DAiSES: Dynamic Adaptivity in Support of Extreme Scale
Department of Energy Project No. ER25622
Prime Contract No. DE-FG02-04ER25622
Final Report for September 15, 2004-September 14, 2008
PI: Patricia J. Teller, Ph.D.
University of Texas-El Paso
Department of Computer Science

1. Project Goals

The DAiSES project [Te04] was focused on enabling conventional operating systems, in particular, those running on extreme scale systems, to dynamically customize system resource management in order to offer applications the best possible environment in which to execute. Such dynamic adaptation allows operating systems to modify the execution environment in response to changes in workload behavior and system state. The main challenges of this project included determination of what operating system (OS) algorithms, policies, and parameters should be adapted, when to adapt them, and how to adapt them. We addressed these challenges by using a combination of static analysis and runtime monitoring and adaptation to identify *a priori* profitable targets of adaptation and effective heuristics that can be used to dynamically trigger adaptation. Dynamic monitoring and adaptation of the OS was provided by either kernel modifications or the use of KernInst and Kperfmon [Wm04]. Since Linux, an open source OS, was our target OS, patches submitted by kernel developers and researchers often facilitated kernel modifications. KernInst operates on unmodified commodity operating systems, i.e., Solaris and Linux; it is fine-grained, thus, there were few constraints on how the underlying OS can be modified.

Dynamically adaptive functionality of operating systems, both in terms of policies and parameters, is intended to deliver the maximum attainable performance of a computational environment and meet, as best as possible, the needs of high-performance applications running on extreme scale systems, while meeting system constraints. DAiSES research endeavored to reach this goal by developing methodologies for dynamic adaptation of OS parameters and policies to manage stateful and stateless resources [Te06] and pursuing the following two objectives:

1. Development of mechanisms to dynamically sense, analyze, and adjust common performance metrics, fluctuating workload situations, and overall system environment conditions.

2. Demonstration, via Linux prototypes and experiments, of dynamic self-tuning and self-provisioning in HPC environments.

From a high level, the DAiSES methodology, depicted in Figure 1, includes characterization of application resource usage patterns, identification of candidate (profitable) adaptation targets, determination of feasible adaptation ranges, definition of heuristics to trigger adaptation, design and implementation of OS monitoring, triggering, and adaptation code, and quantification of performance gains.

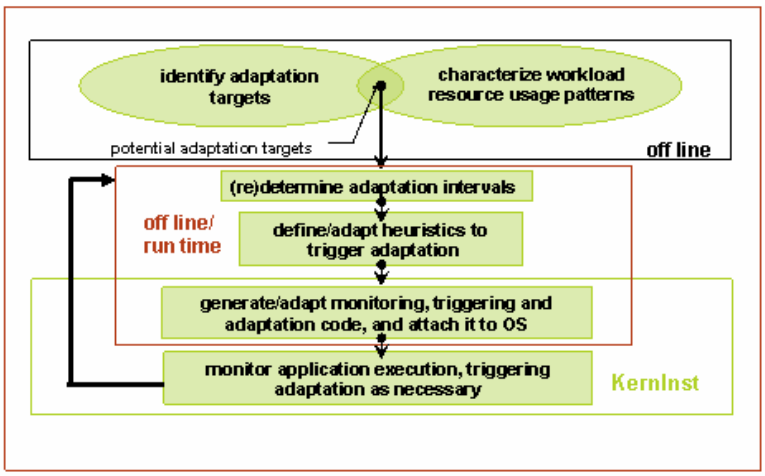


Figure 1: DAiSES Methodology

2. Participants

The research associated with this grant was performed at the University of Texas-El Paso (UTEP), Computer Science and the University of Wisconsin-Madison (UW), Computer Sciences. Although there was some collaboration between the research teams, most of the effort of the UW team was directed at the first objective mentioned above, i.e., development of mechanisms to support dynamic adaptation of operating systems, while most of the effort of the UTEP team was directed towards the second objective, i.e., demonstration of self-tuning and self-provisioning.

The research described in the following three Ph.D. dissertations is associated with the grant:

Mitesh R. Meswani, *Improving Throughput of Simultaneous Multithreaded (SMT) Processors using Shareable Resource Signatures and Hardware Thread Priorities*, Ph.D. Dissertation, University of Texas at El Paso, Computer Science, Anticipated May 2009.

Alexander Mirgorodskiy, *Automated Problem Diagnosis in Distributed Systems*, Ph.D. Dissertation, University of Wisconsin-Madison, Computer Sciences, October 2006 (VMWare).

