer.

In this research the DEM is used to investigate heat transfer in fluid particulate systems. The DEM operates at an individual particle level, and thus provides high fidelity. The method itself has been in use for some time but never applied to investigate heat transfer at the scale of $O(10^5\text{-}10^6)$ particles in three-dimensional (3D) bed configurations because of the high computational complexity. Most previous work with this method, has been in two-dimensions (2D) with $O(10^3\text{-}10^4)$ particles due to high CPU and memory requirements. Since the current work involved 3D DEM simulation studies with $O(10^5\text{-}10^6)$ particles, it is essential to have highly parallelizable code for the multiphase problem. In fluid particulate systems such as rotary kilns and fluidized beds, particles are

heavily concentrated in a small part of the full computational domain. If the work load associated with these particles is large, as is often the case, then treating them in the domain decomposition framework of MPI can lead to severe load imbalances and inefficiencies. By introducing the OpenMP parallelization paradigm the above load imbalance is addressed. The OpenMP parallelization involves domain decomposition for the fluid field and N-body decomposition for the particulate phase. This flexibility offered by OpenMP parallelism will help accelerate complex computations such as fluidized bed heat transfer characterization. With the anticipated massive growth in core count per node as well as accelerating units with shared memory architecture such as General Purpose Graphic Processing Units (GPGPUs) and Many-Integrated Cores (MICs), OpenMP parallelism will also see wide spread utility in high performance computing.

Contributions of this Work

The main scientific and engineering contributions of this work are the development of a parallel framework to simulate fluid-particulate systems with particle scale heat transfer capabilities. The in-house code GenIDLEST was parallelized and the then the parallel performance of the code was fine-tuned using profiling and other tools. The heat transfer capability was achieved by implementing particle scale heat transfer models in the framework of the GenIDLEST code. To our knowledge, this is the first study to have used OpenMP parallelism for coupled fluid-particulate systems on a large scale effectively. This advance allowed the application of DEM to investigate 3D fluid particulate systems with heat transfer. While DEM has been applied in the past to investigate heat transfer, the current capability allows the simulation of larger more realistic non-canonical systems. To the best of our knowledge, this is the first DEM investigation of heat transfer in a rotary kiln with a distribution of particle diameters, unlike existing studies with mono-disperse particles. An additional contribution of this work has been to overcome the deficiency imposed by the volume-averaged nature of the fluid equations on the minimum grid size which is limited to 2.5-3.0 times the particle diameter. While this requirement is not very restrictive in hydrodynamic studies of fluidized beds, it severely limits the grid resolution near heat transfer surfaces and grossly

under predicts convective heat transfer. In this work it is established that an LES approach with a wall model (WMLES) can make up for the lack of grid resolution near heat transfer surfaces.

The following journal articles and conference papers are an integral part of this dissertation:

- OpenMP parallelism for fluid and fluid-particulate systems, Amit Amritkar, Danesh Tafti, Rui Liu, Rick Kufrin, Barbara Chapman, Parallel Computing, Volume 38, Issue 9, September 2012, Pages 501–517
- Efficient parallel CFD-DEM simulation of fluid-particulate system using OpenMP, Amit Amritkar, Surya Deb, Danesh Tafti, Journal of Computational Physics, *Under review*
- Particle scale heat transfer analysis in rotary kiln, Amit Amritkar, Danesh Tafti, Surya Deb, Proc. of ASME HT2012, Puerto Rico, July 8-12 2012.
- Heat transfer analysis in a rotary furnace with a poly-disperse particle distribution, Amit Amritkar, Danesh Tafti, Powder Technology, *to be prepared and submitted*
- Wall modeled LES for heat transfer in fluidized bed with a horizontal tube heat exchanger, Amit Amritkar, Danesh Tafti, Journal of Heat Transfer, *to be prepared and submitted*

Organization of Thesis

The rest of the manuscript is organized as follows. The second chapter introduces the application of OpenMP parallelism to the in-house code GenIDLEST and discusses the code performance. The application of OpenMP parallelism to tightly coupled fluid-particulate system is discussed in Chapter 3. Chapter 4 presents the governing equations and methods used for particle scale heat transfer. Chapter 5 presents results of heat transfer analysis for a rotary kiln with poly dispersed particles followed by results of heat transfer analysis in a fluidized bed with a horizontal tube heat exchanger. Finally, conclusions and future scope of this work are presented in Chapter 6.

2. OpenMP parallelism for fluid flow [1]

Introduction

High-end applications have relied on the Message Passing Interface (MPI) over the last two decades for programming parallel applications. MPI has provided scalability on large applications by forcing data locality. Data locality together with SPMD style programming using spatial domain decomposition has been a very successful model for high-end computing. While this model is inevitable in a clustered environment, it has also proved its mettle on large SMP (Shared-memory MultiProcessor) architectures, in spite of the shared cache-coherent memory model and the added overhead of MPI calls, by forcing an explicit link between processor and memory and eliminating references to remote memory, except through explicit message passing.

The often quoted drawback of MPI is the high programming, development, and maintenance costs. Additionally, a major drawback stems from what gives MPI its strength – explicit array partitioning. Within this framework, any parallelism which deviates from this model incurs heavy costs in performance. In engineering computations, one example is dispersed two phase flow in which solid particles are individually tracked in a fluid domain. The particle distribution is a function of the physical attributes of the solid-fluid system and could well lead to all particles accumulating on a few processors leading to severe load imbalances [1]. Similar irregularities exist in a number of multiphysics applications in which the explicit static data partitioning becomes inefficient.

An alternative to MPI programming on SMPs is the use of OpenMP. OpenMP directives are easy to implement, and can be used for incremental parallelism in a serial application [2]. It is much more flexible than MPI in that it lends itself to different types of

---

[1] Majority part of this chapter is published in *OpenMP parallelism for fluid and fluid-particulate systems*, Amit Amritkar, Danesh Tafti, Rui Liu, Rick Kufrin, Barbara Chapman, *Parallel Computing*, Volume 38, Issue 9, September 2012, Pages 501–517. *Used with permission of Elsevier, 2013*

parallelism. It can exploit SPMD type parallelism (similar to that in MPI), functional parallelism, and task parallelism in a single program unit and is not tied down to any single mode. For instance in the above example, in an OpenMP code, the fluid domain could be parallelized using a domain decomposition style of programming similar to what would be used in an MPI decomposition, whereas N-body parallelism could be used for the dispersed phase. Undoubtedly, OpenMP is much more suitable for dynamic irregular applications [3, 4]. However, it has not seen widespread use in high-end HPC applications because of its inability to scale to a large number of processors and also to a large extent, the lack of portability (re-compile and run) across distributed clusters.

Efforts have been made to combine the advantages of both paradigms (MPI/OpenMP) on DSM (Distributed Shared Memory) architectures. The hybrid paradigm strives to take advantage of the scalability of MPI together with the flexibility of OpenMP. Early studies [5] used the hybrid paradigm to implement embedded parallelism at a coarse level via MPI and at a fine level via OpenMP threads. It was shown that it was possible to combine the two paradigms within the framework of a single program. [4] also showed that the hybrid paradigm could be used for treating dynamic irregular applications, by using dynamic OpenMP threads to balance the computational needs in a MPI process. In two-phase dispersed particulate flows, OpenMP helper threads could be invoked on heavily loaded MPI processors where the particles tend to accumulate.

In the past, many studies have been conducted in which the performance of OpenMP has been compared with MPI. The average filtered timing data for seven simple test programs (two communication oriented and five kernels) was measured by [6]. MPI performed better than OpenMP for most of the cases. In an OpenMP study on a CFD code [7] it was found that the OpenMP results showed poor scalability compared to MPI for 8 processors. They also did studies involving scheduling strategies and critical versus reduction operations to conclude that the static scheduling performed the best and critical sections are more time consuming. In another study [8] on ocean models, OpenMP performance was found to be competitive on shared memory platforms since the parallelization strategy was domain decomposition. On a Sun Microsystems machine, [2] performed OpenMP and MPI runs for up to 144 processors on a molecular modeling code of about 2000 lines. They established that the Sun studio's OpenMP implementation

scaled better because the memory bandwidth for MPI communications was limited. In other research, a comparison of MPI and three different OpenMP parallelization approaches on the NAS Parallel Benchmarks was done [9]. They found the OpenMP SPMD programming style and the optimized loop level OpenMP programming was competitive with MPI but overall MPI still performed better. In a scaling study of an unstructured fluid solver [10], the OpenMP and MPI performances were observed to be 10% apart for upto 128 processors. Recently, performance analysis of a finite element based CFD code called FEFLO for upto 96 processing cores was performed [1]. The study showed good OpenMP scaling for edge based parallelization in finite element discretized space. The performance characterization of the Columbia cluster at NASA was carried out using NAS parallel benchmarks and 3 CFD applications [11]. The Cart3D fluid solver which solves the Euler equations showed excellent scaling for both MPI and OpenMP for 474 CPUs. The performance results of INS3D (incompressible Navier Stokes equation solver) and OVERFLOW-D (compressible Navier Stokes equation solver) codes in hybrid execution mode (OpenMP+MPI) show performance degradation after about 144 processors for INS3D and 64 processors for OVERFLOW-D.

In an alternate study, MPI and hybrid performance results were compared for the NAS Parallel Benchmarks and four CFD applications (one structured CFD application (OVERFLOW-2), one Cartesian grid application (CART3D), one unstructured tetrahedral CFD application (USM3D), and one application from climate modeling (ECCO)) for up to 1024 cores on different architectures with the help of performance measurement tools [12]. The study indicated that overall the MPI performance is better than the hybrid (MPI+OpenMP) execution except when the number of cores per node was increased to 32. The MPI performance was poor due to limited memory bandwidth available for the MPI communications. Another CFD application called TAU, developed by the German aerospace agency (DLR) to solve the compressible Navier Stokes equations on unstructured grids [13] was tested using hybrid parallelization up to O(1000) processors.

The lack of data placement directives among threads and processor cores for data locality in the OpenMP standard can be addressed by either compiler directives or runtime systems for data distribution. Nonlinear Euler equations in 3D with MPI, hybrid

(MPI+OpenMP) and different OpenMP parallelization strategies were solved by [14]. The work concluded that using the first touch placement policy was essential on NUMA machines and in the absence of this policy; the page replication technique performed better than page migration. An alternate approach of copy-inside-copy-back (CC) to achieve data locality was used by [15]. The CC approach is applicable only for coarse grain parallelism where the ratio of computation time to copy operation time is high.

The literature also describes different strategies and methods used for porting MPI codes or legacy serial codes to OpenMP with the help of different performance tools. Code porting strategies were suggested with code tuning using prof, ssrun, perfex on a single processor for a Large Eddy Simulation code [16]. In their study, the use of first touch policy for OpenMP execution was emphasized along with a demonstration of load balancing in arrays for cache optimization. Similarly Hackenberg et al [17] recommended the first touch implementation for data initialization in OpenMP.

The conversion of combustion codes to run in parallel using OpenMP was done by [18]. They tested the memory usage, scheduling strategies, first touch placement policy, loop fusion (combining loops together), loop collapsing for better load balancing and the usage of parallel regions with the 'nowait' OpenMP clause. It was concluded that the parallel region in OpenMP lacked a reduction clause outside of the context of a work-sharing loop and OpenMP I/O was 100 times slower than MPI. It was shown that a large page size improved OpenMP performance by 30% along with performance optimization using 'nowait' loop blocking for efficient cache utilization and processor binding using environmental variables [19]. Large pages of 64 kb resulted in reducing the overhead of translating a program page address to a memory page address (reducing the TLB miss rate and L1 cache miss). Simulations were performed on a large benchmarking code SPECseis, which is a seismic process analysis suite [20]. OpenMP and MPI performance was found to be similar for up to four processors.

More recently hybrid code usage is increasing across various scientific applications due to the shift toward the multi-core architecture. The performance study of the NAS benchmarks for four different parallelization strategies (MPI, OpenMP, MPI-OpenMP and two hybrid methods) on symmetric multiprocessors (SMP) [21] indicated that pure MPI performed best with high speed interconnects but on slow networks hybrid codes

performed better than MPI. In another investigation [22] on a 3D image construction code, hybrid code (MPI+OpenMP) performed 10% better than MPI as the OpenMP part took advantage of the lower latency of shared memory threads across processors. On contrary, it was found that MPI outperformed hybrid (MPI+OpenMP) runs, with static OpenMP scheduling, for a CFD solver [23].

The current work is motivated by the flexibility afforded by OpenMP in multiphysics applications where the physics does not permit a single mode of parallelism but to multiple modes for optimal parallel efficiency. The solution of grid based field equations like the Navier-Stokes equations map to the domain decomposition mode of parallelism, whereas discrete N-body type of computations map to the discrete particle numbers. When both are combined in a single multiphysics code, domain decomposition type parallelism for the field equations is not an efficient choice for parallelizing the N-body problem. In such situations OpenMP is more flexible with less overhead in changing from one mode of parallelism to the other – that is if OpenMP can be made to efficiently scale to a large number of processors. Thus, the objective of this work is to parallelize a large production CFD code (>100,000 lines) with OpenMP and show that its scalability can match that of MPI over O(100) processors. In this research fine grain OpenMP parallelism has been implemented and the performance on 256 cores investigated. The simplicity of the loop level OpenMP parallelism is maintained in the implementation by prudently using parallel initialization of data and process placement tools. Finally, the potential use of OpenMP is presented in a multiphysics fluid-particulate system by comparing its scalability to MPI.

In this chapter, the structure and capabilities of the GenIDLEST code are listed. Performance measurement and optimization techniques used are briefly discussed followed by the description of the system and the test problems used for performance testing. The scalability results of OpenMP, MPI and Hybrid execution of GenIDLEST are discussed and finally the scalability results for a multiphysics fluid-particulate system are enumerated. In this chapter the dual core system refers to two individual processing units (called cores in this sense) on a chip whereas single core refers to a single processing unit on the chip.

Methodology

The scalability study and testing of the OpenMP API on a real world CFD simulation code called GenIDLEST [24, 25] (Generalized Incompressible Direct and Large Eddy Simulation of Turbulence) is performed in this study. GenIDLEST is a computational fluid dynamics package that solves for the velocity, pressure, temperature and species fields in turbulent multi-phase flows. It solves the time-dependent Navier-Stokes and energy equations in a multiblock generalized body-fitted coordinate system and is used extensively in propulsion, energy and biology related applications to complex multi-physics flows [26-29]. At its core, the code uses a finite volume formulation with second order central difference discretization scheme. A fractional step algorithm using semi implicit Adams-Bashforth/Crank-Nicolson, or a fully-implicit Crank-Nicolson method for a predictor step, with the corrector step solving a pressure Poisson equation to satisfy mass continuity [24, 25] is implemented. The core algorithmic features are shown in Figure 2.1. Supplementing the core algorithm are different options for discretization, linear solvers and boundary conditions. Algorithmic modifications are also implemented for incorporating property variations with temperature, for dynamic moving grids, for incorporating coupled fluid-particle transport physics, turbulence models, structural models, and coupled solid-fluid heat transfer models. The code has been under development since the mid-90s at the National Center for Supercomputing Applications (NCSA) and has undergone a number of rewrites to adjust the data structures, memory allocation procedures, I/O policies, parallelization strategies to take maximum advantage of current day hierarchical memory, and parallel architectures mostly within the context of MPI. The code spans over 300 subroutines and more than 100,000 lines. It uses an overlapping multiblock grid framework and its computational identity is best characterized by structured grids, sparse linear algebra, coupled with N-body computations for dispersed dense particulate flows. In the past, GenIDLEST has been used for various tests including power modeling, OpenMP barrier algorithms, etc. [30-32].

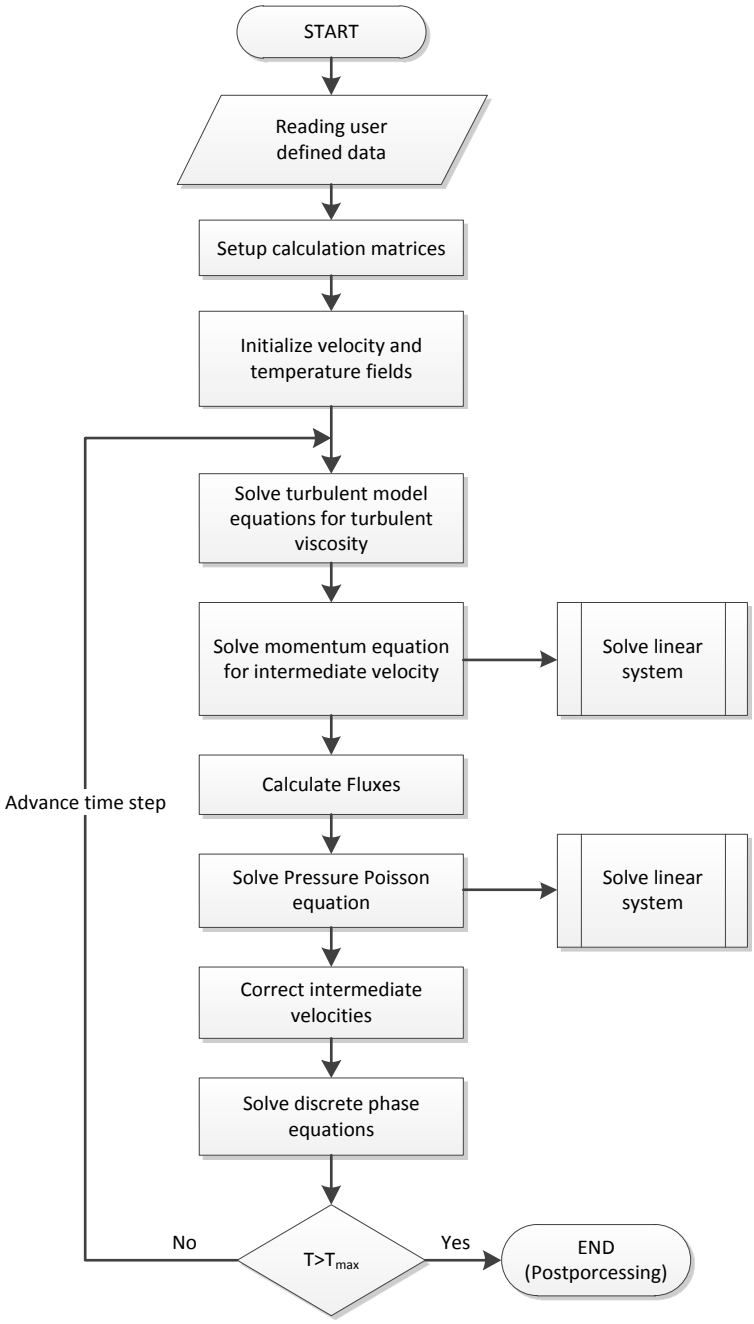

Figure 2.1 GenIDLEST computational structure for solving the Navier-Stokes and energy equations using a fractional step method.

Data distribution and parallelization

The overlapping multiblock framework used in GenIDLEST provides a natural framework for parallelization. The degree of overlap between adjoining blocks in GenIDLEST is dictated by the order of spatial discretization used and is one

computational cell wide. This offers the framework within which independent computations can be performed in each block, provided that the ghost cell has been updated at inter-block boundaries by a suitable data transfer from the adjoining block. Within this framework, Figure 2.2 illustrates the data structure and the multiple levels of parallelism which can be extracted. The mesh generation process has two implicit constraints imposed on it: the number and size of blocks dictated by the physical complexity of the geometry; and by the degree and efficiency of parallelism sought. Depending on the total number of mesh blocks and the degree of parallelism sought, each node can have multiple blocks residing on it as shown in the Figure 2.2. It is to be noted that the total number of blocks is always dictated first and foremost by the geometrical complexity of the computational domain, with the degree of parallelism sought being a secondary but important consideration. Hence, multiple overlapping blocks are the norm even though pure OpenMP parallelism does not explicitly require this mapping. In the context of Figure 2.2, all blocks in a pure OpenMP would map to a single shared memory node with multiple processors or cores, whereas with MPI, the blocks would be spread across multiple nodes or processors or cores as the case may be. Further within each block, “virtual cache blocks” are used. The 'virtual' blocks are not explicitly reflected in the data structure but are used only in the solution of linear systems, which are the most time consuming part of the fluid phase calculations (between 50-90% of computational time). The motivation to construct much smaller 'cache' blocks is to extract performance on cache based hierarchical memory systems by using them as the basic sub-structures in a two–level domain decomposition additive Schwarz algorithm to precondition Krylov based solvers [5]. In this method, the full system of equations is sub-structured into smaller sets of overlapping domains, which are then solved individually in an iterative manner, updating the boundaries periodically such that the global system is driven to convergence. Each subdomain is smoothed with an iterative method such as the Jacobi method or Symmetric Successive Over-relaxation (SSOR) or Incomplete LU (ILU) decomposition. By sub-structuring the large system into smaller systems that are designed to fit into cache memory, main memory accesses are minimized, and result in large single processor performance gains.

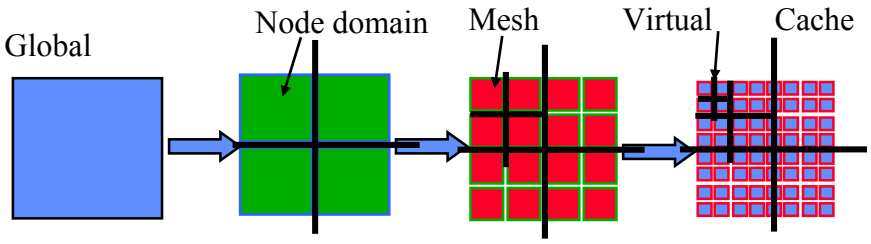

Figure 2.2 Data structure and mapping to cores and threads with different programming paradigms used in GenIDLEST

The data structure in GenIDLEST lends itself to multiple modes and levels of parallelism. For example, a 256 block geometry can be spread across a single shared memory node on a large SMP with OpenMP threads acting independently on each block or a collection of blocks, or spread across multiple processors on a distributed memory architecture using MPI. It can also use hybrid MPI-OpenMP parallelism across blocks. The virtual cache blocks provide a further level of parallelism which can be exploited by using multi-level parallelism in OpenMP.

Communication

In GenIDLEST, the inter block boundary communication to exchange variable values among processors is done in a separate subroutine called *exchange_var*. The subroutine *exchange_var* collects the block boundary values of a variable from all the blocks sharing the same memory into a buffer. The data exchange is performed in a similar manner with MPI and OpenMP. For MPI transfers to remote memory, MPI isend/ireceive operations are used, whereas inter-block transfers in local memory are done using copy operations. Hence, in the MPI framework, a combination of local copies and MPI message passing is used compared to pure OpenMP operation which uses only local copies. Using a ghost cell in the block topology of OpenMP has several advantages and chief among them is complete portability between an OpenMP run, a MPI run, and a hybrid run. Also, the ghost cell data is accessed multiple times by the OpenMP thread on which the block resides and by eliminating the ghost cell, the thread would be forced to fetch ghost cell data from another thread adding to the computational overhead. The cost of multiple remote thread accesses during computations overshadows the cost of retaining the ghost

cell and doing a single copy in *exchange_var*. These communication overheads are reflected within data generated by performance tools under the time spent in the calls to the subroutine *exchange_var* for both MPI and OpenMP.

Performance measurement and optimization

The OpenMP run time of the GenIDLEST application is measured using MPI function MPI_Wtime and cross checked with FORTRAN function date_and_time. The performance optimization tools are used iteratively, to check for code consistency, accuracy and efficiency. These tools and their use for GenIDLEST performance tuning is discussed in this section. Intel FORTRAN compiler, version 11.1.038, with the compiler optimization option –O3 is used.

Code consistency

The parallel execution of GenIDLEST using MPI has been verified in the past in various publications [33-35]. In this study the emphasis is on using OpenMP in a scalable, accurate and consistent manner which is tested using different test cases and software tools.

The TotalView debugger is used for finding memory related issues including checks for memory leaks and stack memory usage. OpenUH is an open source compiler suite for OpenMP 2.5 in conjunction with C, C++, and Fortran 77/90/95 and the IA-64, x86, x86_64 Linux ABI and API standards [36]. The OpenUH FORTRAN compiler is mainly used for checking the scope of variables in OpenMP. Intel Thread Checker is used for checking parallel consistency of the code in both MPI runs and OpenMP runs.

Performance tools

Monitoring the performance of parallel computing applications needs specialized tools. In this study, hardware counter based performance tools PerfSuite and TAU are used for analyzing the performance of GenIDLEST.

NCSA developed PerfSuite [37] which is a small set of light weight performance monitoring tools and libraries based on PAPI (Performance Application Programming Interface). PerfSuite operates in one of two primary modes: in “counting mode”, one or more user-selected hardware performance events are activated and reported at the end of

monitoring as aggregate event counts along with associated derived metrics; in “profiling mode”, the hardware performance event under measurement periodically triggers an interrupt on a user-selected overflow interval, resulting in a source-level profile gathered through statistical sampling. This study employed PerfSuite in both counting and profiling modes in order to gain a comprehensive view of the performance characteristics of GenIDLEST.

In profiling mode, event based sampling over total number of cycles is performed to obtain detailed information about the time spent in various subroutine calls and line summary of different function files. This performance data is used to identify the bottlenecks in the subroutines. Subroutine by subroutine comparison between MPI and OpenMP results is helpful in understanding the code signatures.

In counting mode, the overall performance statistics of the program is obtained (cache miss ratio, memory bandwidth usage, etc). The performance statistics obtained in the counting mode are used for the code behavior monitoring and optimization.

As an example of the usefulness of PerfSuite analysis, several rounds of counting runs for both MPI and OpenMP were done to identify the most significant stall event (BE_L1D_FPU_BUBBLE_L1D) which was then used to profile GenIDLEST application. Intel's documentation for Itanium 2 hardware performance events defines this as "the number of full-pipe bubbles in the main pipe due to stalls caused by either the floating point unit or L1 data cache". A "bubble" refers to a condition that prevents the processor from making forward progress. In both MPI and OpenMP, the subroutine line contributing the largest number of counting samples is in the pre-conditioning function, indicating that the most stalls occurred there. The stall information directly relates to the subsystem “cache” block size in GenIDLEST since the CPU is data starved because of the latency and bandwidth associated with memory access. Thus by adjusting the virtual cache blocks higher cache hit ratio can be achieved.

Detailed information about the MPI communication time and time spent in OpenMP parallel regions is not available through PerfSuite. Due to these limitations, the TAU (Tuning and Analysis Utilities) performance system [38], developed at the University of Oregon, is currently used to obtain the detailed performance regression analysis. Primarily the results from TAU are used as a check for the PerfSuite results.

The performance tools generate data for each thread and to analyze the system wide parallel performance of the application it is important to be able to analyze potentially large amounts of performance data effectively. ParaProf, a performance profile visualization tool that is part of TAU, is used in this study to obtain a graphical visualization of the vast amounts of performance data.

Placement and locality issue

Two key issues with applications parallelized with OpenMP are process/thread and data placement. Keeping the data local to the core gives vastly improved performance. This can be controlled using SGI's dplace tool and SGI's first touch policy and SGI's thread affinity tools – dplace/omplace. Since the parallelization strategy used is fine grain parallelized or fork and join parallelization [39], there is a possibility that data assignment to threads could migrate from one core to another during the execution of the code. To avoid this cost, the thread affinity is kept in check by using static scheduling, which is the default on most of the systems.

First touch placement

On SGI Altix systems, the data placement is done by first touch placement policy. According to the policy, the data is placed within a node that contains that core which allocates and initializes the memory block first. Applications using OpenMP require that the data initialization be done in parallel in order to implement the first touch placement. The initialization of arrays is performed in parallel such that each core initializes data that it is likely to access later for calculation. This configuration ensures that the data is placed where it is most frequently accessed. This placement policy has no effect on the applications using MPI parallelization since the data is distributed manually to each core in MPI. The example of this code change is shown in Table 2.1 and shows that the original FORTRAN 90-style array syntax was changed to an OpenMP parallel loop in order to effect the proper per-thread initialization.

Table 2.1 Code snippet of the modifications to implement first touch policy

Original Code	Code modified for first touch policy implementation
buf2ds=0.0 buf2dr=0.0	c$omp parallel do private(m) do m = 1, m_blk(myproc) buf2ds(:,:,:,m)=0.0 buf2dr(:,:,:,m)=0.0 enddo c$omp end parallel do

SGI tools for processes placement

dplace and omplace

Thread affinity restricts execution of certain threads to a set of the CPUs in a multiprocessor system. Depending on the topology of the system, thread affinity can have a remarkable effect on the execution speed of an application. omplace and dplace are tools provided by SGI for process or thread placement on NUMA systems. The omplace tool is particularly applicable for hybrid MPI/OpenMP codes where successive threads are placed on unique CPUs. omplace is easier to use and has stricter placement policies compared to dplace, even though it is essentially a wrapper around dplace.

After a few experiments with thread placement, it was concluded that the option dplace –s1 –x2 gave the best results for OpenMP execution. For this placement to work, two additional cores are requested for correct dplace functioning during OpenMP runs. Proper thread placement was verified by observing CPU/thread assignments at runtime through standard Linux facilities (e.g., the 'ps' command and the /proc filesystem). Under SGI's ProPack software stack and MPT MPI library, most MPI applications are launched by mpirun and use N + 1 processes. The first process is the MPI helper process which is mainly inactive and usually does not need to be placed. The option -s1 causes dplace to not place the MPI helper process that is mostly inactive. The option –x provides the ability to skip placement of processes and it is recommended for Intel OpenMP applications be placed using -x2 when using the NPTL POSIX threads implementation under Linux.

In addition to the SGI's placement tools, the OpenMP runtime library of the Intel compiler has the ability to bind OpenMP threads to CPUs. The KMP_AFFINITY is an

environmental variable for setting the thread binding. It is set to “disabled” to avoid interfering with the correct functioning of SGI’s dplace utility.

dlook

dlook is an SGI tool for showing process memory map and cpu usage. This tool is used to probe and verify the correctness of CPU and data placement across nodes in terms of number of memory pages.

Memory management

Stack size

The stack memory size available varies with different OS (operating system) distributions. In OpenMP framework, there are multiple thread private copies of arrays created during execution of an OpenMP parallel loop. This puts additional demands on memory and makes the memory management more difficult. Considering the limited amount of stack memory available due to the above mentioned constraints, the majority of the variables need to be allocated on the heap memory. The heap memory allocation allows the execution of large problems using GenIDLEST but slows the execution due to the inherently slow nature of heap memory. On the other hand excessive use of stack memory with OpenMP private arrays leads to stack overflows. In the Intel Fortran compiler, when -openmp compile flag is used, the local arrays become automatic arrays and are placed on the stack memory by default. To add to this, the Intel Fortran compiler uses stack space to allocate a number of temporary or intermediate copies of array data which piles up in the stack memory. Hence, memory management for a large code such as this is a challenge and a careful balance has to be struck between stack and heap memories.

When allowed, the stack memory size limit is removed so that more variables can be allocated on the stack memory. The Linux command 'ulimit –s unlimited' is used in this study to obtain the maximum possible stack memory size.

The environmental variables OMP_SLAVE_STACK_SIZE and KMP_STACKSIZE which govern the thread private memory and thread private stack size, respectively, are

used for allowing better memory management of thread private data. The sizes of these memories vary as per the application.

diff_coeff subroutine

The diff_coeff subroutine calculates the gradient operator coefficients at cell faces for momentum and energy equations. The gradient operators are then used to construct the Laplacian operator. The code snippet of the diff_coeff subroutine is shown in the Table 2. In the diff_coeff module a large number of temporary arrays are created.

To overcome the above challenges a strategy of converting local arrays to global arrays is applied. Thus the local arrays are selectively made global by allocating them in heap memory as shown in the Table 2.2. These changes in the code considerably reduce the demands on stack memory by allocating multiple arrays in the heap memory. The option of additional compiler flag of '–heap-arrays' also locates all the local arrays in heap memory but can have unwanted effects on data sharing and was found to slow the execution of GenIDLEST.

