Tuning Complex Systems by Sonifying Their Performance Data

Cody Henthorne
Raytheon BBN Technologies
chenthor@bbn.com

Eli Tilevich
Department of Computer Science
Virginia Tech
tilevich@cs.vt.edu

Ivica Ico Bukvic
Department of Music
Virginia Tech
ico@vt.edu

Pardha S. Pyla
Senior Interaction Designer
Bloomberg L.P.
pardha@pyla.name

Abstract

In the modern computing landscape, the challenge of tuning software systems is exacerbated by the necessity to accommodate multiple divergent execution environments and stakeholders. Achieving optimal performance requires a different configuration for every combination of hardware setups and business requirements. In addition, the state of the art in system tuning can involve complex statistical models, which require deep expertise not commonly possessed by the average software developer. This paper presents a novel approach to tuning complex software systems by leveraging sound to convey performance information during execution. We conducted a scientific survey to determine which sound characteristics (e.g., loudness, panning, pitch, tempo, etc.) are most accurate to express information to the average programmer. As determined by the survey, the characteristics that scored the highest across all the participants were used to create a proof-of-concept demonstration. The demonstration showed that a programmer who is not an expert in either software tuning or enterprise computing can configure the parameters of a real world enterprise application server, so that its resulting performance surpasses that exhibited under the standard configuration. Our results indicate that sound-based tuning approaches can provide valuable solutions to the challenges of configuring complex computer systems.

Categories and Subject Descriptors D.2.8 [*Software Engineering*]: Metrics—Performance measures; D.2.9 [*Software Engineering*]: Management—Software configuration management; C.4 [*Computer Systems Organization*]: Performance attributes

General Terms Experimentation, Performance, Human Factors

Keywords Performance Tuning, Sonification, Empirical Studies, J2EE, Enterprise Application Servers

1. Introduction

A system's performance depends as much on the efficiency of its design as it does on the precision of its configuration. Performance tuning, a key facet of configuration, is the process of choosing those input parameters that achieve the desired level of some performance characteristic. For example, properly choosing the size for a server's thread pool can increase throughput or decrease latency.

Performance tuning is notoriously hard due to the need to accommodate multiple deployment environments and business requirements. Furthermore, pinpointing the exact source of inefficiency in a complex system can be non-trivial. Finally, it is often unclear how the value of a configuration parameter affects system performance.

Existing approaches to performance tuning take advantage of performance profiling [16, 30], visualization [29], and automation using complex statistical models [4]. Because performance tuning remains a standing challenge in the face of the increasing complexity of modern systems, there is great potential benefit in ex-

ploring novel tuning approaches that utilize untapped resources. One such resource is using sound to convey performance information.

Programmers have long recognized the utility and value of sound as a cognitive tool for understanding their programs. When dealing with data intensive applications, programmers often listen to the sounds made by hard drives to determine if thrashing is present. When dealing with processor intensive applications, programmers listen to the sounds made by CPU cooling fans to determine if a number crunching phase is in progress. It is worth noting, however, that the overall trend in hardware design aims at silent execution. For example, solid state hard drives are silent, as they do not contain any moving parts.

But what if executing programs could provide meaningful performance information to the programmer through sound? Furthermore, what if the sounds used were specially designed, so that multiple performance indicators could be effectively discerned by the programmer? This way, the sound would indicate the level of different performance characteristics. Further, in response to the programmer tuning a configuration parameter, the provided sound would immediately reflect if performance was affected by the change.

In this paper, we report on the results of an investigation that we conducted to explore these questions. For the purposes of our investigation, we have tapped into the ability of sound to convey information, otherwise known as *sonification* [19]. Additionally, in order to sonify multiple concurrent streams of performance data, we relied upon the concept of aural focus [23], or the human ability to segregate different aural stimuli in noisy environments (e.g., having a conversation in a loud public space).

Nevertheless, how performance information can be represented effectively within the auditory modality has not been sufficiently investigated. Although prior research has explored using sound and music to represent program execution [26], such results have to be confirmed experimentally to determine their effectiveness specifically as tuning aids.

To obtain reproducible results that can be used by us and other researchers, we conducted a scientific study that evaluated the accuracy with which programmers can perceive sound properties as the means to convey performance information. The survey involved more than 30 programmers with different levels of programming and music expertise. A statistical analysis of the study's results revealed that there are indeed sound properties that are more effective in general sonification.

With these results in hand, we architected, implemented, and evaluated a proof-of-concept performance tuning system that uses sonification to guide the programmer in tuning their systems for improved performance. For our evaluation, we focused on tuning a real world enterprise system—the GlassFish Application Server [22], which is widely used in real world installations. Equipped with our tuning system, all the evaluation's subjects were able to successfully tune GlassFish to achieve a level of performance higher than that provided by the standard configuration.