Seetharami R. Seelam, *Towards Dynamic I/O Scheduling in Commodity Operating Systems*, Ph.D. Dissertation, University of Texas at El Paso (UTEP), Computer Engineering, ETD Collection for UTEP, http://digitalcommons.utep.edu/dissertations/AAI3214017, May 2006. (IBM TJ Watson Research Center)

In addition, the following seven students, who conducted research associated with this grant, attained Master's degrees:

University of Texas at El Paso (UTEP):

Muthuveer Somanathan, *An Initial Operating System Adaptation Heuristic for Swap Cluster Max (SCM)*, Master's Thesis, UTEP, Computer Science, ETD Collection for UTEP, http://digitalcommons.utep.edu/dissertations/AAI1456754, May 2008. (Intel)

Ricardo A. Portillo, *Fine-grain Dynamic Adaptation of the Linux 2.6 Virtual Memory Manager: A First Step*, Master's Thesis, UTEP, Computer Science, ETD Collection for UTEP,

PaperAAI1436513, /http://digitalcommons.utep.edu/dissertations/AAI1436513, May 2006. (Ph.D. Candidate, UTEP)

Jayaraman Suresh Babu, *Coarse-Grain Dynamic Adaptation for Asynchronous I/O Scheduling: Is it needed?*, Master's Thesis, UTEP, Computer Science, ETD Collection for UTEP, Paper AAI1435308, http://digitalcommons.utep.edu/dissertations/AAI1435308, May 2006. (Cisco)

University of Wisconsin-Madison:

Gregory Cooksey
Grenory Quinn
Igor Grobman
Todd Miller

In addition, as a result of the research being done under the grant, several collaborations developed with:

- IBM-Austin Linux Technology Center, in particular, Bill Buros
- Linux developers, especially Jens Axboe (responsible for device drivers and I/O schedulers)
- Sandia National Laboratories-Albuquerque, in particular, Ron Oldfield and Rolf Riesen
- Los Alamos National Laboratory, in particular, John Daly (now at the DoD Center of Excellence)

3. Accomplishments

The accomplishments during the grant period, reported in Sections 3.1 and 3.2, are partitioned in terms of the two objectives mentioned above, i.e., development of mechanisms to support dynamic adaptation of operating systems (Section 3.1) and demonstration of self-tuning and self-provisioning (Section 3.2). The research endeavors described in Section 3.2 provided us with insights concerning the original goal of the proposed research, i.e., development of a general-purpose methodology for dynamic adaptation of commodity operating systems, which is not feasible because as the methodology depends on the nature of the adaptation target. With respect to this, we learned the following:

- Identifying promising adaptation targets is a challenging and time-intensive task, which includes identifying applications/workloads that will be affected by the targeted adaptation, using static adaptation and a variety of workloads to quantify potential performance gains, and performing a feasibility study of the dynamic adaptation.
- The complexities associated with parametric and policy adaptations differ significantly. The original DAiSES methodology was more directed at parametric adaptation.
- Adaptations can be classified according to whether they are meant to meet an application performance objective or a system performance objective, e.g., one that may require concurrent tuning of multiple applications that share resources. The latter is decidedly more difficult.
- Improved execution-time performance is not the only objective, i.e., in some cases system constraints must be met, e.g., fairness and/or latency.
- Different methodologies/strategies are needed for resources with state and resources without state.
- Control theoretic models should be considered for implementing feedback loops that are integral in adaptation strategies.

In [Te06] we (1) discuss the potential value of and challenges associated with providing dynamic operating systems (OS) policy and parameter adaptation, (2) revisit the initial DAiSES methodology, elucidating the methodology in terms of two rather different adaptation targets, i.e., process scheduling and I/O scheduling, and (3) abstract the challenges associated with dynamic adaptation of stateless systems and provide solutions to these challenges, exemplifying them in the context of I/O schedulers.

3.1. Mechanisms to Support Dynamic Adaptation

3.1.1. Foundation for Scalable Tools: Group file operations are a new, intuitive idiom for scalable tools and middleware - including parallel debuggers and runtimes, performance measurement and steering, and distributed resource management - that require scalable operations on large groups of distributed files. The idiom provides new semantics for using file groups in standard file operations to eliminate costly iteration. A file-based idiom promotes conciseness and portability, and eases adoption. With explicit semantics for aggregation of group results, the idiom addresses a key scalability barrier. We have designed TBON-FS, a new distributed file system that provides scalable group file operations by leveraging tree-based overlay networks (TBONs) for scalable communication and data aggregation.

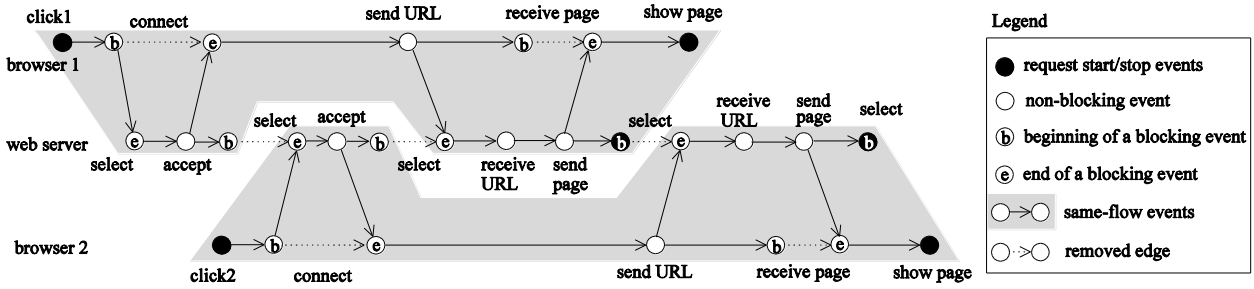
We integrated group file operations into several tools: parallel versions of common utilities including `cp`, `grep`, `rsync`, `tail`, and `top`, and the Ganglia Distributed Monitoring System. Our experience verifies the group file operation idiom is intuitive, easily adopted, and enables a wide variety of tools to run efficiently at scale.

