HPC Colony II
Award No. ERKJT17
Consolidated Annual Report
July-2010 to June-2011

Terry Jones[1], Laxmikant Kalé[2], Gennady Laventman[3], Benjamin Mandler[3]
Celso Mendes[2], Esteban Meneses[2], José Moreira[3], Yoav Tock[3], Lukasz Wesolowski[2]

[1]Oak Ridge National Lab
Mailstop 5164
Oak Ridge, TN 37831

[2]University of Illinois
201 N. Goodwin Avenue
Urbana, IL 61801

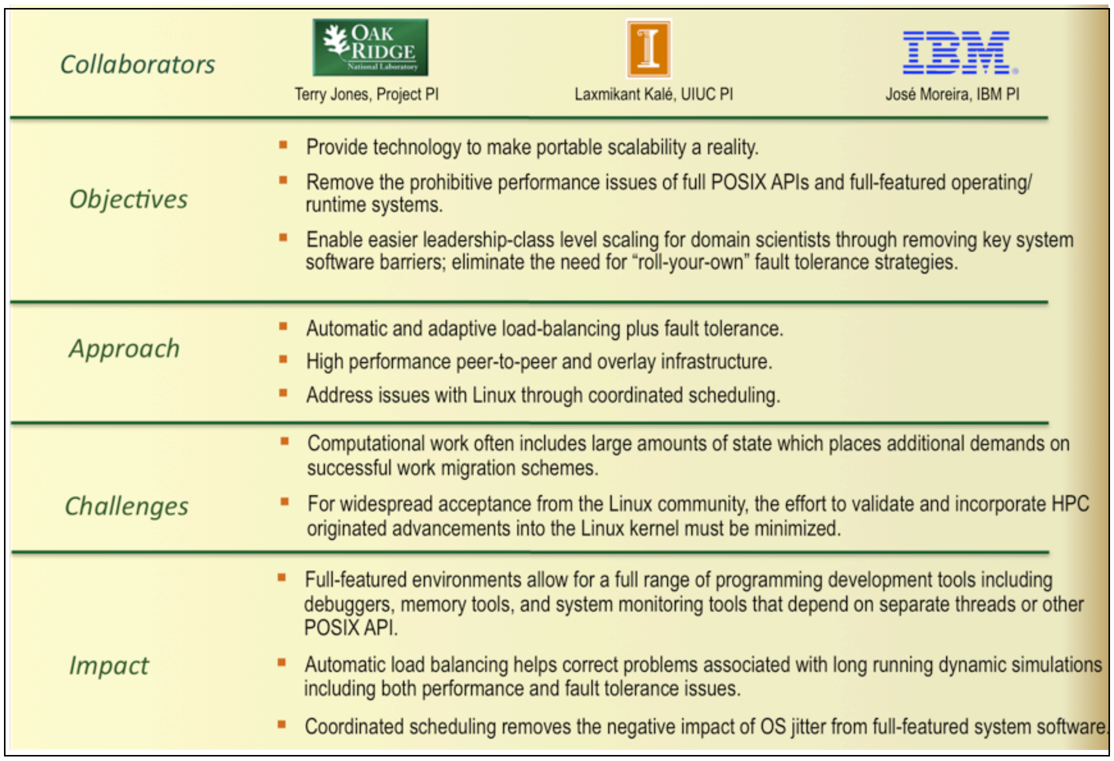
[3]International Business Machines
1101 Kitchawan Rd
Yorktown Heights NY 10598

Submitted by: **Terry Jones, PI**
June 10, 2011

Summary
This report provides a brief progress synopsis of the HPC Colony II project for the period of July 2010 to June 2011. HPC Colony II is a 36-month project and this report covers project months 10 through 21. It includes a consolidated view of all partners (Oak Ridge National Laboratory, IBM, and the University of Illinois at Urbana-Champaign) as well as detail for Oak Ridge. Highlights are noted and fund status data (burn rates) are provided.

1 Background

Colony II is a three-year Computer Science oriented project funded from the Fast-OS Re-compete program. The primary goal for Colony II is to enable portable performance on leadership-class machines—a task made difficult by the emerging variety of more complex computer architectures. Colony II attempts to move the burden of portable performance to system software, thereby allowing domain scientists to concentrate on their field rather than the fine details of a new leadership class machine. An overview matrix is provided in Table 1.



Collaborators	Terry Jones, Project PI	Laxmikant Kalé, UIUC PI	José Moreira, IBM PI
Objectives	▪ Provide technology to make portable scalability a reality. ▪ Remove the prohibitive performance issues of full POSIX APIs and full-featured operating/ runtime systems. ▪ Enable easier leadership-class level scaling for domain scientists through removing key system software barriers; eliminate the need for "roll-your-own" fault tolerance strategies.		
Approach	▪ Automatic and adaptive load-balancing plus fault tolerance. ▪ High performance peer-to-peer and overlay infrastructure. ▪ Address issues with Linux through coordinated scheduling.		
Challenges	▪ Computational work often includes large amounts of state which places additional demands on successful work migration schemes. ▪ For widespread acceptance from the Linux community, the effort to validate and incorporate HPC originated advancements into the Linux kernel must be minimized.		
Impact	▪ Full-featured environments allow for a full range of programming development tools including debuggers, memory tools, and system monitoring tools that depend on separate threads or other POSIX API. ▪ Automatic load balancing helps correct problems associated with long running dynamic simulations including both performance and fault tolerance issues. ▪ Coordinated scheduling removes the negative impact of OS jitter from full-featured system software.		

