The Cellular Automata Paradigm for the Parallel Solution of Heat Transfer Problems

Bruce B. Lowekamp Layne T. Watson[†]
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106

Mark S. Cramer
Department of Engineering Science and Mechanics
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0219

Running Title: Parallel Cellular Automata for Heat Transfer Problems

The authors acknowledge support under NSF grant CTS-8913198, AFOSR grant F49620-92-J-0236, and DOE grant DE-FG05-88ER25068. Additionally, they would like to express their gratitude to the Mathematical Sciences Section at Oak Ridge National Laboratory and the Advanced Computing Research Facility at Argonne National Laboratory for their generosity in allowing the use of their iPSC/860 computers.

[†]Author to whom correspondence should be addressed

Abstract

This paper describes the numerical solution of heat transfer problems using cellular automata. While traditional methods offer high performance on uniprocessor machines, their performance is limited on distributed memory multiprocessors by communication bottlenecks caused by the interdependence of the equations. Using a cellular automata formulation, these bottlenecks can be avoided, and performance greater than that obtained by parallelizing traditional algorithms can be achieved. This paper gives an overview of the cellular automata paradigm and specific examples of solutions to a hyperbolic and a parabolic problem. The accuracy of the method is verified by comparisons of the results with analytical solutions and with results produced by other techniques.

Nomenclature

E	the set of cells on the border of P
e	$\|E\|$
k	$e/3$
n	number of cells used in the simulation
P	the set of cells assigned to a processor
p	$\|P\|$
Q	nondimensional heat flux
t	elapsed time
T	temperature
x	cartesian coordinate
y	cartesian coordinate
α	ratio of wave period to relaxation time
β	temperature dependence of thermal conductivity
Δx	mesh spacing
Δt	time step
ϵ	nondimensional wave amplitude
λ	integration variable
ν	viscosity coefficient
Θ	nondimensional temperature

Subscripts

j	$x/\Delta x$
L	data from the left neighbor
R	data from the right neighbor
t	differential in t
x	differential in x

Superscripts

k	$t/\Delta t$

1 INTRODUCTION

Heat transfer problems arise frequently in many scientific and engineering disciplines. Classically, the equations involved are parabolic second-order differential equations. However, for extremely high frequency sources, short time periods, or for temperatures approaching 0 K, the parabolic Fourier approximation breaks down and hyperbolic equations must be used to model the finite transport speeds observed in real systems [1, 2, 3, 4, 5]. This paper presents the cellular automata paradigm for modeling both these systems. It should be noted, however, that the cellular automata paradigm is by no means limited to heat transfer problems and can, in fact, be applied to a wide variety of partial differential equations, including elliptic systems [6].

Several methods are presently in common use for solving heat transfer problems. Three popular techniques are finite differences, finite elements, and collocation. Each of these has varying tradeoffs in terms of implementation time, execution time, memory requirements, stability, and accuracy. Some variants also vectorize well. However, a major disadvantage of these techniques is that they do not work well on massively parallel distributed memory computers [6]. For a method to work well in such an environment, it must be possible to partition the steps in the method into small computational units which require little intercommunication. Since these techniques generally involve solving systems of implicit equations, it is impossible to partition the computational effort in such a way that each portion meets the requirement for low intercommunication; therefore, these techniques perform poorly on such machines.

This difficulty is the primary reason for examining the cellular automata paradigm as a solution technique for heat transfer problems. By their very definition, automaton cells are separate entities with clearly defined rules governing how they operate and interact. Because the cells function independently and synchronously, an algorithm using cellular automata should perform well in a massively parallel environment.

The goal of developing the cellular automata paradigm is not to produce a method which is faster or more accurate than traditional techniques on a serial machine. Because the equations used for the cellular automata must be explicit, it is impossible to obtain the *serial* speed of other techniques while achieving sufficient accuracy. However, cellular automata perform significantly faster on massively parallel distributed memory computers. This is because cellular automata can utilize almost 100 percent of the processors' power for any number of processors, while traditional algorithms can only make use of a significantly lower percentage of the processors' power. This makes cellular automata preferable to many other techniques for parallel machines.

Section 2 will define cellular automata and give a simple example of how they can be applied to a problem. Section 3 discusses the heat transfer problems which were considered in the development of this technique. These problems were chosen for their utility in developing and demonstrating the uses of cellular automata; they are certainly not illustrative of the full range of its capabilities. Section 4 describes the results obtained using cellular automata on these problems, including results obtained using an Intel iPSC/860 parallel supercomputer.