3.1.2. Diagnosis of Large-Scale Distributed Systems: We developed a three-part approach for diagnosing bugs and performance problems in production distributed environments. First, we introduce a novel execution monitoring technique that dynamically injects a fragment of code, the *agent*, into an application process on demand. The agent inserts instrumentation ahead of the control flow within the process and propagates into other processes, following communication events, crossing host boundaries, and collecting a distributed function-level trace of the execution. Second, we developed an algorithm that separates the trace into user-meaningful activities called *flows*. This step simplifies manual examination and enables automated analysis of the trace. Finally, we created an automated *root cause analysis* technique that compares the flows to help the analyst locate an anomalous flow and identify a function in that flow that is a likely cause of the anomaly. We demonstrated the effectiveness of our techniques by diagnosing two complex problems in the Condor distributed scheduling system.

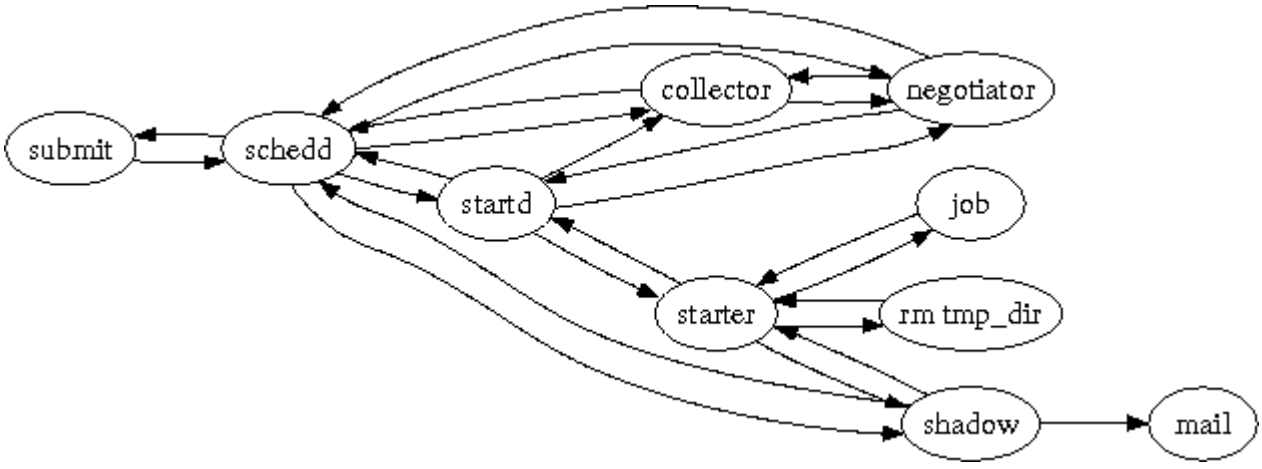
Bugs and performance problems in complex systems are often manifested by deviations of control flow from the common path. To identify such deviations and determine the root cause of a problem, we collect control-flow traces and analyze them manually or automatically. In distributed environments however, collected traces may contain events that correspond to different concurrent activities such as HTTP requests, possibly of multiple users. Events that correspond to one request are not explicitly labeled in the trace and can appear interleaved with events from other requests. Such interleaving of unrelated activities complicates manual trace analysis. It also increases trace variability and thus presents challenges for automated analysis.

Events from different activities may appear in a different order in different runs. As a result, a normal trace may appear substantially different from previous ones and can be marked as an anomaly.

To overcome these problems, we decompose the execution of an application into units that correspond to different semantic activities, referred to as flows. Flows are easier to analyze manually and automatically than the original trace. They contain logically-related events and there is little trace variability within a flow. For example, a Web server process can execute requests from two users in an interleaved order, as shown in the figure below. Flows are shown as disjoint shaded areas in that figure. Unlike previous techniques for flow construction, our approach uses general rules to construct flows but can incorporate application-specific knowledge into analysis when the rules are insufficient.



The figure below shows an example of a diagram of the structure and flow of control in a complex system that was extracted automatically by our execution trace and flow analysis system. The diagram shows the Condor system tracing activity from submission on the user's desktop computer, across the scheduling cluster, including user job execution and email notification of job completion. No knowledge of the system structure was known in advance, and all instrumentation and analysis took place on the executing binary code.



3.1.3. KernInst Ports, Enhancements, and Related Tools: KernInst ports to IA32 for Linux 2.6 and POWER4 for Linux 2.4 were accomplished under this grant, in addition to the testing of the IA32 for Linux 2.4 port on Pentium 3 and Pentium 4 processors. Note that the POWER4 for Linux 2.4 port includes functionality for code analysis and generation, KernInst's application programming interface (API), hardware performance counter access, and the Kperfmon

performance profiling tool. A source of complexity of the POWER4 port was that it must support both kernels that run stand-alone and kernels that run under the Hypervisor virtual machine layer, which is not transparent and requires explicit support. (Note that little public documentation on this layer was available.)

The POWER4 for Linux 2.6 port, which was accomplished under this grant:

- Allows memory allocation for code patches in the region close to kernel code — this allocation technique permits the use a single jump to the patch area (instrumentation). The kernel has no published interface for this operation and, hence, required an intricate design (a redesign from the Linux 2.4 version).
- Improves register analysis robustness.
- Includes TOC (procedure linkage table pointer) recovery routines because this information is no longer provided by the kernel's symbol information.