Table 1: Overview Matrix for Colony II

2 Project Progress over Last 12 Months (months 10 through 21)

To help realize our project goal of ***portable performance***, the HPC Colony II project is focused on six interrelated topic areas which utilize adaptive technology to make portable scalability much more feasible. Our six topic areas are:

- Reduce performance consequences from fault tolerance
- Provide scalable membership, monitoring, & communication services
- Investigate innovative ways to provide dynamic load balancing
- Improve the resource management interface between center batch scheduling & on-node system software
- Enable broad application sets on most capable machines
- Enable Linux kernel advances for extreme scale systems

2.1 Kernel & Kernel Support Advances (Colony lead: ORNL)

During this period, we developed the first co-scheduling Linux kernel designed for High Performance Computing. Results were obtained on ORNL's Jaguar XT5 system; first with the normal scheduling Linux 2.6.16.60 operating system, then with the coordinated scheduling Linux 2.6.16.60 operating system described above. Results for normal Linux scheduling showed noticeable variability from test to test at scales of 10K cores. This was expected and coincides well with results obtained from Hoefler et al. [Hoefler10] and Sottile et al. [Sottile04], as well as our previous work. Performance measurements were then obtained using the coordinated-scheduling policy and modified operating system. It was immediately clear for this workload that the coordinated scheduling provided a significant performance improvement, both in terms of average execution time and in terms of variability between runs. Finally, an additional set of performance numbers with the normal operating system were measured. The last set of normal operating system results closely matched the set of results obtained before the co-scheduled kernel results taken in the middle.

The left half of Figure 1 depicts the normal Linux results while the right half of Figure 1 depicts the co-scheduled results. As described earlier, the benchmark employed for this testing results in a single number corresponding to a unit of execution time, the lower the better. The graphs indicate shorter durations (better performance) for the co-scheduled kernel.

The best observed time from all experiments was 0.44. The average execution time for the co-scheduled kernel was **0.56**, which compares to **1.60** with the Normal Scheduled kernel. An improvement of 285%. Moreover, the variability was *much improved* with the co-scheduled kernel. The standard deviation for the Normal Scheduled samples was **5.32**; this compares to **0.20** for the co-scheduled kernel.

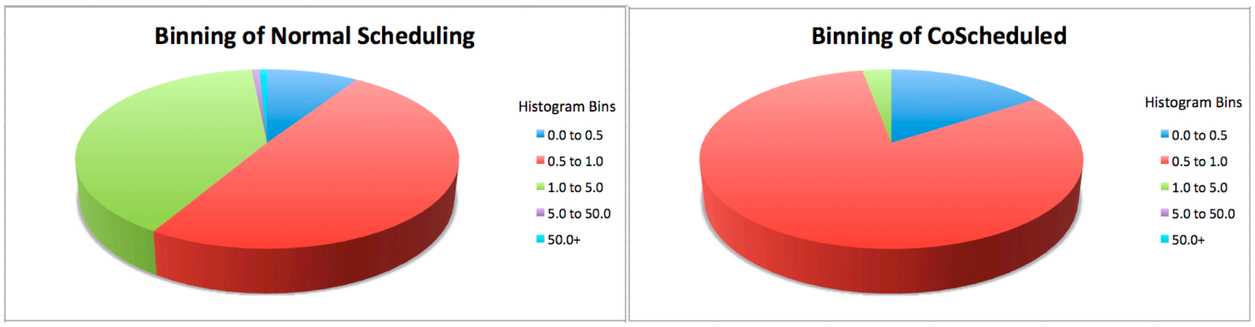


Figure 1. Coordinated and uncoordinated schedulings. The above figure portrays histogram bins in a pie-chart to provide an indication of the relative timing of runs. The top chart gives results without scheduling, and bottom chart gives results for coordinated scheduling.

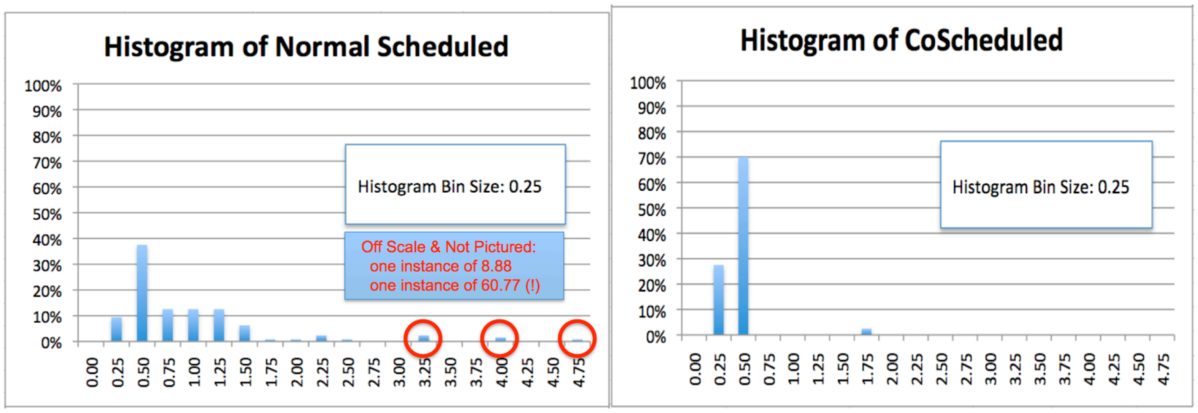


Figure 2. Coordinated and uncoordinated schedulings. The above figure portrays a histogram of runs with and without coordinated scheduling. The lower histogram includes coordinated scheduling.