2 CELLULAR AUTOMATA

The cellular automata paradigm was first suggested in 1946 by Wiener [7]. Unfortunately, there was no practical application for it at the time, given the current state of computing technology. Presently, the cellular automata paradigm is used to solve a wide variety of problems, from simulation of nerve depolarization [8] to graphic imaging [9]. It is used because it is simple to implement but at the same time can represent complex systems. Cellular automata can be used to represent any system which can be viewed as a set of small, identically functioning units. A large number of these units, following simple rules, can model a very complex system. A general overview of cellular automata is presented in [9].

The easiest way to explain cellular automata is by example. Consider a metal sheet divided into a two-dimensional mesh. Each square on this mesh is referred to as a cell. Each cell has a set of internal properties, such as temperature, and a set of rules which define how it interacts with its neighbors. Typically, these neighbors are the bordering cells, although the

0	1	0
1	-4	1
0	1	0

Figure 1: The mask applied to the grid to calculate ΔT in the steady state two-dimensional diffusion problem.

neighborhood is sometimes extended to include all cells within a stencil of some shape, commonly referred to as a mask.

One practical application of cellular automata is heat diffusion. Consider the problem of calculating the equilibrium temperature, $T(x, y)$, of the metal sheet with constant temperature borders. The equilibrium temperature satisfies the Laplace equation,

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \tag{1}$$

which is the mathematical model of steady state two-dimensional diffusion.

To compute the equilibrium temperatures, we begin with an arbitrary set of initial temperatures for each cell and calculate the change in temperature per iteration, ΔT, of each cell using a weighted mask, as shown in Figure 1. Equilibrium is reached when $\Delta T \approx 0$ for each cell.

The rules for the interior cells are:

1. Get the temperatures T_N, T_S, T_E, T_W from the north, south, east, and west neighboring cells.

2. Calculate the new temperature using the formula

$$T_{\text{new}} = T_{\text{old}} + \frac{1}{4}(T_N + T_S + T_E + T_W - 4T_{\text{old}}). \tag{2}$$

3. Send T_{new} to the north, south, east, and west neighbors.

This process is continued with each cell acting independent of, but synchronous with, the rest of the cells, until ΔT becomes sufficiently close to zero for all cells.

The mask is determined by finding an explicit discrete formulation for the diffusion problem. An ab initio technique is generally preferred, as it allows us to fully exploit the capabilities of cellular automata. Using random walks, a discrete formulation of the diffusion problem is derived in [10, 11]. The temperature is represented as $T(j\Delta x, k) = T_j^k$, where Δx represents the distance between cells and k is used to index the iterations. The equation is

$$T_j^{k+1} = \frac{1}{2}(T_{j+1}^k + T_{j-1}^k). \tag{3}$$

Rearranging this equation and combining the equations for both dimensions, we obtain the mask used for this system. Numerically, this technique is identical to Jacobi relaxation, but the cellular automata paradigm makes implementation in parallel straightforward.

3 NUMERICAL SOLUTION

The cellular automata paradigm is demonstrated on two problems: the parabolic heat equation with nondimensional space and time,

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2}, \tag{4}$$

and the nonlinear telegraph equation [2, 12]

$$\begin{aligned} \frac{\partial \Theta}{\partial t} + \frac{\partial Q}{\partial x} &= 0, \\ \frac{\partial Q}{\partial t} + (1 + \epsilon\beta\Theta)\frac{\partial \Theta}{\partial x} &= -\alpha Q, \end{aligned} \tag{5}$$

where Θ is the nondimensional temperature, Q is the nondimensional heat flux, x represents nondimensional space, and t is nondimensional time. The quantity $\alpha > 0$ is a constant measuring the relative sizes of the wave period and the thermal relaxation time. A temperature dependent thermal conductivity has been employed. This temperature dependence is taken to be linear, with the material constant β being a measure of the strength of the dependence. The positive constant ϵ is a measure of the amplitude of the signal. $\partial\Theta/\partial t + \partial Q/\partial x = 0$ is recognized as the energy equation, whereas the second equation is a nonlinear scaled form of the Maxwell Cattaneo relaxation law.