Enhancements made to KernInst during the grant period include the following:

- An option to Kerninstd that allows it to run as a real-time process with a specified priority. This reduces missing instrumentation samples on a highly loaded machine.
- The ability to use process-id predicated virtual timers in the value-added user timer library for the KerninstAPI.
- Functionality to KerninstAPI that allows for reading a specific hardware counter across all processors in a SMP machine at the same time.
- Functionality to KerninstAPI that allows for process-id predicated function replacement, to support future use of value-specialized replacement functions.

In addition, the following tools were developed:

- KerninstAPI program to measure total process-specific virtual time (processor ticks or hardware counter events) incurred during execution of a specified program with arguments.
- KAPI program to measure total global virtual time (hardware counter events) incurred during execution of a specified program with arguments.
- Preliminary versions of KerninstAPI programs to dynamically read/write the value of function local variables and Linux process task_struct member variables.
- Linux 2.4/2.6 kernel profiler that uses the KerninstAPI and identifies kernel functions invoked on behalf of a specific process by tracing call path execution starting at the system call interface. This profiler extends the implementation of the CrossWalk tool for tracing application performance problems across the user/system boundary. A UNIX dot graph generation utility input file is generated as output, allowing for the user to generate a graphical representation of the observed control flow. The profiler was further extended to allow for collection of execution counts of edges in the call graph in order to help identify "hot" functions and paths.
- KerninstAPI program to measure the total time spent in a kernel function on behalf of a specific process.

3.1.4. Dynamic Code Optimization: We investigated dynamic code optimization strategies. In the past, dynamic reorganization of the basic block layout was successfully used in parts of the Solaris kernel on SPARC platforms to improve performance by reducing I-cache misses

[Ta01b]. There was discussion in the research community as to whether such analyses are workload dependent and needed to be done dynamically, or are mostly independent of workload and can be done statically. There were no studies that provided conclusive evidence either way. We performed a general survey of code optimization techniques as candidates for use in kernel-level code optimization and identified value-specialization of kernel function arguments as a possible target optimization. Value specialization works by observing that one or more function argument values are the same across the majority of calls made to the function, and then generating an optimized version of the function that assumes these values. Given a number of "constant" arguments, optimization techniques can be applied such as constant and copy propagation, load/store forwarding through registers and the stack, dead-code elimination, branch inversion, and function in-lining. These techniques may be applied recursively to functions called by the optimized function if their arguments also become constant. Value-specialization is valuable when the benefit from specialization exceeds the cost to check that the arguments match the values used to specialize.

The KerninstAPI program that measures the total time spent in a kernel function on behalf of a specific process (described in the previous section) was used to identify the maximum benefit achievable if value-specialization could completely optimize-away functions. For all of the profiled applications, none of the "hot" kernel functions accounted for more than 5% of total application CPU time. Since the 5% represents the maximum achievable benefit, and the expected benefit from value-specialization is much lower, no further work was done to collect parameter value profiles.

3.1.5. File System Optimization: We performed a small study on the possibility of file system optimizations with a genetics application called *blast*. It turns out that the file cache miss rate is small enough to render possible optimizations virtually negligible.

3.1.6. OS Profiling: The profiler described in Section 3.1.3 was used to analyze the call graphs of three representative HPC applications and identify "hot" paths and functions. The first application studied was MADBench, a lightweight version of the MADCAP CMB power spectrum estimation code implemented using MPI. This code was compiled with the '-DIO' option, which eliminates the computational portions of the benchmark and allows observation of kernel processing due to I/O. The second application studied was from the MILC Collaboration software collection Version 6 for simulations of SU3 lattice gauge theory on MIMD parallel machines. The specific application was pure gauge su3_rmd, compiled using the default Linux with MPI configuration. The third application studied was OM3, a full-blown global ocean general circulation model from the University of Wisconsin Atmospheric Sciences Department and implemented using MPI.

3.2. Demonstration of Self-tuning and Self-provisioning

During the grant period, we made progress towards dynamic adaptation of (1) single-disk and RAID I/O scheduling (which includes a pending patent, publications, a Ph.D. dissertation, and a Master's thesis), (2) checkpoint/restart (which includes publications and the development of collaborations with technical staff at Sandia National Laboratories, Los Alamos National Laboratory, and the DoD Center of Excellence, (3) virtual memory management, in particular the

dynamic adaptation of a parameter that controls the size of SCM (Swap Cluster Max) (this includes two Master's theses), and (4) process and SMT hardware thread scheduling (this includes a Ph.D. dissertation). The related work is summarized below.

3.2.1. I/O Scheduler Optimization: The most significant accomplishment of our work in I/O scheduling is a proof of concept of dynamic policy adaptation. This work focuses on the fairness (proportional sharing) of I/O scheduling algorithms, where fairness is measured by a new metric, i.e., disk time usage [Se06a]. In previous related work, fairness was measured in terms of number of requests or bytes and, as a result, true proportional sharing or differentiated I/O service was not achieved.