With a standard deviation larger than the average, it is clear that the samples do not follow a Gaussian distribution. In fact, the distribution of samples for the co-scheduled kernel has a very prominent peak near the average measurement, and a short tail of longer times. However, the distribution for the Normal Scheduled kernel has a much broader peak and a very long tail of outlier samples with much longer times. These results can be seen in Figure 2. In the left histogram, the worse performing outliers are circled in red, and the two most are off the charts at 8.88 and 60.77. This variability is in stark contrast to the co-scheduled results in the right histogram of Figure 2.

In context, the 285% speedup is good news for that class of applications impacted by synchronizing collectives, but it should be noted that overall application performance will depend upon many factors beyond synchronizing collective performance. Yet the 30% overall application slowdown reported by Nataraj et al. [Nataraj07] and Ferreira et al. [Ferreira08] indicates a significant amount of speedup may be realized by an entire application when noise effects are minimized.

2.2 Fault Tolerance Advances (Colony lead: Univ. of Illinois)

Also during this period, we continued our fault tolerance research by investigating a more advanced form of the message-logging scheme that we had studied initially. The new technique is based on a variant of the *causal* message logging protocol that seems to be a promising alternative to provide fault tolerance in large supercomputers. This study was conducted over three phases: first, we analyzed various scenarios that make pessimistic (i.e. conservative) message logging compromise the performance in order to keep consistency in an execution; next, we did a performance comparison of pessimistic and causal approaches for message logging with different applications; then we conducted a performance evaluation of the simple causal message logging protocol for applications that scale up to 1024 processors.

In contrast to pessimistic message logging, this new causal approach has low latency overhead, especially in collective communication operations. Besides, it reduces the number of messages when more than one thread is running per processor. In our tests, we demonstrated that a simple causal message logging protocol has a faster recovery and a low performance penalty when compared to checkpoint/restart.

As an example of the performance achieved with the new approach, Figure 3 shows the execution time observed on the NAS Benchmarks running on 1,024 processors of NCSA's Abe cluster. This figure shows the times for executions in the forward path, i.e. under no faults. The causal scheme presents significantly lower overhead than pessimistic message logging, and achieves performance very similar to the traditional checkpoint-restart case.

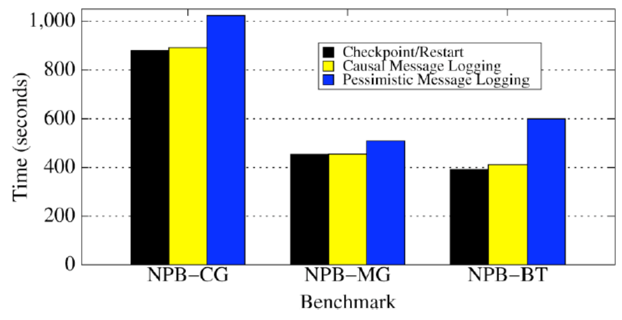


Figure 3. Results with NAS Benchmarks running on 1,024 processors of NCSA's Abe cluster

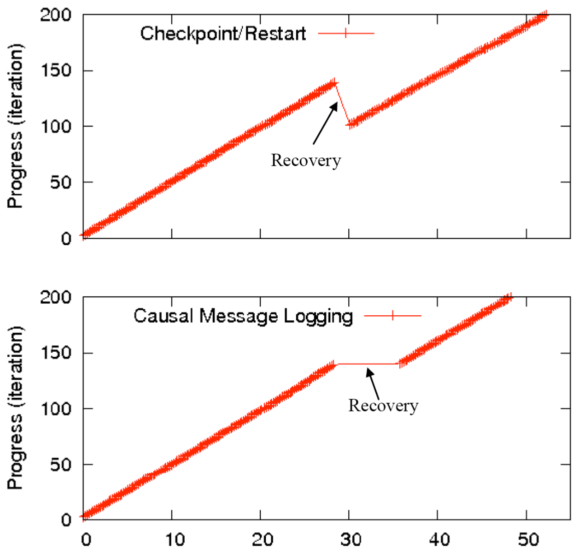


Figure 4. Effect of a failure on execution of a 7-point stencil.

To better evaluate the new approach when failures occur, we employed a 7-point stencil, and forced the recovery to happen after an external failure was introduced in the execution. The code executes 200 iterations, and we introduced a failure at iteration 140. Figure 4 shows the performance under the causal message logging protocol and under checkpoint-restart; a checkpoint was taken at iteration 100. The figure plots the application progress, in terms of completed iterations, as a function of elapsed time. In the checkpoint-restart case, the work of a few iterations (i.e. 100 to 140) needs to be redone when the failure occurs; meanwhile, with causal message-logging, only the failing processor requires its work to be repeated, and other processors that do not depend on it can proceed. Hence, the interruption is less severe, and the overall execution is allowed to complete faster than in the checkpoint-restart case. Notice also the significant energy savings of the causal message logging protocol over checkpoint-restart, as only a few processors are affected by the occurrence of the failure and its recovery.