Table 2.2 Code snippet showing the modification in the diff_coeff subroutine for efficient memory management

Original Code	c$omp parallel do private(m,i,j,k,tauc,ii,jj,kk,n,nf), c$omp+ private(ib,ie,jb,je,kb,ke,i_f,i_l,j_f,j_l,k_f,k_l) c$omp+ private(ad_ci_xi_e, ad_ci_xi_w, ad_ci_eta_nw, ad_ci_eta_sw, c$omp+ ad_ci_eta_w, ad_ci_eta_ne, ad_ci_eta_se, ad_ci_eta_e, c$omp+ ad_ci_zeta_wh, ad_ci_zeta_wl, ad_ci_zeta_w, ad_ci_zeta_eh, c$omp+ ad_ci_zeta_el, ad_ci_zeta_e, ad_cj_xi_se, ad_cj_xi_sw, c$omp+ ad_cj_xi_s, ad_cj_xi_ne, ad_cj_xi_nw, ad_cj_xi_n, c$omp+ ad_cj_eta_n, ad_cj_eta_s, ad_cj_zeta_sh, ad_cj_zeta_sl, c$omp+ ad_cj_zeta_s, ad_cj_zeta_nh, ad_cj_zeta_nl, ad_cj_zeta_n, c$omp+ ad_ck_xi_el, ad_ck_xi_wl, ad_ck_xi_l, ad_ck_xi_eh, c$omp+ ad_ck_xi_wh, ad_ck_xi_h, ad_ck_eta_nl, ad_ck_eta_sl, c$omp+ ad_ck_eta_l, ad_ck_eta_nh, ad_ck_eta_sh, ad_ck_eta_h, c$omp+ ad_ck_zeta_h, ad_ck_zeta_l) ……. c$omp end parallel do
Modified code for efficient memory management	c$omp parallel do private(m,i,j,k,tauc,ii,jj,kk,n,nf), c$omp+ private(ib,ie,jb,je,kb,ke,i_f,i_l,j_f,j_l,k_f,k_l) ……. c$omp end parallel do

Computational details

The lid driven cavity problem [40] is a common test case in mechanical engineering simulations. The problem has a simple geometry but still has complex flow features. From the computational viewpoint, the problem set up is done such that during computations most of the subroutines in the time integration loop are used to obtain a true measure of the speedup and scaling capabilities of the code.

The total number of processors used in the experiments is always a power of two as the speedup study and scaling analysis with load balanced problems can be easily performed with this configuration. The performance testing was done on NCSA's SGI-Altix system which has a Single System Image (SSI) and a DSM architecture with Intel Itanium-2 processors. It consists of three partitions, which is refereed in this chapter as compute-1, compute-2 and compute-3. The important differences between the various partitions are the number of cores and memory per core. Of the three systems, compute-2 (Altix 3700 with 2 cores/node and 12 Gbytes memory/node) has the highest memory per core, whereas compute-3 (Altix 4700 with 4 cores/node and 12 Gbytes memory/node), the only dual core system, has slightly higher memory per node than compute-1 (Altix 3700 with 2 cores/node and 4 Gbytes memory/node).

The speedup and scalability experimentations are done using two different problem types, fixed problem and scaled problem. In a fixed problem, the problem size remains constant and the workload per core decreases with increase in core count to obtain the speedup characteristics (also known as "strong scaling"). The problem size is about 16 million grid nodes for compute-1 and compute-2. On compute-3, due to a limit on the maximum cores available, the problem is scaled down to 8 million grid nodes in order to solve the problem on up to 128 cores. In a scaled problem, the data used for calculations per core remains constant (also known as "weak scaling"). The weak scaling study of GenIDLEST is performed on compute-2. In this study, each core is assigned a computational block. Each computational block consists of 65536 grid nodes. Unidirectional stacking of computational blocks in the z-direction is done to construct the problem geometries up to 32 blocks. A 64 block geometry is constructed by stacking 32 similar blocks in the y-direction to the existing 32 block geometry. Similarly, for constructing a 128 block geometry, 64 more blocks are stacked in the x-direction to the

existing 64 block geometry. Finally, for the 256 block problem another 128 blocks are stacked in the z-direction. This particular stacking strategy was chosen in order to keep the physical dimensions of the problem the same for different number of blocks while avoiding cells with high aspect ratios.

Scaling results and discussion

The scaling performance of the GenIDLEST code for 10 time steps of fluid simulations using MPI, OpenMP and hybrid parallel programming paradigms is discussed in this section. The tuning of the MPI version of the GenIDLEST code has been carried out over the years including employing an efficient communication strategy by overlapping computations with communications. Single core performance optimization has been done in the past and is not included in this study since the performance of GenIDLEST using various parallel programming models is the focus here. Some details about performance optimization on a single core can be found in other GenIDLEST studies including [5, 41].

Initial results

Initially the performance of the GenIDLEST code was observed to be significantly worse for OpenMP execution compared to MPI. For a 2 million grid cell geometry executed with 8 OpenMP threads and default Intel compiler version 10.1.017, the OpenMP execution took 498 seconds of wall clock time, whereas the MPI execution took about 69 seconds. It was identified that the Intel compiler version 10.1.017 had a compatibility problem with *dplace* that effectively serialized thread execution within parallel regions. The compiler was then replaced with version 11.1.038 and the resulting OpenMP wall clock time decreased to around 169 seconds. The OpenMP version was then profiled with PerfSuite to identify the top CPU time consuming subroutines and lines. Based on the profiling results, parallel initialization of some arrays was done to ensure favorable data locality through first touch data placement, and the wall clock time decreased further to around 84 seconds. Further tuning of the code decreased the execution time to about 70 seconds. This performance evolution shown in Figure 2.3 clearly shows the importance of thread affinity or the process placement and first touch data placement.

After all such modifications, the study of performance comparison between MPI, OpenMP, and hybrid models on NCSA's SGI Altix system is carried out.

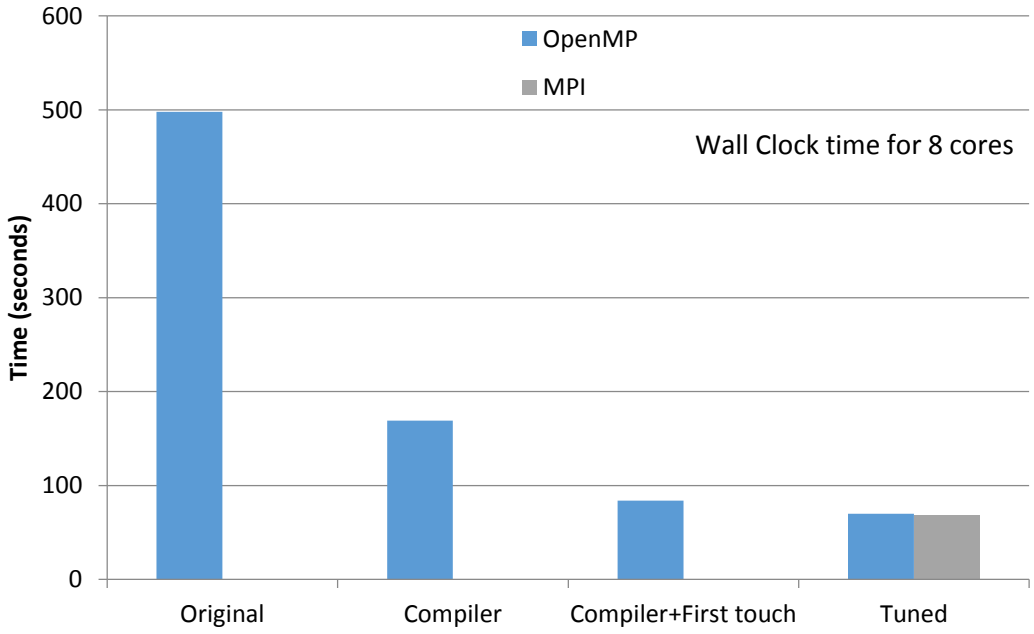

Figure 2.3 Wall clock time for a 2 million grid cell geometry executed using 8 OpenMP threads depicting GenIDLEST performance evolution with various modifications for OpenMP parallelism.

GenIDLEST profiling

The GenIDLEST code is fairly large and the identification of the code signature based on time spent in various subroutine calls is vital in performance prediction [42]. The analysis of individual code subroutines is obtained in the profiling mode of PerfSuite. In Figure 2.4, the most time consuming subroutines on a single core of compute-2 for the two extreme problem sizes are shown. The two problem sizes are compared to gauge the relative time spent in subroutine calls. These fifteen subroutines cover over 90% of the total run time for the small problem (65536 grid nodes in a single block geometry) and about 80% for the large problem (16 million grid nodes in a 256 block geometry).

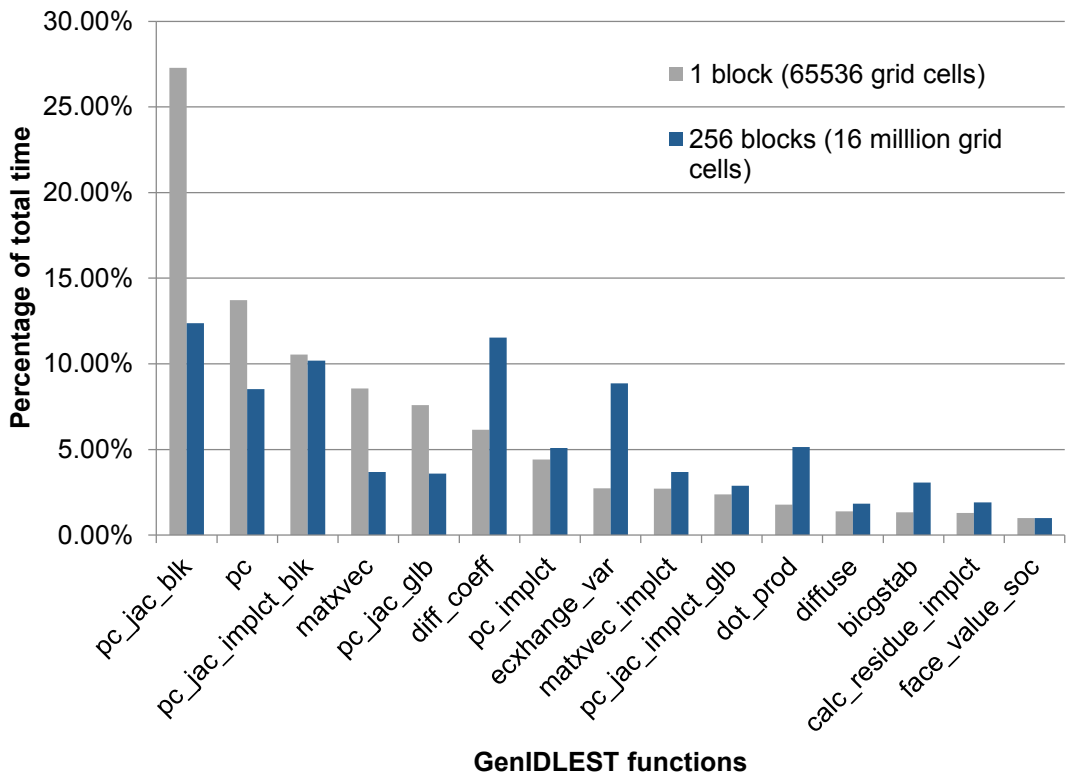

Figure 2.4 Percentage time spent in important GenIDLEST subroutines on a single core of compute-2. The two cases of 65,536 grid cells and 16 million grid cells are compared.

In the subroutine labeling convention used in GenIDLEST, subroutine names with '*pc*' indicate that it is a subroutine in the application of the preconditioner in the Krylov method (BiCGSTAB in this case) for the iterative solution of the pressure and momentum equations. Subroutines with '*implct*' in the name represent the subroutines associated with the solution of linear systems generated in the momentum equation only. The subroutine *matxvec* is a sparse matrix-vector multiplier called by the Krylov method used in the solution of the momentum and pressure equations.

The *diff_coeff* subroutine calculates the gradient operator coefficients at cell faces for the momentum and energy equations followed by the calculation of the Laplace operator. The *diff_coeff* subroutine is not only memory intensive, but also performs a large number of floating point operations, and thus takes comparatively more execution time for the large problem size. For the larger problem size, the memory access cost for inter block data copy in the *exchange_var* subroutine is more, resulting in higher time consumption. It is noted that *exchange_var* is called multiple times to check if any boundary updates

are required even in a single block geometry. Due to the increase in memory access cost for memory intensive subroutines, the relative time spent in the preconditioner is less for the larger problem.

Single core system performance

Weak scaling study

The weak scaling study is done with multi block problems, where each block consists of 65,536 grid nodes and is assigned to a processor core. The performance results for constant load per processor case (scaled problem) on the compute-2 system are shown in Figure 2.5. The trend in the execution time for both MPI and OpenMP is almost identical and constant indicating good scaling up to 32 cores. Beyond 32 cores, both OpenMP and MPI performance deteriorate as the communication overhead increases, yielding longer execution times. The PerfSuite performance regression analysis shows that the OpenMP runs spend an increasingly longer time in the OpenMP library calls and that the *exchange_var* subroutine takes a higher percentage of total execution time with the increasing core count, signifying an increase in the communication cost with number of cores.

The jump in the execution time from one core to two cores is analyzed. The performance regression analysis using PerfSuite revealed that almost all the subroutines take more time to execute for both MPI and OpenMP when the number of cores is changed from one to two. In a single core problem, there is no message passing between blocks as there is only one block, whereas for a two core problem, unidirectional data communication between blocks is introduced. The stall cycles waiting on any resource increased by almost 50 % along with an increase in cache miss ratios supporting the idea of communication overheads. Figure 2.6 shows the overall time consumed in the major modules, consolidated from different subroutines to identify segments of computation. The momentum and pressure linear solvers comprise of the subroutines which solve the momentum and pressure equations, respectively, using iterative Krylov subspace methods with additive Schwarz preconditioners. The diffusion term represents the calculation of diffusion coefficients for all momentum equations including the calculation of the gradient and Laplacian operator in *diff_coeff*. As mentioned earlier, the exchange

variable module updates the boundary surface values between adjacent block boundaries mainly for the primitive variables (velocities, pressure, and temperature). “Interpolation” is used to calculate the finite-volume cell-face values of variables using the second-order central difference approximation.

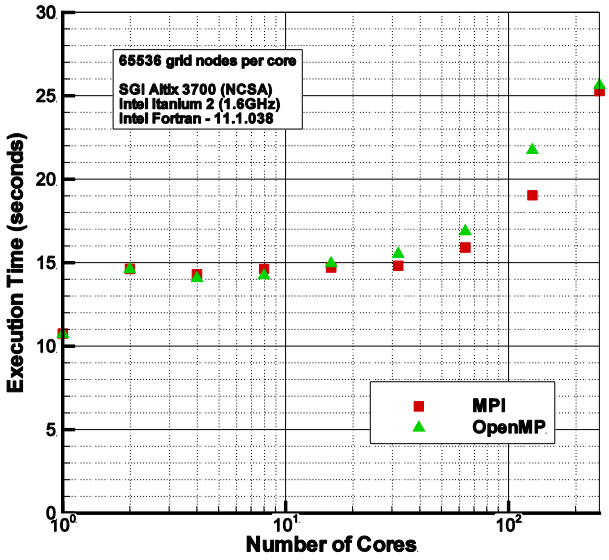

Figure 2.5 GenIDLEST weak scaling performance on compute-2 for simulation of a lid driven cavity problem with 65,536 grid nodes per core comparing MPI versus OpenMP parallelism.

The observed increase in execution time due to communication overheads can be extended to explain the degradation of performance at higher core counts in the weak scaling study. The unidirectional stacking of computational blocks introduces message passing with an average message size of 16,384 double precision words for both sends and receives in the z-direction. As mentioned in section 4, for the 64 block geometry the blocks are stacked bi-directionally, thus messages are now exchanged in two directions. The message size in the z-direction is 16,384 words, and 512 words in the y-direction. Similarly for 128 blocks tri-directional stacking requires data communication in 3 different directions with a message size of 512 words in the x- and y-direction and 16,384 words in the z-direction. This increase in message passing traffic is reflected in the scaling performance. With introduction of message passing there is an increase in run time from 1 to 2 cores. From 32 cores to 64 cores, the bidirectional communication per core further increases the execution time. For 128 cores, the tri-directional message passing increases the communication overhead further, resulting in an increase in

execution time compared to the 64 core problem. Finally, for 256 blocks there is twice the communication traffic as compared to 128 blocks resulting in a further increase in run time.

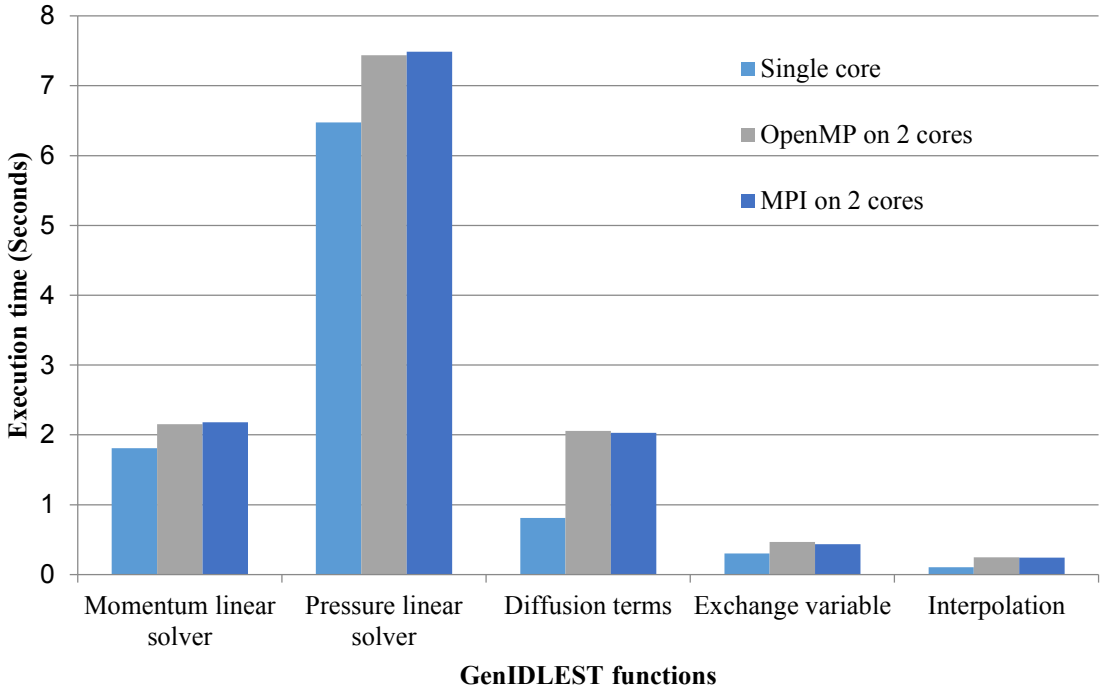

Figure 2.6 Comparison of time spent in important GenIDLEST functions on compute-2 for different core counts and parallelization paradigms for simulation of a lid driven cavity problem with 65,536 grid nodes per core.

Strong scaling study on Compute-1

The speedup results on the Altix 3700 – compute-1 are shown in Figure 2.7. For a fixed-size problem (16 million grid nodes), the strong scaling results can be divided into 3 regions.

1. The sub-linear region
2. The linear region
3. The communication controlled region.

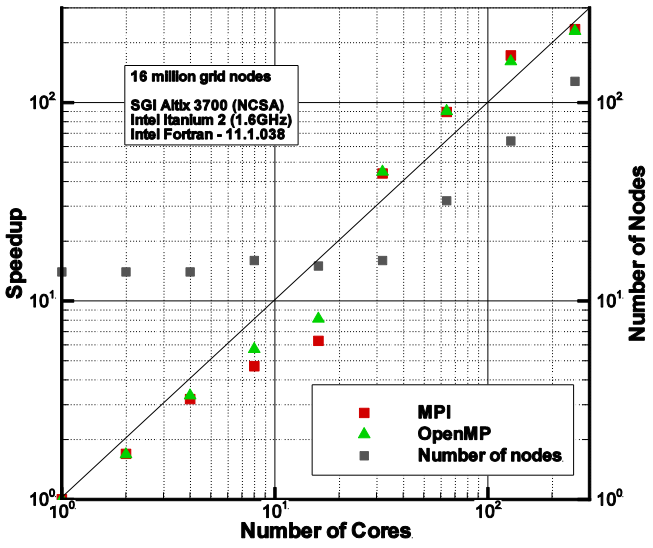

Figure 2.7 GenIDLEST strong scaling performance on compute-1 for simulation of a lid driven cavity problem with 16 million grid nodes. Speedup is shown on left axis with node count on the right axis.

In the sub-linear region, both OpenMP and MPI performances fall below linear scaling. In this region, the total memory required for the problem per node is larger than the system memory available per node. To compensate, the scheduler allocates more nodes for the problem and uses memory from additional nodes. The allocation of remote-to-node memory results in slow remote memory access which is a typical characteristic of a NUMA machine, resulting in performance degradation for up to 16 cores. Performance tools indicate that the floating point units are data starved as a result of the slow remote memory access. That is, the cache miss ratio is high up to 16 cores as the processors have to wait longer for the data from remote memory to be loaded into the cache. The data distribution across nodes was examined using the "*dlook*" utility, which verified that the overall memory consumption of 50GB required a minimum of 13 nodes contributing memory for runs up to 16 cores. Consequently, in the single-core run, 12 nodes (23 CPUs) were not active in the computation but only contributed their local memories. At higher levels of parallelism, data distribution and locality improve accordingly for the strong scaling runs. With an increase in core count beyond 16, the performance abruptly jumps above the linear performance curve. This region, where the performance is above linear performance, is categorized as the linear scaling region. The reason for this abrupt

improvement lies in the fact that the whole problem is accommodated on the local memory of nodes, eliminating the need for data fetching from the remote memory of distant inactive nodes. At 128 cores, the communication overheads increase slightly causing both MPI and OpenMP speedup to decline slightly. This communication controlled region is predominantly observed for the code execution with 256 cores. The overall OpenMP speedup performance on compute-1 is slightly better than MPI.

Strong scaling study on Compute-2

To validate the observations made on compute-1, a similar strong scaling study is conducted on compute-2 which has 12 GBytes of memory per node versus 4 GBytes on compute-1. As seen in Figure 2.8, beyond 8 cores the problem can be accommodated in a core's local memory resulting in both MPI and OpenMP scaling slightly better than linear up to 128 cores. In this region, the memory bandwidth usage is highest for both OpenMP and MPI, achieving linear performance. Beyond 128 cores, the communication overhead starts increasing and there is a slight performance drop with additional cores. Both MPI and OpenMP performance at high core count is similar to compute-1 in the communication controlled region.

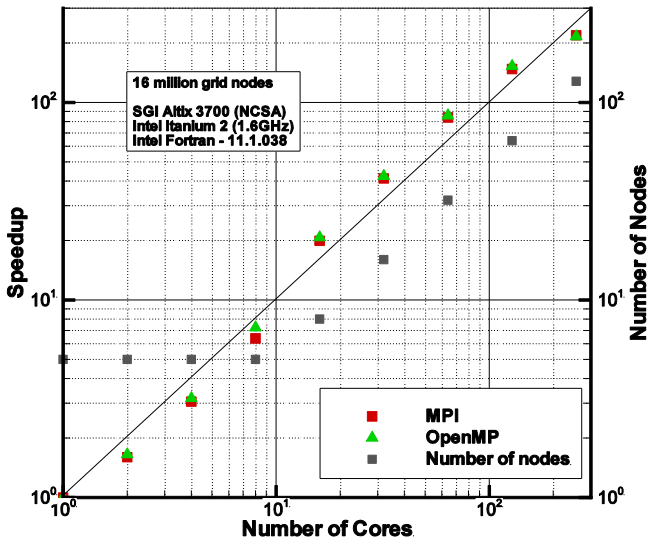

Figure 2.8 GenIDLEST strong scaling performance on the larger memory compute-2 for simulation of a lid driven cavity problem with 16 million grid nodes comparing MPI and OpenMP on left dependent axis. The total number of compute nodes used is listed on right dependent axis.

The average memory bandwidth usage across all processors for OpenMP and MPI paradigm is illustrated in Figure 2.9 with standard deviation bars. The exact counts for the hardware performance events PAPI_L2_TCM, PAPI_L3_TCM and PAPI_TOT_CYC were measured. Here, PAPI_L2_TCM is Level 2 cache misses, PAPI_L3_TCM represents Level 3 cache misses and PAPI_TOT_CYC is total cycles. The Itanium-2 CPUs on the compute-2 system have four hardware performance counters which are the same as the number of events measured. Thus, the counts were exact without any event multiplexing.

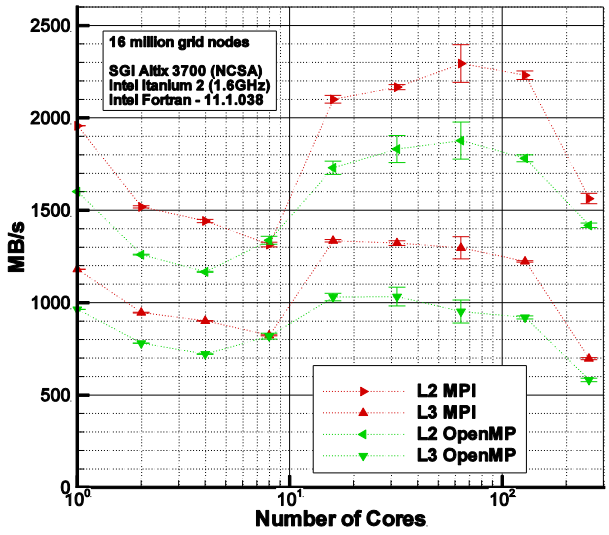

Figure 2.9 Average memory bandwidth usage with standard deviations on compute-2 for different number of cores for a lid driven cavity problem with 16 million grid nodes comparing MPI and OpenMP parallelism.

The bandwidth was determined using the following formulas:

Memory bandwidth used to L2 cache (MB/s)	PAPI_L2_TCM * L2_cache_line_size / (PAPI_TOT_CYC / CPU_MHz)	2.1
Memory bandwidth used to L3 cache (MB/s)	PAPI_L3_TCM * L3_cache_line_size / (PAPI_TOT_CYC / CPU_MHz)	2.2

The (PAPI_TOT_CYC / CPU_MHz) is equivalent to the CPU time spent by the thread. The cache line sizes have constant values for a given type of CPU; for the Itanium-2 CPUs on compute-2, the L2 and L3 cache line sizes were 128 bytes.

Both the paradigms show similar trends of bandwidth usage but MPI has higher L2 and L3 cache memory bandwidth usage compared to OpenMP throughout. When the problem is run on 8 cores, OpenMP uses higher or almost the same bandwidth as compared to MPI and thus the OpenMP speedup performance is better than MPI for this case.

As shown in Figure 2.10, the trends in average (across all cores) floating point operations per cycle explain the speedup performance of GenIDLEST. For OpenMP the floating point operations per cycle decrease up to four cores and then increase up to 64 cores. Whereas, the floating point operations per cycle decrease up to 8 cores for MPI. Then there is a sudden jump in the floating point operations at 16 cores and the curve levels out beyond that. This behavior is consistent with the performance characteristics linked to remote memory access cost.

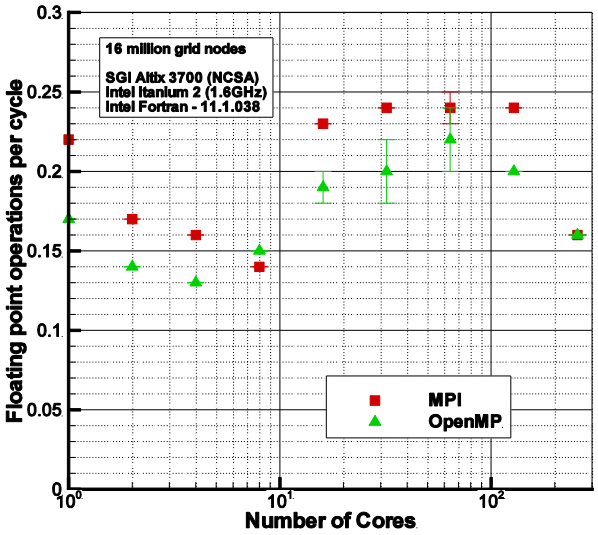

Figure 2.10 Average (across all cores) floating point operations per cycle with standard deviations on compute-2 for a lid driven cavity problem with 16 million grid nodes comparing MPI and OpenMP parallelism.

The average L3 cache miss ratio for both MPI and OpenMP is approximately constant up to 16 cores as shown in Figure 2.11. Beyond 16 cores the cache miss ratio drops resulting in a linear speedup performance.

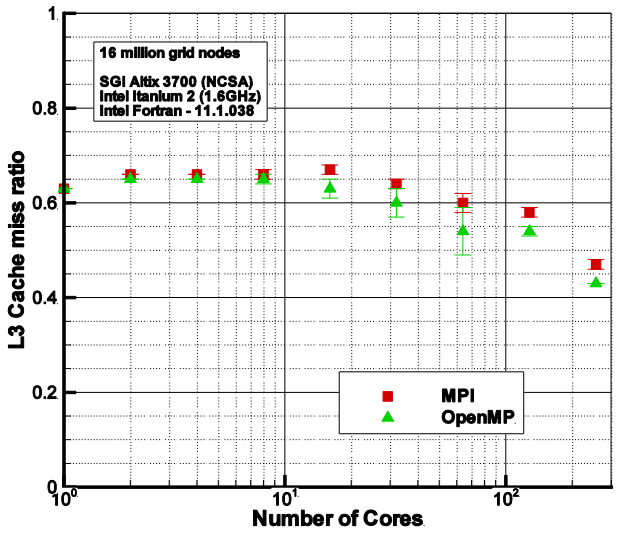

Figure 2.11 Average L3 cache miss ratio with standard deviations on compute-2 for a lid driven cavity problem with 16 million grid nodes comparing MPI and OpenMP parallelism.

Dual core system performance

To contrast the single core performance characteristics, the same strong scaling study is carried out on the dual core Altix 4700 system (compute-3) as shown in Figure 2.12 for MPI, OpenMP and a hybrid study in which OpenMP threads are mapped to each core on a node. The performance trends of OpenMP and MPI are very similar to that obtained on compute-1 and -2. The time required for OpenMP runs is slightly lower than MPI runs for the entire speedup range except for the 128 core run. The hybrid execution time falls between that of MPI and OpenMP, MPI being on the higher side. This behavior is expected since OpenMP has lower latency at the multi-core level across a processor [22].

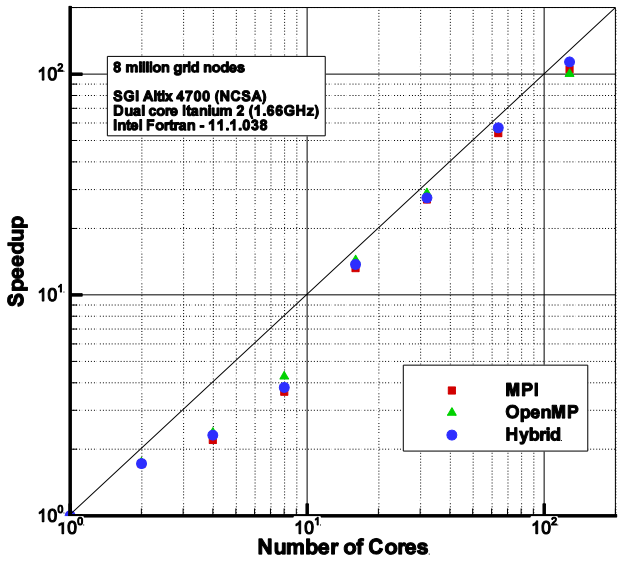

Figure 2.12 Strong scaling performance on dual core compute-3 system for hybrid (OpenMP+MPI), OpenMP and MPI parallelism. Speedup is reported for a lid driven cavity problem with 8 million grid nodes.

Fluid-particulate system

In the previous section it is shown that OpenMP performance can be tuned to be at par with MPI, when both are restricted to the SPMD mode of parallelism. In this section, the flexibility offered by the OpenMP paradigm is highlighted in discrete phase particulate-fluid systems.

In fluid particulate systems (e.gs. circulating fluidized beds (CFBs), rotary kilns, and pneumatic transport) particles are often heavily concentrated in a small part of the full computational domain. In these applications, the spatially decomposed fluid field is calculated in an Eulerian framework, while the particles are treated in a Lagrangian framework. Since the particles are tracked individually in the Lagrangian framework, data associated with each particle (location, mass, properties, velocities, temperatures) needs to be communicated from one fluid domain to the next as they traverse the computation domain.