This is an important problem because storage consolidation has become a popular trend for reasons of economy of scale and reduction of storage management complexity. The sharing of consolidated storage among various workloads is facilitated by storage virtualization. In such an environment, competing workloads are likely to have different service requirements and, therefore, it is important to be able to provide differentiated service, i.e., each application receives an I/O service that is proportional to the priority of the application class to which it belongs. Differentiated service and performance isolation together provide performance virtualization, i.e., each application class experiences I/O performance similar to what it would experience with a dedicated storage utility of a certain fixed capacity.

Performance virtualization may be provided by scheduling algorithms. In the last decade, several sophisticated I/O scheduling algorithms that enable differentiated service (proportional sharing) and performance isolation have been designed. However, most of these algorithms facilitate such sharing of storage resources with respect to number of requests, which translates to proportional sharing with respect to number of bytes when all requests are assumed to be of equal size. Our approach is to provide differentiated service and facilitate proportional sharing irrespective of request characteristics, i.e., using disk time as the metric of fairness.

We designed two novel scheduling algorithms, CFQ-CRR (Complete Fair Queuing with Compensated Round Robin) and CFQ-CRR(p) (Complete Fair Queuing with Compensated Round Robin with device queuing), which demonstrate the effectiveness of this new notion of fairness in single-disk storage systems. Although, these scheduling algorithms were designed primarily for ensuring the fairness of a storage system comprised of a single disk, the underlying concept, as presented in the related patent [Se08] and dissertation [Se06a], could be applied to more complex I/O systems. This work, in its various stages of development, is described in six publications at various venues [Se05a, Se05b, Se06b, Se06c, Se07a] and is supported by a Master's thesis [Ba06].

Currently, except when the storage utility is a single disk, there are no scheduling algorithms that provide proportional sharing with respect to service time. Thus, we have been working (both towards the end of the grant period and after) towards the application of this concept of fairness to the design of an I/O scheduling algorithm for a RAID storage system [Ar07, Ar09a]. It will provide best effort guarantees for (a) differentiated service based on the priority of the request class, (b) good device throughput, and (c) latency requirement of request classes. In addition, the scheduling algorithm is designed to dynamically adapt to changes in request characteristics and number of application classes. In the future we plan to work towards an algorithm to provide

differentiated I/O service at the file system level [Ar07]. Although this research is of value in itself, it could be used to improve system throughput in the presence of multiple concurrent checkpointing applications, described in Section 3.2.5.

3.2.2. File I/O Optimization: To increase the scale and performance of high-performance computing (HPC) applications, it is common to distribute computation across multiple processors. Often without realizing it, file I/O is parallelized with the computation. An implication of this is that multiple compute tasks are likely to concurrently access the I/O nodes of an HPC system. When a large number of I/O streams concurrently access an I/O node, I/O performance tends to degrade, impacting application execution time. In [Se07b], through experimental results, we show that *controlling* the number of file-I/O streams that concurrently access an I/O node can enhance application performance. We call this mechanism file-I/O stream throttling. The paper (1) describes this mechanism and demonstrates how it can be implemented either at the application or system software layers, and (2) presents results of experiments driven by the cosmology application benchmark MADbench, executed on a variety of computing systems, that demonstrate the effectiveness of file-I/O stream throttling. The I/O pattern of MADbench resembles that of a large class of HPC applications.

3.2.3. Process Scheduler Optimization: Inspired by [Bo03, Lo04a, Mo02], it was noted that the general process scheduling policy has an effect on non-real-time, non-interactive applications—a description that fits the high-performance applications targeted by DAiSES research. Accordingly, we analyzed the Linux 2.4 process scheduler and identified two orthogonal heuristics that could be used to make it adapt to process loads. Using the Hackbench benchmark [Cr04], we demonstrated that heuristic-driven dynamic adaptation of the Linux 2.4 process scheduler can enhance application performance up to 160%. This adaptation is independently driven by heuristic thresholds that monitor the number of processes eligible to run and the execution time of the process scheduler. Note, however, that the process scheduler was significantly changed in the Linux 2.6 kernel and no longer suffers the performance problems exhibited by our experimentation. This research was facilitated by the development of KernInst-based tools.

The introduction of the Linux 2.6 kernel together with a pluggable process scheduler framework [Pl05] made it possible to perform experiments to compare the performance impact of various process schedulers on a set of benchmarks. Test workloads associated with these benchmarks allow measurement of process scheduler effects on latency and time to completion of applications that are highly interactive, heavily threaded, or scientific with heavy I/O. A technical report [Ro06] documents the comparison of the performance of the schedulers.

This work was not continued because we felt that the workload characteristics that map well to a process scheduler may be too high level to differentiate.

3.2.4. Virtual Memory Management Optimization: After identifying and analyzing the most influential parameters that modify the behavior of the Linux Virtual Memory Manager (VMM) in a parallel environment, we focused our attention on one parameter, in particular, SCM or Swap Cluster Max, and analyzed its effect on workload performance. First, we developed a methodology that can be used to quantify application performance gains or losses attributable to

changing the value of a VMM parameter [Po06]; this methodology uses RAMDISK to artificially induce memory pressure. Next, using this methodology and building upon the work of Kandiraju [Ka04], who designated SCM as a potential candidate for dynamic adaptation, we conducted a more thorough study of this parameter and concluded that SCM, indeed, is a good candidate for dynamic adaptation [Po06].