The cellular automata rules for the heat equation can be derived in the same manner as the solution for the steady state diffusion problem, Eq. (3). The rule for the heat equation is

$$T_j^{k+1} = T_j^k + \Delta t \frac{T_{j+1}^k - 2T_j^k + T_{j-1}^k}{\Delta x^2}. \tag{6}$$

The formula for the telegraph equation is more complicated to derive. Using the definition of a differential, the partial differential equations are converted into difference equations:

$$\begin{aligned} \frac{\Delta_t \Theta}{\Delta t} &\approx -\frac{\Delta_x Q}{\Delta x}, \\ \frac{\Delta_t Q}{\Delta t} &\approx -(1+\epsilon\beta\Theta)\frac{\Delta_x \Theta}{\Delta x} - \alpha Q, \end{aligned} \tag{7}$$

where Δ_t represents a time difference and Δ_x represents a space difference. By rearranging these equations and providing appropriate definitions for the differences involved, rules suitable for use in cellular automata can be derived.

Using the following definitions: $Q_j^k = Q(j\Delta x, k\Delta t)$, $\Theta_j^k = \Theta(j\Delta x, k\Delta t)$, $\Delta x > 0$, and $\Delta t > 0$, the most obvious approach is to use first order forward differences. However, these definitions result in inherently unstable calculations [13, 16]. Guided by the classical theory of differential equations, the leapfrog scheme is used instead:

$$\begin{aligned} \Delta_t Q &= Q_j^{k+1} - Q_j^{k-1}, \\ \Delta_t \Theta &= \Theta_j^{k+1} - \Theta_j^{k-1}, \\ \Delta_x Q &= Q_{j+1}^k - Q_{j-1}^k, \\ \Delta_x \Theta &= \Theta_{j+1}^k - \Theta_{j-1}^k. \end{aligned} \tag{8}$$

This technique is described in [16, 13].

Because the telegraph equation employed here is nonlinear and hyperbolic, the cellular automata solution will be unstable due to discontinuities which appear in the solution. To compensate for this, an artificial viscosity is added to Eq. (7) [13], producing the rules for the cells:

$$
\begin{aligned}
Q_j^{k+1} &= Q_j^{k-1} - 2\alpha Q_j^k \Delta t - (1 + \epsilon\beta\Theta_j^k)(\Theta_{j+1}^k - \Theta_{j-1}^k)\frac{\Delta t}{\Delta x} + \\
&\quad \nu\Delta x \frac{(Q_{j-1}^k - 2Q_j^k + Q_{j+1}^k)}{\Delta x^2}, \\
\Theta_j^{k+1} &= \Theta_j^{k-1} - (Q_{j+1}^k - Q_{j-1}^k)\frac{\Delta t}{\Delta x} + \nu\Delta x \frac{(\Theta_{j-1}^k - 2\Theta_j^k + \Theta_{j+1}^k)}{\Delta x^2},
\end{aligned}
\tag{9}
$$

where ν is an arbitrary small positive constant. With this addition, the solutions can be calculated accurately and stably using cellular automata.

The development of the cellular automata formulation of the nonlinear telegraph equation indicates the care which is necessary during the development of the rules for cellular automata. Poorly designed equations can easily destroy the solution due to numerical difficulties, but by giving careful attention to the numerical aspects of the solution, it is possible to perform the calculations stably and accurately. Information needed to handle the calculations correctly is given in many standard texts [13, 16, 17].

The rules for the cells are similar to the rules for the diffusion problem:

1. Get the temperature and heat flux Θ_L, Θ_R, Q_L, and Q_R from the left and right neighboring cells.

2. Calculate new values for Θ and Q from Eq. (9).

3. Send Θ and Q to the left and right neighbors.

The parallel implementation of this code is simple. To insure the highest level of performance possible, the processors are organized in a ring. Each processor receives the same number of cells. Any interconnection network in which there is an embedded ring works well for this application—this includes meshes, hypercubes, and toruses.

The boundaries between each processor's cells are the only places where difficulties arise. The communication involved in trading information between processors must be handled with care to avoid creating a bottleneck.