If the workload associated with these particles is large, as is often the case, then treating them in the domain decomposition framework of MPI can lead to severe load imbalances and inefficiencies. In these systems, while the data structure of the fluid field variables

map to the domain decomposed framework, the particle data structure maps to particle number. In the OpenMP framework, by parallelizing the discrete phase computational loops over the total number of particles and not over the computational grid, the workload can be evenly distributed across all the threads in OpenMP. Whereas in MPI, only those cores on which the particles exist can be used for particle related computations. To parallelize the particulate phase uniformly in the MPI framework, all particle data (which includes fluid field data at particle location) needs to be gathered onto a single processor in order to evenly scatter the particle workload across all the processors. After the particulate phase calculations are performed, the particle data again needs to be gathered and scattered to perform the fluid field calculation, which depends on spatial particle concentration. Thus, at every fluid time step, the entire data structure has to be reshuffled twice. This method of dealing with the discrete phase in the MPI framework leads to large overheads and inefficiencies, to the extent of making the parallelization futile. Other alternate strategies can be devised, but which would be equally complex with large overheads, particularly when the two phases are tightly coupled and interdependent. Hence, in such cases OpenMP has a clear advantage over MPI, in spite of some inefficiency introduced by the non-locality of fluid data which the particles require for their computations. The mismatch in data locality arises because the fluid flow data is distributed by first touch based on the domain decomposition, whereas the particles are distributed by first touch based on particle number. However, the relatively large amount of work done on the dense particulate phase offsets the remote memory access costs associated with the fluid data.

Loosely coupled fluid-particulate system in a lid driven cavity

To exemplify the previous point, a 256 computational block geometry, same as in the previous strong scaling study, is used on compute-1 for the fluid-particulate simulations. In this geometry, the particulate phase was introduced locally in a single computational block. These are point mass particles and the interaction between particles is not considered. The performance of OpenMP and MPI is compared by varying the number of particles and keeping the number of cores constant at 32. Figure 2.13 shows the performance of the GenIDLEST code for 10 time steps of localized dense discrete phase

simulations. As the number of particles is increased, the advantage of being able to switch the mode of parallelism in OpenMP becomes evident – the large number of particles increases the workload on a single MPI process, while the other MPI processes wait for the particle computations to complete. In OpenMP, however, the particle workload is distributed evenly between all the threads yielding much faster execution.

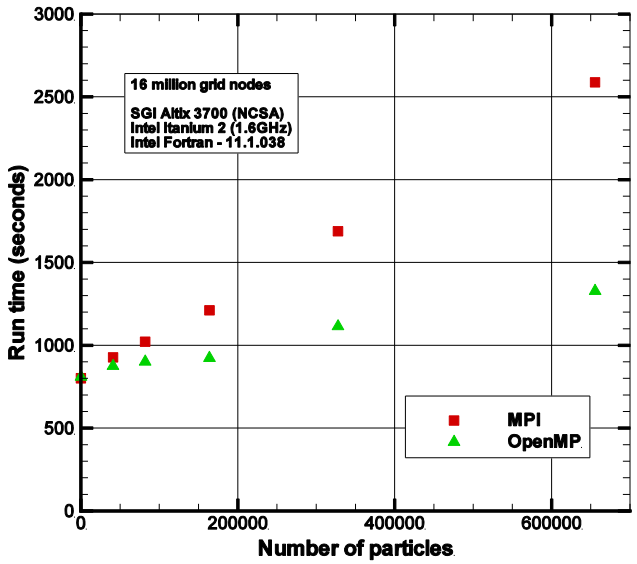

Figure 2.13 Dense discrete phase simulations on compute-1for different number of particles injected locally on single core. A lid driven cavity problem with 16 million nodes is executed on 32 cores with MPI and OpenMP parallelism.

In order to investigate the impact of communication costs for a fluid-particulate system, a lid driven cavity test case with a four block geometry is used. In this problem, 10,000 particles are injected on two blocks each. Figure 2.14 shows OpenMP and MPI TAU profiling data which compares exclusive timings for various subroutines and function calls for 6000 time steps of simulation. The OpenMP profiling data shows almost uniform timings for most of the subroutines across all the threads. Column 1 shows the time spent in particle calculations. In the MPI framework, particle calculations are performed on only two blocks, thus the workload is on only two cores as indicated in column 1 in Figure 2.14. On the other hand, the particle calculations are uniformly distributed on all four cores in the OpenMP framework. While the particulate phase calculations are performed, the other two cores wait which is represented by the MPI_waitall and MPI_allreduce calls taking longer on processor 1 and 4 as depicted in

columns 2 and 3, respectively. The corresponding timings for OpenMP are absent. Column 4 represents the MPI_isend and MPI_irecv operations which take longer on processors 2 and 3, because of the additional particle data which needs to be sent and received between the two processors. Column 5 represents the time spent in the rest of the subroutines. Coalescing the above data, the overall time taken for OpenMP communications (inclusive of time taken in the *exchange_par* subroutine where particle data is exchanged between blocks) was found to be less than 0.01% of total time whereas for MPI it was 10% of the total wall clock time, including wait time.

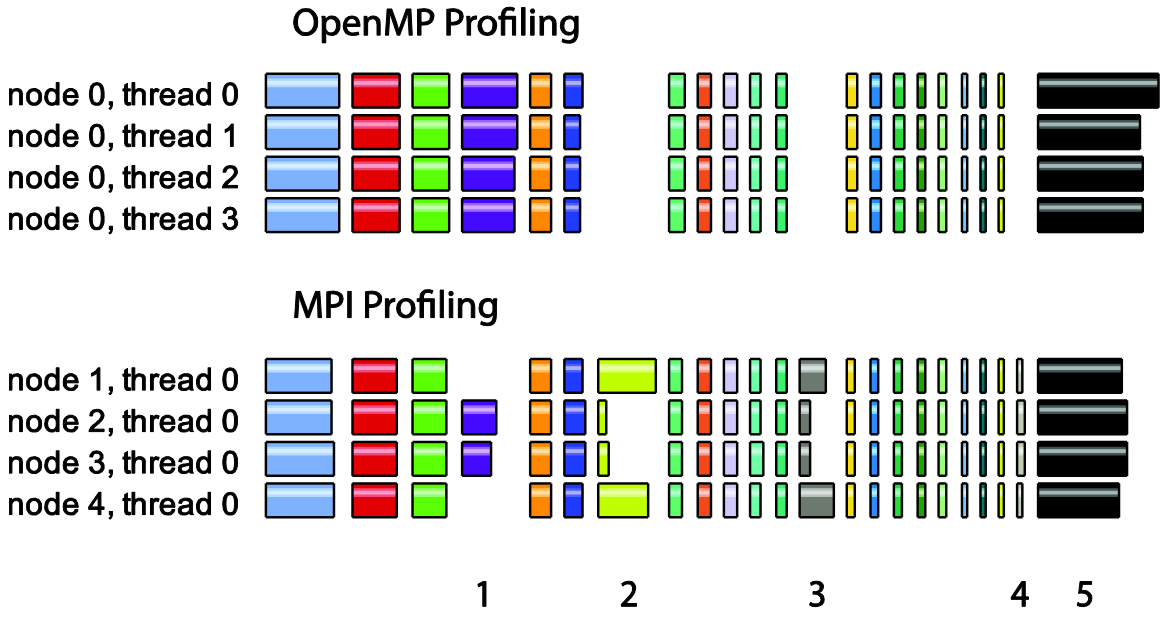

Figure 2.14 TAU profiling analysis of GenIDLEST code for a fluid particulate system on four cores with MPI and OpenMP parallelism. Columns represent time spent in (1) particle calculations; (2) MPI_waitall; (3) MPI_allreduce; (4) MPI_isend and MPI_irecv; (5) other subroutines.

Applicability and future

The results from this study clearly show the utility of the first touch policy, consistent data placement at runtime, and memory management in the OpenMP context, all of which have a direct impact on parallel efficiency of any code. With the anticipated massive growth in core count, more control of data locality is likely to be critical for all OpenMP codes. Data locality and affinity is one of the topics being debated in the context of preparation for OpenMP 4.0. Current work favors both an ability to perform a "next touch" for migrating data explicitly as well as the use of locality specifications rather like

those offered in the HPCS languages X10 (“places”) and Chapel (“locales”). This approach was explored by one of the authors and her group and is the subject of [43]. It is also established that in the realm of solving any type of field or Eulerian equations which are spatially decomposed for parallelization, the flexibility of OpenMP’s light-weight threads have an advantage over MPI because they can be applied to different parallel tasks in a system.

In this study an example of fluid-particulate flow is given, but the same arguments can be applied to multi-physics field equations in which the physics is different in different parts of the computational domain. For example, in atmospheric codes, there might be ice formation in one part of the domain which might require additional local computations. Task based parallelism in OpenMP, which simply cycles through parallel tasks is more appropriate in this case than MPI based spatial decomposition. The same is true for adaptive meshes [3].

Heterogeneous architectures including systems that combine CPUs with graphical processing units (GPUs) and many-integrated cores (MIC) seem to be the future platforms for high performance computing. GPUs require a host CPU, and data must be transferred to and from the GPU explicitly. Their programming requires very careful partitioning of data and work, an overlapping of GPU computations with asynchronous data copying between GPU and CPU, and carefully memory allocation and mapping on the GPU itself. Two different vendor proposals for OpenMP-like directives to support the specification of GPU code and data transfer [44, 45] were input to an OpenMP subcommittee that was formed to define OpenMP extensions for heterogeneous systems. An early outcome is the announcement of OpenACC (see http://www.openacc-standard.org); an initiative led by NVidia that exploits this work by providing GPU-specific extensions that can be easily combined with OpenMP features in an application. Experiences gained from the use of OpenACC directives are expected to contribute to the effort to integrate such features into OpenMP itself.

On these newer architectures, the fluid-particulate flows can be load balanced by offloading the linear system (fluid phase) onto the GPUs and solving the irregular part (particulate phase) on the CPU for simultaneous computations. This particular work sharing strategy should be useful because all the computations for a domain decomposed

fluid phase involve a stencil consisting of data from neighboring cells on the Eulerian grid and this data structure fits the new GPU based architectures. Other strategies such as offloading all the work including particulate workload to GPUs with a modified neighbor search algorithm for faster GPU computations can also be applied. With such work load decomposition strategies and the advent of OpenMP features for the newer architectures, OpenMP would be an attractive parallelization paradigm for irregular applications like fluid-particulate system.

Summary

The OpenMP API gives excellent scalability and speedup when implemented carefully with the use of first touch placement policy and appropriate thread affinity, both of which are critical for scalability over more than a few processors/cores. In addition to these factors, application code scalability also depends on the memory signature of the code relative to the hardware. Strong and weak scaling results are compared for both MPI and OpenMP. The results from weak scaling studies on the GenIDLEST code show the effect of increasing communication overhead as the problem is scaled with the number of cores for both MPI and OpenMP. The strong scaling results show the effect of memory usage on scalability. The parallel performance of OpenMP and MPI paradigms on a single core system is almost identical on ccNUMA architectures. The dual core system shows similar trends as well. The hybrid code (MPI+OpenMP) execution yields almost identical results compared to OpenMP and MPI since both MPI and OpenMP scale closely. In this study for a CFD application, OpenMP performance is shown to be a competitive alternative to MPI on different SGI Altix shared memory machines for up to 256 processing cores.

It is also established that OpenMP threads offer considerable advantages over MPI processes in multiphysics applications which do not adhere to a single mode of parallelism. This is highlighted in fluid-particulate systems, in which the best parallel performance is obtained by switching the mode of parallelism from domain decomposition for the fluid calculations to N-body parallelism for the particles.

3. Parallelism for tightly coupled fluid-particulate system

Introduction

Dense fluid-particle systems are encountered in a wide range of applications in the pharmaceutical and chemical processing industry. The complexity of these multiphase flows makes it difficult to study them experimentally. To gain insight into the internal dynamics of these systems, experimental techniques have to be intrusive because common non-intrusive optical techniques have limitations due to the opaque nature of the particulate phase in three dimensional (3D) flows. Because of such measurement restrictions, it becomes essential to model such multiphase flow using high fidelity computational techniques.

There are two approaches of modeling fluid-particulate multiphase flows, Euler-Euler (E-E) and Euler-Lagrangian (E-L). In the E-E approach or the two fluid model, the solid and fluid phases are treated as interpenetrating media interacting through interphase momentum and energy exchange terms. Only volume or ensemble average information of flow quantities is obtained by the E-E approach which lacks the detailed description of physics at the particle scale. On the other hand, the E-L approach solves the particulate phase in the Lagrangian frame where each particle is tracked, giving details of individual particle behavior, while the fluid is treated in an Eulerian frame. Also commonly referred to as the Discrete Element Method (DEM) [46], this method is widely used in the numerical analysis of dense particulate systems in which the solid volume fractions are typically greater than 40%. In the DEM, each individual particle in the bed is tracked by the application of Newton's laws of motion which calculates particle acceleration due to fluid-to-particle forces, multi-particle collisions, particle-wall collisions and particle body forces. The solid and discrete phases are tightly coupled through interphase exchange of momentum and energy in their respective governing equations [47]. Additionally, a volume fraction of fluid is used in the continuum fluid equations to account for the presence of particulate phase [48].

Particle-particle and particle-wall collision forces are calculated using either a hard sphere model or a soft sphere model. The hard sphere model assumes binary

instantaneous collisions between particles at a single point of contact between collision free periods of flight [49]. The soft sphere model on the other hand takes into account a finite collision time with inelastic deformation of the colliding particles with the inclusion of frictional sliding forces [46]. This model is more appropriate in dense beds with long duration multiple-particle contacts. The DEM model has the advantage that it provides a more fundamental high fidelity approach in calculating bed dynamics at the scale of each individual particle. However, it is computationally expensive. The calculation of collision forces increases the computational complexity of the calculation and also restricts the time step, while introducing additional overheads in a parallel computing environment. Just like ghost cells or overlap regions are required for the parallel solution of the fluid equations, similarly, the calculation of collision forces requires a list of ghost or halo particles at each time step. Since particle locations are dynamic, the list has to be constructed at each time step.

For parallelization, the solution of grid based field equations like the Navier-Stokes equations are best mapped to the spatial decomposition mode of parallelism [1, 10, 13, 50], whereas discrete N-body type of computations are best mapped to discrete particle numbers [51-54]. Thus when grid based methods have to interface with N-body methods in a tightly coupled framework, careful consideration has to be given to the mode of parallelism used in the calculation for optimal performance. There are various algorithms to accelerate parallel N-body calculations of which the most common are mirror domain technique [55, 56], particle subset method [57, 58] and domain decomposition [59, 60]. In the mirror domain technique each CPU has a copy of all the particle data but only works on part of it. The advantage of this method is that there is no communication during computations but it has large memory foot print. The particle subset method involves an even distribution of particle workload amongst CPUs. This method has ideal load balancing at the cost of data communication during computations. For the domain decomposition based parallelization method, spatial decomposition of the computational domain is performed irrespective of number of particles in each domain. This method is easy to implement but does not address load balancing issues when they exist.

The individual components of the CFD-DEM system have been researched extensively. Whereas the work on code parallelization of coupled system of CFD-DEM is limited.

Few techniques for running these components in parallel computing environment have been studied. So far MPI has been the choice for parallelizing coupled E-L kind of systems until recently when a hybrid MPI-OpenMP approach was implemented [61, 62]. Using one-dimensional domain decomposition for both fluid and particulate phase in the MPI framework, a large 3-D fluidized bed was analyzed with 4.5 million particles on 16 CPUs [60]. Domain size dependency study was performed which was inconclusive. Almost perfect speed up was achieved when the processors were increased from 4 processors to 16 processors. In an effort to implement the parallelism of E-L method into commercial packages, Kloss et al. [63] coupled two commercial packages, EDEM for particles and FLUENT for fluid flow using an MPI based domain decomposition strategy for both. The DEM model in which inter-particle collision forces are neglected was shown to give 4 times quicker solution compared to DEM due to the computational savings. In a similar effort, Goniva et al. [64] coupled open source software LAMMPS (discrete phase) and OpenFOAM® (fluid phase).

In order to parallelize Euler-Lagrange model more efficiently, Darmana et al. [56] used domain decomposition for fluid phase using the PETSc libraries and N-body simulations for the disperse phase composed of a maximum of 10^5 bubbles. The mirror domain technique was used for the dispersed phase where the entire disperse phase data was available on each processor at every time step. In contrast to this work, the parallelizing of fluid-particle system by Kafui et al. [58] use the mirror domain technique for the fluid field data and a processor ring communication algorithm in MPI framework. In this effort, Parawise parallelization environment and Parawise communication libraries were used to parallelize an existing CFD-DEM code. The SPMD (single program multiple data) technique was used in which parallelization was done by domain decomposition of fluid domain over the k-direction and N-body decomposition for the particle phase. The particle interactions at the inter-processor boundaries were modeled using min-cut decomposition based on a graph partitioning algorithm for further performance improvement. It was shown that the various strategies used in this work improved the parallel performance on 32 processors for 50,000 particles as compared to work by Darmana et al. [56]. The mirror domain technique where either the fluid phase or the particulate phase is replicated after every time step on all the processors has its

limitations. It is a memory intensive technique and limits the largest problem size that can be computed. The overheads associated with data synchronization at the end of every time step are also significant.

Using MPI parallelism, another study [65] investigated two parallelization strategies for dynamic load balancing. In the first strategy, sub-domains partitioned for CFD and DEM were not identical and coupling between them needed data from other processors which incurred large overheads. For the second strategy, the data partitioning was based on the DEM work load while the CFD data was divided dynamically since the DEM calculations dominated the computation time. The dynamic load balancing using the second strategy gave linear performance till 8 processors and limited performance gains after that till 16 processors. This strategy created additional overheads of repartitioning and redistributing the fluid grid on processors at every time step.

More recent work has taken advantage of hybrid programming. Yakubov et al. [61] used this approach for solving the fluid flow with the dispersed phase of bubbles (without inter-bubble interaction). Domain decomposition for fluid flow in the MPI framework and bubble number (N-body) for dispersed phase using OpenMP at the node level was used. This strategy partially helped to reduce the load imbalance of the system. Using a similar approach, hybrid parallelization on a multi-core cluster was used for a fluid-particulate system [62]. Domain decomposition for fluid phase and N-body decomposition for particulate phase was used. In fluid particulate systems, particles are often heavily concentrated in a small part of the full computational domain which limits the load balancing capability of the hybrid approach.

To summarize, there have been various attempts to combine the domain decomposition strategy with N-body simulation for efficient parallelization of E-L type systems but with limited success. All previous efforts have been fundamentally limited by the static nature of MPI decomposition and the high cost of implementing dynamic modes of parallelism into this framework. Thus, the current work is motivated by the flexibility afforded by OpenMP [3, 4, 50] in multi-physics applications where the physics dictates the use of multiple modes of parallelism for optimal parallel efficiency. When domain decomposition and N-body mode are combined in a single multi-physics code, spatial decomposition type parallelism is not an efficient choice for parallelizing the N-body

problem. In such situations OpenMP is more flexible with less overhead in changing from domain decomposition mode of parallelism to the particle subset type of parallelism. Thus, the objective of this work is to instrument loop level OpenMP parallelism by prudently using parallel initialization of data and process placement tools for a CFD-DEM code. The concept is highlighted in two systems, particle dynamics in a fluidized bed with uniform particle distribution and heat transfer in a rotary kiln with a non-uniform particle distribution.

In this chapter, the methodology used for coupled CFD-DEM code is detailed followed by parallelization and the structure of the code GenIDLEST (Generalized Incompressible Direct and Large Eddy Simulation of Turbulence). Finally the simulation details along with the parallel performance results for fluidized bed and rotary kiln geometries are discussed.

Methodology

As discussed in the previous chapter, GenIDLEST [24, 25] is a computational fluid dynamics package that solves for the velocity, pressure, temperature and species fields in turbulent dispersed-phase flows and is used in this study. Algorithmic modifications are implemented for incorporating coupled fluid-particle transport physics which demand modifications in the parallelism used and are discussed in this chapter.

CFD-DEM Coupling Algorithm

At the start of a time step, first the fluid velocity field and temperature are advanced in time using a fractional-step method with void fractions and interphase exchange terms calculated at the new particle locations from the previous time step. The discrete phase calculation is then invoked using the following steps which are applicable in domain decomposed framework:

1. Locate particles with known (x,y,z) coordinates by assigning them (i,j,k) values on the background grid of each block. During this step, particles which have travelled to another block or processor and cannot be found are packed and sent to the appropriate neighboring block and/or processor to which they have moved.

2. Particles which lie in overlap or ghost cells in blocks are exchanged between adjoining blocks to construct a list of ghost particles on each processor. The ghost particles are used to construct the neighbor list for collision force and inter-particle heat transfer calculation.
3. The neighbor list of colliding particles is constructed by binning the particles in individual particle cells and then cycling through all particles in neighboring cells to identify overlapping or colliding particles.
4. Fluid velocity and temperature are interpolated from the fluid grid to particle locations for calculation of interphase momentum and energy transfer.
5. Particle-particle collision forces and particle-wall collision forces and heat transfer are calculated based on the soft sphere model.
6. Other forces characterizing interphase drag and energy transfer and gravitational forces are calculated.
7. Particle acceleration is calculated and new particle (x,y,z) locations are calculated.
8. Interphase momentum and energy terms are transferred to the fluid grid for inclusion in the fluid momentum and energy equations respectively.
9. Void fractions are calculated on the particle grid and transferred to the fluid grid for inclusion in the fluid momentum and energy equations.

Parallelization and data distribution

This section describes parallelism used in the GenIDLEST framework for fluid phase and particulate phase. The central idea is to use OpenMP parallelization for the CFD-DEM scheme. In order to make sure that the OpenMP performance was optimal, process/thread and data placement was done as follows. Keeping the data local to the core gave optimal performance which was achieved using the first touch placement policy. In first touch placement policy, the data is placed within a node that contains the core which allocates and initializes the memory block first. Hence, the initialization of all arrays is performed in parallel ensuring that the data is placed where it is most frequently accessed. This is supplemented with additional placement tools which during runtime ensured thread/process affinity to a particular processor for the duration of the run.

Fluid field parallelism

The overlapping multiblock framework used in GenIDLEST provides a natural framework for parallelizing the fluid field solver based on spatial domain decomposition. The degree of overlap between adjoining blocks in GenIDLEST is dictated by the order of spatial discretization used and is one computational cell wide for second-order accuracy. This offers the framework within which independent computations can be performed in each block, provided that the overlapping or ghost cell has been updated at inter-block boundaries by a suitable data transfer from the adjoining block. Within this framework, multiple levels of parallelism can be extracted.

The mesh generation process has two implicit constraints imposed on it: the number and size of blocks dictated by the physical complexity of the geometry; and by the degree and efficiency of parallelism sought. Depending on the total number of mesh blocks and the degree of parallelism sought, each node can have multiple blocks residing on it. It is to be noted that the total number of blocks is always dictated first and foremost by the geometrical complexity of the computational domain, with the degree of parallelism sought being a secondary but important consideration. Hence, multiple overlapping blocks are the norm even though pure OpenMP parallelism does not explicitly require this mapping. All blocks in pure OpenMP map to a single shared memory node with multiple processors or cores, and as such do not need an overlap layer because they are mapped to a shared address space. In addition to providing complete portability between MPI, OpenMP and hybrid MPI-OpenMP calculations, since OpenMP threads are applied across blocks, having an overlap region for each block allows complete localization of the data pertaining to that block, thus increasing parallel performance. However, the overlapping block framework requires the overlap regions to be updated in the OpenMP framework by copying data from one memory location to another as opposed to explicit message passing in the MPI framework.

Particulate phase parallelism

The particle phase is best parallelized by distributing the work load based on the number of particles. This method requires a change in the mode of parallelism and will incur large inter-processor communication overheads in the domain decomposed framework

(mostly used in MPI framework) in which particles are attached to blocks decomposed spatially over different processors. Hence the principal strategy is to default to spatial decomposition of the particles based on the fluid grid, but which could lead to large load imbalances if the particles are not uniformly distributed across blocks, which is often the case. In this situation, the flexibility offered by the light weight OpenMP threads to switch parallelism from across spatial blocks to particle numbers (particle subset method), gives OpenMP a distinct advantage over MPI. In this method, the particle phase work load is evenly divided based on number of particles and not based on the spatial decomposition. Under the OpenMP framework all data can be seen by all the threads. Thus, it is possible to separate the particulate phase calculations from grid based calculations and apply a different parallelism scheme.

The particle data distribution introduces some inefficiency by the non-locality of fluid data which the particles require for their computations. The mismatch in data locality arises because the fluid flow data is distributed based on the domain decomposition, whereas the particles are distributed based on particle number, irrespective of their spatial location [50]. Additionally, when the particle neighbor is associated with a different thread it necessitates remote memory access. The non-locality of particle data could happen due to their initial distribution or as a result of the flow physics. However, the inherent load balancing of the particulate phase calculation and the relatively large amount of work done on the dense particulate phase largely offsets the remote memory access costs associated with the fluid and particle data.

Modification for discrete phase under OpenMP framework

Based on the performance analysis, the most time consuming subroutine in the CFD-DEM calculations of dense particulate flow is the 'search for colliding neighbor particles'. In the GenIDLEST framework, the DEM neighbor search calculations are efficiently done by first binning the particles in the fluid cells in which they belong. This is followed by a search in the neighborhood cells (27 in 3D) for particles which overlap or collide with each particle in cell (i,j,k) to construct the neighborhood list. Prior to the neighbor search however, the block boundaries or overlap cells have to be populated with ghost or halo particles from the adjoining blocks which could potentially be in contact

with an internal particle in the block. This involves packing relevant particle information from the adjoining sending block and copying it to the receiving block in the ghost layer. This is a very time-consuming operation because unlike a few particles moving from one block to another in step 1 (CFD-DEM coupling algorithm), a substantial number of ghost particles exist at any given instant. While the packing-sending-receiving-unpacking of ghost particle data is unavoidable in the MPI framework, in the OpenMP framework it is precipitated from the overlapping block data structure used and can be eliminated by a suitable mapping of blocks to a global data structure. This is made possible because all the data is available in shared memory. Once the grid map is created the particles are binned in various computational cells, the particle's neighbors are searched through the neighboring grid cell stencil using the global map. Thus, a separate update of ghost particles at each time step and for each block is not needed in the OpenMP framework.

Results and Discussions

For a comparative performance evaluation of MPI and OpenMP, simulations of strongly coupled fluid-particulate system in fluidized beds and in a rotary kiln are discussed in this section. The fluidized bed simulation was used for validating the hydrodynamics whereas the particle scale heat transfer was validated against rotary kiln experiments. The fluidized beds have near perfect load balancing between MPI processes and performance degradation compared to OpenMP can be attributed to the identification and communication of ghost particles in the MPI framework. The rotary kiln calculations introduce substantial load imbalances in the MPI framework and performance degradation can be attributed to both ghost particles and the ensuing load imbalance between MPI-processes. All the computations were run on Virginia Tech's shared memory resource called 'HokieOne'. HokieOne is a shared memory SGI UV system with Intel Xeon X7542 processors and 5.3GB of memory/core.

Application to fluidized bed

A small scale fluidized bed was simulated to validate the CFD-DEM implementation. The validated code was then used to study parallel performance of a large scale fluidized bed with MPI and OpenMP parallelism.

Small Scale fluidized bed

Validations for the CFD-DEM solver were carried out with a uniformly fluidized bed having a porous distributor plate. The dimensions of the fluidized bed in the experiment by Müller et al. [66] were 44 mm×10 mm×1500 mm (width, depth/transverse thickness, height). The height of the bed in the simulation was reduced to 160 mm in order to reduce the computational time involved. The experimental technique used in their analysis to measure void fractions was Magnetic Resonance (MR). Superficial gas velocities of 0.6 m/s and 0.9 m/s were used to investigate the bed behavior. Poppy seeds were used as particles in the experiment. Numerical simulations were also performed by [66] to validate their DEM code. The initial static height of the bed was 30 mm with 9240 particles. Both the experiments and the simulations were time averaged for 23 seconds.

For the superficial velocity of 0.9 m/s, the voidage values obtained from the simulations compare very well with the experimental values [66] both near the center of the bed as well as near the walls. Details of this validation study can be found in [67].

For parallelization of the validation case, the domain was decomposed in the x-direction balancing the fluid and particle workload almost evenly across the processors. The particles were fluidized for 0.5 second for initial mixing with 0.9 m/s as the superficial velocity. Parallel performance in the form of time to solution from 0.5 to 0.6 seconds or 20,000 time steps is reported on 2 and 4 processors.

The run time for 20,000 time steps of fluidized bed simulation on 2 and 4 cores is listed in Table 3.1. It can be noted that the total time taken by OpenMP parallel code is about 10% less than the MPI run on 2 cores, while OpenMP-DEM run time is 20% less than MPI-DEM. The reason for the performance improvement, though modest, is that there are no ghost or halo particles created in the OpenMP framework which reduces the workload per processor. This is countered by an increase in the cost resulting from remote memory data fetches for neighboring particles at inter – processor boundaries. Since particle drag calculations need the fluid velocity at the particle location, it could be possible that the fluid velocity resides at a distant memory location. Similarly the calculation of a number of terms which couple the two phases could have similar overheads. On increasing the core count from two to four processors, perfect scaling is observed in the OpenMP-DEM calculation. This effect can become significant on a large

number of cores spread across computational nodes. With the increase in number of cores from 2 to 4, there is still improvement in performance owing to the fact that the particle workload is the dominant part of the computations. OpenMP-DEM scales linearly with the increase in core count whereas MPI-DEM does not. The main reason for this is that the number of ghost particles increase in MPI-DEM (~1500 on two cores versus ~4600 on four cores) which need to be identified and communicated to the adjoining processor. Because of the global mapping used in OpenMP-DEM, no ghost particles need to be identified. In both paradigms, the fluid calculation time is high and takes up a much larger fraction of the total compute time than on 2 cores. This is because the fluid grid is quite coarse and on four cores, fluid communication costs start to dominate. Thus the fluid calculation has very poor scaling when the core count is further increased to four.

Table 3.1 Runtime taken to run 0.1 second of fluidized bed simulation after 0.5 second of initial fluidization for 9240 particles

	Number of thread/processor	Fluid phase time		Particulate phase time	
		OpenMP	MPI	OpenMP	MPI
Time (seconds)	2	1550	1585	1715	2075
Time (seconds)	4	1235	1265	820	1310

Simulation of large scale fluidized bed

The scalability of the code was evaluated with a large fluidized bed calculation. The fluidized bed consisted of 1.3 million particles and 1 million fluid cells. The dimensions of the bed and properties of the particles used are listed in Table 3.2. The 3D bed was spatially decomposed along the x- and z-directions. This decomposition strategy was chosen to distribute the fluid load evenly across all the processors/threads and also to distribute the particle load almost evenly among all the spatial domains. This ensured that there is load balance among the MPI processes similar to the OpenMP runs which also were load balanced.

The boundary conditions for the problem were similar to the smaller fluidized bed with the superficial velocity being higher at 2.6 m/s. The critical time step was computed to be 7.445×10^{-4} seconds [47], and a time step of 6×10^{-5} seconds was used for the calculations.

The fluidized bed was initially allowed to mix for about 1 second and the strong scalability results were obtained for 10 time steps of run time (0.6 milliseconds) as shown

in Figure 3.1. There is a clear advantage with OpenMP parallel runs over MPI runs as the runtime for fluid phase is almost the same for both the parallel paradigms but the particle calculations approximately take half the time. There are multiple reasons as to why OpenMP outperforms MPI in these calculations. Firstly, in the MPI framework there are ghost particles created and in this large calculation the number of ghost particles is about 50% of existing particles in the internal mesh blocks. This leads to a significant increase in calculations (mainly building close to half a million ghost particles and associated computations) as well as communication costs at each time step. On contrary, in the OpenMP framework only the overhead associated with remote memory access (communication cost) increases as there is only a onetime cost associated with the global grid map creation. Even though the OpenMP runtime is less than MPI, both OpenMP and MPI runs scale well till 32 cores. In the OpenMP case, the performance levels out at 64 cores whereas for MPI it deteriorates. The performance drop in MPI runs is a result of the increase in the MPI communication overheads at higher core count.