In summary, our evaluations so far identified multiple performance problems of pessimistic message logging and showed that causal message logging has better performance and scalability for all the programs we ran in our experiments. Full results of these studies were reported in [Meneses2011]. There are, however, remaining challenges for causal message logging. Specifically, it imposes a higher latency on communication, which can be a problem for strong scaling and collective operations, and it requires a modest amount of additional memory to store determinants, when compared to executions without any fault tolerance provision. As we proceed in our research, we are addressing these issues and exploiting ways to alleviate them on large scale systems.

2.3 Scalable Load Balancing (Colony lead: Univ. of Illinois)

During this reporting period, we significantly expanded our research on dynamic load-balancing techniques. First, we consolidated our studies of applying a hierarchical load-balancing scheme that we had developed in the previous year; those studies were reported in [Zheng2010]. Using this hierarchical load balancer more recently, combined with optimizations added to the SMP version of Charm++, we were able to scale the NAMD molecular simulator to the entire extent of Jaguar, a Cray XT5 at ORNL, running on 224,000 processors. Part of the obtained results, which we reported in [Mei2011], is shown in Figure 5, corresponding to NAMD's performance under different configurations on Jaguar with a 100 million-atom dataset. As shown, scaling is excellent for the no-cutoff case.

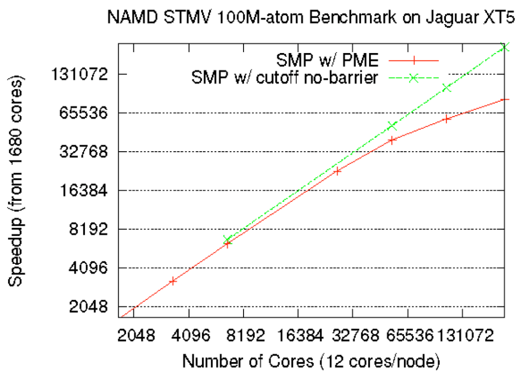


Figure 5. Scaling of NAMD on ORNL's Jaguar under different configurations.

The areas of weather and climate prediction pose a hard challenge for the efficient use of large systems. One of the major factors limiting performance of forecasting models in current machines is load imbalance. Besides the static causes of such imbalance, such as topography, there are dynamic factors that may affect a weather simulation, like the movement of clouds and of thunderstorms. Due to that imbalance, scalability of those models suffers when they are executed on a large number of processors.

We investigated the use of our Adaptive MPI (AMPI), an implementation of the MPI standard based on Charm++, on BRAMS, an existing production-level weather forecasting model. BRAMS is written in Fortran90 and uses MPI for parallelization. As an example, Figure 6 shows the result of a real BRAMS forecast and the corresponding load observed on the 64 processors executing that forecast. The color coding scheme represents rain intensity, in the forecast, and processor load, in the grid of processors. As one can see, there is a big and clear correlation between more rain and higher computational load.

We conducted several tests with BRAMS on Kraken, a Cray XT5 at ORNL. In those tests, we assessed the effects of virtualization and of load balancing on BRAMS executions. Our first observation was that simple AMPI virtualization already improved BRAMS performance. This was due to a combination of (a) better overlap between computation and communication, and (b) better cache utilization, since the over-decomposition of AMPI produces sub-domains that more naturally fit the sizes of the machine's caches (we measured such cache improvements and reported results in [Rodrigues2010]).

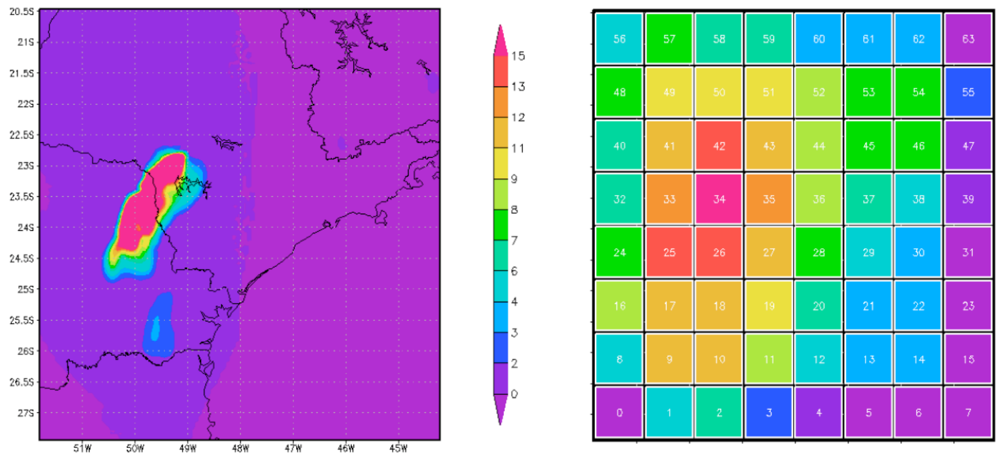


Figure 6. Results of a BRAMS weather forecasting and corresponding load on the 64 used processors.