Based on our results, this paper presents the following novel contributions:

- A systematic user study that assessed the accuracy of sonification to convey information; the results of the study can inform software system designers who want to integrate sonification into their systems.
- A software architecture for tuning software systems with sound; our architecture harmoniously integrates a system under test, a workload generator, a tuning user interface, and a sound rendering engine.
- A proof of concept prototype that demonstrates the utility of our approach and its applicability to real world systems.

The rest of this paper is structured as follows. In Section 2, we describe our user study, and in Section 3 we analyze the results. Then in Section 5, we describe our proof-of-concept tuning system that follows our architecture and present the system's initial results. In Section 6, we compare this work to the related state of the art, and finally in Section 7, we discuss future work directions and present concluding remarks.

2. Investigating Underlying Sonification Parameters

First, we lay out the questions our user study was designed to answer. Then we describe the study's methodology and implementation.

2.1 Study Objectives and Questions

An individual's response to particular sounds is subjective; it depends on factors including one's cultural background, level of music sophistication or training,

and frequency of exposure to non-speech audio [11]. Since professional software developers come from all walks of life and have varying cultural and ethnic backgrounds, a sonification approach must be as broadly applicable as possible to be beneficial. This entails choosing those sound characteristic which are interpreted uniformly by a substantial percentage of software developers.

Therefore, one must ask which sound characteristics are most suitable for the task at hand, performance tuning. To that end, we conducted a systematic study that gathered empirical evidence assessing the fundamental affordances of sound characteristics and their effectiveness when applied to performance tuning. Although several prior studies assessed the fitness of sonification techniques for various software engineering tasks [2, 9, 26], our study focuses specifically on how sonification can help performance tuning. In particular, our intent was to design a study that can answer the following questions: (1) Which sound characteristics are most suitable to facilitate perception and comprehension in a performance-tuning context? (2) Which sound characteristics are most effective in expressing whether a value is increasing or decreasing, so maximum accuracy is achieved across different users? (3) How effectively can a user understand a mapping of sound characteristics to the performance metrics of a running application?

2.2 Study Methodology and Design

Our online survey consisted of five sections: Demographic, General, Panning, Mapping, and General Feedback. Each section was specifically designed to help answer the questions above.

The Demographic Section gathered demographic information including years of programming experience, programming environments experience, music background, genre preferences, music listening habits, and hearing impairments.

The General Section determined which sound attributes participants naturally mapped to increasing or decreasing values. Participants listened to eight 10-second audio clips and indicated if they felt the clip represented "something" increasing a lot, increasing a little, staying constant, decreasing a little, or decreasing a lot. The eight clips modified the following sound characteristics: pitch, loudness, panning, and timbre. The questions did not inform the participant which sound characteristic was being exercised or what should be interpreted as increasing or decreasing.

The Panning Section enforced a mapping of sound coming from the left channel as low values and sound coming from right channel as high values. The participant was asked to listen to five short sound clips and indicate on a scale of 1 (lowest) to 100 (highest) what value they felt the clip represented.

The Mapping Section asked participants to interpret a 30-second sonification of program performance data including disk usage, memory usage, and network usage. Each mapping question highlighted different sonification attributes and provided sample sonifications for the participant. The participant listened to sample sonifications and then the full performance data sonification. As the sonification played the participant was asked to indicate on a scale of 1 to 10 where they felt the usage level for each resource (disk, memory, and network) was during the beginning, middle, and end of the clip.

The General Feedback Section collected comments and suggestions about the survey. This feedback was used to gather subjective data from software engineers about using sonification in general and specifically for performance tuning.

2.3 Study Implementation and Deployment

To gather inputs from a large and diverse population of participants, we created a Web-based survey. Participants were recruited by posting to graduate student mailing lists at five universities (Virginia Tech, Georgia Tech, University of Maryland, University of California in Irvine, and University of Rochester) and professional software developer mailing lists at four companies (Google, Amazon, Microsoft, and Raytheon BBN).

The survey was implemented as a Rich Internet Application constructed using the Adobe Flex Framework [1]. The sonification files used for the survey's questions were custom designed using Max/MSP [6] and saved as MP3s. To achieve meaningful results, a comprehensive configuration section guided participants in selecting appropriate volume levels and headphone orientation to be used throughout the survey.

3. Case Study Analysis and Results

In this section, we first analyze the data produced by the study that we described in Section 2. We then explain

how the results of this analysis can inform the design of novel software engineering techniques and tools that use sonification.