Table 3.2 Particle properties and parameters used in the large fluidized bed simulations

Simulation parameters	Notation	Al particles
Bed		
Width (m)	W	3.072
Transverse Thickness (m)	T	3.072
Height (m)	H	0.768
Particles		
Sphericity	Sp	1
Number	N	5308416
Diameter (mm)	d_p	4
Density (kg/m^3)	ρ	2700
Elastic modulus (GPa)	E	69
Poisson's ratio	σ_p	0.3
Coefficient of normal restitution	e_n	0.9
Coefficient of friction	μ_{p-p}	0.3
Spring stiffness coefficient (N/m)	K	800
Time step (seconds)	Δt	$1x10^{-6}$

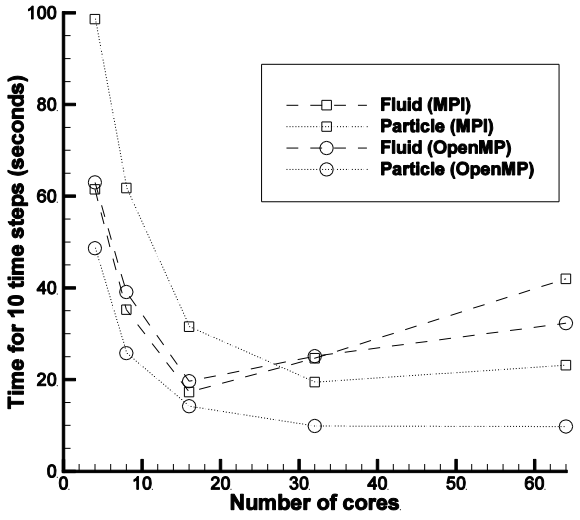

Figure 3.1 Strong scaling study of a fluidized bed with 1.3 million particles and 1 million fluid cells

The Figure 3.1 also indicates that for the fluid phase, both the MPI and OpenMP perform almost the same with a slight increase in runtime for MPI case on 64 cores. The fluid runtimes are almost the same because the same domain decomposition strategy is used for parallelizing both OpenMP and MPI runs. This implies that the cost of inter-block MPI communication is almost the same as that is needed for remote memory fetching in OpenMP framework. The optimal fluid grid size per process/thread is clearly achieved at 16 cores (65,000 fluid cells) after which the parallel performance drops. Overall, the total runtime taken for the fluidized bed calculations with OpenMP parallelism is almost 20 to 30% less than the MPI calculations.

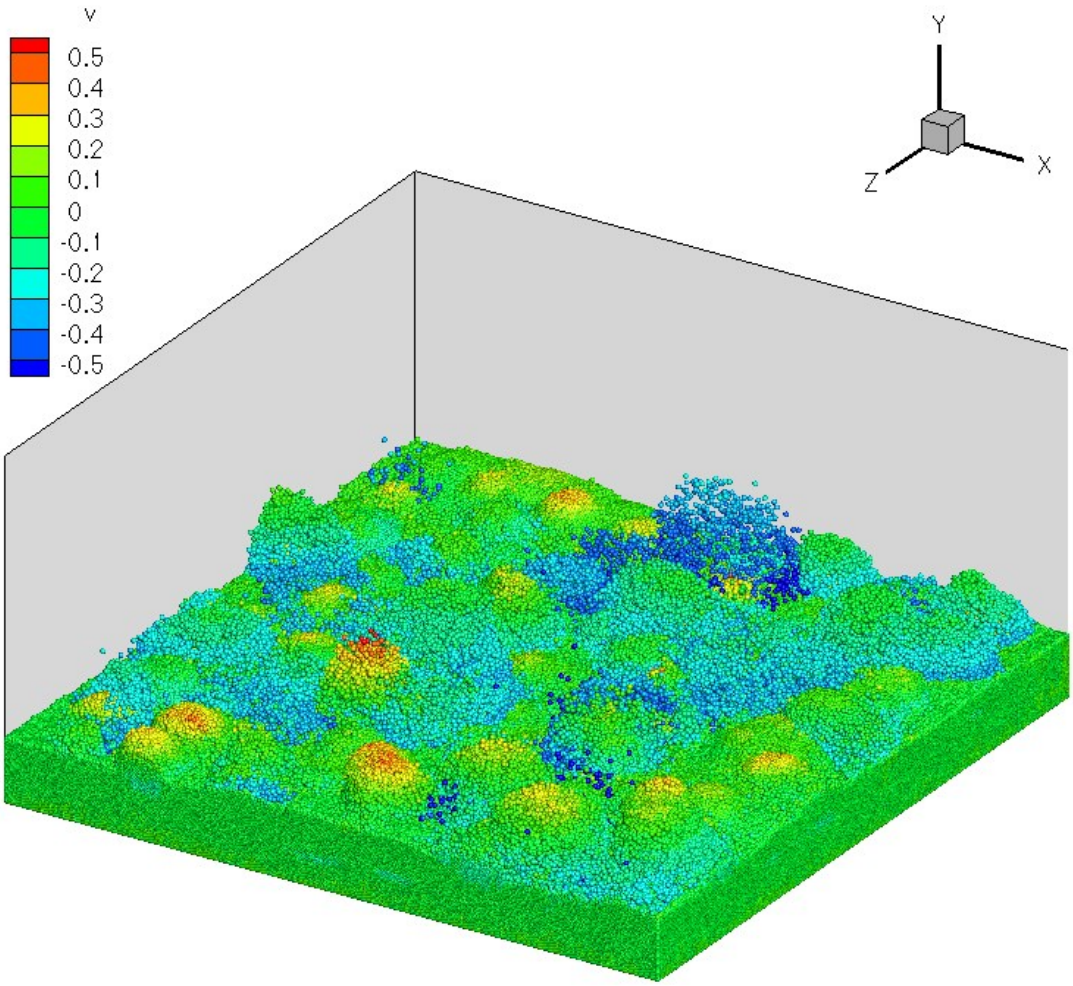

Figure 3.2 Fluidized bed with 5.3 million particles colored by vertical velocity component

In order to test the scalability on even larger particle counts, a larger fluidized bed problem was also simulated. This fluidized bed, shown in Figure 3.2, had the same characteristic as that of the 1.3 million particle bed except the particle count in this case was 5.3 million and the time step was $1x10^{-6}$ seconds. For this case the time required to run 50 time steps on 32 processors was compared after the bed was allowed to mix for 5 seconds. The domain decomposed MPI implementation took 330 seconds of runtime whereas the OpenMP parallel code took 170 seconds in the particle calculations. The main reason for particle calculations with OpenMP taking about 50% of the MPI runtime is the absence of ghost particles. The total ghost particles were approximately 17% of the total particle count and increased the computations as well as communication overheads in the MPI framework. During the bubbling of the bed, there is movement of particles

across block boundaries and thus the particles migrate from one mesh block to the other. This leads to minor load imbalance of about 5% in the domain decomposed framework (MPI paradigm) for the particulate phase work load. This could also lead to some additional slowdown of MPI parallel calculations. This additional overhead in MPI is not significant in the fluidized bed calculations where there is minor load imbalance but it becomes more critical when there is higher load imbalance, such as in rotary kiln case. On the other hand, the particle movement causes non-locality of fluid data needed for particle calculations in the OpenMP framework. Thus there is an overhead associated with remote data fetching in the OpenMP framework. The remote data fetching overhead is highly dependent on the initial distribution of the particles and the ensuing flow physics. Similar to load imbalance for MPI case, remote data fetching overheads for fluidized bed simulations are relatively small as there are many particles which don't leave the spatial domain on which they initially were introduced. This overhead would become limiting when all the particles leave the spatial domain in which they were introduced initially.

Application to a rotary kiln

The partially filled rotary kiln creates a natural load imbalance due to the presence of localized particulate phase. Simulations were performed for such tightly coupled fluid-particulate system to measure parallel performance. A cylindrical rotary kiln with dimensions of 0.1524m diameter and 0.0762m axial length was simulated. A thin section of length 0.01524m of the same kiln geometry was simulated as well. In these three dimensional simulations, 900 fluid cells in a body fitted mesh with 20,000 alumina particles were used for rotary kiln section and 4500 fluid cells and 100,000 particles for the full scale kiln. The body fitted mesh consisted of 5 mesh blocks for the thin section and 25 mesh blocks for the full scale rotary kiln. The fluid calculation was parallelized using domain decomposition by assigning each block to a MPI process or OpenMP thread. The details of these computations are discussed in Chapter 5.

Figure 3.3 (A) compares the uneven work load for a domain decomposed problem (thin section of rotary kiln) unlike the fluidized beds which have fairly even load distribution of both particulate and fluid phase. The particles are colored based on the

processor/thread to which they are assigned. In this initial state of the bed, fluid block 1 has approximately 8000 particles, with approximately 4000 particles each in blocks 2, 4 and 5, with no particles in block 3. After rotation of the bed, under domain decomposition, fluid blocks 2 and 4 have approximately 7800 particles, with approximately 4100 particles in block 5 and a few hundred particles in each of block 1 and 4. Whereas, in OpenMP framework all the particles and fluid mesh blocks are evenly distributed amongst the OpenMP threads at both the instances.

After initial gravitational settling of the particles in the kiln, it is rotated at 20 RPM and the simulation run for 5 milliseconds or 1000 time steps to collect performance data with both MPI and OpenMP. The MPI run calculates both the fluid and the particles in the domain decomposed framework, with block 3 having no contribution to particle computations, which in this case is significantly more than the fluid calculations. In the OpenMP framework, however, the particle calculations are distributed across 5 threads in chronological order from 1-4000 on block 1, 4001-8000 on block 2, and so on. Figure 3.3 (A) shows the initial distribution of the particles (colored by particle number) in the fluid blocks. Thus, thread 1 works on particle numbers 1-4000, most of which are concentrated in fluid block 1, and thread 2 works on particle numbers from 4001-8000, which also are mostly concentrated in block 1. Unlike in the MPI framework in which process 3 has no particles to work on, thread 3 is assigned particle numbers 8001-12000, which mostly exist on fluid block 2. Since particle calculations require local fluid velocity field data, there is some additional overhead of fetching fluid data from non-local thread memory as is clearly the case for particle numbers 4001-16000 in Figure 3.3 (A). This overhead increases as the kiln continues to rotate and particles continue to mix as shown in Figure 3.3 (B). However, the additional overhead is miniscule, compared to having an idle processor in MPI. This is reflected in Figure 3.4, which demonstrates the advantage of using the task based parallelism of OpenMP threads over MPI. With MPI, process 1 works on 8000 particles, whereas process 3 remains idle for most of the computation, since particle calculations dominate the overall computational time. On the other hand, with OpenMP, the particle calculations are more uniformly distributed across all threads, barring the larger percentage time taken by thread 1 which has to compute the serial parts of the particle algorithm. This flow is particle collision dominated and thus the

computations per particle are much higher and the benefit of OpenMP parallelism can be observed even for a small number of particles.

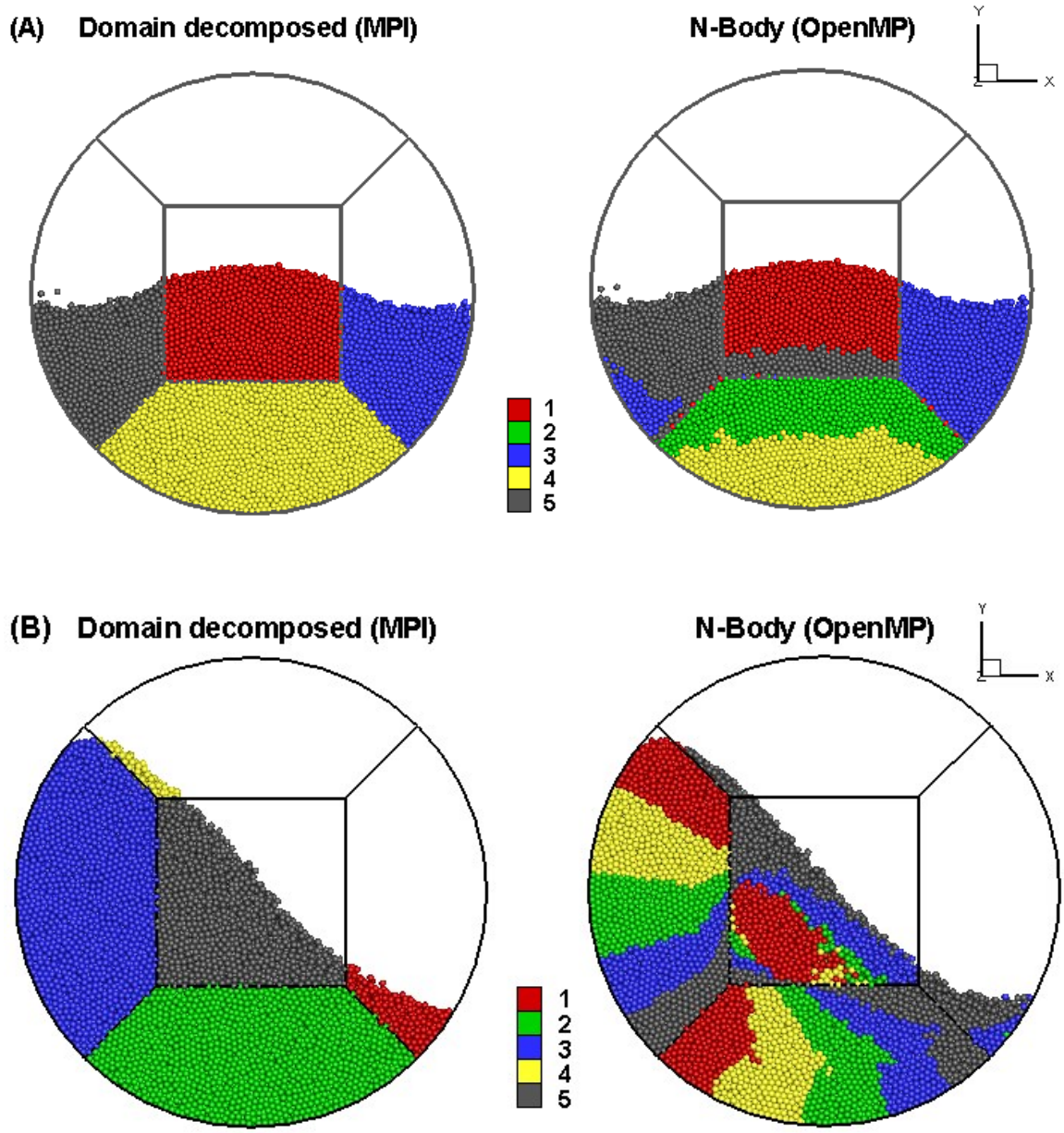

Figure 3.3 (A) Initial distribution of particles in a rotary kiln (thin section) showing domain decomposition used in MPI framework and N-body particle decomposition for OpenMP. (B) Comparison of particle workload division after roation of the kiln. Different particle colors represent the workload assignment to various cores in the two modes.

Table 3.3 summarizes the timings for MPI and OpenMP for the 5 and 25 processor calculations with 20,000 and 100,000 particles corresponding to the rotary kiln section

simulation and full scale simulation respectively. The load imbalance in the MPI run is reflected in Table 3.3, which shows that the time taken is between 60 and 100% higher than the OpenMP parallel run for the 5 and 25 block calculations, respectively. This flow is particle collision dominated and thus the computations per particle are much higher and the benefit of OpenMP parallelism can be observed for the relatively modest number of particles per core. Since the 5 block geometry calculations were performed on a single node the remote data fetching communication time was minimal but when the larger problem was run across multiple nodes, communication overhead of remote data fetching increased. In spite of this, the OpenMP parallel version ran 40% faster than the domain decomposed (MPI) run mostly because of the load balance achieved. Even though the workload per processor was the same for the two different sized problems listed in Table 3.3, there is a much higher cost associated with message passing in MPI and remote memory accesses in OpenMP when the data is distributed across multiple nodes as is the case for the larger problem. In the domain decomposed particle workload partitioning scheme, when a particle crosses mesh block boundaries, data associated with that particle is packed and sent to the mesh block which receives the particle. Whereas in the OpenMP framework once a particle is assigned to a particular thread, the particle stays with that thread irrespective of its location. Because of this, the fluid data needed by the particle may only be available at a memory location associated with another thread, which has to be fetched at an increased communication latency cost. Additionally, based on the domain decomposition strategy selected for the fluid domain, the smaller geometry has message passing in two directions as compared to three directions for the larger case, increasing the communication costs in the larger geometry.

Table 3.3 Total runtime taken to run 0.01 second of rotary kiln simulation on HokieOne for 20,000 and 100,000 particle cases after 1 second of initial rotation

	Number of Mesh Blocks	OpenMP	MPI
Time (seconds)	5	160	330
Time (seconds)	25	275	445

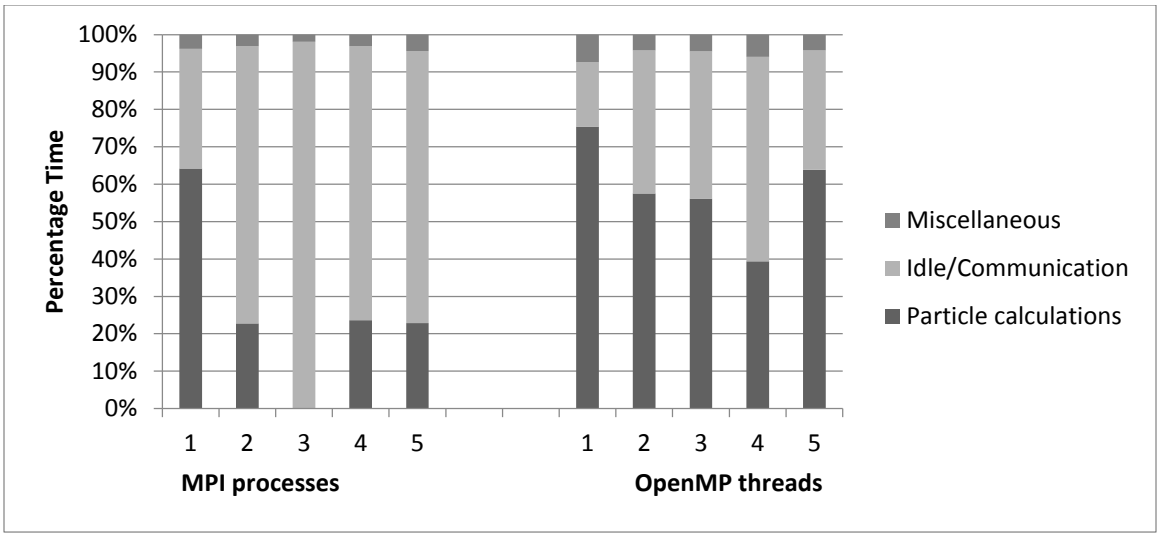

Figure 3.4 Comparision of time spent by MPI-parallel and OpenMP-parallel paradigms in communications, particle, and fluid (miscellaneous) computations in a rotary kiln (thin section) simulation with 900 fluid cells and 20,000 particles.

Additional strong scaling results in which the problem size is kept constant and the number of cores is varied is shown in Figure 3.5 for MPI and OpenMP for 25 blocks and 100,000 particles. It is noted that the majority of the computational time is spent in the DEM for the rotary kiln. The benefits of using the OpenMP framework where the particle work load is evenly distributed across all the processors can be observed. As the number of processors is varied from 5 to 25, the time required to perform the calculations drops but not linearly. This indicates that the inter-processor communication overheads in MPI and remote data fetches in OpenMP become significant as the number of cores increase and also establishes that a minimum of approximately 10,000 particles per core are necessary to offset the communication overheads.

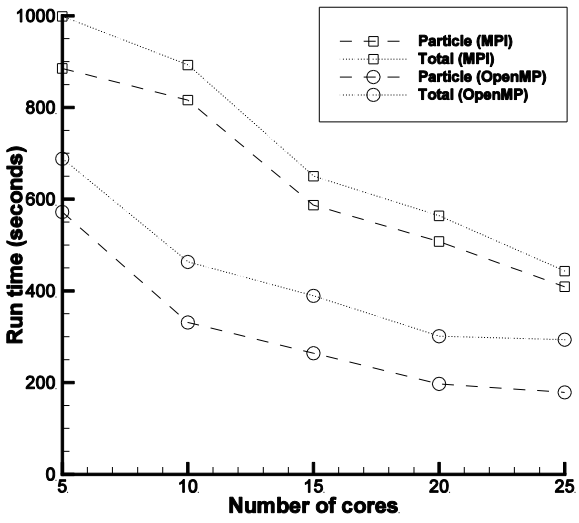

Figure 3.5 Scaling study of rotary kiln case with 100,000 particles for 10 milliseconds of runtime comparing domain decomposed parallelism against the hybrid of particle subset parallelism for particulate phase and domain decomposition for the fluid phase. The OpenMP parallel version outperforms MPI parallel version for different number of cores.

Summary

It is established that OpenMP threads offer considerable advantages over MPI processes in multiphysics applications which do not adhere to a single mode of parallelism. MPI, while ensuring data locality has large overhead associated with changing modes of parallelism. On the other hand, OpenMP does not ensure data locality, but is very adaptable to different modes of parallelism. By carefully constructing OpenMP code to increase data locality for scalable performance, its adaptability can be exploited effectively. This is highlighted in tightly coupled fluid-particulate systems (DEM-CFD), in which the best parallel performance is obtained by switching the mode of parallelism from domain decomposition for the fluid calculations to N-body parallelism for the particles. In DEM, building ghost particle lists at process boundaries is a very time consuming communication-heavy operation, which is eliminated in the OpenMP parallel framework. For a 1.3 million particle uniformly fluidized bed system it is shown that OpenMP-DEM is twice as fast as MPI-DEM on up to 64 processors or cores. The adaptability of OpenMP is illustrated in a rotary kiln application in which particles are not uniformly distributed across the domain decomposed computational blocks and suffer

large load imbalances when parallelized in the MPI framework. Changing modes of parallelism in this framework with MPI would involve very large overheads. It is shown that in spite of OpenMP suffering from decreasing data locality on large core counts, it is 50-90% faster than MPI. These developments are very relevant to recent advanced co-processing architectures with a large number of shared memory cores.

4. Methodology and validation for heat transfer analysis [2]

Methodology

Computational methods for predicting large-scale fluid-particulate flows have been developed over the last 20 years. The in-house code GenIDLEST [24] was used for this work. The overall algorithm, data structure and capabilities of the code are mentioned in chapter 2 and 3. GenIDLEST uses an incompressible variable property fractional step algorithmic model solving the mass, momentum and energy conservation equations which are listed in this chapter. Particle scale validation studies are also discussed later in this chapter.

Governing equations

Built on top of the existing capability, a number of multiphysics modules were developed to attack the full range of physics in multiphase flows. Notable among these are particle-particle and particle-wall collisions and the ensuing transfer of momentum and heat between solid-solid and gas-solid-gas. For this work, particle-particle and particle-wall collision momentum transfer using DEM, which is a fundamental capability required for simulation of particulate flow dynamics, was used together with appropriate particle-fluid, particle-particle and particle-wall heat transfer models [68].

Fluid Flow and Energy Governing Equations

The governing equations for incompressible variable property unsteady viscous flow in a generalized coordinate system consist of mass, momentum and energy conservation laws. The equations are mapped from physical $(\vec{x})$ to logical/computational space $(\vec{\xi})$ by a boundary conforming transformation $(\vec{x}) = \vec{x}(\vec{\xi})$, where $(\vec{x}) = (x, y, z)$ and $(\vec{\xi}) =$

[2] Majority part of this chapter is published in *Particle scale heat transfer analysis in rotary kiln,* Amit Amritkar, Danesh Tafti, Surya Deb, Proc. of ASME HT2012, Puerto Rico, July 8-12 2012. *Used with permission of ASME, 2013*

(ξ, η, ζ). The equations are converted to non-dimensional form using the following parameters,

$$\rho = \frac{\rho^*}{\rho^*_{ref}} \qquad \mu = \frac{\mu^*}{\mu^*_{ref}} \qquad \kappa = \frac{\kappa^*}{\kappa^*_{ref}} \qquad c_p = \frac{c_p^*}{c_{p_{ref}}^*} \qquad \underset{x}{\rightarrow} = \frac{\vec{x}^*}{L^*_{ref}}$$
$$\underset{u}{\rightarrow} = \frac{\vec{u}^*}{U^*_{ref}} \qquad t = \frac{t^* U^*_{ref}}{L^*_{ref}} \qquad P = \frac{P^* - P^*_{ref}}{\rho^*_{ref} {U^*_{ref}}^2} \qquad T = \frac{T^* - T^*_{ref}}{T_0^*} \tag{4.1}$$

The equations for incompressible variable property flow coupled with a solid discrete phase in dimensionless form are written as:

Continuity:

$$\frac{\partial}{\partial \xi}\left(\rho \epsilon \sqrt{g} U^j\right) = 0 \tag{4.2}$$

Momentum:

$$\frac{\partial}{\partial t}\left(\rho \epsilon \sqrt{g} u_i\right) + \frac{\partial}{\partial \xi_j}\left(\left(\rho \epsilon \sqrt{g} U^j\right) u_i\right) = -\frac{\partial}{\partial \xi_j}\left(\sqrt{g}\left(\vec{a}^j\right)_i p\right) + \frac{1}{Re}\frac{\partial}{\partial \xi_j}\left(\epsilon(\mu + \mu_t)\sqrt{g} g^{jk} \frac{\partial u_i}{\partial \xi_k}\right) + S_{fp_i} \tag{4.3}$$

Energy:

$$\frac{\partial}{\partial t}\left(\rho \epsilon \sqrt{g} T\right) + \frac{\partial}{\partial \xi_j}\left(\left(\rho \epsilon \sqrt{g} U^j\right) T\right) = \frac{1}{RePr}\frac{\partial}{\partial \xi_j}\left(\epsilon(\kappa + \kappa_t)\sqrt{g} g^{jk} \frac{\partial T}{\partial \xi_k}\right) + Q_{fp} \tag{4.4}$$

Equation of State:

$$\rho = \frac{P}{RT} \tag{4.5}$$

where $\sqrt{g} U^j = \sqrt{g}\left(\vec{a}^j\right)_k u_k$ is the contravariant flux vector, $\vec{a}^i$ are the contravariant basis vectors, $\sqrt{g}$ is the Jacobian of the transformation, g^{ij} is the contravariant metric tensor, u_i is the Cartesian velocity vector, p is the kinematic pressure and P is the total pressure, T is the temperature, ε is the void fraction, S_{fp} and Q_{fp} are the total non-dimensional interphase momentum and energy transfer terms in a given computational cell,

respectively. Both, the Reynolds number (*Re*) and Prandtl number (*Pr*) are defined based on the reference quantities. μ_t is the non-dimensional turbulent eddy-viscosity given by,

$$\mu_t = \rho C_s^2 \left(\sqrt{g}\right)^{\frac{2}{3}} |\overline{S}| \tag{4.6}$$

where $|\overline{S}|$ is the magnitude of the strain rate tensor given by $|\overline{S}| = \sqrt{2\overline{S_{ik}}\overline{S_{ik}}}$; and the strain rate tensor is given by,

$$\overline{S_{ij}} = \frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) \tag{4.7}$$

and the Smagorinsky constant C_s^2 is calculated using the Dynamic subgrid stress model [69], κ_t is the turbulent conductivity or reciprocal of the turbulent Prandtl number times the turbulent viscosity.

The above formulation takes into account property variation with temperature. The dynamic viscosity and thermal conductivity variations are calculated based on Sutherland's law for gases whereas the specific heat is assumed constant as it has a much weaker dependence on temperature.

Further details about the algorithm, functionality, and capabilities can be found in [24, 25]. The software has been applied to various turbulent flow and heat transfer problems [26]. In this study, the fluid equations are solved by a semi-implicit version of the fractional-step method.

Particle Scale Modeling

The DEM operates at the particle scale providing a framework to investigate the hydrodynamics and heat transfer mechanisms in detail. Central to the DEM is the treatment of multiple particle-particle and particle-wall interactions of spherical smooth particles. A soft sphere methodology has been implemented to model these interactions. The idea of discrete particle modeling using the soft sphere method was originally developed by [46]. In their approach, they use a linear spring-dashpot system to model the inter particle interactions in dense granular flows. In the soft sphere methodology, multiple particle interactions can be taken into consideration unlike the hard sphere model [70], where only binary collisions are considered. The soft sphere model was first applied by [47] to a 2D fluidized bed. The forces acting on any individual particle *i* are calculated based on Newton's second law as follows,

$$m_{p,i}\frac{d\vec{u}_{p,i}}{dt} = (\rho_p - \rho_f)V_{p,i}\vec{g} + \frac{V_{p,i}\beta}{1-\varepsilon}(\vec{u}_f - \vec{u}_{p,i}) + \sum \vec{F}_{p,ij} \qquad 4.8$$

where $m_{p,i}, \rho_p, \rho_f, V_{p,i}, \vec{u}_f and\ \vec{u}_{p,i}$ are the particle mass, particle density, fluid density, particle volume, fluid velocity and particle velocity, respectively, ε represents the void fraction, and β represents the momentum exchange coefficient between the solid and the fluid phase. $\sum \vec{F}_{p,ij}$ represents the net contact forces due to collisions with other particles and with walls. The first term on the right hand side accounts for the buoyancy of the particle in the fluid medium. The second term accounts for the particle-fluid coupling through the drag formulation. The inter phase momentum exchange coefficient β is modeled by combining correlations given by [71] for dense regimes ($\varepsilon < 0.8$) and by [72] for dilute regimes ($\varepsilon > 0.8$). The combined drag forces of all the particles inside a fluid grid and the calculated voidage are transferred to the fluid equations to couple the fluid and particle phase.

Particle rotation is also considered in the calculations. The angular acceleration of each particle *i* is computed as follows,

$$I_{p,i}\frac{d\vec{\omega}_{p,i}}{dt} = \sum_j \vec{\tau}_{p,ij}\,; \qquad \vec{\tau}_{p,ij} = \vec{r}_{p,i} \times \vec{F}_{p,ij\_to} \qquad 4.9$$

where, $I_{p,i}, \vec{\omega}_{p,i}, \vec{\tau}_{p,ij}, \vec{r}_{p,i} and \vec{F}_{p,ij\_to}$ are particle moment of inertia, angular velocity, torque acting due to collision with a neighboring particle *j* , radius of the particle and tangential force acting due to collision with a neighboring particle *j*.

The collision forces ($\vec{F}_{p,ij}$) are calculated using the soft sphere model. The contact forces acting on a particle in collision with its neighbor is modeled as a linear spring dashpot system in the normal and tangential directions. In the tangential direction, there is an additional sliding element that controls the magnitude of the tangential force acting on the particle and enables sliding. Figure 4.1 shows the normal and the tangential spring systems involved in soft sphere modeling. The details of this spring dashpot slider model can be found in [47].

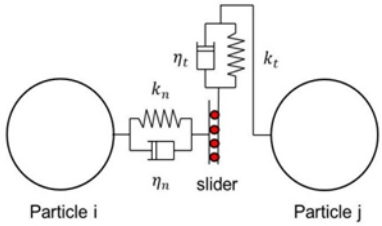

Figure 4.1 The soft sphere spring - dashpot - slider model

Methodology for Thermal DEM

There are different modes of heat transfer in a dense particulate system. The most common modes of heat transfer pertaining to the particulate phase are the particle-particle and particle-wall conduction heat transfer, thermal conduction through the gas between the particles (gas lens and liquid bridge effect), convective heat transfer with the surrounding gas, radiative heat transfer with the gas phase and the bed walls and frictional heating between the particle and a wall or another particle. Usually radiation heat transfer can be neglected at low temperatures, typically <700K [73].

Prior to explaining the microscopic models, the characteristic equation describing heat transfer for the dispersed phase is given by,

$$m_{p,i}c_{p,i}\frac{dT_{p,i}}{dt} = Q_{fp,i} + Q_{p,i} + Q_{fric,i} + Q_{rad,i} \qquad 4.10$$

where, $Q_{fp,i}$ is the convective heat transfer between particle and fluid, $Q_{p,i}$ is the source term arising from inter-particle and particle-wall interactions and $Q_{fric,i}$ is the frictional heating. The convective heat transfer ($Q_{fp,i}$) between particle and fluid is calculated by assuming a lumped capacitance system (Bi<0.1) and using one of many correlations available in the literature for dense particulate systems [74-77]. An equal and opposite convective heat transfer source term is transferred to the fluid energy equation.

Particle – fluid convection heat transfer:

The convective heat transfer coefficient between particles and fluid phase is calculated based on Nusselt number correlations. There are many correlations in the literature which describe the heat transfer rates between solid and fluid phase. Some of these correlations involve the void fraction whereas some of them are dependent on the particle properties

and applicable only in a certain range of flow parameters. In this study, a widely employed correlation proposed by [77] is used which is as follows,

$$Nu = (7 - 10\varepsilon + 5\varepsilon^2)\left(1 + 0.7Re_p^{0.2}Pr^{\frac{1}{3}}\right) + (1.33 - 2.4\varepsilon + 1.2\varepsilon^2)Re_p^{0.7}Pr^{\frac{1}{3}} \quad 4.11$$

where, $Re_p = \varepsilon u_p d_p/\nu$

Friction heating

Particle flow in the rotary kiln is governed by large particle contact times with each other and with the walls of the kiln, making it essential to include the heating due to friction. The frictional heating [78] between a particle and another particle or a surface is calculated as,

$$Q_{fric,i} = \gamma_{i,j}\mu\left|\vec{V}_T\right|\left|\vec{F}_N\right| \quad 4.12$$

where the partition coefficient of generated heat flow is given by $\gamma_{i,j} = \frac{\kappa_{p,i}}{\kappa_{p,i}+\kappa_{p,j}}$ for inter-particle collision and $\gamma_{i,j} = 0.5$ for particle wall collision, μ is coefficient of friction, $\vec{V}_T$ is the tangential velocity, $\vec{F}_N$ is the normal force, and κ_p is the thermal conductivity.