Next, we applied several load balancers to BRAMS. Besides testing various load balancers already available in Charm++, we also developed a new balancer based on the distribution of sub-domains to processors according to a space-filling curve defined by to a Hilbert function. This distribution seems to be very appropriate for the two-dimensional domain decomposition employed by BRAMS, and preserves some of the locality of communication across sub-domains, even when some of those sub-domains migrate across processors due to load balancing. As a brief sample of our obtained results, Figure 7 shows the original processor utilization in BRAMS before any virtualization was applied, and the resulting utilization obtained with a virtualization factor of eight (i.e. AMPI divides each original domain into eight sub-domains) and the Hilbert load balancer. There is a much higher utilization, and we observed a reduction of more than 30% in the total execution time.

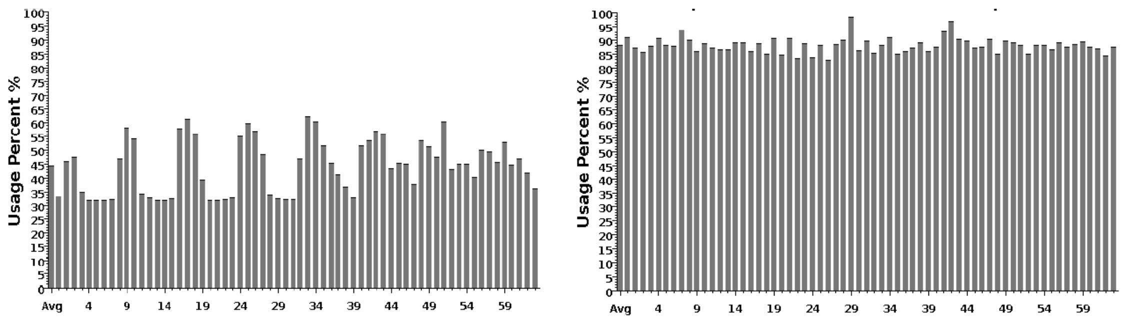


Figure 7. Processor utilization in BRAMS: pre-virtualization (left) and after virtualization and load-balance (right).

The more recent work in this area has focused on attempts to provide more automation to the entire optimization process via load balancing, such as finding automatically the best load balancing period, based on balancing costs and imbalance penalties. Our studies indicated that, for codes with a large memory footprint such as BRAMS, assessing the degree of imbalance is relatively cheap compared to actually migrating sub-domains across processors. Hence, one can exploit techniques that monitor the degree of imbalance closely, and only allow migrations that would produce performance gains higher than the penalties associated to the current imbalance.

The present grant also partly funded preliminary work on power-aware load-balancing techniques. It is now well known that increasing the number of cores and clock speeds on a smaller chip area implies more heat dissipation and an increased heat density. This increased heat, in turn, leads to higher cooling costs and the possible occurrence of hot spots. Effective use of dynamic voltage and frequency scaling (DVFS) can help to alleviate this problem. However, there is an associated execution time penalty, which can get amplified in parallel applications. In high performance computing, applications are typically tightly coupled and even a single overloaded core can adversely affect the execution time of the entire application. We have started to investigate a temperature-aware load-balancing scheme that uses DVFS to keep core temperatures below a user-defined threshold, with minimal timing penalties. While doing so, it also reduces the possibility of hot spots. We tested our scheme with three parallel applications having different energy consumption profiles.

Results from our initial experiments show that it is possible to save up to 14% in execution time and 12% in machine energy consumption as compared to frequency scaling without using load balancing. As an example, Figure 8 shows the measured execution time of a Jacobi-2D code on 128 processors, as a function of the temperature set for the machine room's air conditioner. When our temperature-aware load balancer (TempLDB) is used, the effects of a slowdown due to pure DFVS are not as strong, resulting in better overall performance. In other tests, we are also able to bound the average temperature of all the cores and reduce the temperature deviation amongst the cores by a factor of three. A full description of our initial results in this area was reported in [Sarood2011].

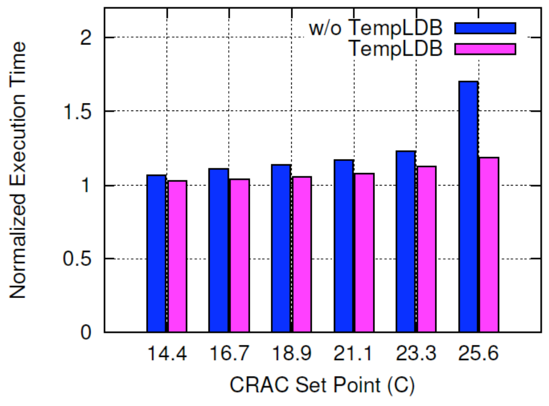


Figure 8. Effects of temperature-aware load balancing on Jacobi-2D execution with 128 cores under DVFS

2.4 Task Mapping (Colony lead: Univ. of Illinois)

The third focus of our study was the problem of task mapping on large parallel machines. Network contention has a significantly adverse effect on the performance of parallel applications with increasing size of parallel machines. Machines of the current petascale era are forcing application developers to map tasks intelligently to job partitions to achieve the best performance possible. We have developed a framework for automated mapping of parallel applications with regular communication graphs to two and three dimensional mesh and torus networks. This framework can save much effort on the part of application developers to generate mappings for their individual applications.