3.1 Analysis

Overall we had 42 participants in our study. Of these, 33 participants successfully completed the survey. The survey tracked the number of times a participant listened to a sound clip. If a participant did not listen to all the sound clips, all of their answers were excluded from analysis. Except for this restriction, no other data was removed from analysis.

The data collected in the survey was multi-faceted and large. We decided to focus our analysis on primarily finding the three strongest sound characteristics for sonifying performance information. Focusing our analysis allowed us to find the information needed to build a proof-of-concept tuning system quickly. After performing a cursory analysis of all the data, we focused on the general and mapping sections, as they showed the most promise.

Analysis of the general section provided us with a distribution of participants who were able to map increasing and decreasing trends to particular changes in sound characteristics. As described in Section 2.2, the general section of our survey was a collection of multiple choice questions which asked participants to indicate how they felt changes in basic sound characteristics mapped to "something" increasing and decreasing. Analysis of this section was fairly straightforward, as we calculated the distribution of participants that selected each choice for a given question. See Figure 1 for a summary of the analysis of the general section.

The mapping section consisted of 6 major questions each with nine subquestions. These questions asked the participant to rate their observed usage levels of three resources (disk, memory, and network) on a scale from 1 to 10 during the beginning, middle, and end of a 30 second program sonification clip. For simplicity, the nine subquestions when referenced individually, will be referred to as B-DISK, B-MEM, B-NET, M-DISK, M-MEM, M-NET, E-DISK, E-MEM, and E-NET for remainder of this paper. Each question enforced a mapping of particular sound characteristics to resource usage. Since the data collected from this section was large, we have only included the analysis of one question answered with the highest accuracy. The question utilized panning from left to right and from right to left

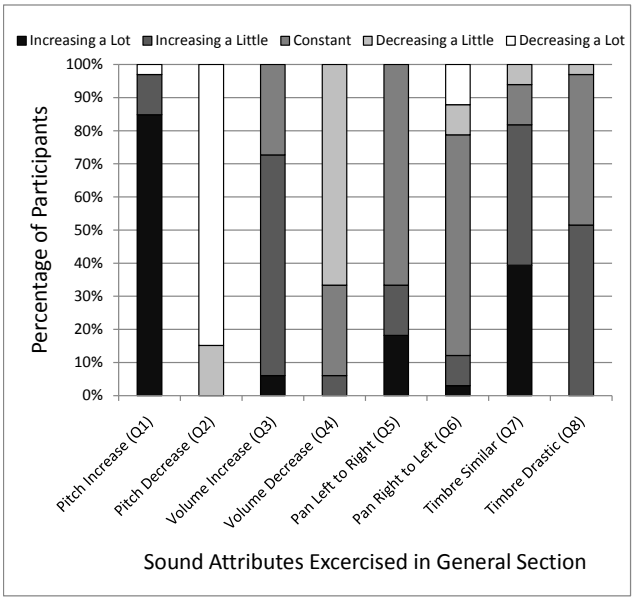

Figure 1. Distribution of participants' responses to all questions in the general section.

to show increasing and decreasing resource usage, respectively.

To analyze the panning question within the mapping section we calculated the average error for each resource at each location in the clip. The average error was calculated by taking the absolute value of the difference between the average of all the participants' responses and the expected answer for each subquestion. The error analysis is summarized as a column graph in Figure 2.

While the error analysis provides an estimate of the accuracy of using panning to sonify resource usage, it does not indicate any trends across the participant population. We calculated 95% confidence intervals for the average response of each subquestion. Figure 3 shows the confidence interval as error bars for the average participants' response for each subquestion along with the expected response.

3.2 Results

As is evident from the analysis of the general section, 85% of the participants mapped increasing pitch to "something" increasing a lot and 12% mapped increasing pitch to increasing a little. Additionally, 85% of participants mapped decreasing pitch to "something" decreasing a lot and 15% mapping decreasing pitch to decreasing a little. This is a strong indication that

most participants already map increasing and decreasing pitch to increasing and decreasing values.

The low error for using panning is an indicator that panning can provide a high level of accuracy for representing changes in sonified data, as long as the participant is aware of the mapping. Analysis of the general section, provided evidence that there is no apparent natural mapping from panning to resource usage; however, if a mapping of panning location to usage level is enforced, panning performs well. Panning is further supported by the confidence interval analysis, as it shows a trend that users will answer consistently within an acceptable level of accuracy.