Particle/surface – particle conduction heat transfer:

The literature suggests mainly two approaches for calculating inter-particle and particle-surface collision heat transfer. The first method, which is used in this study, is based on the quasi steady state solution of the collisional heat transfer between two spheres [79]. The other approach is based on the analytical solution of the one dimensional unsteady heat conduction between two semi-infinite objects [68].

The quasi steady state solution approach has been widely used for many granular flow applications in the pharmaceutical, petrochemical, and mineral industries, energy conversion, gaseous and particulate pollutant transport in the atmosphere, and heat exchangers amongst others [80]. This modeling approach is particularly useful for applications where the collision between particles involves more than two particles and the time of collision between any two particles is influenced by the presence of additional colliding particle(s) as is the case in dense particulate flows. The approximate analytical

solution of contact conductance [81] from one stationary particle center line to the other *stationary particle in vacuum* is given by,

$$H_{pc,ij} = 2\kappa_{p,i} r_{c,ij} \tag{4.13}$$

where $\kappa_{p,i}$ is the thermal conductivity and $r_{c,ij}$ is the contact radius between the colliding particles i and j (assuming $r_{c,ij} \ll R^*_{ij}$). It is implicitly assumed that the thermal conductivity ratio of particle to fluid is large for the above contact conductance to be valid. The instantaneous conductance $H_{pc,ij}$ is calculated as a function of the material properties and the actual contact force based on Hertz's theory as follows,

$$H_{pc,ij} = 2\kappa_{p,i} \left[\frac{\vec{F}_{p,ij,n} R^*_{ij}}{E^*_{ij}} \right]^{1/3} \tag{4.14}$$

where, R^*_{ij} is the geometric mean of the particle radii, $\vec{F}_{p,ij,n}$ is the normal contact force calculated from the DEM simulation and E^*_{ij} is the equivalent Young's modulus given by,

$$E^*_{ij} = \frac{4/3}{\frac{1-\sigma_{pi}^2}{E_{pi}} + \frac{1-\sigma_{pj}^2}{E_{pj}}} \tag{4.15}$$

where, E_{pi} and E_{pj} are the elastic moduli of the colliding particles and σ_{pi} and σ_{pj} are the Poisson's ratio, respectively. The amount of heat transported across the collisional interface per unit time ($Q_{pc,ij}$) is thus given by,

$$Q_{pc,ij} = H_{pc,ij}(T_j - T_i) \tag{4.16}$$

The above formulation was modified by [82] to account for cases with different material properties of colliding particles or particle and surface.

$$Q_{pc,ij} = \frac{4 r_{c,ij}(T_j - T_i)}{\left(1/\kappa_{pi} + 1/\kappa_{pj}\right)} \tag{4.17}$$

The other approach (unsteady approach) of solving the inter-particle conduction is by solving the unsteady heat transfer equation in the direction normal and parallel to the contact area. There are a number of implicit assumptions in this formulation, namely, that the contact area is much smaller than particle diameter, time of contact is short such that the two particles can be treated as infinite mediums, and that the particles are perfectly smooth with no contact resistance. The analytical solution of this equation exists for the

assumption of one dimensional heat transfer where the Fourier number $Fo_{pij} = \alpha_i t_{c,ij}/r_{c,ij}^2$ approaches zero.

$$q_{0,ij} = \left(0.87\beta_{p,i}\beta_{p,j}\left(T_{p,j} - T_{p,i}\right)A_{c,ij}{t_{c,ij}}^{-0.5}\right)/\left(\beta_{p,i} + \beta_{p,j}\right) \quad 4.18$$

where, $\beta_{p,i} = (\rho_{p,i}c_{p,i}\kappa_{p,i})^{0.5}$, $\rho_{p,i}$ is the density of the particle material, $\kappa_{p,i}$ is the thermal conductivity of the particle, $A_{c,ij}$ is the maximum contact area based on Hertz's theory [68] given by,

$$A_{c,ij} = \pi\left(5m{R^*}_{ij}^2/4{E_{ij}}^*\right)^{2/5}\left(\vec{u}_{p,ij\_n}\right)^{4/5} \quad 4.19$$

where m is the reduced mass and t_c is the total time of collision. The estimation of this time is done using the elastic collision time as given by,

$$t_{c,ij} = 2.94\left(5m/4{E_{ij}}^*\right)^{2/5}\left({R^*}_{ij}\vec{u}_{p,ij\_n}\right)^{-1/5} \quad 4.20$$

A correction term is proposed to compensate for radial heat conduction in cases where $Fo_{pij} > 1$.

$$Q_{pc,ij} = C q_{0,ij} \quad 4.21$$

The correction coefficient 'C' is obtained by solving the complete heat equation numerically [68].

The calculation of the collisional heat transfer is done by numerically integrating the analytical solution of the 1D heat conduction equation as mentioned above. This involves obtaining A_c, $r_{c,ij}$ and t_c from the soft sphere collision model. The soft sphere model slows down the process of collision by reducing the normal spring constant associated with the particle material. Computational efforts show that the softening treatment has no effect on the overall movement of the particles since the particle velocity is independent of the normal spring stiffness. However, this leads to a higher estimation of heat transfer between the particles as the heat transfer process is highly sensitive to the contact time and area of contact. [83] proposed area and time restoration factors for A_c and t_c based on the actual normal spring constant. The restoration factors are listed below as,

$$A_{ca}/A_{c,ij} = \sqrt[4]{k_{ni}/k_{na}} \quad , \quad t_{ca}/t_{c,ij} \approx \sqrt{k_{ni}/k_{na}} \quad 4.22$$

The restoration is then applied to the $A_{c,ij}$ and $t_{c,ij}$ as mentioned above to obtain the corrected area and time of contact A_{ca} and t_{ca} respectively.

Zhou et al. [84] proposed another method of finding the actual area of contact which implicitly accounts for contact time correction. They propose a correction factor for the contact radius as,

$$c = r_{ca}/r_{c,ij} = \left(E_{ij}/E_{ij}^{*}\right)^{1/5} \quad 4.23$$

where, E_{ij} is the equivalent Young's modulus calculated based on the material properties used in soft sphere modeling of the colliding particles. Instead of using an explicit correction factor for contact time as done by [83], this approach uses the E_{ij} [85] obtained from the DEM simulations as follows,

$$E_{ij} = 3k_{p,i}\left(1 - \sigma_{p,i}^{2}\right)/\sqrt{2r} \quad 4.24$$

The collisional heat transfer is then summed over all the collisions ($n_{p,i}$) a particle is undergoing at a given instant with its neighboring particles,

$$Q_{pc,i} = \sum_{j=1}^{n_{p,i}} Q_{pc,ij} \quad \& \quad Q_{pcw,i} = \sum_{j=1}^{n_{w,i}} Q_{pcw,ij} \quad 4.25$$

The heat transfer source terms from particle-particle and particle-surface interactions are corrected and then summed together to obtain the source term in the particle energy equation for particle i.

$$Q_{p,i} = Q_{pc,i} + Q_{pcw,i} \quad 4.26$$

Radiation heat transfer

At temperatures higher than 700K, radiative heat transfer has larger contribution to the heat transfer and can no longer be neglected. To calculate the radiation between the particle and its local environment, an enclosed domain is considered around each particle. For closely packed bubbling fluidized beds this consideration is reasonable due to the closely packed nature of the bed particles [86]. In this study the size of the enclosing domain for radiative heat transfer is considered to be the same as the computational cell which could range between 2.5 to 3 times the particle diameter depending on the body fitted grid size. Thus the radiative heat transfer between a particle and its surrounding is given by,

$$Q_{rad,i} = \sigma \epsilon A_s (T_{bed}^{4} - T_i^{4}) \quad 4.27$$

$$T_{bed} = \varepsilon_f T_{f,grid} + \frac{(1-\varepsilon_f)}{n}\sum_{i=1}^{n} T_{p,i} \qquad 4.28$$

where n is the number of particles in the fluid grid. To simplify the computations the radiative heat transfer between fluid and its surrounding is neglected in this work.

Particle scale validation studies

The validation of the code was done using various experimental studies. The particle surface collision heat transfer was validated with experiments by Ben-Ammar et al. [87] and inter-particle collision was validated using experiments by Kuwagi et al. [88]. The heat transfer validation for multi-particulate system was accomplished by comparing the experimental data [89] of hot sphere cooling in a packed bed. Finally, a rotary kiln validation is performed by comparing the computational results with the experimental [90] bed temperature.

Particle-surface collision simulations

Collision between a particle and a surface was simulated for calculation of heat transfer based on thermal DEM and compared with experiments. Ben-Ammar et al. [87] performed experimental measurements by colliding heated stainless steel particles of diameter 0.00476 m with a surface at 2.3 m/s relative velocity. The experiments provide an average range of heat transferred per collision. The experiments were performed in vacuum which made sure that the convection heating is completely eliminated and the maximum temperature of the particles was also below 360K, which allowed neglecting radiative heat transfer in the simulations [73].

As listed in Table 4.1, the heat transfer calculations based on the quasi steady approach and the unsteady approach are within the experimental uncertainty.

Table 4.1 Validation of particle-surface heat conduction with experiments

Particle-surface collision	Total energy transferred per impact (J)
Experimental	1-3 E-04
Unsteady approach	1.106E-04
Quasi steady approach	1.220E-04

Particle-particle collision simulations

Kuwagi et al. [88] performed experiments to measure the dynamic heat transfer during the collision of hot (423K) and a cold (294K) stainless steel particles at a relative velocity of 2.81 m/s. The experiments used electrical current as a measure of the heat transfer by using a curve fitted function to convert the electric current to heat flux.

The result of the unsteady approach with the correction for the actual contact area and time is compared with results of the experiment and the quasi steady state solution approach. As listed in Table 4.2, the heat transfer calculation with the unsteady approach is within the estimated experimental uncertainty whereas the calculation based on the quasi steady approach under predicts the heat transfer. The difference in prediction can be attributed to the uncertainty associated with the experimental correction used for converting the electric current into energy which was based on static contact observations. In addition to the experimental uncertainty, for the particle diameter of 0.0198m, the Biot number is approximately 0.1 where the lumped capacitance assumption of the quasi-steady model might not hold true. In contrast, the Fourier number is of $O(10^{-3})$ for which the unsteady heat transfer model is more appropriate.

Table 4.2 Validation of particle-particle heat conduction with experiments

Particle-particle collision	Total energy transferred per impact (J)
Experimental	2.788E-02
Unsteady approach	2.289E-02
Quasi steady approach	3.949E-03

In the packed bed and rotary kiln simulations, the quasi steady state approach is used for particle-particle and particle-wall heat transfer since the time of contact between particles acutely exceeds the collisional time estimated by Hertzian contact theory [68].

Hot particle cooling in packed bed

The validity of the proposed heat transfer model in a multi-particle scenario is examined by considering the cooling of a hot sphere in a fluidized bed. In the experiments performed by Collier et al. [89] the temperature of the hot sphere was measured by a thermocouple directly attached to the particle whereas in the CFD-DEM simulations the

temperature of any particle in the bed is readily available. The properties of the particles used and simulation parameters are listed in Table 4.3.

Table 4.3 Particle properties and parameters used in the fluidized bed simulations

	Notation	Fluidized bed
Number of particles	N	16000
Density (kg/m^3)	ρ	420
Hot particle diameter (mm)	d_{ph}	2
Bed particle diameter (mm)	d_{pb}	3
Poisson's ratio	σ_p	0.3
Young's Modulus (GPa)	E	5.0
Friction Coefficient	μ	0.4
Thermal conductivity (W/m-K)	κ	0.84
Heat Capacity (J/Kg-K)	Cp	800.0
Coefficient of restitution	e	0.9
Spring stiffness coefficient (N/m)	K	2000
Time step (seconds)	Δt	1x10^{-6}
Fluidization velocity (m/s)	U_{mf}	0.74
Fluidized bed size (mm)	W,H,T	90,900,24
Grid size (mm)	$\Delta x,\Delta y,\Delta z$	9,9,8

All the external bounds of the computational domain consist of no-slip walls. The top surface of the domain was an outlet boundary condition whereas the bottom surface was wall boundary with superficial inlet velocity. Initially all the particles in the bed were allowed to attain equilibrium under the influence of gravity. Since the exact location of the hot particle in the experimental setup was not known, for computations an approximate location was chosen in the center of the bed. Once the packed bed was formed, as shown in Figure 4.2 where all the particles are stationary, fluid with a superficial velocity of $0.58U_{mf}$ was introduced from the bottom wall. Simultaneously a particle in the middle of the bed at about 168 mm height from the bottom of the bed was instantaneously heated to 180 ^oC and then allowed to cool. The cooling curve of the particle was recorded and is reported in Figure 4.3. The 3D calculations took about 17 days of wall clock time on 2 CPU cores for 8 seconds of physical time.

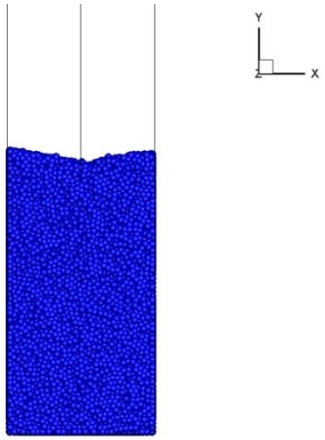

Figure 4.2 Validation setup for cooling of a hot sphere in a packed bed

As shown in Figure 4.3, the computational temperature is comparable with the measured one. The cooling curve of the hot sphere is slightly different than the experiments, indicating that the thermal behavior of hot sphere varies with their initial locations. These minor variations can be attributed to the different packing fractions in various regions of the bed once the air flow has started. Also, the use of lumped capacitance to model the cooling of a single particle in a flow yields a slower cooling rate as it doesn't account for the additional cooling through particle-particle conduction.

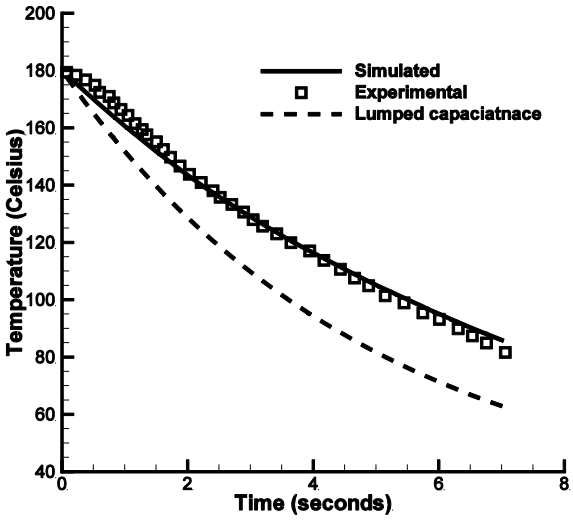

Figure 4.3 Cooling curves for a single hot sphere cooling in a packed bed

Summary

The particle scale heat transfer models including conduction, convection, radiation and friction heating were incorporated in the in-house code GenIDLEST. Two types of conduction models were incorporated namely, quasi-steady model (equation 4.14) and dynamic collision model (equation 4.18) with corrections for actual time of contact and area of contact. Validation studies at the particle scale were conducted at single particle scale for various models used. Finally, the usefulness of having collisional conduction model over lumped capacitance calculations was illustrated.

5. Heat transfer studies in fluid-particulate systems

Introduction

The effects of mono dispersed as well as poly dispersed particles in a rotary kiln are presented in this chapter. These results are followed by heat transfer analysis in fluidized bed with an immersed tube heat exchanger to study the average heat transfer coefficient around the immersed tube. LES wall function is used to resolve the near wall coarse grid. The results focus on identifying primary mechanism of particle scale heat transfer in such systems.

Heat transfer analysis in rotary furnace

Rotary kilns are widely used reactors in metallurgical and chemical industries to handle bulk material. The reactor is a cylindrical vessel rotated around the drum axis with bulk material loaded inside for processing as shown in Figure 5.1. Typically the rotary kiln is rotated either by a friction drive wheel system or a positive rack/pinion or chain drive depending upon the size and production requirements. Considering the wide range of high temperature applications of the rotary furnaces, it is essential to understand the bed hydrodynamics along with the mechanisms of heat transfer occurring inside the reactor.

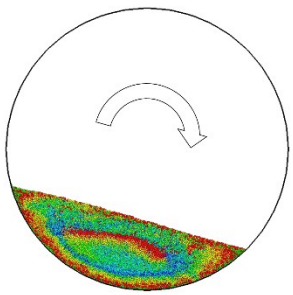

Figure 5.1 Rotary furnace rotating clockwise

To investigate the heat transfer mechanisms, several experiments have been performed in rotary furnaces for a wide range of process parameters including the effects of bed material properties like specific heat capacity and thermal conductivity, vessel rotational speeds, fill levels and so on. In one such experimental study [91] using Ottawa sand in a rotary furnace for convective heat transfer effects of hot air on the rolling solids and walls, it was found that the air to solids heat transfer coefficient was about an order of

magnitude higher than that from air to walls. In another study [92], theoretical analysis of the heat transfer mechanisms in a rotary furnace was performed giving an overview of various heat transfer coefficients along with experiments for contact heat transfer coefficient determination. To quantify particle-particle conduction heat transfer, experimental measurements [89] of cooling of a hot particle, directly connected to a thermocouple, in packed as well as fluidized bed was performed. It was found that when the hot particle was much smaller than the bed particles the particle-particle conduction heat transfer is negligible. Unlike fluidized beds, continuous temperature measurement in the rotary furnace is complicated due the continuous moving bed and the kiln. Hence, the data reported in the literature has been limited to the bed surface or to near the kiln walls. Recent advancements in measurement technology have helped overcome these limitations. A new measurement technique [93] employing radio transmission of temperature measurements from the rotating furnace was used to record 3D temperatures within a rotating sand bed as well as the temperature of freeboard gas (The freeboard region of a rotary reactor refers to the free space above the particulate bed.) and kiln walls. The authors observed large temperature gradients in the radial directions of the kiln whereas the axial variation was minimal and approached the wall temperature.

The heat transfer study of rotary kiln gets more complicated when the granular bed has particles of different sizes. The hydrodynamic behavior of such poly-dispersed particles in rotary kiln was studied experimentally by many previous studies [94-97]. They observed that there is clear segregation of smaller particles at the core of the bed and the larger particles surround them. Based on the percolation of particles in the bed, Boateng et al. [97] came up with a mathematical model to predict the preferential particle motion in poly-dispersed rotary kiln. Experimental study by Dhanjal et al. [98] looked at the effects of poly-dispersed particles on heat transfer in rotary kiln. It was observed that there was little influence of particle segregation on overall heat transfer and there was inadequate particles mixing which caused radial temperature gradients.

Experiments still lack the ability to capture the details of all the heat transfer mechanisms; thus there have been efforts to numerically simulate granular flows. Without considering interstitial gases, DEM simulations were performed with alumina powder in rotary kiln [90]. Additional DEM simulations with copper particles were

performed to explore the parameter space. Chaudhuri et al. [99] in their previous work also investigated the effects of thermal conductivity, specific heat capacity, vessel rotational speeds, fill levels and baffle sizes for different particles. It was observed that the heating was faster in high conductivity and lower heat capacity materials. To further enhance numerical capabilities, inclusion of fluid becomes essential. A two-dimensional rotary kiln was studied using CFD-DEM by Schmidt et al. [100]. In that study, a hard sphere model was used for inter-particle and particle-wall interactions. The DNS study involved modeling the large individual particles of 28mm diameter for temperature distribution inside particles; which allowed them to show that the lumped capacitance assumption was not valid. In another work [101] using CFD-DEM, heat transfer calculations for a rotary kiln were performed. They investigated particle of 3 different materials with 3 mm diameter for 2 different kiln rotational speeds with periodic boundaries axially in the rolling bed mode. Non-uniform grids were used along with a fictitious or mathematical wall to model the curved boundary. It was found that the ratio of heat transfer coefficient of convection to conduction for aluminum particles was almost 1.0, whereas for steel it was higher, and even higher for glass.

There are six different bed behavior modes which are documented in the literature based on the operating conditions of the kiln [97]. These modes are, slipping, slumping, rolling, cascading, cataracting and centrifuging. The computational heat transfer studies of rotary kiln so far have concentrated on the rolling mode and the literature lacks comprehensive studies in other bed modes. Thus, the objective of this work was to segregate the different modes of heat transfer that dominate the heating of particulate phase in a rotary kiln running in the cascading mode. The dominant modes of heat transfer are identified for a mono-dispersed particulate flow.

Additionally, there are no computational studies in the literature known to the author that have studied the effects of particle size distribution on bed heating in a rotary kiln. Thus a rotary kiln was studied to segregate various modes of heat transfer with poly-disperse distribution of particles in a slipping bed regime. There are few additional challenges associated with such computational study. Some of these challenges are, 1. Small computational time step is required since it is governed by the smallest particle diameter. 2. Identification and storing of neighbors becomes memory intensive as the maximum

number of neighbors can easily exceed those in a closely packed bed with a single particle size. These issues are addressed using OpenMP parallelism to accelerate the computations and use of shared memory systems.

Problem setup for mono-dispersed particulate flow

Using the validated code, DEM simulations were run for a rotary kiln. The computational geometry, boundary conditions and physical parameters considered for rotary kiln calculations are summarized in this section. As discussed in chapter 3, two different geometries, the full scale rotary kiln and a thin section of the same kiln, were simulated. The thin section kiln was used only for parallel performance study. The thin section geometry is described in chapter 3 whereas the particle properties and other computational details are same as the full scale rotary kiln as listed below. The following chapter deals only with full section results.

The model assumptions require that the resistance to heat transfer inside the particle is significantly smaller than the resistance between the particles. The Biot number [102] using the quasi steady conduction model (equation 4.14) assumptions require that the resistance to heat transfer inside the particle is significantly smaller than the resistance between the particles. The Biot number [102] is calculated using,

$$\mathrm{Bi} = \frac{2}{\pi}\left(\frac{r_{c,ij}}{\mathrm{R}}\right) \qquad 5.1$$

where R is the particle radius. The maximum Bi number was found to be Bi~0.06. Thus the assumption of lumped heat capacitance at particle scale for these calculations is valid since Bi<0.1.

Computational Grid

The full scale cylindrical rotary kiln was simulated for validation and heat transfer studies. The dimensions of the horizontal cylinder were 0.1524m diameter with 0.0762m axial length. A 50% fill level was used based on the experiments performed by Chaudhuri et al. [90]. To avoid discontinuities in the void fraction profiles, the fluid cell size selected was 2.5 to 3 times the particle diameter [48]. Thus in this simulation, 4500 fluid cells in a body fitted mesh with 100,000 alumina particles were used. The number of fluid cells in the axial direction was 15 cells with a total axial length of 40 particle

diameters, making the calculations fully three-dimensional (3D) for both fluid and particulate phase.

Boundary and initial conditions

All the external bounds of the kiln consist of no-slip walls. Initially all the particles were introduced uniformly throughout the cylinder and then allowed to reach mechanical equilibrium at the bottom of the stationary kiln, under the influence of gravity. After settling, the curved walls were instantaneously heated and maintained at a temperature of 373K and the kiln was rotated at a constant rate of 20 RPM. The flat axial walls of the kiln were adiabatic to both fluid and particulate phase. Initially all the particles, air and the side walls are assumed to be at the room temperature of 298K.

Computational details

All the fluid pertinent equations were solved semi-implicitly with the Crank-Nicolson scheme. The particle equations were solved explicitly. The coupling terms computed as per equation 4.6 and 4.9 were introduced in the equations 4.3 and 4.4 respectively as source terms. The quasi-steady model of heat transfer as listed in equation 4.14 for conduction between particle-particle and particle-wall was used. The turbulent viscosity and turbulent conductivity were neglected in equations 4.3 and 4.4 as the fluid flow in the kiln falls in the laminar flow regime.

The critical time step or the time of elastic collision between particles, based on the work by Tsuji et al. [103] and given in equation 5.2, was computed to be $1x10^{-4}$ seconds. In order to resolve the particle collisions, $1/20^{th}$ of the critical time step was used for the simulations. The material properties of the bed particles are listed in Table 5.1.

$$\Delta t_{critical} = \frac{\pi}{\sqrt{K\left(\frac{1-\delta^2}{m}\right)}} \qquad 5.2$$

where δ denotes a constant given by,

$$\delta = \frac{-\ln(e)}{\sqrt{(\pi^2 + \ln(e)^2)}} \qquad 5.3$$

The average temperature of all the particles in the bed is reported as average bed temperature in the computations and was used for comparison with the experimental

results [90]. In order to quantitatively describe the dynamics of temporal evolution of the particle temperature field, the conduction heat transfer flux, convective heat transfer flux and average bed temperature were computed. These variables were used to examine heat transport mechanisms in the system of interest here. Since the maximum possible temperature difference is 75 K for the rotary kiln, the temperature for calculations of air properties is assumed to be constant at room temperature.

Table 5.1 Particle properties and parameters used in the rotary kiln simulations

	Notation	Al particles
Number of particles	N	100000
Density (kg/m^3)	P	3900
Diameter (m)	d_p	0.002
Poisson's ratio	σ_p	0.3
Young's Modulus (GPa)	*E*	193.05
Friction Coefficient		
particle-particle	$\mu_{p\text{-}p}$	0.5
particle-wall	$\mu_{p\text{-}w}$	0.7
Thermal conductivity (W/m-K)	*K*	36.0
Heat Capacity (J/Kg-K)	Cp	875.0
Coefficient of restitution	E	0.8
Spring stiffness coefficient (N/m)	K	6000
Time step (seconds)	Δt	5x10^{-6}

Results and discussion

The 3D calculations of the full scale kiln took about 4.5 days of wall clock time on 25 CPU cores for 12 seconds of physical time or 4 rotations of the rotary kiln after initial particle settling. The results of the full scale rotary kiln are discussed in this section.

This work uses non-dimensional temperature based on equation 5.4, where the reference temperature used was the ambient temperature of 298K.

$$T = \frac{T^* - T^*_{ref}}{T^*_w - T^*_{ref}} \qquad 5.4$$

Figure 5.2 shows the variation of non-dimensional average bed temperature for full scale kiln simulation and experiment. The computational average bed temperature under predicted the experimentally measured bed averaged temperature by Chaudhuri et al. [90]. A number of experimental and computational deficiencies can potentially lead to these differences. The experimental study reports the average bed temperature by measuring temperature at ten locations in the bed along a vertical line as opposed to the

average temperature of all the bed particles. In the experiments, when the bed is stopped for measurements the thermocouple locations align with the maximum bed depth direction and the reading of temperature is taken in that direction. During thermocouple insertion in the bed in the experiments, it is very likely that the temperature recorded by the thermocouple is an average measure of the fluid and particle temperatures. The difference between average bed temperature and experiment can also be attributed to the coarse fluid grid which is a requirement for the volume averaged nature of the gas phase equation to be valid. The coarse fluid grid near the heated kiln walls cannot resolve the thermal boundary layer with precision and subsequently could reduce the heat transferred from the wall-to-fluid-to-particles.

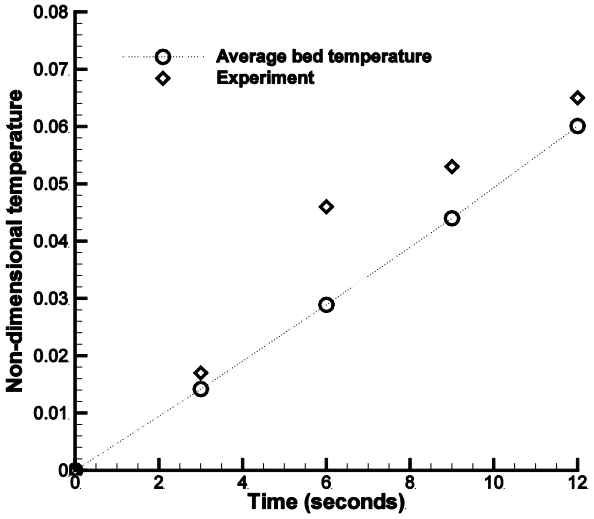

Figure 5.2: Average bed temperature in a rotary kiln running at 20 RPM compared with experimental data.

Hydrodynamics and thermodynamics

Figure 5.3 shows the flow field of air in the laboratory sized rotary kiln in hydrodynamic equilibrium after 12 seconds of rotation. The lip in the particle flow, a characteristic of cascading flows, can be seen in the figure. The stream traces show the presence of two counter-rotating vortices, one of which is present in the particle bed, and the other in the free board region. Since air flow is obstructed in the particle bed, convective velocity of air is small and is not particularly effective in facilitating convective transfer of heat to the fluid in the particle bed. On the other hand, the rotating flow cell in the freeboard is

more effective in convecting heat from the wall into the interior as seen in the figure. Axial transport of fluid at the core of these counter-rotating vortices is negligible.

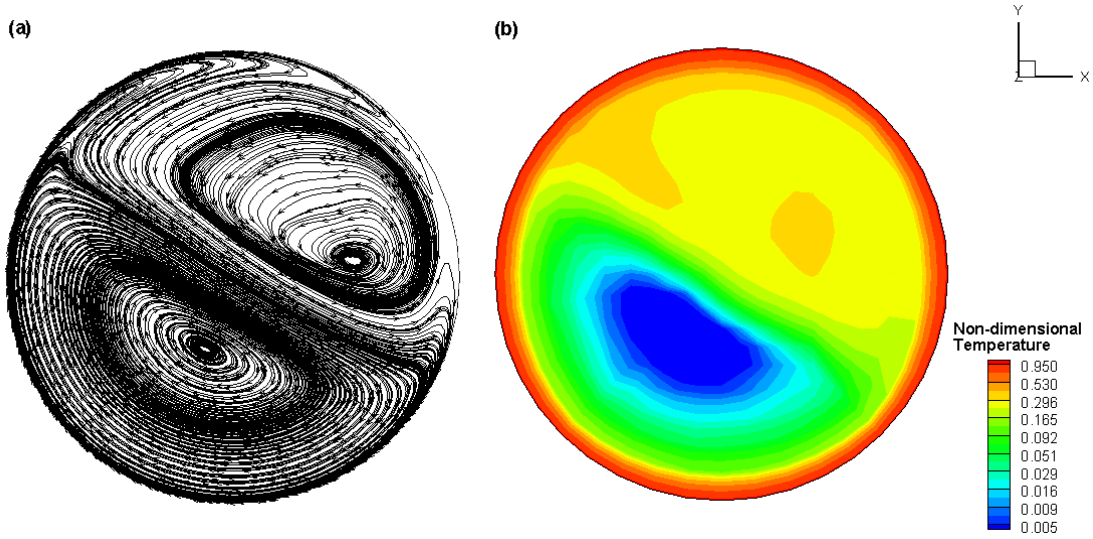

Figure 5.3: (a) Air stream traces and (b) Non-dimensional air temperature in the full scale rotary kiln after 12 seconds from the stationary position

The granular flow with temperature evolution of particles is shown in Figure 5.4. The cascading flow simulations are shown after every revolution of the kiln starting from the stationary bed position. In Figure 5.4 (a), the internal core of particles which remains quasi static during the calculations can be distinctly seen at the middle of the bed. As time progresses, the near wall particles heat up due to both conduction heat transfer as well as convection heat transfer. These particles are transported to the freeboard as the kiln rotates. With subsequent rotations, the solid's thermal boundary grows as the heat penetrates to include more heated particles. This happens as heated particles from near the wall transmit heat to surrounding particles as they move toward the freeboard region and as particles from the core slowly diffuse toward the wall under the action of gravity. Figure 5.4 (c) indicates that there are fewer hot particles on the freeboard side since only a thin layer of particles adjacent to the rotating walls are transported to the freeboard.