One component of our framework is a process topology analyzer to find regular patterns and, when found, to determine the dimensions of the communication graphs of applications. The other component is a suite of heuristic techniques for mapping 2D object grids to 2D and 3D processor meshes. The framework chooses the best heuristic from the suite for a given object grid and processor mesh pair based on the *hop-bytes* metric. We obtained performance improvements using the framework, for a 2D Stencil benchmark in MPI and for the Weather Research and Forecasting model (WRF) running on the IBM Blue Gene/P.

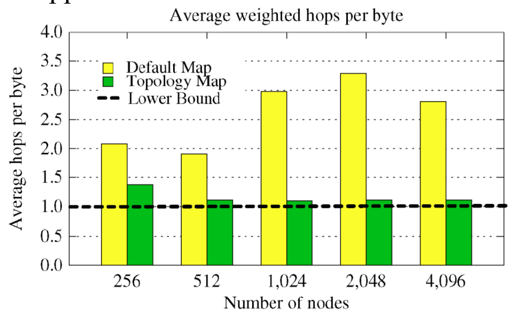


Figure 9. Results from topology-aware mapping of the WRF model on Blue Gene/P

For WRF, some of our results are shown in Figure 9; on 1,024 nodes, the average hops per byte reduced by 63% and the communication time (not shown in the figure) reduced by 11%. We measured an overall performance improvement of 17% for the application. At 4,096 nodes, there is a reduction in total execution time by 5%. Such performance improvements can be quite significant for the overall completion time of long running simulations. We also compared our algorithms with others discussed in the literature, as described with the full results of this study in [Bhatele2010].

2.5 Scalable membership, monitoring, & communication services (Colony lead: IBM Research)

We continue our activities on the HPC Colony II project for the second year, in cooperation with our partners from Oak Ridge National Laboratory (ORNL) and University of Illinois at Urbana-Champaign.

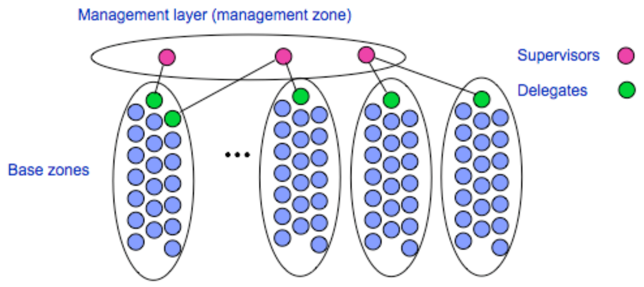
As originally planned, we conducted research on scalable membership, monitoring and communication services that will enable sophisticated applications and general purpose cluster computing on high-performance computing systems with a very large numbers of processors. The SpiderCast system, that will provide these services, will be based on overlay and peer-to-peer technologies.

Membership services enable the discovery of both active group processes as well as failed nodes and processes, thus facilitating fault tolerance implementations [Renesse98, Ganesh03, Allavena05, Varma06]. The Attribute-service allows each node to declare runtime attributes on itself, which facilitates easy integration of cluster services and a distributed mechanism for service location and discovery. Monitoring services enable the collection and aggregation of statistics from nodes and processes thus supporting the implementation of dynamic load balancing schemes [Renesse03]. Group communication services provide groups of processes with the means to efficiently communicate using topic-based publish/subscribe, which greatly helps developing clustered applications [Chockler01, Eugster03, Chockler07]. SpiderCast will also provide a DHT (distributed hash table) implementation, which provides services for storing and looking up key-value pairs in a distributed manner [Stoica01, Cass10].

SpiderCast will, on the one hand, utilize the unique architecture and networking features of Blue Gene [BGP08] to achieve top performance, and on the other hand, develop scalable technologies for systems with tens of thousands of processors, which can be deployed on general clustered systems.

The following specific activities were conducted in this funding period:

1. We completed the implementation of the multi-zone scalable membership service (see [TR1]). This service supports a two level hierarchy of SpiderCast zones, which allow an order of 1000 zones, each holding and order of 1000 nodes. We have a software prototype that demonstrates these capabilities.



2. We completed the low level design of the Distributed Hash Table (DHT, see [TR2]). This service allows storage and lookup of key-value pairs in the memory of SpiderCast nodes. It is exposed as a service to the application using SpiderCast and is also used as an internal building block of the publish-subscribe component.
3. We completed a prototype implementation of the DHT.
4. We completed the low level design of publish-subscribe component.
5. We expect to finish the implementation of the publish-subscribe component by end of funding period, according to plan.
6. By the end of the current funding period we will complete a software prototype of the DHT and Publish-Subscribe services. We will present the results of our experiments and provide additional documentation.

3 Project Status and Future Plans

Progress for Colony II has proceeded according to plans and without unresolved problems. During this reporting period, we maintained bi-weekly teleconferences with our project collaborators from UIUC and IBM, and had two face-to-face team meetings: one at the Supercomputing'10 conference in November'2010 at New Orleans, Louisiana, and another in April'2010 during the 9th Charm++ Workshop at Urbana, IL. Finally, a project website is maintained at http://www.hpc-colony.org

3.1 Overall Assessment of Progress With Respect to Project Plan

To date, all project FWP milestones have been realized on time.