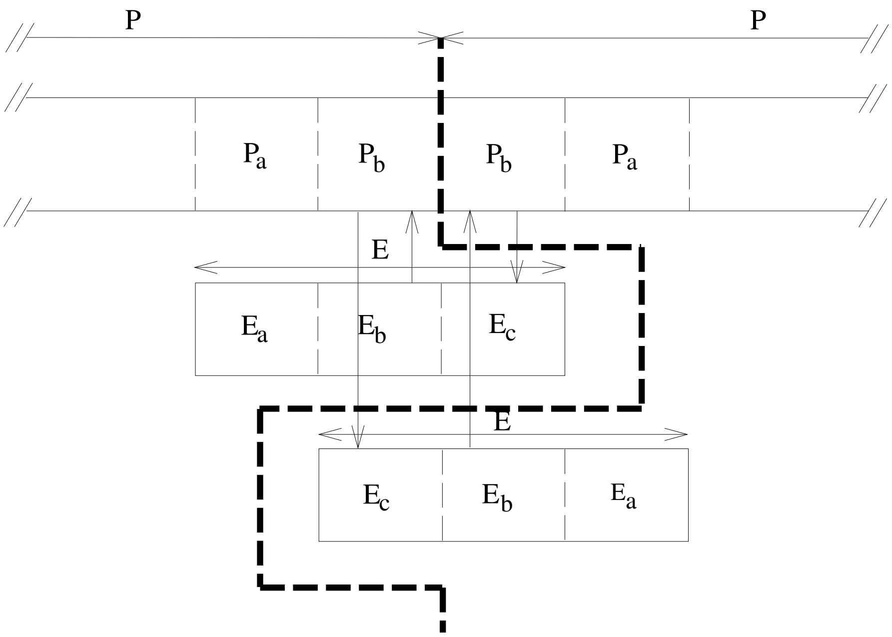

Figure 2: Organization of data at processor boundaries. The heavy dashed line represents the boundary between two processors. Regions on the figure above or below each other each represent the same area. The arrows indicate how the data is transferred between processors.

The communication overhead can be limited by overlapping the cells on each end of the processors' cells. For simplicity, only one end will be described here, although this technique is applied to both ends of all internal processors.

The set of p cells allocated to each processor is referred to as P. Each processor also reserves an additional area, referred to as E, for the cells on each side of the border between its own and the neighboring processor's cells. This is illustrated in Figure 2. To facilitate the description of this technique, portions of P and E are given separate labels. P_a, P_b, E_a, E_b, and E_c are all k cells long, $p \gg k$. Upon starting the simulation, the processors copy the data in P_b into E_b. The steps which the processors use to organize the computation are as follows:

1. Save the data in P_a into E_a.

2. Begin transmitting the data in P_b to the neighboring processor's E_c. This transmission continues in the background while the computation for the next step is being performed.

3. Perform k iterations of the cellular automata rules on P. Because the data for the neighboring processor's P_b is not yet available, after k steps the data stored in P_b is incorrect.

4. Receive the data for E_c, which arrived during the previous step.

5. Perform k iterations of the cellular automata rules on E.

6. Copy the data in E_b to P_b.

Note that the interprocessor communication in Step 2 completes while Step 3 is being performed, effectively hiding the communication cost. The intraprocessor data transfer between buffers in Steps 1 and 6 cannot be hidden, but with appropriate choice of k, it represents a negligible portion of the execution time.

4 RESULTS

4.1 Numerical Results

The cellular automata algorithms were tested on several different problems. Figure 3 gives results for the heat equation solved with the cellular automata rule given in Eq. (6) with $T(t, x_0) = 1$ and $T(t, x_\infty) = 0$ compared to the analytical result, which is the error function [14]:

$$T(x,t) = \frac{1}{\sqrt{\pi}} \int_{x/\sqrt{t}}^{\infty} e^{-\lambda^2/4} d\lambda. \tag{10}$$

The results obtained using the cellular automata and the analytical solution are virtually indistinguishable.