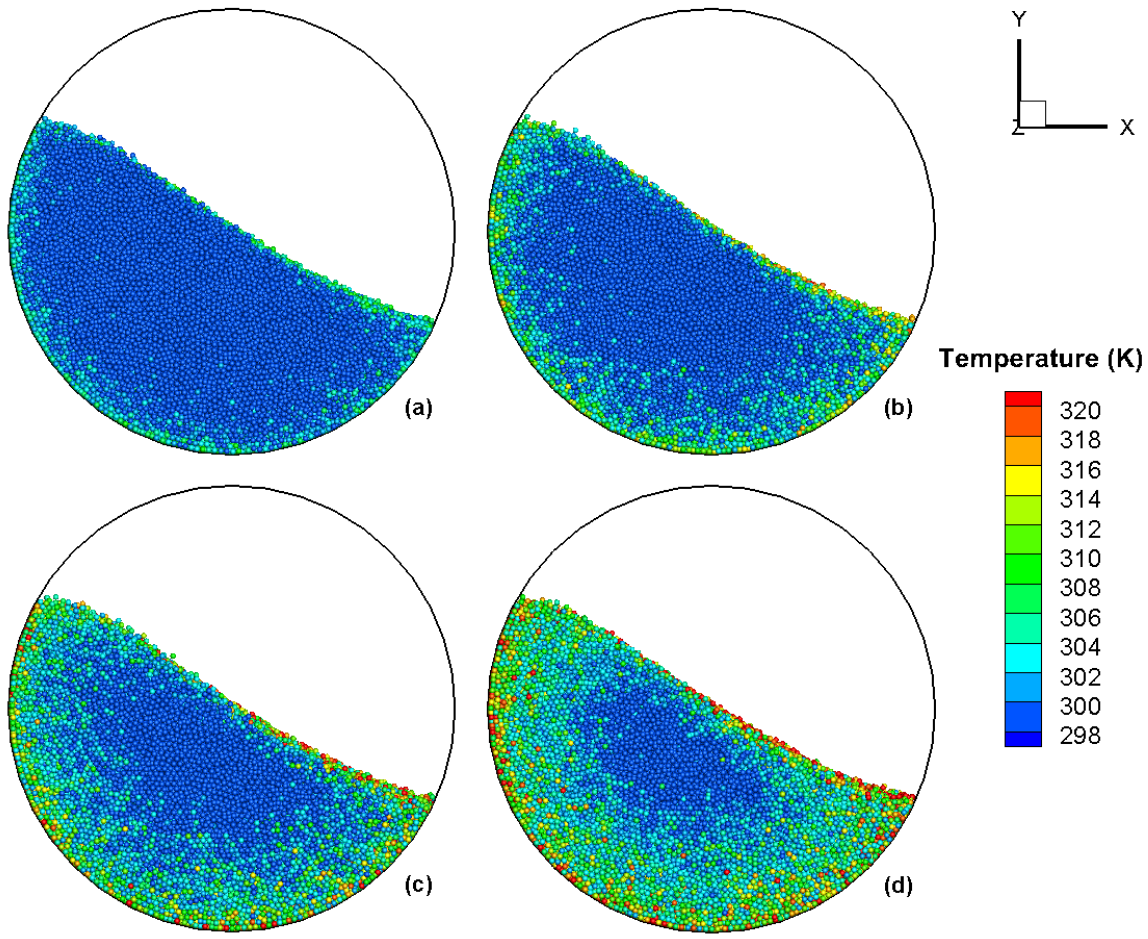

Figure 5.4: Particle temperatures in the full scale rotary kiln (a) after 3 seconds, (b) after 6 seconds, (c) after 9 seconds and (d) after 12 seconds from stationary bed position

To separate out the heat transfer mechanisms of particle heating in the cascading kiln, particle-wall conductive heat transfer fluxes and convective heat transfer fluxes between fluid-particles are recorded. The average of these heat fluxes over all the particles was calculated and is reported in Figure 5.5. The contribution of convective heat transfer from fluid-to-particle in the near wall region dominates the direct transfer of heat from wall-to-particles by conduction. Conduction heat transfer is about 10% as compared to about a 90% contribution from convective heat transfer.

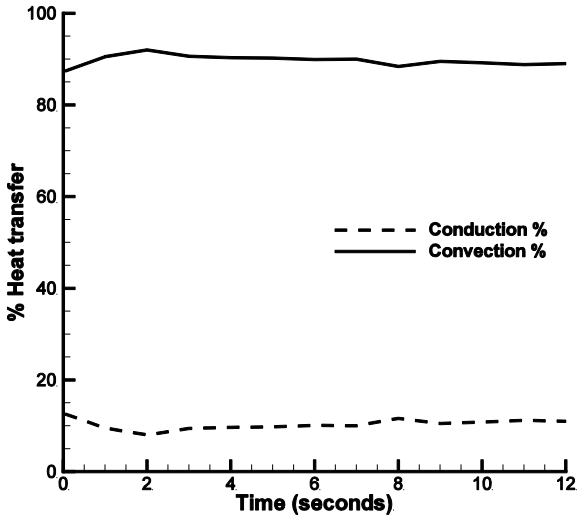

Figure 5.5: Conduction heat transfer between particle-wall and convection heat transfer between air-particles in the rotary kiln

The full scale rotary kiln was approximately divided into 5 axial sections and the average particle temperature in these sections was calculated. The axial variation in temperature can be seen in Figure 5.6. Sections 1 and 5 both encompass the wall region and have the end wall effects, whereas the temperature variation between sections 2 – 4 is comparatively negligible. Thus the axial temperature varies only near end wall whereas in the middle of the particle bed, there is not much variation.

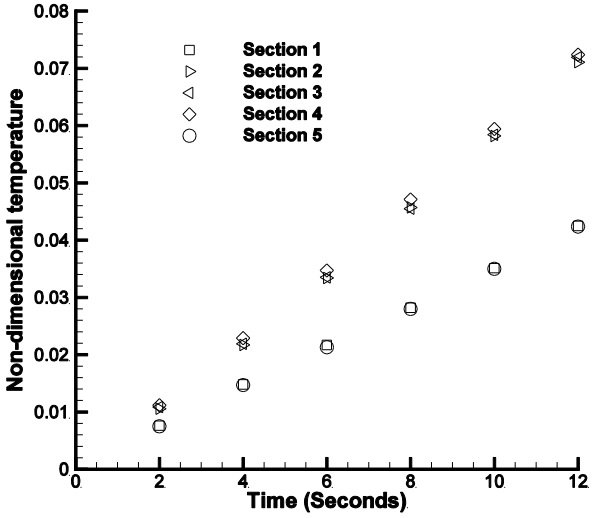

Figure 5.6 Axial variation of average particle temeprature in the full scale rotary kiln

Heat transfer in poly-dispersed rotary kiln with effect of modulus of elasticity

Additional simulations were performed to study the effects of particle size distribution on heat transfer in a rotary kiln. The geometry selected for these simulations was based on the experimental setup of [98]. In order to reduce the computational expenses, a thin section of the rotary kiln with periodic boundaries in the axial direction was used. As observed in the previous study with the full scale rotary kiln, there are minimal axial variations of average bed temperature in the middle of the kiln and thus a thin section is found to be sufficient. The kiln dimensions were 0.4 m of diameter and 9 times average particle diameter thickness. The bed had about 20% fill level. The fluid domain was decomposed into 5 body fitted mesh blocks. The properties of sand particles used are listed in Table 5.2. The properties of the kiln wall material are assumed to be the same as the particles. Note that the modulus of elasticity of the sand particles is about 20000 times less than that of the Aluminum particles used in the previous section.

Table 5.2 Particle properties and parameters used in the rotary kiln simulations with particle size distribution

	Notation	Sand particles
Number of particles	N	58000
Density (kg/m^3)	P	1428
Average particle diameter (m)	d_p	0.0019
Diameter range with normal distribution (m)		0.000988-0.002811
Poisson's ratio	σ_p	0.3
Young's Modulus (MPa)	*E*	10 (70, 700)
Friction Coefficient particle-particle	$\mu_{p\text{-}p}$	0.4
particle-wall	$\mu_{p\text{-}w}$	0.4
Thermal conductivity (W/m-K)	*K*	2.9
Heat Capacity (J/Kg-K)	Cp	733
Coefficient of restitution	E	0.9
Spring stiffness coefficient (N/m)	K	800
Time step (seconds)	Δt	$6x10^{-6}$

Boundary conditions

Initially the particles were uniformly distributed in the rotary kiln and allowed to settle under the influence of gravity. The particles and the air in the bed were initially at ambient temperature of 25 Celsius. After the particles reached a mechanical equilibrium

(particle velocity < $1x10^{-6}$ m/s approximately), the kiln was rotated with 1 RPM and the kiln walls were instantaneously heated to 800 Celsius.

All the fluid equations were solved semi-implicitly whereas the particle equations were solved explicitly. The quasi-steady model (equation 4.14) of conduction heat transfer between particle-particle and particle-wall was used. The temporal evolution of the particle temperature field is described using contribution from the conduction heat transfer flux, convective heat transfer flux, and radiation heat transfer. Since the maximum possible temperature difference is 775 K for the rotary kiln, variable air properties were used based on Sutherland's law as discussed in chapter 4 .

Parallel performance

Using OpenMP parallelism, 12 seconds of simulation took about 10 days where as with MPI it took 65 days. Here the advantage of OpenMP parallelism gets magnified due to the fact that even though the fluid domain is decomposed into 5 fluid blocks the particles can be decomposed in as many threads as the number of particles would allow to sustain a high parallel efficiency. For parallel efficiency, 16 maximum OpenMP threads were specified compared to 5 static MPI processes yielding over 600% time savings.

Results and Discussion

The experiments by Dhanjal et al. [98] report the bed temperature at 4 different probe locations. The temperatures recorded by the probe placed 0.01 m away from the wall in the deepest bed section, is used for comparing the results. The temperature reported in the computations is the average of fluid and particulate temperature at the probe location calculated based on equation 4.28. The comparison with probe temperature is shown in Figure 5.7. The experimental results are reported for a much longer duration and due to the high computational cost associated with such long durations, only simulation results up to 74 seconds are included in Figure 5.7. The computational results match within 10% of the polynomial curve fit to the experimental data.

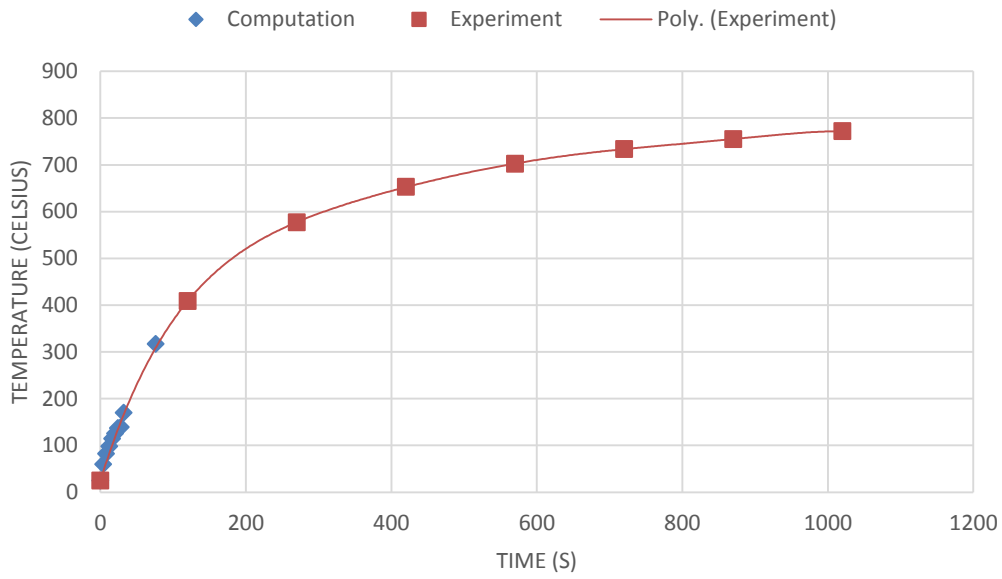

Figure 5.7 Temperature comparison with experiments for a poly dispersed rotary furnace

Instantaneous particle temperatures in rotary kiln are shown after every 20 seconds for first minute of calculations in Figure 5.8. As it can be seen, the internal core of particles remains quasi static during the single revolution simulated. The heating from the wall is dominant but as the time progresses, the heating of particles near the freeboard becomes more prominent. Since the particles are in slipping bed mode, the particles near the freeboard do not get transported along the freeboard as they experience negligible rolling motion. Thus the shrinkage of the colder particle core from the freeboard side is dominated by convective heat transfer from the air which gets heated in the freeboard cavity with time.

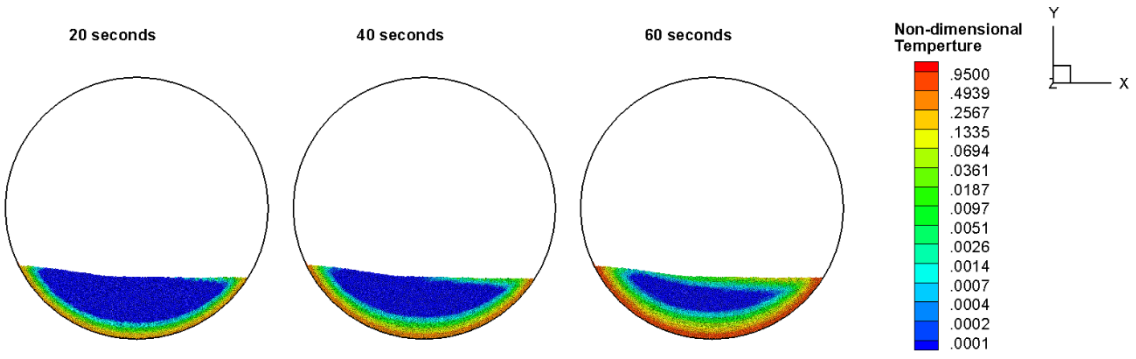

Figure 5.8 Non-dimensional particle temperature after 20, 40 and 60 seconds

The average contribution of various modes of heat transfer is shown in Figure 5.9. At early times, conduction heat transfer from the wall is the dominating mode of heat

transfer. However, after about 40 seconds, there is a sharp increase in the percentage of heat transferred to the particles by convection and a modest increase in the heat transferred by radiation. The increase in convective heat transfer is because the air in the freeboard region gets heated enough that the convective heat transfer from air to particles at the freeboard interface increases. The increase in radiative heat transfer as time progresses is due to the increase in particle and fluid temperature and the fourth power dependence of radiation heat transfer on these quantities. This behavior has also been observed in other fluid-particulate systems such as fluidized beds [86].

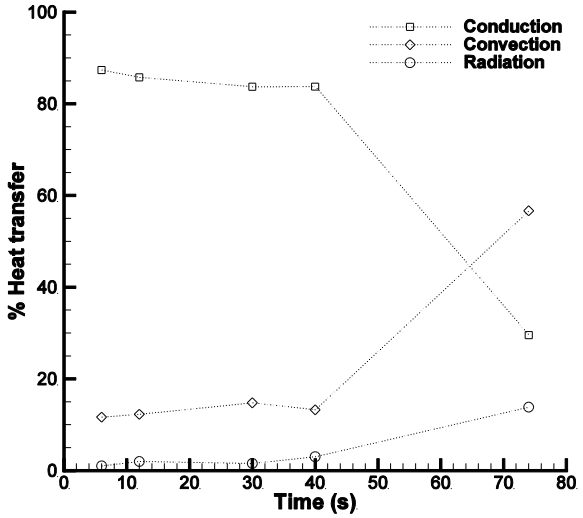

Figure 5.9 Decomposition of various modes of heat transfer in the rotary kiln

The Figure 5.10 shows aggregate contribution of various modes of heat transfer for different particle sizes. The overall range of diameters was subdivided into 9 different sizes and the average heat transfer contribution for the first 74 seconds was calculated. Since radiation heat transfer was negligible during the initial period, its contribution is not included in the Figure 5.10. As seen from the figure, both conduction and convection are relatively less in particles with small diameter except for the smallest particles (<1.4 mm) whereas the larger particles tend to dominate the heating of the particulate phase. The main reason for this to happen is because the large particles settle at the bottom of the bed initially and are in the proximity of the heated kiln walls for most of the time averaging duration. The smallest particles reside in the voids of these larger particles near the wall and also get heated due to wall conduction and near wall convection as time progresses. When looking at the larger particles (>2.6 mm), the conduction heat transfer

is high as the area of contact between the particle and the heated wall is much larger than the smaller particles.

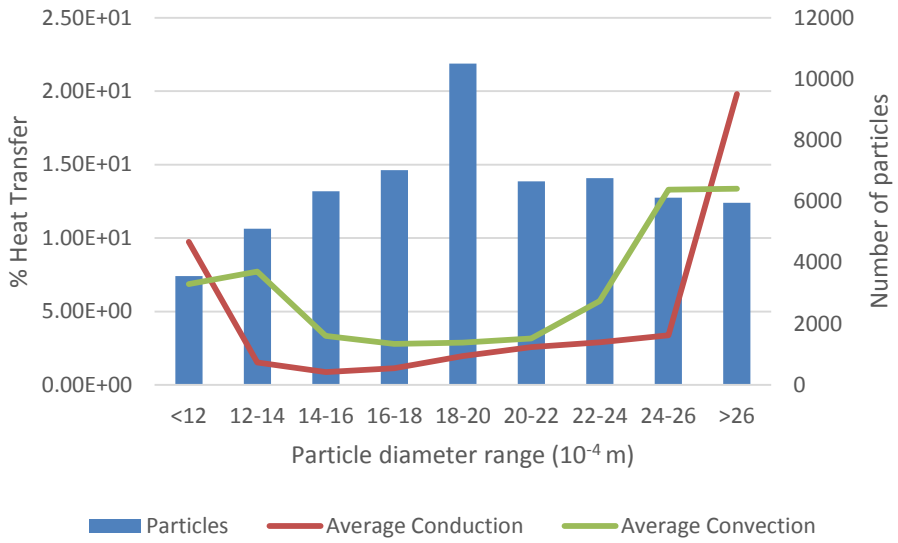

Figure 5.10 Effect of particle size distribution on heat transfer modes

This behavior of larger particles segregating towards the outer walls and subsequently to the free board is also observed in the literature [97]. The overall effect of the particle size distribution on the flow field is that the smaller particles occupy the voids created in between the larger particles as shown in Figure 5.12. This gives rise to low void fraction or a tightly packed bed and thus low convective heat transfer. Also, there is higher particle-particle and particle-wall contact area giving rise to higher conductive heat transfer. This is because of the lower modulus of elasticity. The coefficient of conduction heat transfer in the quasi steady model is inversely proportional to the modulus of elasticity. In the poly dispersed case the modulus of elasticity of particles is four orders of magnitude smaller than the mono dispersed case. Thus conduction plays a much larger role in the poly dispersed rotary kiln. The effect of variation of modulus of elasticity is shown in Figure 5.11 by comparing the actual values and percentage heat transferred by different modes of heat transfer during the first 5 seconds of rotation. The conduction heat transfer is larger for low modulus of elasticity and it drops with an increase in the modulus of elasticity whereas the convective heat transfer remains almost the same. Thus, with the increase in modulus of elasticity the contribution of conduction heat

transfer reduces as the contact area between particle and heated walls reduces and consequently the percentage of convection heat transfer increases.

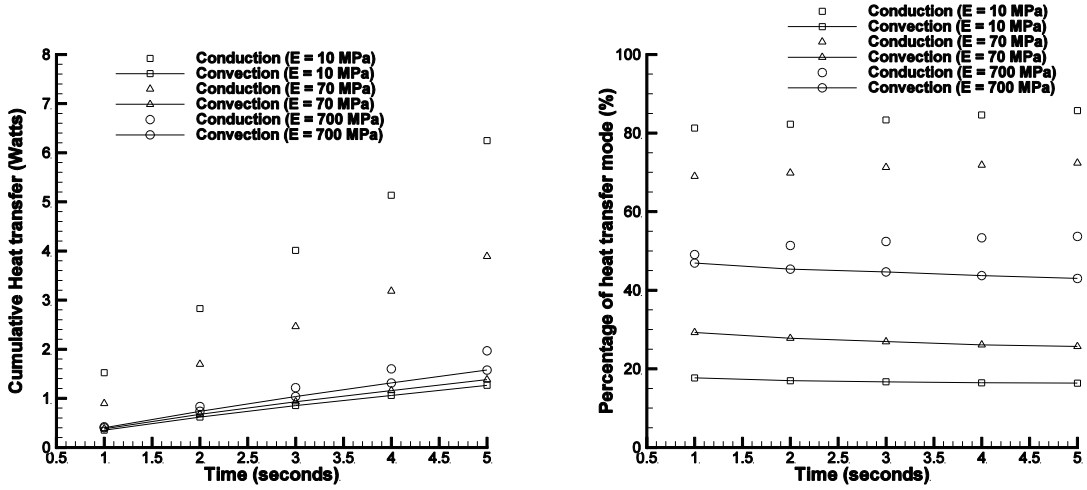

Figure 5.11 Variations in heat transfer mechanisms with change in modulus of elasticity

The thermal boundary layer growth for fluid and for particulate phase can be noted in Figure 5.12. The fluid field develops the boundary layer more rapidly in the free board area where as it is predominantly absent in the particle bed as the particles take away most of the heat from fluid via convection.

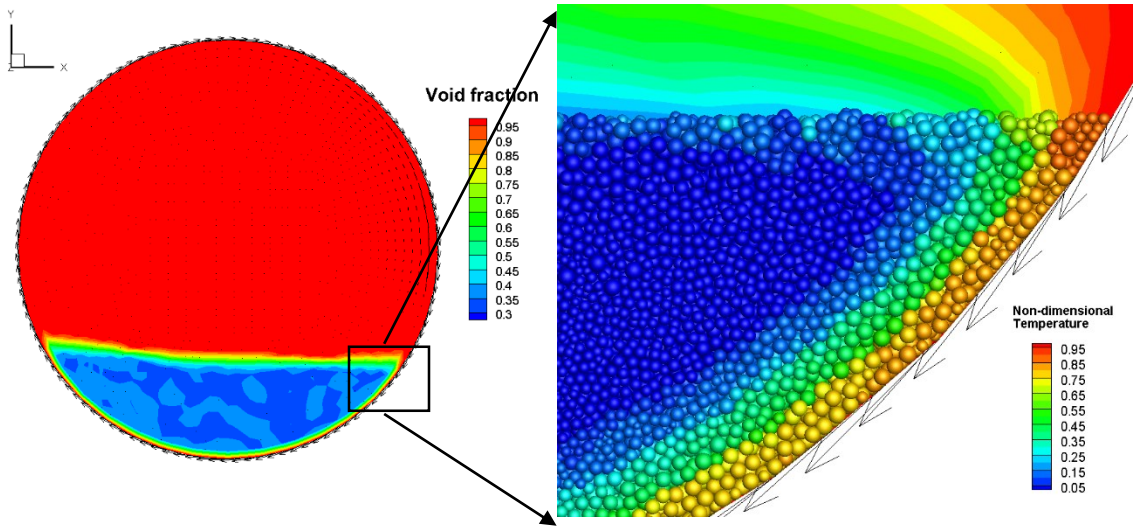

Figure 5.12 Void fraction profile in the rotary kiln with magnified view of particle size distribution colored by temperature after 74 seconds. The arrows represent fluid velocity vectors

Summary

In this study, numerical simulations were performed using coupled Discrete Element Method (DEM) and Computational Fluid Dynamics (CFD) to analyze heat transfer in a non-reacting rotary kiln.

Microscopic models of particle-particle, particle-fluid, particle-surface and fluid-surface heat transfer are used in the analysis. For the mono-dispersed aluminum particles the results show that the convective heat transfer between particle and air dominates the overall heat exchange. Particles are heated near the rotary kiln walls by convection heat transfer as they pass through the thermal boundary layer of the heated fluid. These particles are transported to the center of the kiln where they transfer heat to the cooler particles in the quasi static core of the kiln and back to the cooler fluid at the center of the kiln. It is found that 90% of the heat transferred to particles from the kiln walls is a result of convection heat transfer, whereas only 10% of the total heat transfer is due to direct conduction from the kiln walls.

When particle with different sizes are introduced in the kiln, the bed becomes tightly packed. The modulus of elasticity of the poly dispersed sand particles is much smaller than the mono dispersed aluminum particles and thus the convective heat transfer becomes secondary to conduction heat transfer as the main mode of bed heating at the beginning of the kiln rotation. Convective and radiative heat transfer play a larger role as the bed temperature increases.

Heat transfer in fluidized bed with a tube heat exchanger

Introduction

Fluidized beds heat exchangers have been used to enhance heat transfer capacity which has led to many studies of fluidized beds with immersed heat transfer tubes. In these exchangers, hydrodynamic and thermodynamic bed characteristics around the tube surface are critical in understanding the heat transfer mechanisms to or from the tube with the bed. These characteristics have been obtained through several experimental studies. They have led to development of mechanistic models and subsequently numerical correlation for calculating heat transfer coefficient in such systems. More recently

computational studies have been used to analyze the heat transfer coefficients due to the presence of immersed tubes in fluidized beds. A brief overview of the literature is presented below.

Experimental studies

Heat transfer on immersed surfaces in a fluidized bed is a function of the thermal properties of the solid and gas, the state of fluidization in the bed or fluidization velocity, particle size, and orientation of the surface in the bed. The transfer of heat from the bed to any surface takes place through direct particle contact with the surface through unsteady conduction and through gas convection for bed temperatures below 700 K [104] for which radiative effects can be neglected. Hence the dominant mode of heat transfer is dependent on the solid and fluid fraction in the vicinity of the surface of interest.

The majority of experimental studies have quantified heat transfer to walls and tubes through measurement of water inlet and exit temperatures and surface temperatures [105, 106] or electrically heated probes in a room temperature bed [107, 108]. Other specially designed probes were used to measure local heat transfer at different locations around an immersed tube as well as local particle solid fraction [107].

Experiments suggest that the heat transfer coefficient initially increases with particle diameter up to about 100 microns and then decrease as the size increases further up to about 1 mm, after which there is a gradual increase [109-114]. For very fine particle (< 50-100 micron diameter), the heat transfer is mostly controlled by the rate of exchange of particles between the bulk and the wall and on contact the particles reach thermal equilibrium with the wall very quickly transporting heat to or from the surface. The decrease in heat transfer coefficient for d_p>100 micron is due to an increase in the gas fluidization velocity, a decrease in solids fraction near the heat transfer surface, and an increase in gas resistance to heat flow between the particle and surface. In this regime, the particles that come in contact with the wall do not change their temperature substantially during contact. For larger particles, because of the higher fluidization gas velocity, the bed transitions to turbulence and heat transfer due to gas convection increases at the surface, increasing the heat transfer coefficient. Consequently, particle heat capacity has a substantial effect on the heat transfer coefficient at small particle

diameters but a small effect for large particle diameters. The particle thermal conductivity has a significant influence on heat transfer coefficient only when kf/kp > 10% [115]. On the other hand, the heat transfer coefficient increases linearly with gas thermal conductivity [115].

Mechanistic models and numerical correlations

A number of mechanistic models have been proposed to describe the heat transfer in fluidized bed heat exchangers. The earliest and the most studied is the "packet" model of Mickley et al. [116] for bubbling beds. In this model they use a picture of fluidization in which a small group or assembly of particles ("packets") move as individual homogeneous units through the bed as the dense phase is stirred and that the principle resistance to the transfer of heat from surfaces to dense fluidized beds exists in the layers of solid particles nearest the surface. The packets are not permanent but are accorded a finite persistence in time given by a root mean square mean residence time and a time fraction of contact by the dense phase. During contact, the void fraction, density and heat capacity are those of the quiescent bed and are assumed to be homogeneous. The model disregards any changes in the structure of the packet during contact, disregards any convective heat transfer between the surface and the gas, disregards any change in temperature of the packet for long residence times, and disregards any contact resistance between the packet and the surface and the orientation or geometry of the surface.

Subsequently a number of modifications have been proposed to this model to account for these deficiencies. Baskakov [117] and Koppel et al. [118] introduced an empirical, time-independent contact resistance at the wall-packet interface. Gelperin et al. [119] solved the basic packet model in terms of two heat transfer resistances, one due to the increased voidage in the vicinity of the wall that extended to one-half the diameter of a particle, and the other due to an adjoining two-phase packet. Baskakov [114] introduced gas to surface convection in the model. Antonishin et al. [120] analyzed heat conduction by allowing for local temperature relaxation in the heterogeneous medium. Kubie at al. [121] accounted for the wall effect by introducing a property boundary layer. Ozkaynak et al. [122] used the concept of penetration depth to make an allowance for the influence of the wall on void fraction at short residence times. They also experimentally determined

empirical curves for the mean residence time and time fraction as a function of excess gas velocity over the minimum fluidization velocity.

While the packet or emulsion model worked reasonably well in dense beds with small particles (typically less than 1 mm), it breaks down for large particle diameters and high fluidization velocities, when gas convection starts playing a dominant role. Many investigators [110], [123-132] have investigated convective gas-surface heat transfer and have developed correlations for large particle systems.

Numerous correlations have been developed from the mechanistic models and theories put forth in the literature for vertical surfaces e.g. [133-137] and single tubes e.g. [108, 138-144] with diameter D_T placed in a fluidized bed with particle size d_p and superficial fluidization mass flux given by G. Typically the correlations are expressed in terms of $Nu = f\left(Ar, Re, Re_p, Pr, \varepsilon, d_p, D_T, fluid\ and\ solid\ properties\right)$ where $Nu = hD_T/k_f$, $Ar = \frac{g d_p^3 \rho_f (\rho_p - \rho_f)}{\mu_f^2}$ is the Archimedes number, $Re = GD_T/\mu$ and $Re_p = Gd_p/\mu$, Pr is the Prandtl number, and ε is the void fraction. Often, due to the lack of data in the immediate vicinity of the tube, ε is assumed to be that of a packed bed at about 0.40, while others use correlations relating ε to the fluidization velocity. A sample study of the variation in heat transfer coefficients for an immersed tube in fluidized bed is listed in Appendix A.

Computational studies

The most common computational approach used in the past has been the Eulerian-Eulerian two fluid approach in which the solid phase is treated as a continuum. With respect to heat transfer calculations, one of the major sources of uncertainty is the calculation of the effective thermal conductivity of solid and fluid phase used in the calculations. The effective thermal conductivity depends on the thermal properties of the gas and solid, on the void fraction, and on the collision between particles in the bed. There are many approaches cited in the literature to find the effective thermal conductivity. The standard approach is that given by [145] which calculates the bulk conductivity based on spherical packing in a stationary bed and as a function of the fluid and solid thermal properties [146]. In another approach, the kinetic theory of granular

media is applied to estimate the effective solid and gas conductivity taking into account the collision between solid particles [147].

Schmidt et al. [148] used two different effective thermal conductivity models for Geldart B particles for 2D simulations of fluidized beds on a structured grid to observe the effects on a single immersed tube and multi immersed tube heat exchangers. They observed that the particle renewal at the wall led to instantaneous high heat transfer coefficients. There was a significant gap in the experimental results and the simulations in the average temperature profiles. This was attributed to the short simulation periods (2s) compared to the experiments (60s). Another recent study with the two fluid model approach has been performed by [149] for an immersed tube bank in a fluidized bed. They used the standard approach for calculating effective thermal conductivity in the bed but used a modified approach near the wall [150]. In their 2D fluidized bed simulations, the immersed tubes are modeled with a square cross-section. A deviation of 20% from the experimental results was observed. The difference has been attributed to the 2D simulation, no turbulence models, and the square tube geometry and low averaging times. Armstrong et al. [151] have also carried out heat transfer simulations for 1, 2 and 3 tubes immersed in fluidized beds using the two fluid model approach. The effective thermal conductivity model used in their study is simply the addition of the two different approaches taken by [148]. 2D simulations with Geldart B particles are performed and compared with another simulation study [152] showing over 300% differences. The authors attributed the differences to the symmetry conditions used by [152]. Armstrong et al. [153] used the same computational set up as their immersed tube paper [151] for calculating the heat transfer coefficients between a fluidized bed and walls. Geldart B particles in 2D bed simulations were used and 25-30% difference in estimation of the heat transfer coefficient compared to experiments was observed. This difference in results was ascribed to the difference in the initial conditions and also to the short duration of simulations (2s). Patil et al. [154] use Geldart B particles in 2D fluidized bed simulations using a two fluid model. Effective thermal conductivity is modeled based on a combination of the standard approach and kinetic theory of granular flow and the results are compared with experiments. Due to inaccuracies in the description of the hydrodynamics near the wall, differences in the wall heat transfer coefficients are

observed. Another study by Sun et al. [155] using two fluid model observed the effects of superficial velocity on the heat transfer coefficient for an immersed tube and found linear correlation between effective heat transfer coefficient and the superficial velocity albeit without experimental validation. Study by Dong et al. [156] over predicted the heat transfer coefficient around a circular and square tube as compared to the experiments as they found that the porosity model used in the two fluid modeling approach to be inadequate.