Table 2. HPC Colony FWP Milestones for months 10 through 21

PROJECT Milestones (months 10 through 21) July 2010 – June 2011	Planned	Actual
Load-balancing framework capable of accepting application knowledge information	07/2010	07/2010
Message-logging protocol extended with integration to load-balancing	09/2010	01/2009
Demonstration of a basic overlay-based communications stack that runs on both Blue Gene networks (torus, collectives)	09/2010	01/2009
An initial prototype of the standalone distributed communication infrastructure	09/2010	01/2010
Documentation plus manual of the communication infrastructure	09/2010	09/2009
Initial demonstration of Oak Ridge National Laboratory and UIUC selected applications	09/2010	09/2009
Annual report of activities submitted to U.S. Department of Energy (DOE)	09/2010	06/2010
New class of hierarchical load balancers based on topology-aware algorithms	12/2010	12/2010
Demonstration of a basic SpiderCast system functionality in Blue Gene	03/2011	03/2011
Evaluation and testing of the standalone communication infrastructure on a large cluster of computers	03/2011	03/2011
Research paper/report	03/2011	11/2010
Message-logging and proactive fault-tolerance schemes tested on full-scale applications and integrated to Charm++ distribution	03/2011	03/2011
Paper reporting new Fault Tolerance features	03/2011	09/2010
Integration of optimized communications stack / distributed communication infrastructure	06/2011	06/2011

3.2 Selected Overall Project Highlights

a. Completed a 2010 INCITE Allocation Award. We plan to apply for a follow-up INCITE Allocation Award as well as Blue Waters allocation.
b. During this period, we developed the first co-scheduling Linux kernel designed for High Performance Computing. A bulk-synchronous-parallel benchmark improved 285% in execution time performance under the new kernel.
c. Developed a new power aware load balancing strategy which has shown improvements for both execution time and power consumption. The new scheme takes advantage of dynamic voltage and frequency scaling (DVFS) hardware capabilities.

d. We completed the initial implementation of a multi-zone scalable membership service as well as the low level design of the new Distributed Hash Table to be used for key-value pairs within SpiderCast.
e. Our new adaptive task mapping strategies show improvements for the Weather Research and Forecasting (WRF) model. For 1,024 nodes, the average hops per byte reduced by 63% and the communication time reduced by 11%..
g. Developed new causal-based message logging scheme with improved performance and scalability.
h. We also completed the design and implementation of a new dynamic load-balancing technique. Results for the BRAMS weather forecasting model show much higher machine utilization and reduction of more than 30% in execution time.

3.3 Future Plans and Ongoing Activities

a. Continue research on our coordinated kernel scheduler and measure the effectiveness on both synthetic benchmarks and real applications.
b. Explore adaptive+ganged fault tolerance and load balancing strategies.
c. Measure the effectiveness of the Colony system software stack for several applications on ORNL's Jaguar system.
d. During the next funding period, we will continue the development of SpiderCast services that will deliver the level of functionality and scalability necessary to accomplish the goals of the HPC Colony II project. We will also improve the scalability and performance of the services already developed during the previous funding periods. In particular, we will focus on the development of (1) a ConvergeCast service, and (2) a demonstration of SpiderCast on a large scale system.
e. At the end of the next funding period we plan to have a software prototype that includes all services we set out to implement at the beginning of the project. We plan to present the design of those services, results of our experiments, a user manual and additional documentation.
f. By the end of the current funding period we will complete a software prototype of the membership and attribute services. We will present the results of our experiments and provide additional documentation.

3.4 Cost Status

ORNL: Due to the Continuing Resolution, we still have not received our full FY11 funds. Our finance people anticipate a reduction of between 2% and 5% from the planned FY11 Colony ORNL allocation ($185K), or between $175,750 and $181,000, plus our carry-over of $24,287. We anticipate total expenditures of about $198K. This would leave a small amount of carry-over (between 1% and 4%) if cuts follow our best current guess.

UIUC: By the end of our second project year (Sep.15, 2011) we expect to have approximately $100,000 of unused funds, which corresponds to 20% of our total budget for the overall project period. This is still the result of our very late start in 2009, caused by the long inactive period between the project's approval and its actual implementation and contract agreement between DOE and Illinois. During the ongoing second year (i.e. Sep.2010 to Sep.2011), we are on track to spending almost exactly the expected amount for this period ($250,000). We plan to shift part of our personnel to the tasks in this project, such that we can complete, during its third year, all the activities that we had listed in our proposed plan. That move is expected to require employing both the new funds for the third/final year ($250,000) and the unused funds mentioned above.

IBM: We anticipate no unexpended funds for this period.