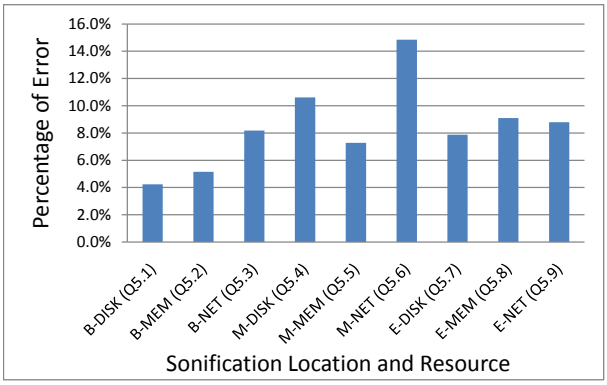

Figure 2. Error in mean participant vs. expected response for each panning subquestion. The persistently low error indicates that panning could provide a high level of accuracy for sonifying information.

The general feedback section of the survey provided a form for participants to provide subjective comments about the survey. While this data is subjective, multiple participants indicated a strong preference for panning based sonifications. We believe this strong preference for panning stems from the user's having been bounded within the left and right limit.

4. Software Architecture for Performance Tuning with Sound

We now present our general software architecture that can be used to create performance tuning systems that utilize sonification. Then we describe our software engineering process for developing, refining, and deploying sonification based tuning systems.

The primary goal of our proposed architecture is to codify how one can create a modular and extensible system for tuning complex computer systems using sonification. Using our architecture, one should be able

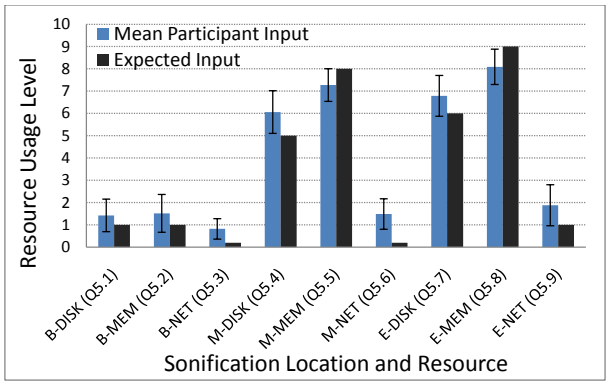

Figure 3. Expected vs. mean participant response with 95% confidence intervals as error bars for each panning subquestion. Since most expected responses fall within the confidence intervals, it shows that users will likely answer consistently and accurately.

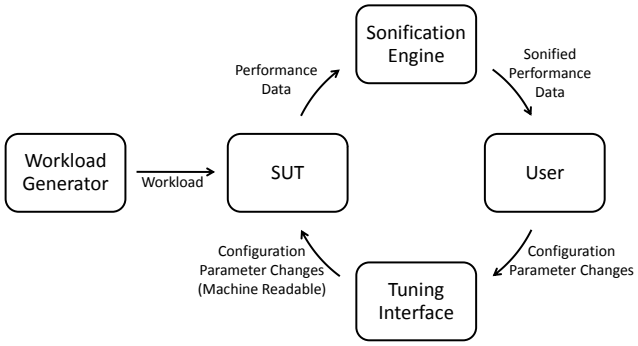

Figure 4. Instantiation of our software architecture for our proof-of-concept demonstration.

to select component implementations to accommodate the requirements of different application domains.

Figure 4 depicts our proposed architecture for interactive tuning systems that use sonification as a cognitive guide. The architecture integrates the following conceptual entities: system under test (SUT), workload generator, sonification engine, and tuning interface.

System Under Test (SUT)

The SUT is the system to be tuned. For a SUT to be tunable, it must expose modifiable configuration parameters that can potentially affect some system performance characteristic. The SUT must also be equipped with performance data gathering facilities that can provide runtime information accurately without interfering with the SUT's execution. A trend in modern system design is to include such data gathering facilities as a standard feature.

Workload Generator

The SUT is exercised by the workload generator, which emulates realistic usage scenarios. The workload generator and SUT are tightly coupled with respect to the type of interaction that may occur between them. The purpose of generating workloads is to be able to tune a realistic system in an artificial (lab) environment, without compromising the accuracy of the end result.

Sonification Engine

The sonification engine continuously takes, as input, the performance data from the SUT and, as output, renders the appropriate sonifications. The designer of the tuning system has significant leeway both in terms of what data they choose to sonify and which sonifications to employ. In addition, sonification can be optionally combined with other cognitive aids, such as visual and haptic displays.

Tuning Interface

Based on the cognitive feedback received from the sonification engine, the user interacts with the tuning interface. The purpose of the tuning interface is to expose to the end user intuitive controls for manipulating configuration parameters and to relay the user's input into the actual SUT's configuration settings.