Recently, studies using coupled computational fluid dynamics and discrete particle method have also been applied to investigate heat transfer in fluidized beds. Di Maio et al. [104] performed 2D simulations using 25,000 Geldart B particles to estimate the heat transfer coefficient on a circular probe. The immersed boundary method is used for modeling the tube like probe. Based on the combination of the microscopic heat transfer models used, a range of heat transfer coefficients from 43 to 340 W/m^2-K are obtained as compared to the experimental value of 160 W/m^2-K. A model in which heat transfer through direct contact and conduction through the surrounding gas during contact is included, gives the closest results to the experiments. Zhao et al. [157] used a 2D unstructured mesh to perform simulations of fluidized bed with an embedded tube. A DEM approach is used with particle-particle conduction heat transfer modeled using the quasi steady state solution of contact conductance. The turbulent dense gas–solid two phase flow using the k-ε model and multiway coupling heat transfer model among particles, walls and gas is solved. Up to 68,000 particles of diameter 0.5, 1 and 1.5 mm are used. It is observed that the heat transfer coefficient value is low at the top and bottom of the tube, high at the left and right sides, and the minimum is 70–80% of the maximum, which shows qualitative agreement with experiments. There is no quantitative comparison with experimental studies. Hou et al. [86, 158] have also performed 2D simulations of an immersed tube in fluidized beds using the DEM approach with 30,000 particles of Geldart B type. A combination of both particle-particle conduction heat transfer modes (static and collisional) is applied. The near wall convective heat flux calculations are performed based on correlation instead of solving for the temperature field of the fluid. There is qualitative agreement with experiments but there is over 30-35% difference quantitatively.

In summary, extensive experiments have been done in the past to develop mechanistic models and numerous correlations for prediction of heat transfer coefficients to surfaces submerged in a fluidized bed. Most experiments have used energy balances to quantify the heat transfer coefficient. These efforts have yielded in several numerical correlations with more than 100% deviation in predicted heat transfer coefficients. Computational analysis has been limited on one hand by the incomplete physicality and phenomenological modeling of the two-fluid Eulerian-Eulerian model, and on the other hand by the computational expense of the more physics based discrete particle method based on the scale of individual particles, but which has been limited to two-dimensional calculations with less than 100,000 particles because of computational cost.

Problem description

The heat transfer study between the fluidized bed and a horizontal tube heat exchanger performed was based on the experiments performed by [159]. The effects of heat transfer between the tube and the fluidized bed are localized to the surrounding of the tube and thus a smaller sized geometry was used for computational simplicity [86]. A thin section of the fluidized bed is used with 9 times the particle diameter thickness. Both DEM and CFD calculations are performed in three dimensions (3D) in order to simulate the bed. Figure 5.13 shows a magnified view of the fluidized bed with tube heat exchanger. The domain decomposition used for fluid phase calculations and a body fitted mesh with dimensions of 2.5 to 3 times the particle diameter is also highlighted in the figure. The geometry parameters and the material properties of particles, tube, and walls are listed in Table 5.3.

Table 5.3 Particle properties and parameters used in the fluidized bed with tube heat exchanger simulations

Bed			
Width (m)	W	0.06	
Transverse Thickness (m)	T	0.0054	
Height (m)	H	0.768	
Tube			
Height from bottom of bed (m)		0.03	
Diameter (m)	D_t	0.024	
Simulation parameters	Notation	Sand particles	Tube/Wall properties
Density (kg/m^3)	ρ	2600	
Thermal conductivity (W/m-K)	κ	1.1	380
Heat capacity (J/Kg-K)	Cp	840	24.4
Elastic modulus (MPa)	E	10	10
Poisson's ratio	σ_p	0.3	0.3
Coefficient of normal restitution	e_n	0.9	0.9
Coefficient of friction	μ_{p-p}	0.3	0.3
Spring stiffness coefficient (N/m)	K	800	800
Initial temperature (K)	T_{init}	298	298
Sphericity	Sp	1	
Number	N	67500	
Diameter (mm)	d_p	0.6	
Time step (seconds)	Δt	$2x10^{-5}$	

Methodology

The methodology used for this simulation is discussed in chapter 4. The properties of air are assumed to be constant since the maximum temperature difference is 75K. For particle-particle and particle-surface conduction heat transfer, the quasi-steady formulation given in the equation 4.14 is used. The radiative heat transfer was neglected since the maximum temperature reached was << 700K.

Periodic boundary condition is applied to the front and back of the fluidized bed to avoid end wall effects. Convective outflow boundary condition is applied at the outlet. Ambient air at a predetermined superficial velocity is injected from the bottom wall to fluidize the bed. The immersed tube surface was set to a constant temperature of 373K whereas all the other walls were set as adiabatic for both particle as well as fluid phase.

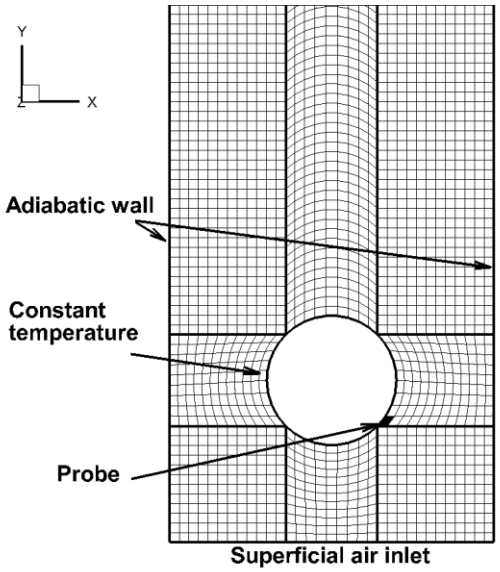

Figure 5.13 Magnified view of body fitted mesh around tube heat exchanger in a fluidzed bed with domain decomposition for fluid phase calculations.

Initially, the particles are distributed uniformly in the fluidized bed and allowed to settle. Once the particles are settled, the tube is instantaneously heated and ambient temperature air with 0.77 m/s superficial velocity is injected in the bed.

The average heat transfer coefficient (HTC) is calculated based on the following formulation,

$$h = \frac{Q_{fluid-tube} + \sum_{i=1}^{n} Q_{pcw,i}}{A_s(T_{tube} - T_{bed})} \qquad 5.5$$

where T_{bed} is calculated based on equation 4.28 for which the domain considered is the computational cell size, $Q_{fluid-tube}$ is the convective heat transfer and is obtained from zonal two layer wall model as listed in equation 5.7, Q_{pcw} is the particle-tube conduction heat transfer and A_s is the surface area of the tube (computational grid).

The angular position used the in the results is measured form the top center of the tube as shown in the insert of Figure 5.14.

Wall modeled LES

The requirement of a coarse grid which is at least 2-3 times the particle diameter due to the volume averaged nature of the governing equations [48] limits the use of DEM for heat transfer studies because the thermal boundary layer cannot be resolved, particularly at high fluidization velocities. Thus there are a limited number of immersed tube heat

transfer studies in fluidized beds using DEM where they use a turbulent Nusselt number correlation to resolve the thermal boundary layer. This work attempts to tackle this issue by using LES with a zonal two layer heat transfer model at the walls. A brief description of this model is given below and further details can be found in [160].

The two layer wall model used in this study solves simplified boundary layer equations in the inner wall layer. These equations are solved on a virtual grid between the wall node and first off-wall node ($y^+<50$). By using the instantaneous outer flow velocity as a boundary condition for the inner layer to solve for wall shear stress in the inner layer and using this wall shear stress as the boundary condition to the outer layer at the first off-wall node, the coupling between the inner and outer layers is achieved.

The momentum and energy equations are solved in local wall coordinates (n,t) in the normal and tangential directions. By neglecting the convection terms and the time derivative terms from the momentum equation, considerable simplifications are obtained as shown in equation 5.6.

$$\frac{\partial}{\partial n}\left[\left(\frac{1}{Re}+\frac{1}{Re_t}\right)\frac{\partial u_t}{\partial n}\right]=\frac{\partial P}{\partial t} \qquad 5.6$$

with $u_t = 0$ at the wall and $u_t = ||U_t||$ at the edge of the inner layer. The turbulent Reynolds number (Re_t) is calculated using the Johnson-King turbulence model [160].

Neglecting the advection term and in absence of any additional source term, the simplified energy equation becomes,

$$\frac{\partial}{\partial n}\left[\left(1+\frac{RePr}{Re_tPr_t}\right)\frac{\partial T}{\partial n}\right]=0 \qquad 5.7$$

where Pr_t is turbulent Reynolds number. The turbulent Prandtl number requires a closure equation and the formulation of Kays is used [161]. This formulation accounts for the higher values of turbulent Prandtl number very close to wall. For the specified temperature boundary condition on the wall node, the boundary condition for the first off-wall node is specified as the heat flux boundary as obtained from the inner layer temperature profile.

Initial estimates of heat transfer coefficient around the tube

The calculation for the local HTC was initially performed without the use of LES and the wall model. The convective heat flux required in equation 5.5 was calculated by assuming a linear temperature profile between wall node (T_{wall}) and the first off-wall node (T_{fluid}) as follows,

$$Q_{fluid-tube} = -kA_s \frac{(T_{fluid} - T_{wall})}{\Delta n} \qquad 5.8$$

where Δn is the normal distance between the wall and the first off-wall node.

The comparison of the simulation results and the experimental results [159] is shown in Figure 5.14. Clearly the predicted HTC values are only about 20% of the measured values.

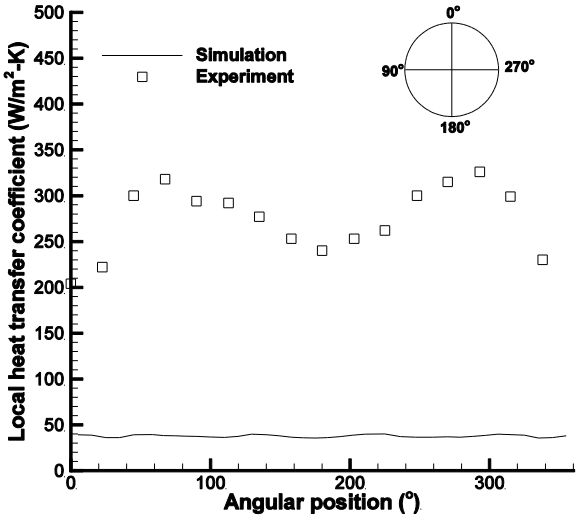

Figure 5.14 Local heat transfer coefficient around immersed tube without wall model

As mentioned previously, this issue is often seen in computational heat transfer studies in fluidized beds due to the coarse nature of the computational grid which is necessitated by the volume-averaged nature of the fluid equations. Most studies in the literature [74, 76, 86, 162, 163] suggest the use of heat transfer correlations for fluid-wall convection instead of actually resolving the thermal boundary layer. Thus in this work a novel approach of solving simplified flow and temperature fields is used to resolve the inner region of the boundary layers.

The use of the wall model to resolve the inner boundary layer

The comparison of the time averaged local HTC, calculated using the wall model, with experiments [159] is shown in Figure 5.15. As it can be seen from the figure the experimental results and the computational results are within reasonable limits. The overall trend also compares qualitatively with other experimental studies of [106, 107]. The computational results give jagged profile compared to experiments possibly because the time averaging of data in simulations is performed for 2.5 seconds; which most probably is much smaller as compared to the experimental data. The HTC is not uniform around the tube and the localized mechanism for heat transfer differs which is discussed later.

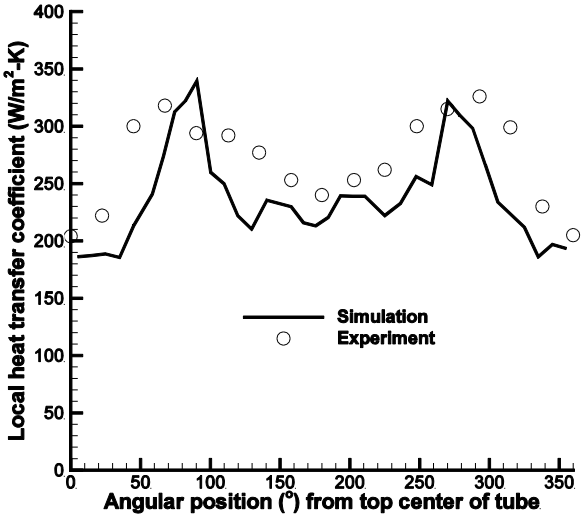

Figure 5.15 Local heat transfer coefficient around the immersed tube in fluidized bed.

The presently used wall model is applicable for fully turbulent flows. In order to justify the use of LES with a wall layer model in this study, velocity signal at a location 45° from the stagnation point of the tube, as shown by the darkened computational cell in Figure 5.13, was recorded. This signal is shown in Figure 5.16 along with the corresponding energy spectrum of the signal. As commonly seen in such energy spectrum [164], with increasing frequency the energy drops. The majority of the energy is associated with the lower frequencies which indicates that the flow is governed by larger flow structures. The lower frequencies associated with a signal in fluidized beds are normally associated with the bubble or particle packet motion and the higher frequencies

are attributes of local particle motion [165]. Such behavior also observed in the current study (based on power spectrum of the HTC) is typically associated with non-laminar flows. This does not necessarily indicate that the flow is fully turbulent but hints at the transitional nature of the flow. The y^+ around the tube was observed to vary between 10 and 50 and occasionally going up to 60.

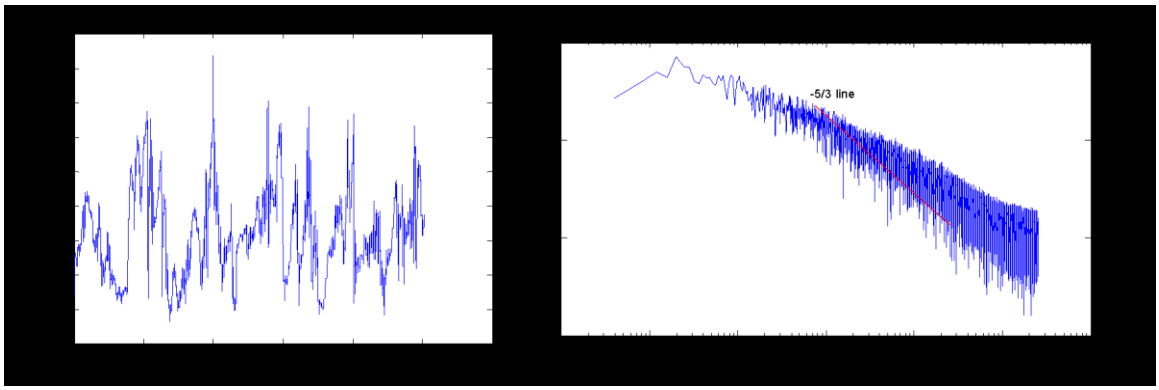

Figure 5.16 Velocity signal at the probe location and its energy spectrum

The Figure 5.17 shows the particle configuration for different instances in time. The particle temperature in the figure is non-dimensional temperature based on equation 5.4. The reference temperature is the ambient temperature of 298K.

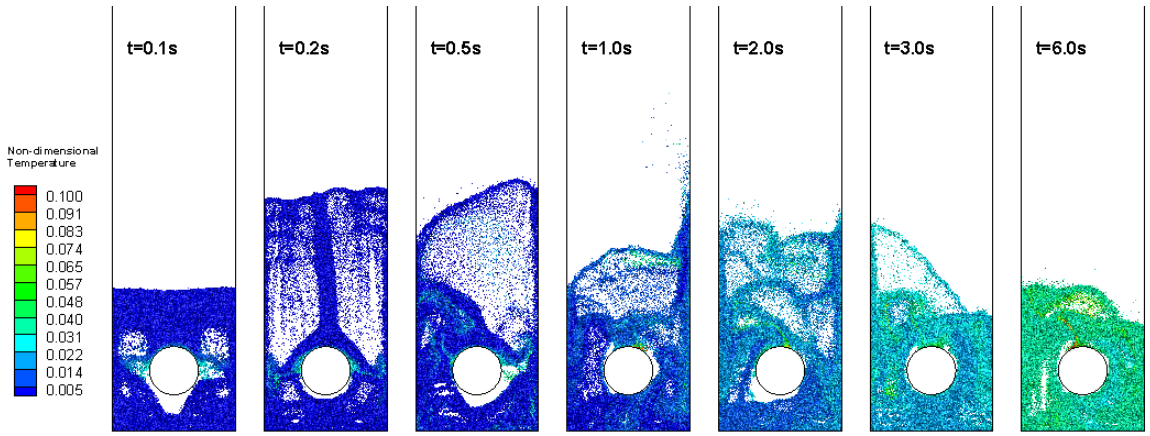

Figure 5.17 Particle positions at different time instantances colored by non-dimensional temperature

The nature of gas bubbles in the fluidized bed is completely changed due to the presence of the immersed tube. The particles in the bed move vigorously with the bubbles in the bed transferring heat from the tube surface. As seen in the Figure 5.17, packets of particles pick up heat from the tube surface and convect into the wake before collapsing

on the tube and into the bed. It can also be noted that except for the last two frames, the stagnation region of the tube is mostly encapsulated in a gas bubble.

The time averaged contribution of conduction and convection heat transfer along with the variations in the void fraction around the tube is shown in Figure 5.18. Convective heat transfer is the dominating mechanism of heat transfer accounting for almost all of the heat transfer in all the regions around the tube except the wake region of the tube. In the wake region, the particle heat conduction increases due to the longer residence times of particles and less violent fluid flow. The void fraction profile around the tube reaffirms the longer residence time of particles in the tube wake region.

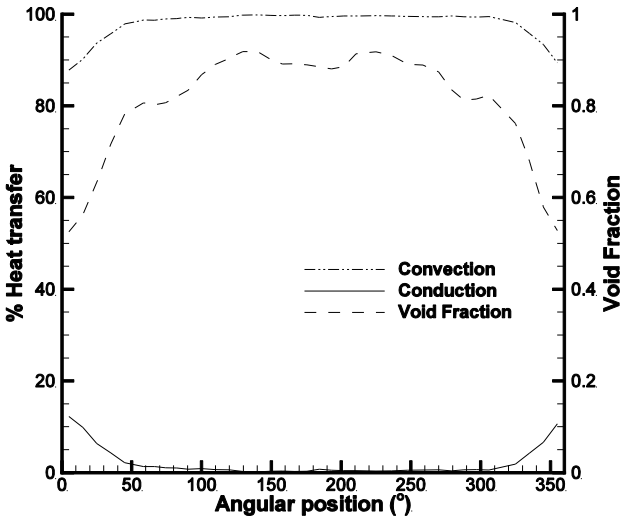

Figure 5.18 Time averaged contributions of conduction and convection heat flux along with average void fraction around the immersed tube

Higher the void fraction around the tube, larger is the probability of the presence of a bubble around the tube. Thus the average void fraction profile can be considered as an approximate indicator of gas bubbles around the tube. The average HTC around the tube is constantly changing based on the bed dynamics. The time evolution of the spatially average HTC and void fraction around the tube is shown in Figure 5.19 (A). The variation in HTC is clearly dominated by low frequencies indicating that the HTC is largely a function of bubble dynamics.

The time evolution of convection heat transfer and conduction heat transfer around the tube is show in the Figure 5.19 (B). The dominant mechanism of heat transfer is clearly convection, accounting for about 95% of the total heat transfer.

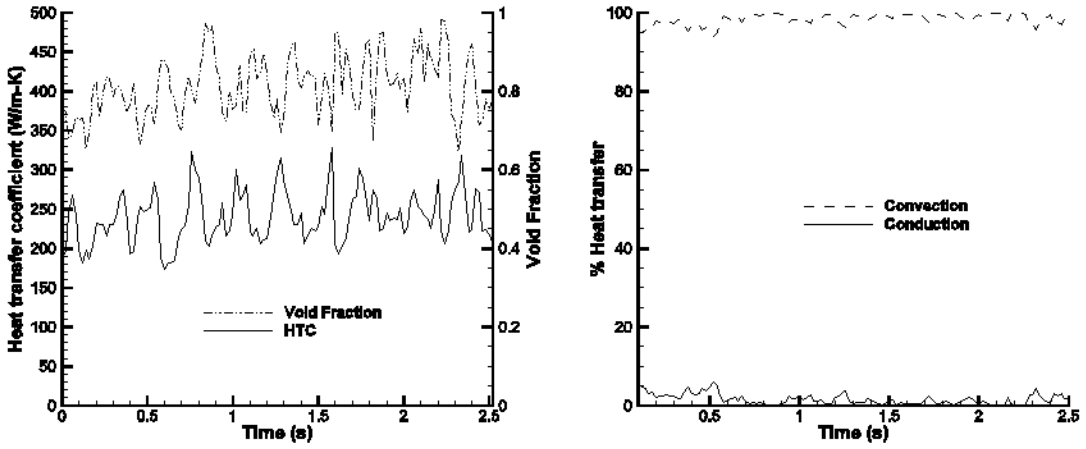

Figure 5.19 Time evolution of (A) void fraction and heat trasnfer coefficient and (B) contributions of conduction and convection heat flux, spatially averaged around the immersed tube

When comparing the results with available correlations in literature as shown in Figure 5.20, it can be noticed that the average HTC is of the same order as that is predicted by many correlations for 0.6 mm particles. Packing fraction is required to predict HTC in majority of these correlations. HTC is highly sensitive to the packing fraction which in this case was assumed to be 0.6.

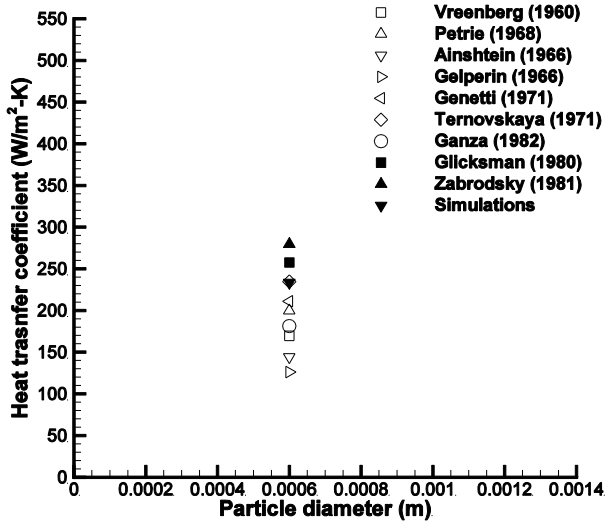

Figure 5.20 Comparison of numerical correaltions of local heat transfer coefficient [113, 131, 166]

Summary

In summary, performing heat transfer studies using coupled CFD-DEM is a challenge due to the coarse grid required to avoid discontinuities in the void fraction profiles. To overcome this issue, LES with a wall model was used in the framework of CFD-DEM. Though the wall model assumes fully turbulent flow, it has been successfully applied to transitional flow regime with presence of particulate phase. The average heat transfer coefficient obtained from the simulations compare within 20% of experiments. In the scope of current study, the dominant mode of heat transfer was observed to be convection.

6. Conclusions and future scope

The main purpose of this work is to illustrate the effectiveness of OpenMP parallelism in computational fluid dynamics along with its superiority over MPI in dense fluid-particulate systems. Using the parallel performance optimized code; heat transfer studies of fluid-particulate system in rotary kiln and fluidized bed with tube heat exchanger are completed. The conclusions from the current work for each of the objectives stated in the introduction are presented below.

OpenMP parallelism for GenIDLEST

With the prudent use of first touch placement policy and appropriate thread affinity, the OpenMP API gives excellent scalability and speedup for fluid phase calculations.

Efficient parallelism of coupled CFD-DEM

Modified algorithm used for tightly coupled fluid-particulate calculations provided OpenMP threads considerable advantages over MPI processes which adhere to a single mode of parallelism.

Heat transfer in rotary kiln – effect of particle size distribution

The study of rotary kiln heat transfer in cascading bed mode yielded the dominant mechanism of heat transfer as convective heat transfer. On the other hand, poly dispersed kiln in slipping bed mode was dominated by conductive heat transfer at the beginning of kiln rotation. Convective and radiative heat transfer played a larger role as the bed temperature increased.

Heat transfer in fluidized bed with tube heat exchanger

Subgrid stress model along with LES wall function (WMLES) at the tube surface was used in the framework of CFD-DEM to analyze heat transfer in fluidized bed with tube heat exchanger. The dominant mode of heat transfer was observed to be convection.

Future scope

On heterogeneous computing architectures including systems that combine CPUs with graphical processing units (GPUs) and many-integrated cores (MIC), MPI has seen little

success as compared to OpenMP. These accelerators have shared memory layout and thus OpenMP seems to be more suitable to such systems for high performance computing. With the coprocessing capabilities it will be apt to apply the flexibility of OpenMP parallelism to offload a set of computations in a multi-physics application to achieve better load balancing and parallel performance. Using such optimized codes, computational studies in the area CFD-DEM can be further accelerated to better understand existing systems and study more complex systems such as industrial scale problems.

References

[1] F.E. Camelli, R. Lohner, J.C. Cebral, E.L. Mestreau, Timings of an unstructured-grid CFD code on common hardware platforms and compilers, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics Inc., Reno, NV, United states, 2008.

[2] R. Brown, I. Sharapov, High-scalability parallelization of a molecular modeling application: Performance and productivity comparison between OpenMP and MPI implementations, International Journal of Parallel Programming, 35 (2007) 441-458.

[3] W. Huang, D.K. Tafti, A parallel adaptive mesh refinement algorithm for solving nonlinear dynamical systems, International Journal of High Performance Computing Applications, 18 (2004) 171-181.

[4] W. Huang, D.K. Tafti, A Parallel Computing Framework for Dynamic Power Balancing in Adaptive Mesh Refinement Applications, in: D. Keyes, A. Ecer, J. Periaux, N. Satofuka, P. Fox (Eds.) Parallel Computational Fluid Dynamics' 99: Towards Teraflops, Optimization and Novel Formulations, Williamsburg, VA, 1999, pp. 249-256.

[5] G. Wang, D.K. Tafti, Performance enhancement on microprocessors with hierarchical memory systems for solving large sparse linear systems, International Journal of High Performance Computing Applications, 13 (1999) 63-79.

[6] G.R. Luecke, L. Wei-Hua, Scalability and performance of OpenMP and MPI on a 128-processor SGI Origin 2000, Concurrency and Computation Practice & Experience, 13 (2001) 905-928.

[7] M. Resch, S. Bjorn, L. Isabel, A comparison of OpenMP and MPI for the parallel CFD test case, in: Proc. of the First European Workshop on OpenMP, 1999.

[8] A.J. Wallcraft, SPMD OpenMP versus MPI for ocean models, Concurrency: Practice and Experience, 12 (2000) 1155-1164.

[9] G. Krawezik, F. Cappello, Performance comparison of MPI and three OpenMP programming styles on shared memory multiprocessors, in: Annual ACM Symposium on Parallel Algorithms and Architectures, Association for Computing Machinery, San Diego, SA, United states, 2003, pp. 118-127.

[10] D.J. Mavriplis, Parallel performance investigations of an unstructured mesh Navier-Stokes solver, International Journal of High Performance Computing Applications, 16 (2002) 395-407.

[11] M. Aftosmis, M. Berger, R. Biswas, M.J. Djomehri, R. Hood, H. Jin, C. Kiris, A detailed performance characterization of Columbia using aeronautics benchmarks and applications, in: 44th AIAA Aerospace Sciences Meeting 2006, January 9, 2006 - January 12, 2006, American Institute of Aeronautics and Astronautics Inc., Reno, NV, United states, 2006, pp. 1084-1100.

[12] S. Saini, D. Talcott, D. Jespersen, J. Djomehri, H. Jin, R. Biswas, Scientific application-based performance comparison of SGI Altix 4700, IBM POWER5+, and SGI ICE 8200 supercomputers, in: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, IEEE Press, Austin, Texas, 2008.

[13] P.A. Thomas Alrutz, Achim Basermann, Kim Feldhoff, Thomas Gerhold, Jörg Hunger, Jens Jägersküpper, Hans-Peter Kersken, Olaf Knobloch, Norbert Kroll, Olaf Krzikalla, Edmund Kügeler, Ralph Müller-Pfefferkorn, Mathias Puetz, Andreas Schreiber, Christian Simmendinger, Christian Voigt and Carsten Zscherp., HICFD - Highly efficient implementation of CFD codes for HPC-many-core architectures, in: PARS-Workshop, Parsberg, Bavaria, Germany, 2009.

[14] M. Norden, S. Holmgren, M. Thune, OpenMP versus MPI for PDE solvers based on regular sparse numerical operators, Future Generation Computer Systems, 22 (2006) 194-203.

[15] A. Marowka, L. Zhenying, B. Chapman, OpenMP-oriented applications for distributed shared memory architectures, Concurrency and Computation Practice & Experience, 16 (2004) 371-384.

[16] P. Satya-narayana, R. Avancha, P. Mucci, R. Pletcher, Parallelization and Optimization of a Large Eddy Simulation code using OpenMP for SGI Origin2000 Perforamance, in: J.P. D. Keyes, A. Ecer, N. Satofuka and P. Fox (Ed.) Parallel computational fluid dynamics towards teraflops, optimization, and novel formulations : proceedings of the Parallel CFD '99 Conference, Elsevier, Amsterdam, 2000, pp. 371-379.

[17] D. Hackenberg, R. Schone, W.E. Nagel, S. Pfluger, Optimizing OpenMP parallelized DGEMM calls on SGI Altix 3700, in: Lecture Notes in Computer Science, Springer Verlag, Lisbon, Portugal, 2006, pp. 145-154.

[18] J. Hoeflinger, P. Alavilli, T. Jackson, B. Kuhn, Producing scalable performance with OpenMP: Experiments with two CFD applications, Parallel Computing, 27 (2001) 391-413.

[19] X. Wu, V. Taylor, Using large page and processor binding to optimize the performance of OpenMP scientific applications on an IBM POWER5+ system, in: Proceedings 2009 International Conference on High Performance Computing, Networking and Communication Systems, ISRST, Worthington, OH, USA, 2009, pp. 65-71.

[20] B. Armstrong, K. Seon Wook, R. Eigenmann, Quantifying differences between OpenMP and MPI using a large-scale application suite, in: Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2000, pp. 482-493.

[21] G. Jost, H. Jin, D. an Mey, F.F. Hatay, Comparing the OpenMP, MPI, and Hybrid Programming Paradigm on an SMP Cluster, in, Technische Hochschule Aachen, Germany, 2003, pp. 14p.

[22] M.D. Jones, R. Yao, Parallel programming for OSEM reconstruction with MPI, OpenMP, and hybrid MPI-OpenMP, in: Nuclear Science Symposium Conference Record, IEEE, Piscataway, NJ, USA, 2004, pp. 3036-3042.

[23] E. Yilmaz, R.U. Payli, H.U. Akay, A. Ecer, Hybrid parallelism for CFD simulations: Combining MPI with openMP, in: Lecture Notes in Computational Science and Engineering, Springer Verlag, Antalya, Turkey, 2009, pp. 401-408.

[24] D.K. Tafti, GenIDLEST - A scalable parallel computational tool for simulating complex turbulent flows, in: ASME-IMECE, American Society of Mechanical Engineers, New York, NY 10016-5990, United States, 2001, pp. 347-356.

[25] D.K. Tafti, Time-accurate techniques for turbulent heat transfer analysis in complex geometries, Advances in Computational Fluid Dynamics and Heat Transfer, in: R. Amano, B. Sunden (Eds.) Computational Fluid Dynamics and Heat Transfer, WIT PRESS, Southampton, UK, 2011, pp. 217-264.

[26] L.W. Zhang, D.K. Tafti, F.M. Najjar, S. Balachandar, Computations of flow and heat transfer in parallel-plate fin heat exchangers on the CM-5: Effects of flow unsteadiness and three-dimensionality, International Journal of Heat and Mass Transfer, 40 (1997) 1325-1341.

[27] K. Nagendra, D.K. Tafti, A.K. Viswanathan, Modeling of soot deposition in wavy-fin exhaust gas recirculator coolers, International Journal of Heat and Mass Transfer, 54 (2011) 1671-1681.

[28] P. Gopalakrishnan, D.K. Tafti, Effect of Wing Flexibility on Lift and Thrust Production in Flapping Flight, American Institute of Aeronautics and Astronautics, Reston, VA, ETATS-UNIS, 2010.

[29] N.K.C. Selvarasu, D.K. Tafti, P.P. Vlachos, Hydrodynamic Effects of Compliance Mismatch in Stented Arteries, Journal of Biomechanical Engineering, 133 (2011) 021008-021011.