Figure 4 presents the cellular automata and analytical solutions for an initial value problem using the linear form ($\beta = 0$) of the telegraph equation, Eq. (9), with $\alpha = 0.1$. The initial value for this simulation was one period

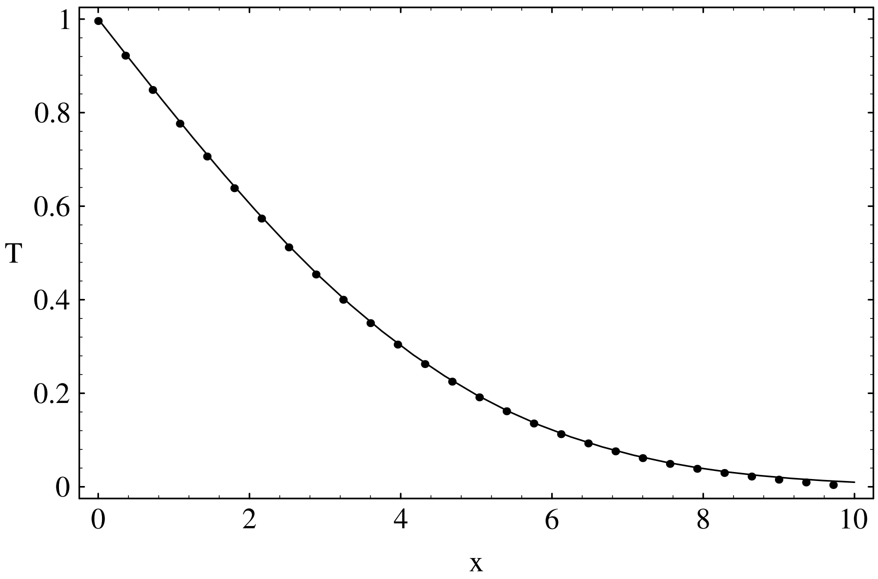

Figure 3: Cellular automata and analytical solutions for the heat equation with boundary conditions.

of a sine wave of wavelength 1 and amplitude 1. Since this form of the telegraph equation propagates the wave forward with unit speed while it decays exponentially, the analytical solution is calculated using the formula

$$\begin{aligned} \Theta_j^k &= \Theta_{j-k}^0 e^{k\Delta t\alpha/2}, \\ Q_j^k &= Q_{j-k}^0 e^{k\Delta t\alpha/2}, \end{aligned} \tag{11}$$

with the speed being taken into account by the $_{j-k}$ term. Agreement between the cellular automata result and the analytical result is very good.

Figure 5 shows the cellular automata solution to Eq. (9) with $\epsilon\beta = 0.1$ and $\alpha = 0.1$. For this problem, the left boundary conditions were $Q = 1$ and $\Theta = 2 - e^{k\Delta t\alpha/2}$. The right boundary conditions were $\partial Q/\partial x = 0$ and $\partial\Theta/\partial x = 0$. As pointed out in [12], the front of the disturbance is a shock having a speed slightly larger than that of the linear ($\beta = 0$) theory. Between the shock and the left boundary, the heat flux and the temperature increase in a nearly linear fashion for the time scales considered. In this figure, we compare the cellular automata results with results produced using

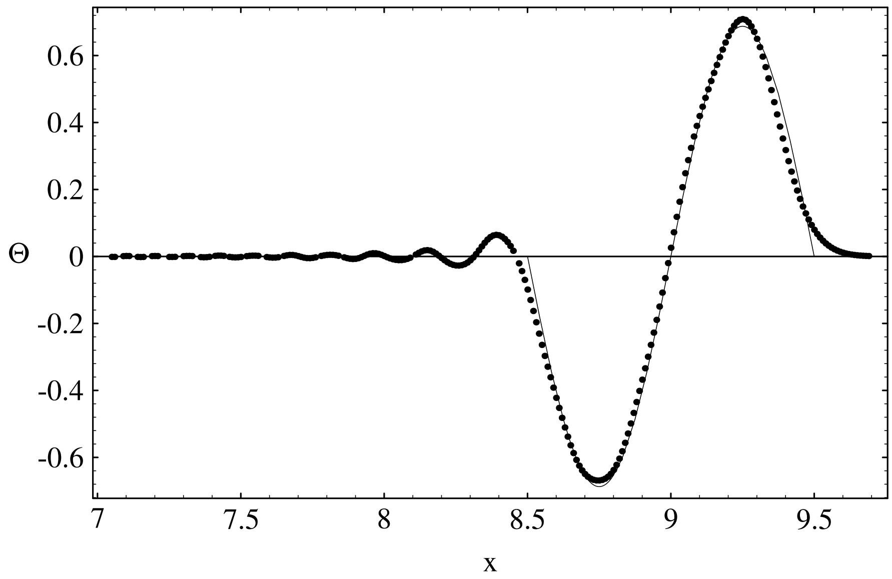

Figure 4: Cellular automata and analytical solutions for the linear telegraph equation initial value problem.

spline collocation. Collocation is a technique for solving differential equations with extremely high accuracy, but at a relatively high cost. It is documented in [15]. The oscillations observed at the shock occur because of the numerical difficulties inherent in modeling waves with discontinuities. These oscillations are produced by nearly all methods when attempting to solve such problems; see, for example, the solutions presented in [2].