5. Proof-of-Concept

Our proof-of-concept sonification tuning system leverages the insights from our study described in Section 2. The four components of our proof-of-concept are instantiated as follows:

System Under Test (SUT)

The SUT is the system to be tuned. For a SUT to be tunable, it must expose modifiable configuration parameters that can potentially affect some system performance characteristic. As our SUT, we chose a real-world commercial application server, an essential and representative part of modern enterprise infrastructure. An application server provides a runtime environment for enterprise applications. The performance and reliability requirements imposed on an application server are comparable to that of an operating system. Specifically, we chose Oracle GlassFish v3.0 [22], which has been successfully deployed in commercial installations.

As our enterprise application, we chose a Blueprints J2EE 5 reference application, PetStore 2.0 [13]. The

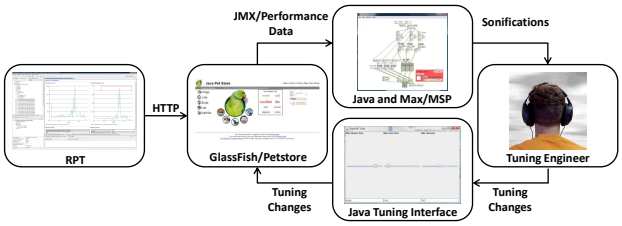

Figure 5. Instantiation of our software architecture for our proof-of-concept demonstration.

PetStore application is representative of modern J2EE technologies and additionally has been used as an evaluation subject in several other performance tuning-related research and demonstrations [4, 17]. PetStore encompasses the functionality of a typical e-commerce application, in which users can browse, select, sell, and purchase pets.

GlassFish exposes an extensive set of monitoring and performance data via Java Management Extension (JMX). This mechanism provides a standardized avenue through which external clients can obtain information about a running system [20]. External applications can access GlashFish performance data, including processor usage, request rate, and connection queue status through a Java API.

Workload Generator

The SUT is exercised by the workload generator, which emulates realistic usage scenarios. Because our SUT is a Web server-based system, our workload generator emulates clients accessing the SUT remotely across the network using the HTTP network protocol. To reproduce a realistic workload for a typical e-commerce application, we used IBM Rational Performance Tester (RPT)[1]. This flagship testing tool from IBM is used by enterprises to verify whether their applications can handle potential workloads. We used the RPT test generation tools to create 75 virtual users who were exercising the functionality of the SUT under various frequencies commonly observed in practice.

Sonification Engine

The sonification engine continuously takes, as input, the performance data from the SUT and, as output, renders the appropriate sonifications. Our sonification engine comprises two components written in Java and

[1] `http://www-01.ibm.com/software/awdtools/tester/performance/`

Max/MSP [6], communicating with each other through TCP sockets. The Java component collected performance data from a running GlashFish instance via JMX. Specifically, the performance data our system collected was the average requests per second, CPU usage, and connection queue length. As the Java component collected performance data it dispatched the data as sequences of integers that were received, interpreted, and sonified by the Max/MSP component.

We pursued two key goals while designing the sonifications for our proof-of-concept tool: (1) they must not be "tiring" for the listener during a prolonged tuning session, and (2) their spectral differences must make them sufficiently distinct from each other. Therefore, for our MAX/MSP component, we chose naturally-occurring sounds that were designed as follows: (1) a sustained chord as if it were played by a string instrument, (2) a low hum of a cooling fan, and (3) a filtered sound of a mechanical hard drive.

As our study indicated, pitch and panning were the two sound characteristics that proved most effective for the definitive majority of the study's participants. To take advantage of this observation, we sonified the range of a performance parameter such that it was mapped to a corresponding pitch range. Thus, there was a direct correspondence between the value of a performance parameter and the pitch of the sound used to represent it. For pitch, we chose to use a range of two octaves that provided significantly noticeable difference in sounds while limiting their distortion.

To improve accuracy, we also sonified the range of a performance parameter by assigning the left channel to reflect the smallest value while the right channel the highest one. The panning location of a given sound between channels thus reflected the sonified value.

Tuning Interface

Based on the cognitive feedback received from the sonification engine, the user interacts with the tuning interface. The purpose of the tuning interface is to expose to the end user intuitive controls for manipulating configuration parameters and to relay the user's input into the actual SUT's configuration settings. JMX is a bidirectional interface. In addition to allowing external clients to obtain information about a running system, one can also use JMX to set values in the system. We designed a Java GUI that exposed three GlassFish configuration parameters, number of acceptor threads, maximum request thread pool size, and idle thread timeout to the user. When the user adjusts a parameter via the GUI, the application sends the configuration change to GlassFish.

5.1 Pilot Study: Tuning GlassFish with Sound

The purpose of this pilot study was to investigate whether *sound-based tuning can enable non-experts in either the tuned application's domain or tuning itself to function as effective systems engineers*. In the following, we describe the design, execution, and analysis of our pilot study.