Building upon this work, [So08] built a workload generator that can simulate phases of a class of applications with simple memory-access patterns, which are characterized by their Page Fault Rates (PFRs). The workload generator was used to refine the original methodology by creating natural memory pressure, rather than artificial pressure induced by a RAMDISK. Using the workload generator, experiments were conducted that demonstrate a correlation between SCM and PFR, and show that the PFR can be used as a basis for a dynamic adaptation heuristic for SCM.

3.2.5. SMT Thread Scheduling Optimization: The work described in this section was started during the grant period. It constitutes the first step towards dynamic adaptation of hardware thread priorities.

Simultaneous multithreading (SMT) allows multiple applications to execute concurrently on a processor core, potentially increasing the utilization and throughput of the processor by a factor of the degree of multithreading. However, such performance gains may not be achieved due to contention for resources shared by the applications executing in SMT mode. The IBM POWER5, which has two hardware threads per core, provides software-controlled hardware thread priorities that facilitate the control of the ratio of decode cycles allocated to the two hardware threads of a core and, therefore, the degree of resource contention between the threads. By default, the priorities of the two hardware threads are set to equal values – each gets half of the decode cycles. Several studies have shown that many scientific applications do not achieve their best throughput at equal priorities. The best priority settings or best priority pair, i.e., the priority settings for a given pair of application threads that give the best throughput, depend on the characteristics of the co-scheduled application threads.

During the grant period, we began the development of a methodology for statically predicting the best priority pair for a given co-schedule of serial applications [Me09a, Me09b, Me09c, Tr09, Me06]. Our approach exploits resource-utilization information collected during an application's single-threaded execution. The utilization information provides insights about the availability of resources that are shared in SMT mode and, thus, available for the co-scheduled application's use. In [Me09] we: (1) Demonstrate that the setting of hardware thread priorities can be used to improve SMT throughput. Using a POWER5 simulator, we show that equal priorities (default) are not best for 82% of application trace-pairs simulated. (2) Introduce the concept of a Shareable Resource Signature, which characterizes an application's utilization of critical processor resources, during a specified time interval, when executed in single-threaded mode. (3) Present a methodology to predict the best priority pair for co-scheduled applications, each with only one signature. (4) Describe the implementation of the methodology on the IBM POWER5 processor, which shows that (a) 17 out of 10,000 signatures are sufficient to represent 95.6% of the execution time of 20 SPEC CPU2006 and 3 NAS NPB3.2 serial benchmarks (3 data inputs), and 10 PETSc KSP solvers (12 data inputs). The 10 PETSc KSP solvers and one of the NAS

NPB benchmarks (bt-mz) have signatures that are independent of input data. (b) For 19 out of the 21 application co-schedules studied, the predicted best priority pairs yield throughput that is greater than or equal to that attained with the default priority pair.

3.2.6. Checkpoint/Restart Optimization: Our research in the area of checkpoint/restart [Ol07, Ar09] began under this grant. Through analytical modeling and simulation, in [Ol07] we demonstrated that for next-generation, capability-class, massively parallel processing systems, application-driven, periodic-checkpointing does not scale. In addition, we adapted the model to investigate a proposed optimization that makes use of lightweight storage architectures and overlay networks to overcome the storage system bottleneck. Our results indicate that (1) as we approach the scale of next-generation systems, traditional checkpoint/restart approaches will increasingly impact application performance, accounting for over 50% of total application execution time; (2) although our alternative approach improves performance, it has limitations of its own; and (3) there is a critical need for new approaches to fault tolerance that allow continuous computing with minimal impact on application scalability.

Further analytic modeling [Ar09] has provided information that can potentially be used to coordinate the I/O operations of concurrently executing applications. Our results indicate that there may be reasonable tradeoffs that can be made w.r.t. execution time and number of checkpoint I/O operations. Application execution time corresponding to checkpoint intervals greater than, and in the vicinity of, the value that minimizes the execution time increase slowly while the number of checkpoint I/O operations decreases drastically. This, certainly, seems to be true for the illustrative Blue Gene /L example. Increasing the checkpoint interval from 9.8 minutes, which is the value that minimizes the execution time, to a value of 67 minutes leads to an increase of execution time by a mere 5%, whereas, the number of checkpoint I/O operations reduces steeply from a value of 3121 to a value of 507 at 67 minutes. This is an 83.78% reduction in the number of checkpoint I/O operations.

4. Patents

[Se08] S. Seelam and P.J. Teller, "Method and Devices for Determining Quality of Services of Storage Systems", see http://www.faqs.org/patents/app/20080244209 , Provisional Patent, March 2007, Patent Published October 2008.

5. Publications (in chronological order, most recent first)

M.J. Brim and B.P. Miller, "Group File Operations for Scalable Tools and Middleware", Under submission.

M.R. Meswani, P.J. Teller, and S. Arunagiri, "On the Use of Shareable Resource Signature Characterization and Hardware Thread Priorities to Improve Simultaneous Multithreaded (SMT) Processor Throughput," In preparation.

[Ar09b] Arunagiri, J.T. Daly, and P.J. Teller, "Modeling and Analysis of Checkpoint I/O Operations," to appear in Proceedings of the 16th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA 2009), Madrid, Spain June 9-12, 2009.