Figure 6 shows the solution to the nonlinear telegraph equation problem, $\epsilon\beta = 0.1$ and $\alpha = 0.1$, along with a solution generated using spline collocation. The initial values are the same as those used to calculate Figure 4. The solutions exhibit excellent agreement over the wave fronts and the body of the wave, with only minor deviations observable at the crests, which is to be expected due to the different methods which collocation and our cellular automata technique use to handle discontinuities.

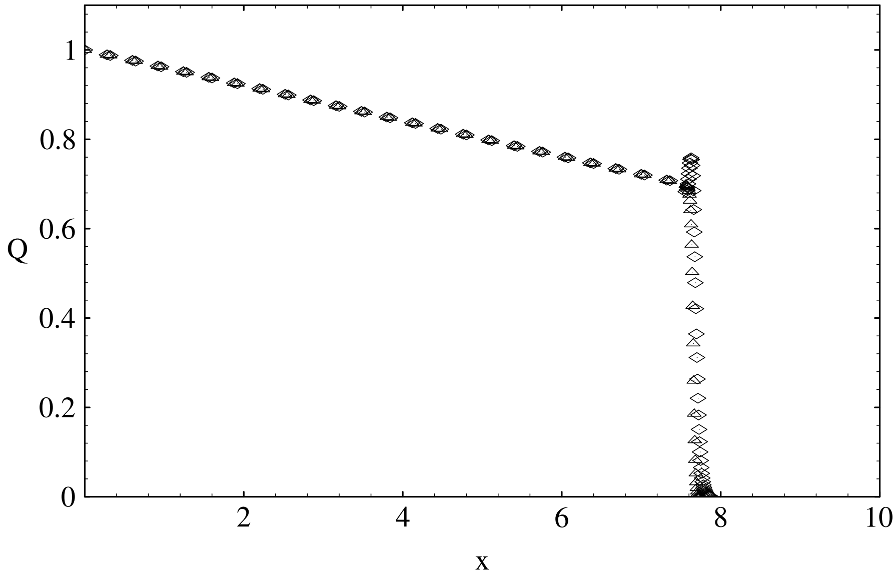

Figure 5: Cellular automata and collocation solutions for the nonlinear telegraph boundary value problem. The collocation solution is indicated with the triangles and the cellular automata solution with diamonds.

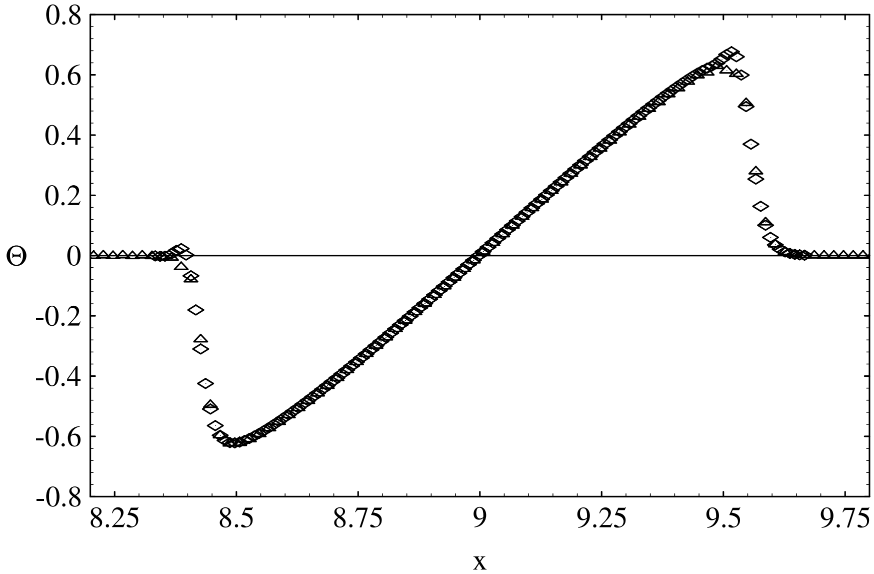

Figure 6: Cellular automata and collocation solutions for the nonlinear telegraph initial value problem. The collocation solution is indicated with the triangles and the cellular automata solution with diamonds.