Our study subjects were four professional software developers, employed by a large technology R&D firm. Although quite experienced in their core domain, they only had a cursory knowledge with enterprise infrastructure and vague familiarity with configuring and tunning of application servers. Because all the subjects happened to have participated in our online survey, they were familiar with the general concept of conveying computing information with sound.

The pilot was executed by first exposing each subject to our tuning interface and the available tuning parameters. We were careful though not to recommend or give specific guidelines regarding how the parameters' values should be set. Then each subject learned which sonifications represented which performance metric and how the changing performance data affected the sonifications. In particular, the sonified performance metrics were: requests per second, CPU usage, and connection queue length (i.e., number of requests waiting to be serviced). Finally, each subject was presented with the tuning goal of maximizing requests per second, and the auxiliary goals of maximizing the CPU usage, and minimizing the connection queue length.

The main procedure of the pilot included two runs, during which each subject attempted to achieve the aforementioned tuning objective while being guided by our sonifications. The reason for including two runs was to present an opportunity for the subjects to become more comfortable with our tuning system. Each run lasted for six and half minutes, and only the second run's results were recorded for future analysis. To ensure greater accuracy, the GlassFish server was restarted between runs.

Figure 6 shows the requests per second measurements for all four subjects during the final two and half minutes of the second run. The vertical dashed line des-

ignates the time point at which the RPT was winding down its simulation, which reduced the number of virtual users accessing the web application.

The analysis of the results shows that all the subjects were able to specify a tuning configuration that led to a performance level higher than that of the standard configuration (SC). However, the degree to which they were able to outperform the SC differed between the subjects. One subject (P2) consistently outperformed the SC, and toward the end of the second run achieved a configuration that was processing over 400 more request per second than the SC. Another subject (P1) also managed to outperform the SC by the end of the second run, but with more quality fluctuations in the intermediate configurations. Being able to specify a high quality configuration at some point, P1 then somewhat worsened the results but was able to recover toward the end of the run.

The other two subjects (P3 and P4) yielded similar results, which were exemplified with a longer learning curve. Although consistently improving their configuration, they took significantly longer to become comfortable with our tuning system. One way to explain this disparity is that programmers' auditory perception and multitasking abilities tend to differ. P3 explicitly stated that he "was not a good multitasker" and was not even able to listen to music while programming.

We conjecture that the performance of P3 and P4 could have been improved to a greater degree had the run been longer. Based on the trajectory of their results, it is possible that they would be able to eventually achieve the level of performance comparable to that of P1 and P2.

Thus, the results of our pilot study indicate that sound can be effective as a means of conveying performance information to aid tuning. It is certainly worth investigating further the exact human and technical factors that affect the effectiveness of sonification as a cognitive tool.

6. Related Work

Performance tuning, long recognized as a formidable software engineering challenge, has been the target of multiple research efforts. In the following discussion, we first present performance tuning approaches that are similar to ours. Then we give an overview of key sonification concepts and other systems that use information sonification.

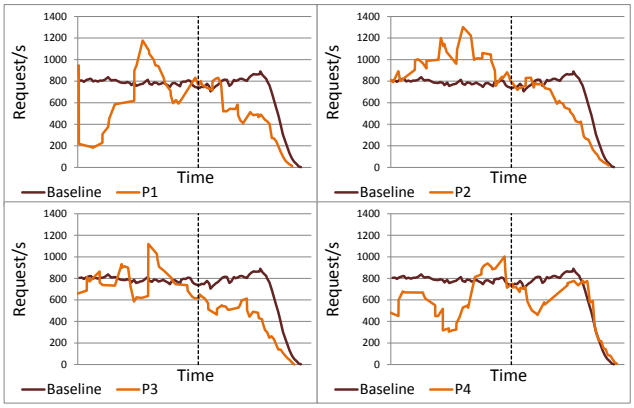

Figure 6. Graphs depicting the four participants' requests/second compared to GlassFish's standard configuration. The data starts at two and half minutes prior to the end of their second run. The vertical dashed line indicates when virtual users began logging off.

6.1 Performance Tuning Approaches and Tools

Jovic and Hauswirth [16] presents an approach to tuning Java GUI applications that profiles event listener latency. Another profiling approach to analyzing performance and tuning applications is presented by Zagha et al. [30]. They provide a framework of counters and profiling tools for tuning systems based on the MIPS R10000 processor.

Other approaches have leveraged visualizations to aid tuning. Walker et al. [29] visualize the large amounts of data collected during the execution of an object-oriented system at the architectural level. Chassin et al. [8] visualize a variety of thread interactions to facilitate tuning multi-threaded parallel applications. Jones et al. [15] used profiling and visualizations to aid tuning of parallel applications.