[Me09c] M.R. Meswani, *Improving Throughput of Simultaneous Multithreaded (SMT) Processors using Shareable Resource Signatures and Hardware Thread Priorities*, Ph.D. Dissertation, University of Texas at El Paso, Computer Science, Anticipated May 2009.

[Me09b] M.R. Meswani, P.J. Teller, and S. Arunagiri, "Improved Throughput of Simultaneous Multithreaded (SMT) Processors using SMT Thread Signatures and Hardware Thread Priorities,", Poster at the 10th LCI International Conference on High-Performance Clustered Computing (LCI'09), Boulder, CO, March 2009.

[Ar09a] S. Arunagiri, Y. Kwok, P.J. Teller, and S. Seelam, "An I/O scheduling Algorithm for Storage Performance Virtualization," Poster at the 10th LCI International Conference on High-Performance Clustered Computing (LCI'09), Boulder, CO, March 2009.

[Me09a] M.R. Meswani, P. J. Teller, and S. Arunagiri, "Measuring and Validating Metrics used to Estimate Microarchitecture Resource Utilization: A Case Study of the IBM POWER5 Processor", *Proceedings of the 2008 Live-Virtual-Constructive Conference* (formerly known as the ITEA Modeling and Simulation Conference), hosted by the White Sands Chapter of the International Test and Evaluation Association, January 12-15, 2009, El Paso, TX.

[Tr09] P.C. Trillo, M. R. Meswani, P. J. Teller, and S. Arunagiri, "A Study of the Influence of the POWER5 Dynamic Resource Balancing Hardware on Optimal Hardware Thread Priorities", *Proceedings of the 2008 Live-Virtual-Constructive Conference* (formerly known as the ITEA Modeling and Simulation Conference), hosted by the White Sands Chapter of the International Test and Evaluation Association, January 12-15, 2009, El Paso, TX. *The paper was selected as the ITEA Best Undergraduate Student Paper and the lead author, Princess Trillo, presented the paper at the conference and received a $1,000 scholarship award at the conference's awards luncheon.*

A.V. Mirgorodskiy and B.P. Miller, "Diagnosing Distributed Systems with Self-Propelled Instrumentation", *ACM/IFIP/USENIX 9th International Middleware Symposium*, Leuven, Belgium, LNCS 5346, December 2008.

[So08] Muthuveer Somanathan, *An Initial Operating System Adaptation Heuristic for Swap Cluster Max (SCM)*, Master's Thesis, UTEP, Computer Science, ETD Collection for UTEP, http://digitalcommons.utep.edu/dissertations/AAI1456754, May 2008.

[Ol07]R.A. Oldfield, S. Arunagiri, P.J. Teller, S. Seelam, M.R. Varela, R. Riesen, and P.C. Roth, "Modeling the Impact of Checkpoints on Next-Generation Systems", *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies, 2007 (MSST 2007)*, San Diego, CA, pp. 30-46, September 2007.

[Se07b] S. Seelam, A. Kerstens, and P.J. Teller, "Throttling I/O Streams to Accelerate File-I/O Performance", *Proceedings of the Third IEEE International Conference on High Performance Computing and Communications (HPCC-2007)*, LNCS 4782, pp. 718-731, September 2007.

[Se07a] S. Seelam and P.J. Teller, "Virtual I/O Scheduler: A Scheduler of Schedulers for Performance Virtualization", *Proceedings of the 3rd International ACM/SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, San Diego, CA, pp. 105-115, June 2007.

[Ar07] S. Arunagiri, S. Seelam, and P.J. Teller, "Storage Performance Isolation: An Investigation of Contemporary I/O Schedulers", *Abstract in Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*, San Jose, CA, pp. 15, February 2007.

A.V. Mirgorodskiy, N. Maruyama and B.P. Miller, "Problem Diagnosis in Large-Scale Computing Environments", *Proceedings of the 2006 International Conference for High Performance Computing, Networking, Storage, and Analysis (SC2006)*, Tampa, FLA, November 2006.

[Se06c] S. Seelam and P.J. Teller, "Fairness and Performance Isolation: an Analysis of Disk Scheduling Algorithms", *Proceedings of the International Workshop on High Performance I/O Techniques and Deployment of Very Large Scale I/O Systems (HiperIO'06)*, Barcelona, Spain, September 25-28, 2006.

[Se06b] S. Seelam, J.S. Babu, and P.J. Teller, "Rate-Controlled Scheduling of Expired Writes for Volatile Caches", P*roceedings of the 3rd International Conference on Quantitative Evaluation of SysTems (QEST'06)*, University of California, Riverside, CA, September 11-14, 2006.

[Me06] M. Meswani and P.J. Teller, "Evaluating the Performance Impact of Hardware Thread Priorities in Simultaneous Multithreaded Processors using SPEC CPU2000", *Proceedings of the Second International Workshop on Operating System Interference in High Performance Applications (OSIHPA)*, Seattle, WA, September 2006.

[Se06a] S. Seelam, *Towards Dynamic I/O Scheduling in Commodity Operating Systems*, Ph.D. Dissertation, University of Texas at El Paso, Computer Engineering, May 2006.

[Po06] Ricardo A. Portillo, *Fine-grain Dynamic Adaptation of the Linux 2.6 Virtual Memory Manager: A First Step*, Master's Thesis, UTEP, Computer Science, ETD Collection for UTEP, PaperAAI1436513, /http://digitalcommons.utep.edu/dissertations/AAI1436513, May 2006.