4.2 Parallel Results

The parallel implementation was done on an iPSC/860 hypercube using cubes ranging in size from 1 to 64 processors. What ultimately determines the success of this algorithm is how well it handles increased degrees of parallelization—in other words, how does the performance of the program change as we add more processors? If a program utilizes a computer's processing power poorly as processors are added, then clearly it is not suitable for massively parallel architectures.

Speedup and efficiency gauge a parallel program's performance as the number of processors increases based on the best known serial algorithm, while parallelizability is based on the parallel code's own performance on a *single* processor [18, 19]. What ultimately determines the suitability of a parallel program is its speedup. As stated in Section 1, cellular automata will always perform at a lower level than traditional methods on a single processor. However, if the cellular automata has a sufficiently high parallelizability, then for a large enough number of processors, the cellular automata algorithm will have a higher speedup than parallelized versions of traditional algorithms and will ultimately exceed their performance.

Figure 7 shows the parallelizability of the cellular automata algorithm, which is the ratio of time on a single processor to time on multiple processors. Excellent results are obtained by the algorithm when run with a large enough problem size to provide sufficient work for the number of processors used. The reason for the slight decrease in efficiency as more processors are added is that the parallelizability is a function of p/k. Adding more processors for a fixed n decreases that ratio and thus requires a larger percentage of the processing effort to be spent maintaining the overlapped cells. The loss shown in Figure 7 is easily avoided on problems for which the use of a massively parallel supercomputer is reasonable, however the problems chosen for examination in this paper were chosen for their convenience in developing and demonstrating the potential of cellular automata, not for their ability to fully utilize a modern parallel supercomputer.

To calculate the speedup, it is necessary to determine the best serial algorithm, implement it on a single processor of the parallel machine, and compare the speed of the new parallel algorithm to the serial algorithm's

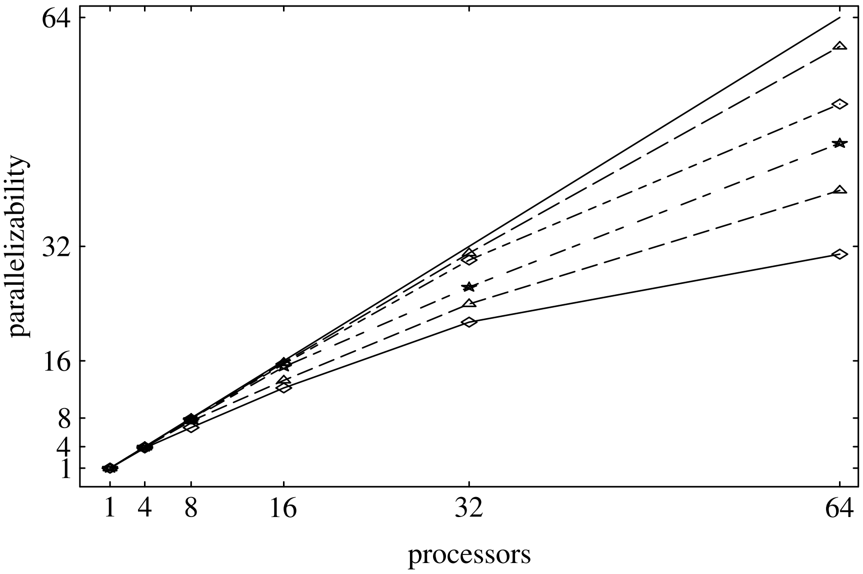

Figure 7: Parallelizability of the cellular automata solution compared to perfect parallelizability. From bottom to top, the lines indicate the parallelizability for systems with 10000, 20000, and 40000, 80000, and 160000 cells. The very top line shows perfect parallelizability.

speed [13]. It is difficult just to determine the best serial algorithm, so we will not attempt to evaluate the speedup of this algorithm analytically. A more meaningful measure is Figure 7, which clearly indicates that the cellular automata paradigm is capable of extracting extremely high efficiency from a massively parallel processor, given a sufficiently large computational domain. On the other hand, an implicit finite difference scheme, for example, will be increasingly unable to make efficient use of a parallel processor as it attempts to utilize more processors [6]. Therefore, even though the cellular automata algorithm is not as fast on a single processor, it overtakes and surpasses the implicit techniques on parallel processors [6].