Other approaches have leveraged statistical and stochastic tools to automate tuning instead of simply aiding a manual process. Chen et al. [4] utilize the Markov decision process and a reinforcement learning strategy to discover a system's optimal configuration. Additionally, similar to our work, they also validate their approach by tuning an older version of the J2EE PetStore application. Tiwari et al. [25] present a parallel algorithm for automatically tuning parallel applications. Their work utilizes the automated tuning framework, Active Harmony [5], presented by Ţăpuş et al. as a framework for automated runtime tuning.

6.2 Information Sonification

A good reference that outlines key concepts of sonification research and design is an article by Walker and Nees [28] that defines an *auditory display* as using sound to convey information and *sonification* as an auditory display consisting of non-speech audio. Another definition of sonification is provided by Kramer at al. [19], who define it as "the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation."

Most of sonification research is concerned with identifying those scenarios for which auditory displays provide effective solutions [14, 18, 24]. For example, auditory displays have been particularly effective in rendering complex data patterns and events requiring the user's immediate attention [10, 21]. This is because human hearing excels at identifying temporal information.

A Sonification Design Map presented by deCampo [7] shows quantitative relationships between non-speech auditory displays. A traditional classification of sonification approaches includes audification [18], parameter mapping, and model-based [12]. Categorizing sonification approaches on the basis of their respective data representations results in continuous, discrete, and model-based data. According to this categorization, the sonification approach that we used in this work is model-based. This representation uses a model based on the properties of the data to mediate between the sonified data and the sound. Because the model captures the domain knowledge of the sonified data, it can be applied to different types of datasets.

Several prior approaches have used auditory displays to convey information about computer applications. Vickers and Alty [26] use music to communicate information about which programming language structures are used, to express how programs behave at runtime, and to find potential program defects [27]. They have created the CAITLIN system to aurolize Turbo Pascal programs. One of the most salient insights of their investigation is that music can successfully communicate useful information about computer programs for all users, even for those who have not been formally musically trained.

When representing program constructs using both speech and non-speech audio, Finlayson and Mellish [9] concluded that the two modalities should be used together for maximum benefit. To improve program understanding, Berman and Gallagher [3] sonify program slices.

Compared to the aforementioned auditory displays approaches to convey program information, this work focuses on understanding the performance of a computer application that we treat as a black-box. That is, we sonify the external behavior of an application without any regard for its source code or any other software artifacts. Furthermore, our sonifications are significantly less-structured than music tunes. Finally, we change our sonifications interactively in response to tuning actions of the user.

7. Future Work and Conclusions

This work has explored sonification as a cognitive aid to assist performance tuning. It also opens up several possible future work directions. In particular, we plan to:

- analyze the results of our case study further; correlating the participants' demographics to their answers looks particularly promising. If noticeable differences exist for identifiable demographics, sonification tools can be customizable to accommodate specific users.
- leverage the results of our studies to create approaches that facilitate software engineering tasks other than tuning. As certain sound characteristics can effectively convey software information, we can revisit important software engineering challenges such as program comprehension, analysis, and bug detection.
- evaluate actual performance tuning tools in greater detail. Now that our proof-of-concept tool shown that our general approach is feasible, we would like to further investigate its effectiveness in relation to other tuning approaches.
- combine sonification with other cognitive aids such as visualization. Sonification and visualization have their respective strengths and weaknesses, and when combined together can convey more information better than either cognitive aid in isolation.

The ever increasing complexity of modern computer systems calls for new and more powerful approaches to accommodate these systems for the needs of real users. In that light, tuning systems for optimal performance has come to the forefront of industrial software engi-

neering. This paper has presented a novel approach that leverages sonification to facilitate performance tuning of complex computer systems. We believe that our approach can lay the foundation for a new generation of approaches and tools that use sonification. This powerful cognitive aid can help tame the complexity of engineering computer systems of today and tomorrow.

Acknowledgements The authors would like to acknowledge VT LISA for their statistical help.

References

[1] Adobe. Adobe Flex. `http://www-sjc0.adobe.com/products/flex/`, 2010.

[2] L. Berman, S. Danicic, K. Gallagher, and N. Gold. The sound of software: Using sonification to aid comprehension. In *14th IEEE International Conference on Program Comprehension (ICPC'06)*, pages 225–229, 2006.

[3] L. I. Berman and K. B. Gallagher. Listening to program slices. In *Proceedings of the 12th International Conference on Auditory Display (ICAD)*, 2006.

[4] H. Chen, G. Jiang, H. Zhang, and K. Yoshihira. Boosting the performance of computing systems through adaptive configuration tuning. In *SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1045–1049, New York, NY, USA, 2009. ACM.