[Ba06] J. Suresh Babu, *Coarse-Grain Dynamic Adaptation for Asynchronous I/O Scheduling: Is it needed?*, Master's Thesis, University of Texas at El Paso, Computer Science, May 2006.

[Te06] P. Teller and S. Seelam, "Insights into Providing Dynamic Adaptation of Operating System Policies", *ACM Operating Systems Review*, April 2006.

[Ro06] R. Romero, S. Seelam, and P. Teller, "Workload Dependent Performance Analysis of Process Schedulers: A Case Study," Technical Report, College of Engineering – Computer Science, University of Texas at El Paso, El Paso, TX, January 2006.

[Se05b] S. Seelam, J. Suresh, and P.J. Teller, "Automatic I/O Scheduler Selection for Latency and Bandwidth Optimization", *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques PACT2005, First Workshop on Operating System Interference in High Performance Applications (OSIHPA)*, St. Louis, MO, September 2005.

[Se05a] S. Seelam, R. Romero, P.J. Teller, W. Buros, "Enhancements to Linux I/O Scheduling", *Proceedings of the 2005 Linux Symposium*, Ottawa, Canada, July 20-23, 2005.

6. Presentations (other than presentations of papers and posters)

University of Texas at El Paso, Computer Science

P.J. Teller, Invited talk, Los Alamos Computer Science Institute (LACSI), Santa Fe, NM, "Towards the Incorporation of Dynamic Adaptation into Operating Systems: Adaptive Disk I/O", October 17, 2006.

P.J. Teller, Invited talk, Oak Ridge National Laboratories, "Towards Dynamic Adaptivity of Operating Systems", September 2006.

P.J. Teller, Invited talk, Sandia National Laboratories-Albuquerque, "Towards Dynamic Adaptivity of Operating Systems", August 28, 2006.

P.J. Teller, Invited talk, Lawrence Berkeley National Laboratory, "Towards Dynamic Adaptivity of Operating Systems", July 28, 2006.

P.J. Teller, Invited talk, Lawrence Livermore National Laboratory, "Towards Dynamic Adaptivity of Operating Systems", July 26, 2006.

P.J. Teller, Invited talk, FastOS Workshop in conjunction with USENIX '06, Boston, MA, "Towards Dynamic Adaptivity of Operating Systems", May 30-31, 2006.

S. Seelam, Invited talk, IBM Petascale Tools Strategy Workshop, T. J. Watson Research Center, Yorktown Heights, NY, "Performance Tools – Thoughts and Experiences", May 2005.

P.J. Teller, Invited talk, the Conference on High Speed Computing – Paths to Petaflops (The Salishan Conference), sponsored by Lawrence Livermore, Los Alamos, and Sandia National Laboratories, Gleneden Beach, Oregon, "Challenges in Scalability of Performance Tools for Petaflops Systems," April 2005.

University of Wisconsin-Madison, Computer Sciences

B.P. Miller, Distinguished Lecturer, College of Computing, Georgia Institute of Technology, Atlanta, April 2007.

B.P. Miller, Distinguished Lecturer, Triangle Computer Science Lecture Series (joint University of North Carolina, North Carolina State and Duke University), Raleigh, October 2007.

B.P. Miller, Invited talk, Los Alamos National Lab, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", University of Tennessee, September 2007.

B.P. Miller, Invited talk, Oak Ridge National Lab, "Tree-based Overlay Networks for Scalable Middleware and Systems", September 2007.

B.P. Miller, Invited Talk, Los Alamos National Lab, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", Los Alamos, NM, January 2007.

B.P. Miller, Invited Talk, University of New Mexico, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", Albuquerque, January 2007.

B.P. Miller, Invited Talk, Sandia National Lab, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", Albuquerque, August 2006.

B.P. Miller, Invited Talk, University of California, San Diego, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", January 2006.

B.P. Miller, Invited Talk, University of California, Santa Barbara, "A Framework for Binary Code Analysis and Static and Dynamic Code Patching", January 2006.

7. References

[Bo03] D. Bovet and Cesati, M., *Understanding the Linux Kernel*, 2nd Edition. O'Reilly, 2003.

[Cr04] T. Craiger, "Linux Process Scheduler Improvements in Version 2.6.0," *http://developer.osdl.org/craiger/hackbench*.

[Ka04] G.B. Kandiraju, *Towards Self-Optimizing Memory Management*, Ph.D. Dissertation, Pennsylvania State University, AAI3141006, 2004.

[Lo04a] R. Love, *Linux Kernel Development*, Sams Publishing, 2004.

[Mo02] "Interview: Ingo Molnar," *http://kerneltrap.org/node/517*, December 2002.

[Pl05] CPU Scheduler Evaluation Project, http://sourceforge.net/projects/cpuse/PlugSched

[Ta01b] A. Tamches and B. Miller, "Dynamic Kernel I-Cache Optimization", *Proceedings of the Workshop on Binary Translation*, Barcelona, Spain, September 2001.

[Te04] P. Teller and B. Miller, *DAiSES: Dynamic Adaptability in Support of Extreme Scale*, DOE Research Proposal, May 2004.

[Wm04] Paradyn Parallel Performance Tools, http://www.paradyn.org/html/manuals.html, 2004.