5 CONCLUSIONS

We have shown that when implemented on massively parallel distributed memory computers, cellular automata algorithms parallelize significantly better than traditional algorithms. This allows them to perform faster on such machines. The accuracy achievable by the technique has been verified by comparisons of cellular automata generated results with analytical and computed solutions. These two factors make cellular automata an attractive alternative to using traditional techniques on the massively parallel computers which are becoming standard in high-performance computing. Cellular automata techniques modeling complex phenomena are easy to implement, requiring care only when handling the boundaries between processors. The rules do not have to meet any specific requirements and can be derived ab initio or obtained using other techniques which produce explicitly solvable equations.

The solutions demonstrated in this paper have indicated that cellular automata are capable of solving a variety of problems from the simple heat equation to wave solutions involving discontinuities. Future work will include applying the techniques developed in this research to more complicated problems and extending the method to two and three dimensions.

References

[1] K. K. Tamma and R. R. Namburu, Hyperbolic Heat-Conduction Problems: Numerical Simulations via Explicit Lax-Wendroff Based Finite Element Formulations, *J. Thermophysics,* vol. 5, pp. 232–239, 1990.

[2] D. E. Glass and D. S. McRae, Variable Specific Heat and Thermal Relaxation Parameters in Hyperbolic Heat Conduction, *J. Thermophysics,* vol. 4, pp. 252–255, 1990.

[3] M. N. Özisik and D. Y. Tzou, On the Wave Theory in Heat Conduction, *Proceedings of the Symposium on Fundamental Issues in Small Scale Heat Transfer, Anaheim, CA,* pp. 13–28, 1992.

[4] D. D. Joseph and L. Preziosi, Heat Waves, *Reviews of Modern Physics,* vol. 61, pp. 41–73, 1989.

[5] D. Y. Tzou, Thermal Shock Phenomena Under High Rate Response in Solids, *Annual Review of Heat Transfer,* vol. 4, pp. 111–185.

[6] D. P. Bertsekas and J. N. Tsitlikis, *Parallel and Distributed Computation,* Prentice Hall, Englewood Cliffs, NJ, 1989.

[7] N. Wiener and A. Rosenbluth, The Mathematical Formulation of the Problem of Conduction of Impulses in a Network of Connected Excitable Elements, Specifically in Cardiac Muscle, *Arch. Inst. Cardiol. Mexico,* vol. 16, pp. 205–265, 1946.

[8] J. R. Weimar, J. J. Tyson, and L. T. Watson, Diffusion and wave propagation in cellular automaton models of excitable media, *Phys. D*, vol. 55, pp. 309–327, 1992.

[9] K. Preston and M. J. B. Duff, *Modern Cellular Automata, Theory and Applications,* Plenum Press, New York, 1984.

[10] Y. Taitel, On the Parabolic, Hyperbolic, and Discrete Formulation of the Heat Conduction Equation, *Int. J. Heat Mass Transfer,* vol. 15, pp. 369–371, 1972.

[11] M. Chester, Second Sound in Solids, *Phys. Rev.,* vol. 131, pp. 2013–2015, 1963.

[12] D. E. Glass, M. N. Özisik, D. S. McRae, and B. Vick, Hyperbolic Heat Conduction with Temperature-Dependent Thermal Conductivity, *J. Appl. Phys.,* vol. 59, pp. 1861–1865, 1986.

[13] C. Hirsch, *Numerical Computation of Internal and External Flows,* vol. 1, John Wiley & Sons, New York, 1991.

[14] M. N. Özisik, *Basic Heat Transfer,* McGraw-Hill, New York, 1977.

[15] Madsen, N. K. and R. F. Sincovec, PDECOL—Collocation Software for Partial Differential Equations, *ACM—Trans. Math. Software,* vol. 5, pp. 326–351, 1979.

[16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computation,* 1st ed., Cambridge University Press, Cambridge, 1989.

[17] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis,* 4th ed., Addison Wesley, Reading, MA, 1989.

[18] M. J. Quinn, *Parallel Computing, Theory and Practice,* McGraw Hill, New York, 1993.

[19] M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers,* McGraw Hill, New York, 1987.

List of Figures