[5] C. Ţăpuş, I.-H. Chung, and J. K. Hollingsworth. Active harmony: towards automated performance tuning. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[6] Cycling '74 Inc. Max/MSP. `http://www.cycling74.com`, 2010.

[7] A. de Campo. Toward a data sonification design space map. *Proceedings of the International Conference on Auditory Display (ICAD)*, pages 342–347, 2007.

[8] J. C. de Kergommeaux, B. Stein, and P. E. Bernard. Paj, an interactive visualization tool for tuning multithreaded parallel applications. *Parallel Computing*, 26(10):1253 – 1274, 2000.

[9] J. L. Finlayson and C. Mellish. The 'audioview' - providing a glance at Java source code. In *Proceedings of the 11th International Conference on Auditory Display (ICAD)*, 2005.

[10] J. Flowers, D. Buhman, and K. Turnage. Cross-modal equivalence of visual and auditory scatterplots for exploring bivariate data samples. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 39(3):341–351, 1997.

[11] D. Hargreaves. *The developmental psychology of music*. Cambridge University Press, 1986.

[12] T. Hermann. *Sonification for exploratory data analysis–demonstrations and sound examples*. PhD thesis, Bielefeld University, Bielefeld, Germany, 2002.

[13] Java BluePrints. Java Pet Store Demo. `https://blueprints.dev.java.net/petstore/`, 2010.

[14] G. Johannsen. Auditory displays in human–machine interfaces. *Proceedings of the IEEE*, 92(4):742–758, 2004.

[15] D. Jones, Jr., S. Marlow, and S. Singh. Parallel performance tuning for haskell. In *Haskell '09: Proceedings of the 2nd ACM SIGPLAN symposium on Haskell*, pages 81–92, New York, NY, USA, 2009. ACM.

[16] M. Jovic and M. Hauswirth. Measuring the performance of interactive applications with listener latency profiling. In *PPPJ '08: Proceedings of the International Symposium on Principles and Practice of Programming in Java*, pages 137–146, New York, NY, USA, 2008. ACM.

[17] K. Juse, S. Kounev, and A. Buchmann. Petstore-ws: Measuring the performance implications of web services. In *Proceedings of the 29th International Conference of the Computer Measurement Group on Resource Management and Performance Evaluation of Enterprise Computing Systems*, pages 113–123, 2003.

[18] G. Kramer. An introduction to auditory display. *Auditory Display: Sonification, Audification, and Auditory Interfaces*, pages 1–78, 1994.

[19] G. Kramer, B. Walker, T. Bonebright, P. Cook, J. Flowers, N. Miner, J. Neuhoff, et al. Sonification Report: Status of the Field and Research Agenda. 1999. *Prepared for the National Science Foundation by members of the International Community for Auditory Display*, 1999.

[20] McManus, Eamonn. Java Management Extensions Specification (JSR 3). `http://www.jcp.org/en/jsr/summary?id=3`, 2010.

[21] B. Moore. *An introduction to the psychology of hearing*. Academic Press San Diego, Calif, 2003.

[22] Oracle. Glassfish - Open Source Application Server, 2010. `https://glassfish.dev.java.net/`.

[23] N. Roman, D. Wang, and G. Brown. Speech segregation based on sound localization. *The Journal of the Acoustical Society of America*, 114:2236–2252, 2003.

[24] M. Sanders and E. McCormick. *Human Factors in Engineering and Design*. McGraw-Hill Science/Engineering/Math, 1993.

[25] A. Tiwari, V. Tabatabaee, and J. K. Hollingsworth. Tuning parallel applications in parallel. *Parallel Comput.*, 35(8-9):475–492, 2009.

[26] P. Vickers and J. Alty. Using music to communicate computing information. *Interacting with Computers*, 14(5):435–456, 2002.

[27] P. Vickers and J. L. Alty. When bugs sing. *Interacting With Computers*, 14:793 – 819, 2002.

[28] B. Walker and M. Nees. *Handbook of Sonification, In T. Hermann, A. Hunt, & J. Neuhoff (Eds.)*. New York: Academic Press, 2009.

[29] R. J. Walker, G. C. Murphy, B. Freeman-Benson, D. Wright, D. Swanson, and J. Isaak. Visualizing dynamic software system information through high-level models. In *OOPSLA '98: Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 271–283, New York, NY, USA, 1998. ACM.

[30] M. Zagha, B. Larson, S. Turner, and M. Itzkowitz. Performance analysis using the mips r10000 performance counters. In *Supercomputing '96: Proceedings of the 1996 ACM/IEEE conference on Supercomputing*, page 16, Washington, DC, USA, 1996. IEEE Computer Society.