[30] V. Bui, O. Hernandez, B. Chapman, R. Kufrin, D.K. Tafti, P. Gopalkrishnan, Towards an implementation of the OpenMP collector API, in: Parallel Computing, Germany, 2007.

[31] V. Bui, B. Norris, K. Huck, Lois Curfman McInnes, L. Li, O. Hernandez, B. Chapman, A component infrastructure for performance and power modeling of parallel scientific applications, in: Proceedings of the 2008 compFrame/HPC-GECO workshop on Component based high performance, ACM, Karlsruhe, Germany, 2008.

[32] R. Nanjegowda, O. Hernandez, B. Chapman, H.H. Jin, Scalability Evaluation of Barrier Algorithms for OpenMP, Lecture notes in computer science., (2009) 42-52.

[33] M.A. Elyyan, D.K. Tafti, Flow and heat transfer charactersitcs of dimpled multilouvered fins, Jouranl of enhanced heat transfer, 16 (2009) 43-60.

[34] A. Rozati, D.K. Tafti, N.E. Blackwell, Thermal performance of pin fins at low Reynolds numbers in mini-micro-channels, in: Proceedings of the ASME/JSME Thermal Engineering Summer Heat Transfer Conference, American Society of Mechanical Engineers, New York, NY 10016-5990, United States, 2007, pp. 121-129.

[35] E.A. Sewall, D.K. Tafti, A.B. Graham, K.A. Thole, Experimental validation of large eddy simulations of flow and heat transfer in a stationary ribbed duct, International Journal of Heat and Fluid Flow, 27 (2006) 243-258.

[36] C. Liao, O. Hernandez, B. Chapman, W. Chen, W. Zheng, OpenUH: An optimizing, portable OpenMP compiler, in: Concurrency Computation Practice and Experience, John Wiley and Sons Ltd, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 2007, pp. 2317-2332.
[37] R. Kufrin, Measuring and Improving Application Performance with PerfSuite, Linux Journal., (2005) 62.
[38] S.S. Shende, A.D. Malony, The TAU parallel performance system, International Journal of High Performance Computing Applications, 20 (2006) 287-311.
[39] R. Chandra, Parallel programming in OpenMP, Morgan Kaufmann Publishers, San Francisco, CA, 2001.
[40] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics, 48 (1982) 387-411.
[41] P. Kang, N. Selvarasu, N. Ramakrishnan, C. Ribbens, D. Tafti, S. Varadarajan, Modular, Fine-Grained Adaptation of Parallel Programs, in: G. Allen, J. Nabrzyski, E. Seidel, G. van Albada, J. Dongarra, P. Sloot (Eds.) Computational Science – ICCS 2009, Springer Berlin / Heidelberg, 2009, pp. 269-279.
[42] J. Lau, J. Sampson, E. Perelman, G. Hamerly, B. Calder, The Strong correlation Between Code Signatures and Performance, in: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, IEEE Computer Society, 2005.
[43] L. Huang, H. Jin, L. Yi, B. Chapman, Enabling locality-aware computations in OpenMP, Scientific Programming, 18 (2010) 169-181.
[44] CAPS Enterprise, HMPP Directives, in.
[45] The Portland Group Inc., PGI Accelerator Programming Model, in.
[46] P.A. Cundall, O.D.L. Strack, Discrete numerical model for granular assemblies, Geotechnique, 29 (1979) 47-65.
[47] Y. Tsuji, T. Kawaguchi, T. Tanaka, Discrete Particle Simulation of 2-Dimensional Fluidized-Bed Powder Technology, 77 (1993) 79-87.

[48] T.B. Anderson, R. Jackson, Fluid Mechanical Description of Fluidized Beds. Equations of Motion, Industrial & Engineering Chemistry Fundamentals, 6 (1967) 527-539.
[49] B.P.B. Hoomans, J.A.M. Kuipers, W.J. Briels, W.P.M. Van Swaaij, Discrete particle simulation of bubble and slug formation in a two-dimensional gas-fluidised bed: a hard-sphere approach, Chemical Engineering Science, 51 (1996) 99-118.
[50] A. Amritkar, D. Tafti, R. Liu, R. Kufrin, B. Chapman, OpenMP parallelism for fluid and fluid-particulate systems, Parallel Computing, 38 (2012) 501-517.
[51] Y. Shigeto, M. Sakai, Parallel computing of discrete element method on multi-core processors, Particuology, 9 (2011) 398-405.
[52] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, P. Koumoutsakos, PPM – A highly efficient parallel particle–mesh library for the simulation of continuum systems, Journal of Computational Physics, 215 (2006) 566-588.
[53] J.H. Walther, I.F. Sbalzarini, Large-scale parallel discrete element simulations of granular flow, Engineering Computations: International Journal for Computer-Aided Engineering, 26 (2009) 688-697.
[54] A.K.C. Algirdas MAKNICKAS, R.B.C. Rimantas KAC˘IANAUSKAS, A. DŽIUGYS, Parallel DEM Simulations of Granular Media, Informatica, 17 (2006) 207-224.
[55] D.W. Washington, J.N. Meegoda, Micro-mechanical simulation of geotechnical problems using massively parallel computers, International Journal for Numerical and Analytical Methods in Geomechanics, 27 (2003) 1227-1234.
[56] D. Darmana, N.G. Deen, J.A.M. Kuipers, Parallelization of an Euler–Lagrange model using mixed domain decomposition and a mirror domain technique: Application to dispersed gas–liquid two-phase flow, Journal of Computational Physics, 220 (2006) 216-248.
[57] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, Journal of Computational Physics, 117 (1995) 1-19.
[58] D.K. Kafui, S. Johnson, C. Thornton, J.P.K. Seville, Parallelization of a Lagrangian–Eulerian DEM/CFD code for application to fluidized beds, Powder Technology, 207 (2011) 270-278.

[59] R. Kačianauskas, A. Maknickas, A. Kačeniauskas, D. Markauskas, R. Balevičius, Parallel discrete element simulation of poly-dispersed granular material, Advances in Engineering Software, 41 (2010) 52-63.
[60] T. Tsuji, K. Yabumoto, T. Tanaka, Spontaneous structures in three-dimensional bubbling gas-fluidized bed by parallel DEM–CFD coupling simulation, Powder Technology, 184 (2008) 132-140.
[61] S. Yakubov, B. Cankurt, M. Abdel-Maksoud, T. Rung, Hybrid MPI/OpenMP parallelization of an Euler–Lagrange approach to cavitation modelling, Computers & Fluids, (2012).
[62] H. Zhang, F.X. Trias Miquel, Y.Q. Tan, Y. Sheng, A. Oliva Llena, Parallelization of a DEM/CFD code for the numerical simulation of particle-laden turbulent flows, in: Parallel CFD 2011 : 23rd International Conference on Parallel Computational Fluid Dynamics, Barcelona, 2011, pp. 1-5.
[63] C. Kloss, C. Goniva, G. Aichinger, S. Pirker, Comprehensive DEM-DPM-CFD Simulations - Model Synthesis, Experiemtnal Validation and Scalability, in: Seventh International Conference on CFD in the Minerals and Process Industries, Melbourne, Australia, 2009.
[64] C. Goniva, C. Kloss, S. Pirker, Towards fast parallel CFD-DEM: An Open-Source Perspective, in: Open Source CFD International Conference 2009 Barcelona, Spain, 2009.
[65] X. Zhao, J. Wang, S. Zhang, Parallel CFD-DEM for Fluid-Particle Systems, ASME Heat Transfer/Fluids Engineering Summer Conference Proceedings, 2004 (2004) 575-584.
[66] C.R. Müller, S.A. Scott, D.J. Holland, B.C. Clarke, A.J. Sederman, J.S. Dennis, L.F. Gladden, Validation of a discrete element model using magnetic resonance measurements, Particuology, 7 (2009) 297-306.
[67] S. Deb, D. Tafti, A Novel Two Grid Formulation for Fluid-Particle Systems using the Discrete Element Method, Powder Technology, (2013).
[68] J. Sun, M.M. Chen, A theoretical analysis of heat transfer due to particle impact, International Journal of Heat and Mass Transfer, 31 (1988) 969-975.

[69] M. Germano, U. Piomelli, P. Moin, W.H. Cabot, A dynamic subgrid-scale eddy viscosity model, Physics of Fluids A (Fluid Dynamics), 3 (1991) 1760-1765.
[70] C.S. Campbell, C.E. Brennen, Computer simulation of granular shear flows, Journal of Fluid Mechanics, 151 (1985) 167-188.
[71] S. Ergun, Fluid flow through packed columns, Chemical Engineering Progress, 48 (1952) 89.
[72] C.Y. Wen, Y.H. Yu, Mechanics of fluidization, Chemical Engineering Progress Symposium Series, 62 (1966) 100-111.
[73] S.S. Zabrodsky, Hydrodynamics and Heat Transfer in Fluidized Beds, MIT Press, Cambridge, MA, 1966.
[74] H. Zhou, G. Flamant, D. Gauthier, DEM-LES simulation of coal combustion in a bubbling fluidized bed part II: Coal combustion at the particle level, Chemical Engineering Science, 59 (2004) 4205-4215.
[75] F.P. Incropera, Fundamentals of Heat and Mass Transfer, John Wiley & Sons, 2006.
[76] K.F. Malone, B.H. Xu, Particle-scale simulation of heat transfer in liquid-fluidised beds, Powder Technology, 184 (2008) 189-204.
[77] D.J. Gunn, Transfer of heat or mass to particles in fixed and fluidised beds, International Journal of Heat and Mass Transfer, 21 (1978) 467-476.
[78] V.D. Nguyen, C. Cogné, M. Guessasma, E. Bellenger, J. Fortin, Discrete modeling of granular flow with thermal transfer: Application to the discharge of silos, Applied Thermal Engineering, 29 (2009) 1846-1853.
[79] W.L. Vargas, J.J. McCarthy, Stress effects on the conductivity of particulate beds, Chemical Engineering Science, 57 (2002) 3119-3131.
[80] A. Amritkar, D. Tafti, S. Deb, Particle scale heat transfer analysis in rotary kiln, in: Proceedings of the ASME 2012 Summer Heat Transfer Conference, ASME, Puerto Rico, 2012.
[81] G.K. Batchelor, R.W. O'Brien, Thermal or Electrical Conduction Through a Granular Material, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 355 (1977) 313-333.
[82] G.J. Cheng, A.B. Yu, P. Zulli, Evaluation of effective thermal conductivity from the structure of a packed bed, Chemical Engineering Science, 54 (1999) 4199-4209.

[83] L.Y. Lu, G. Zhaolin, L. Kangbin, An inter-particle contact area and time restoration for softening treatment in thermal discrete element modeling, Europhysics Letters, 87 (2009) 44004 (44004 pp.).
[84] Z.Y. Zhou, A.B. Yu, P. Zulli, A new computational method for studying heat transfer in fluid bed reactors, Powder Technology, 197 (2010) 102-110.
[85] D.S. Boyalakuntla, Simulation of granular and gas-solid flows using discrete element method, in: Department of Mechanical Engineering, Carnegie Mellon University, Pittsburg, 2003.
[86] Q.F. Hou, Z.Y. Zhou, A.B. Yu, Computational study of heat transfer in a bubbling fluidized bed with a horizontal tube, AIChE Journal, 58 (2012) 1422-1434.
[87] F. Ben-Ammar, M. Kaviany, J.R. Barber, Heat transfer during impact, International Journal of Heat and Mass Transfer, 35 (1992) 1495-1506.
[88] K. Kuwagi, M. Arif, T. Takami, The effects of surface roughness on heat transfer between two contacting particles, in, IEEE, Piscataway, NJ, USA, 2007, pp. 5 pp.
[89] A.P. Collier, A.N. Hayhurst, J.L. Richardson, S.A. Scott, The heat transfer coefficient between a particle and a bed (packed or fluidised) of much larger particles, Chemical Engineering Science, 59 (2004) 4613-4620.
[90] B. Chaudhuri, F.J. Muzzio, M.S. Tomassone, Experimentally validated computations of heat transfer in granular materials in rotary calciners, Powder Technology, 198 (2010) 6-15.
[91] S.H. Tscheng, A.P. Watkinson, Convective heat transfer in a rotary kiln, The Canadian Journal of Chemical Engineering, 57 (1979) 433-443.
[92] F. Herz, Y. Sonavane, E. Specht, Analysis of Local Heat Transfer in Direct Fired Rotary Kilns, ASME Conference Proceedings, 2010 (2010) 175-182.
[93] X.Y. Liu, E. Specht, Temperature distribution within the moving bed of rotary kilns: Measurement and analysis, Chemical Engineering and Processing: Process Intensification, 49 (2010) 147-150.
[94] E. Alizadeh, O. Dubé, F. Bertrand, J. Chaouki, Characterization of Mixing and Size Segregation in a Rotating Drum by a Particle Tracking Method, AIChE Journal, 59 (2013) 1894-1905.

[95] H. Henein, J.K. Brimacombe, A.P. Watkinson, An experimental study of segregation in rotary kilns, Metallurgical Transactions B, 16 (1985) 763-774.
[96] N. Nityanand, B. Manley, H. Henein, An analysis of radial segregation for different sized spherical solids in rotary cylinders, Metallurgical Transactions B, 17 (1986) 247-257.
[97] A.A. Boateng, P.V. Barr, Modelling of particle mixing and segregation in the transverse plane of a rotary kiln, Chemical Engineering Science, 51 (1996) 4167-4181.
[98] S. Dhanjal, P. Barr, A. Watkinson, The rotary kiln: An investigation of bed heat transfer in the transverse plane, Metallurgical and Materials Transactions B, 35 (2004) 1059-1070.
[99] B. Chaudhuri, F.J. Muzzio, M.S. Tomassone, Modeling of heat transfer in granular flow in rotating vessels, Chemical Engineering Science, 61 (2006) 6348-6360.
[100] R. Schmidt, P.A. Nikrityuk, Numerical simulation of the transient temperature distribution inside moving particles, The Canadian Journal of Chemical Engineering, 90 (2012) 246-262.
[101] D. Shi, W.L. Vargas, J.J. McCarthy, Heat transfer in rotary kilns with interstitial gases, Chemical Engineering Science, 63 (2008) 4506-4516.
[102] W.L. Vargas, J.J. McCarthy, Heat conduction in granular materials, AIChE Journal, 47 (2001) 1052-1059.
[103] Y. Tsuji, T. Kawaguchi, T. Tanaka, Discrete particle simulation of 2-dimensional fluidized bed, Powder technology, 77 (1993) 79-87.
[104] F.P. Di Maio, A. Di Renzo, D. Trevisan, Comparison of heat transfer models in DEM-CFD simulations of fluidized beds with an immersed probe, Powder Technology, 193 (2009) 257-265.
[105] B. Andersson, and Leckner, B., Experimental Methods of Estimating Heat Transfer in Circulating Beds Boilers, Int. J. Heat and Mass Transfer, 35 (1992) 3353-3362.
[106] J. Botterill, Teoman, Y. and Yuregir, K., Factors Affecting Heat Transfer Between Gas-Fluidized Beds and Immersed Surfaces, Powder Technology, 39 (1984) 177-189.
[107] S.A. Kim, J. Kim, S. and Lee, D., Heat Transfer and Bubble Characteristics in a Fluidized Bed with Immersed Horizontal Tube Bundle, Int. J. of Heat and Mass Transfer, 46 (2003) 399-409.

[108] N.S. Grewal, Heat transfer between a horizontal tube and a gas-solid fluidized bed, International Journal of Heat and Mass Transfer, 23 (1980) 1505.
[109] J.S.M. Botterill, Fluid-Bed Heat Transfer, Academic Press New York, 1975.
[110] S.S. Zabrodsky, Hydrodynamics and Heat Transfer in Fluidized Beds, in, MIT Press, Cambridge, MA, 1966.
[111] N.S. Grewal, S.C. Saxena, Investigation of heat transfer from immersed tubes in a fluidized bed, in: Fourth National Heat Mass Transfer Conference, India, 1977, pp. 53-58.
[112] N.S. Grewal, S.C. Saxena, Effect of Surface Roughness on Heat Transfer from Horizontal Immersed Tubes in a Fluidized Bed, Journal of Heat Transfer, 101 (1979) 397-403.
[113] N.S. Grewal, S.C. Saxena, Heat-Transfer between a Horizontal Tube and a Gas-Solid Fluidized-Bed International Journal of Heat and Mass Transfer, 23 (1980) 1505-1519.
[114] A.P. Baskakov, Heat transfer to objects immersed in fluidized beds, Powder technology, 8 (1973) 273-282.
[115] H. Martin, Heat transfer between gas fluidized beds of solid particles and the surfaces of immersed heat exchanger elements, part II, Chemical engineering and processing, 18 (1984) 199.
[116] H.S. Mickley, D.F. Fairbanks, Mechanism of heat transfer to fluidized beds, AIChE J., 1 (1955) 374-384.
[117] A.P. Baskakov, The Mechanism of Heat Transfer between a Fluidized Bed and a surface., Int. Chem. Eng., 4 (1964).
[118] L.B. Koppel, R.D. Patel, J.T. Holmes, IV : Wall to Fluidized Bed Heat Transfer Coefficients. , AIChE J., 16 (1970).
[119] N.I. Gelperin, V.G. Einstein, Heat Transfer in Fluidized Beds, in: Fluidization, Academic, 1971, pp. 471-535.
[120] N.V. Antonishin, Hyperbolic equation of heat conduction for dispersed systems, Journal of engineering physics (New York, N.Y.), 26 (1974) 353-356.
[121] J. Kubie, J. Broughton, A model of heat transfer in gas fluidized beds, International Journal of Heat and Mass Transfer, 18 (1975) 289-299.

[122] T.F. Ozkaynak, J.C. Chen, Emulsion phase residence time and its use in heat transfer models in fluidized beds, AIChE J., 26 (1980) 544-550.
[123] J.S.M. Botterill, M. Desai, Limiting factors in gas-fluidized bed heat transfer, Powder Technology, 6 (1972) 231-238.
[124] A.P. Baskakov, V.M. Suprun, Determination of the Convective Component Of The Heat-Transfer Coefficient to a Gas in a Fluidized Bed, 12 (1972) 324-326.
[125] A.O.O. Denloye, J.S.M. Botterill, Bed to surface heat transfer in a fluidized bed of large particles, Powder technology, 19 (1978) 197-203.
[126] G.S. Canada, M.H. McLaughlin, Large Particle Fluidization and Heat Transfer at High Pressures 74 (1978) 27-37.
[127] L. Adams, J.R. Welty, A gas convection model of heat transfer in large particle fluidized beds, AIChE J., 25 (1979) 395-405.
[128] L.R. Glicksman, N.A. Decker, Design Relationships for Predicting Heat Transfer to Tube Bundles in Fluidized Bed Combustors, in: Proc. 6th Int. Fluidized Bed Combustion Conf., 111, U.S. Dept. of Energy, Washington, D.C., 1980, pp. 1152.
[129] N.M. Catipovic, T.J. Fitzgerald, A.H. George, J.R. Welty, Experimental validation of the Adams-Wwlty model for heat transfer in large-particle fluidized beds, A.I.Ch.E.J., (1982).
[130] N.M. Catipovic, G.N. Jovanovic, T.J. Fitzgerald, O. Levenspiel, Model for Heat Transfer to Horizontal Tubes Immersed in a Fluidized Bed of Large Particles Journal of Technical Writing and Communication, (1980) 225-234.
[131] S.S. Zabrodsky, Y.G. Epanov, D.M. Galershtein, S.C. Saxena, A.K. Kolar, Heat transfer in a large-particle fluidized bed with immersed in-line and staggered bundles of horizontal smooth tubes, International Journal of Heat and Mass Transfer, 24 (1981) 571-579.
[132] S.C. Saxena, V.L. Ganzha, Heat Transfer to Immersed Surfaces in Gas-Fluidized Beds of Large Particles and Powder Characterization Powder technology, 39 (1984) 199-208.
[133] C. van Heerden, A.P.P. Nobel, D.W. van krevelen, Mechanism of heat transfer in fluidized beds., Ind. Eng. Chem. Res., 6 (1953) 1237-1242.

[134] W.M. Dow, M. Jakob, Heat transfer between a vertical tube and a fluidised air solid mixture, Chemical Engineering Progress, 17 (1951) 637-648.
[135] R.D. Toomey, H.F. Johnstone, Heat transfer between beds of fluidized solids and the walls of the container, Chem. Eng. Progr. Symposium Ser. No. 5, 49 (1953) 51.
[136] O. Levenspiel, J.S. Walton, Bed-wall heat transfer in fluidised systems, Chemical Engineering Progress Symposium Series, 50 (1954) 1-13.
[137] C.Y. Wen, M. Leva, Fluidized-bed heat transfer: A generalized dense-phase correlation, AIChE J., 2 (1956) 482-488.
[138] H.A. Vreedenberg, Heat transfer between a fluidized bed and a horizontal tube, Chemical Engineering Science, 9 (1958) 52-60.
[139] B.R. Andeen, L.R. Glicksman, Heat Transfer To Horizontal Tubes In Shallow Fluidized Beds, in: ASME-AIChE Heat Transfer Conference, 1976.
[140] J.C. Petrie, W.A. Freeby, J.A. Buckham, In-bed heat exchangers, Chem. Engng. Prog. Symp. Ser. , 64 (1968) 45-51
[141] V.G. Ainshtein, An Investigation of Heat Transfer Process Between Fluidized Beds and Single Tubes Submerged in the Bed, in, 1966, pp. 270.
[142] N.I. Gelperin, V.Y. Kruglikov, V.G. Ainshtein, In Heat transfer between a fluidized bed and a surface, Int. Chem. Engng, 6 (1966) 67-73
[143] W.E. Genetti, R.A. Schmall, E.S. Grimmett, The effect of tube orientation on heat transfer with bare and finned tubes in a fluidized bed, Chem. Engng Prog. Symp., 67 (1971) 90-96
[144] Y. Kurochkin, Heat transfer between tubes with different cross sections and two-phase flow of granulated materials, Journal of Engineering Physics., 6 (1966) 759-763.
[145] P. Zehner, E.U. Schlunder, On the effective heat conductivity in packed beds with flowing fluid at medium and high temperatures., Chem. Ing. Tech., 42 (1970) 933-941.
[146] J.A.M. Kuipers, W.P. Tammes, W.P.M. van Swaaji, Experimental and Theoretical Porosity Profiles in a Two-Dimensional Gas-Fluidized Bed with a Central Jet., Powder technology, 71 (1992) 87-99.
[147] M.L. Hunt, Discrete element simulations for granular material flows: Effective thermal conductivity and self-diffusivity., International Journal of Heat and Mass Transfer, 40 (1997) 3059-3068.

[148] A. Schmidt, U. Renz, Numerical prediction of heat transfer between a bubbling fluidized bed and an immersed tube bundle, Heat and Mass Transfer, 41 (2005) 257-270.
[149] R. Yusuf, B. Halvorsen, M.C. Melaaen, Eulerian-Eulerian simulation of heat transfer between a gas-solid fluidized bed and an immersed tube-bank with horizontal tubes, Chemical Engineering Science, 66 (2011) 1550-1564.
[150] B. Legawiec, D. Ziólkowski, Structure, voidage and effective thermal conductivity of solids within near-wall region of beds packed with spherical pellets in tubes, Chemical Engineering Science, 49 (1994) 2513-2520.
[151] L.M. Armstrong, S. Gu, K.H. Luo, The influence of multiple tubes on the tube-to-bed heat transfer in a fluidised bed, International Journal of Multiphase Flow, 36 (2010) 916-929.
[152] A. Schmidt, U. Renz, Numerical prediction of heat transfer in fluidized beds by a kinetic theory of granular flows, International Journal of Thermal Sciences, 39 (2000) 871-885.
[153] L.M. Armstrong, S. Gu, K.H. Luo, Study of wall-to-bed heat transfer in a bubbling fluidised bed using the kinetic theory of granular flow, International Journal of Heat and Mass Transfer, 53 (2010) 4949-4959.
[154] D.J. Patil, J. Smit, M. van Sint Annaland, J.A.M. Kuipers, Wall-to-bed heat transfer in gas–solid bubbling fluidized beds, AIChE J., 52 (2006) 58-74.
[155] C. Shen, J.X. Guo, L. Li, J.F. Sun, CFD Studies on the Heat Transfer Characteristics of a Horizontal Single-Tube in Fluidized Bed Heat Exchanger, Advanced Materials Research, 374 (2012) 183-186.
[156] N. Dong, L. Armstrong, S. Gu, K. Luo, Effect of tube shape on the hydrodynamics and tube-to-bed heat transfer in fluidized beds, Applied Thermal Engineering, (2012).
[157] Y. Zhao, M. Jiang, Y. Liu, J. Zheng, Particle-scale simulation of the flow and heat transfer behaviors in fluidized bed with immersed tube, AIChE J., 55 (2009) 3109-3124.
[158] Q.F. Hou, Z.Y. Zhou, A.B. Yu, Investigation of Heat Transfer in Bubbling Fluidization with an Immersed Tube, AIP Conference Proceedings, 1207 (2010) 355-357.
[159] Y.S. Wong, J.P.K. Seville, Single-particle motion and heat transfer in fluidized beds, AIChE Journal, 52 (2006) 4099-4109.

[160] S. Patil, D. Tafti, Wall modeled large eddy simulations of complex high Reynolds number flows with synthetic inlet turbulence, International Journal of Heat and Fluid Flow, 33 (2012) 9-21.
[161] S. Patil, Large Eddy Simulations of high Reynolds number Complex Flows with Synthetic Inlet Turbulence, in: Mechanical Engineering, Virginia Tech, Blacksburg, VA, 2011.
[162] Z.Y. Zhou, A.B. Yu, P. Zulli, Particle scale study of heat transfer in packed and bubbling fluidized beds, AIChE Journal, 55 (2009) 868-884.
[163] J. Li, D.J. Mason, A computational investigation of transient heat transfer in pneumatic transport of granular particles, Powder Technology, 112 (2000) 273-282.
[164] C.S. Daw, C.E. Finney, M. Vasudevan, N.A. van Goor, K. Nguyen, D.D. Bruns, E.J. Kostelich, C. Grebogi, E. Ott, J.A. Yorke, Self-organization and chaos in a fluidized bed, Physical review letters, 75 (1995) 2308-2311.
[165] A.I. Karamavruç, N.N. Clark, A fractal approach for interpretation of local instantaneous temperature signals around a horizontal heat transfer tube in a bubbling fluidized bed, Powder technology, 90 (1997) 235-244.
[166] N. Masoumifard, N. Mostoufi, A.-A. Hamidi, R. Sotudeh-Gharebagh, Investigation of heat transfer between a horizontal tube and gas–solid fluidized bed, International Journal of Heat and Fluid Flow, 29 (2008) 1504-1511.
[167] W.C. Yang, J. Hoffman, Exploratory Design Study on Reactor Configurations for Carbon Dioxide Capture from Conventional Power Plants Employing Regenerable Solid Sorbents, Ind. Eng. Chem. Res., 48 (2009) 341-351.
[168] C.Y. Wen, Y.H. Yu, Mechanics of fluidization, Chem. Eng. Prog.Symp.Ser., 62 (1966) 100-111.
[169] V.L. Ganzha, S.N. Upadhyay, S.C. Saxena, A mechanistic theory for heat transfer between fluidized beds of large particles and immersed surfaces, International Journal of Heat and Mass Transfer, 25 (1982) 1531-1540.

Appendices

Appendix A: Heat transfer coefficient calculations based on numerical correlations

To illustrate the large variations between the numerical correlations for heat transfer coefficient calculations as discussed in Chapter 5, polypropylene particles [167] in fluidized bed tube heat exchanger are considered. The mechanical and thermal properties of polypropylene are given in Table A.1. For illustration purposes, the fluid has been assumed to be air at 100 °C. Five particle diameters are considered ranging from 100 micron to 2 mm as given in Table A.2. In calculating the heat transfer coefficients, it is assumed that the particles are spherical, the heat exchanger tube in the fluidized bed has a diameter of 0.02 m, and the superficial mass fluidization velocity, G, is assumed to be twice the minimum fluidization velocity. The void fraction is assumed to be that of a packed bed. Table A.2 lists the Archimedes number and the Reynolds number based on the fluidization velocity and particle diameter, which are used in calculating the heat transfer coefficients. The minimum fluidization velocity is calculated based on the [168] correlation, $Re_{mf} = 33.7[(1 + 3.59 \times 10^{-5} Ar)^{0.5} - 1]$, which is valid in the range $0.01 < \mathrm{Re_{mf}} < 1000$. Approximately 2 times the minimum fluidization velocity was used as superficial velocity for each particle diameter considered. The Archimedes number, is representative of the density difference driven flow set up in the fluidized bed and larger its value, the stronger the fluidization in the bed. According to [132], for 3<Ar<21700, the interstitial gas is in the laminar flow regime and HTC decreases with an increase in particle diameter, whereas for Ar>1.6x10^6, the gas flow is fully turbulent and the HTC is strongly dependent on gas convection, increasing with an increase in particle diameter. Between the two extremes, the flow is transitional during which both solid phase conduction and gas convection to the heat transfer surface are important.

Table A.1 Properties of Polypropylene particles

Density (Kg/m^3)	Specific heat Cp (J/Kg-K)	Thermal conductivity k (W/m-K)	Modulus of Elasticity E (Pa)	Poisson's ratio	Thermal diffusivity α (m^2/s)
882	1926	0.138	2.50E+09	0.25	8.12E-08

Table A.2 Non-dimensional parameters relevant to fluidized bed with tube heat exchanger for polypropylene particles

d_p (mm)	(Geldart class.) & Ar	Re_p	U_{mf} (m/s)
0.10	(A) 17	0.03	0.003
0.25	(A-B) 269	0.32	0.015
0.50	(B) 2149	2.6	0.059
1.00	(B)17188	18	0.211
2.00	(D) 137504	97	0.558

Figure A.1 plots the predicted heat transfer coefficients for the five particle sizes in Table A.2 using different correlations available in the literature. The dark symbols in the figure are from heat transfer correlations for large particles in which gas convection starts dominating the heat transfer [128, 144, 169], whereas most of the other correlations are for particle diameters in the range 200 micron to 1 mm. The predicted heat transfer coefficients exhibit a large scatter with variability ranging over 100% between the minimum and maximum values. The variability is largest for the small particles and this could be because most of the correlations are built on experiments which have been conducted or particle sizes between 200 micron and 1 mm.

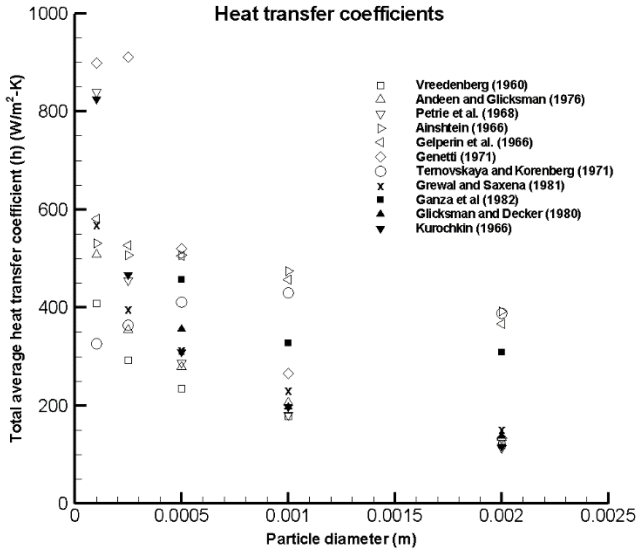

Figure A.1 Average heat transfer coefficients for horizontal tube in a fluidized bed of polypropylene particles using numerical correlations.

Appendix B: Octave code for power spectrum of a signal

Below is the Octave code used for analyzing the input velocity signal from the probe in the fluidized bed flow field.

```
%-----------------------------------------------------------------------%
clear all
close all
clc
u1=load('data.txt'); % load velocity signal
N=length(u1); % number of data points
l_ref=0.06; % reference length
u_ref=0.77; % reference velocity
dtime=2.56e-4; % time step

n=0:N-1;
T=N*l_ref/u_ref*dtime; % Total time of simulation
freq=[0:N/2-1]/T; % sampling frequency
t=[1:N].*(T/N); % time axis
t=t';
plot(t,u1); % original signal
xlabel('Time(s)');
ylabel('Velocity(m/s)');

p=abs(fft(u1))/(N/2);
p=p(1:N/2).^2;

figure();
loglog(freq,p); % power spectrum
xlabel('Frequency(Hz)');
ylabel('Energy');
%-----------------------------------------------------------------------%
```