4 Publications, Talks, and Software Products

4.1 Publications

- Terry Jones, “Linux Kernel Co-Scheduling For Bulk Synchronous Parallel Applications”, *International Workshop on Runtime and Operating Systems for Supercomputers (ROSS 2011)*, Tucson, Arizona, USA, May 2011.
- Terry Jones, Gregory A. Koenig, “Clock Synchronization in High-end Computing Environments: A Strategy for Minimizing Clock Variance at Runtime”. (submitted for publication)
- Jonathan Lifflander, Phil Miller, Ramprasad Venkataraman, Anshu Arya, Terry Jones, and Laxmikant V. Kalé. Exploring Partial Synchrony in an Asynchronous Environment Using Dense LU. (submitted for publication)
- Terry Jones and Gregory Koening, “Providing Runtime Clock Synchronization With Minimal Node-to-Node Time Deviation on XT4s and XT5s”, 2011 Cray Users Group Meeting, Fairbanks, AK, May 2011.
- Terry Jones and Gregory Koening, “A Clock Synchronization Strategy for Minimizing Clock Variance at Runtime in High-end Computing Environments”, *22nd International Symposium on Computer Architecture and High Performance Computing*, Rio De Janeiro Brazil. October 2010.
- Abhinav Bhatele, Gagan Gupta, Laxmikant V. Kale and I-Hsin Chung, "Automated Mapping of Regular Communication Graphs on Mesh Interconnects", Proceedings of International Conference on High Performance Computing (HiPC), Goa-India, 2010.
- Gengbin Zheng, Abhinav Bhatele, Esteban Meneses and Laxmikant V. Kale, "Periodic Hierarchical Load Balancing for Large Supercomputers", Accepted for publication in International Journal for High Performance Computing Applications (IJHPCA), 2010.
- Eduardo R. Rodrigues, Philippe O. A. Navaux, Jairo Panetta, Alvaro Fazenda, Celso L. Mendes and Laxmikant V. Kale, "A Comparative Analysis of Load Balancing Algorithms Applied to a Weather Forecast Model", Proceedings of 22nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Itaipava, Brazil, 2010.
- Eduardo R. Rodrigues, Philippe O. A. Navaux, Jairo Panetta, Celso L. Mendes and Laxmikant V. Kale, “Optimizing an MPI Weather Forecasting Model via Processor Virtualization”, Proceedings of International Conference on High Performance Computing (HiPC), Goa, India, 2010.
- Esteban Meneses, Greg Bronevetsky and Laxmikant V. Kale, “Evaluation of Simple Causal Message Logging for Large-Scale Fault Tolerant HPC Systems”, Proceedings of Workshop on Dependable Parallel, Distributed and Network-Centric Systems (DPDNS) at IPDPS, Anchorage, USA, 2011.
- Aaron Becker, Gengbin Zheng, and Laxmikant Kale, “Distributed Memory Load Balancing”, Encyclopedia of Parallel Computing, David Padua, Ed., 2011 (to appear)
- Osman Sarood, Abishek Gupta and Laxmikant V. Kale, “Temperature Aware Load Balancing for Parallel Applications: Preliminary Work”, Proceedings of Workshop on High Performance Power Aware Computing (HPPAC) at IPDPS, Anchorage, USA, 2011.
- Chao Mei, Yanhua Sun, Gengbin Zheng, Eric J. Bohm, Laxmikant V. Kale, James C. Phillips and Chris Harrison, “Enabling and Scaling Biomolecular Simulations of 100 Million Atoms on Petascale Machines with a Multicore-optimized Message-driven Runtime”, University of Illinois, Urbana, 2011 (submitted for publication).

4.2 Talks

- Celso L. Mendes and Laxmikant V. Kale, “Adaptive MPI”, Blue Waters PRAC Fall Workshop, Urbana, October 2010.
- Abhinav Bhatele, “Mapping your Application on Interconnect Topologies: Effort versus Benefits”, George Michael HPC Fellow Presentation at Supercomputing’10, New Orleans, November 2010.
- Esteban Meneses, “Clustering Parallel Applications to Enhance Message-Logging Protocols”, 4th Workshop INRIA-Illinois Joint Laboratory on Petascale Computing, Urbana, November 2010.
- Eric Bohm, “Scaling NAMD into the Petascale and Beyond”, 4th Workshop INRIA-Illinois Joint Laboratory on Petascale Computing, Urbana, November 2010.
- Eric Bohm, Chao Mei, Yanhua Sun and Gengbin Zheng, “Charm++ Tutorial”, Chinese Academy of Sciences, Beijing, China, December 2010.
- Abhinav Bhatele, “Topology Aware Mapping”, University of Illinois (presented by telecom to the Chinese Academy of Sciences”, December 2010.
- Laxmikant V. Kale, “State of Charm++”, Charm++ Workshop, Urbana, April 2011.
- Osman Sarood, “Temperature-Aware Load Balancing for Parallel Applications”, Charm++ Workshop, Urbana, April 2011.
- Abhinav Bhatele, “New Developments in the Charm++ Load Balancing Framework and its Applications”, Charm++ Workshop, Urbana, April 2011.
- Esteban Meneses and Xiang Ni, “Fault Tolerance Support for Supercomputers with Multicore Nodes”, Charm++ Workshop, Urbana, April 2011.
- Eric Bohm, “Charm++ Tutorial”, Charm++ Workshop, Urbana, April 2011.

4.3 Software Products

- We are discussing our coordinated scheduling Linux kernel with an HPC vendor. Moreover, the work was done as freely available software and may be taken up by additional HPC vendors.

- Some parts of this research have been incorporated to the public distribution of the Charm++ software infrastructure, which is available in both source and binary formats. In particular, a new release of Charm++ (v.6.2.1) was made available recently, through the Charm++ download website: http://charm.cs.uiuc.edu/software/

5 References

[Allavena05] A. Allavena, A. Demers, and J. E. Hopcroft, "Correctness of a gossip based membership protocol," in *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 2005, pp. 292-301.

[BGP08] IBM-Blue-Gene-Team, "Overview of the IBM Blue Gene/P project," *IBM Journal of Research and Development*, vol. 52, no. 1/2, pp. 199-220, 2008.

[Chockler01] G. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: a comprehensive study," *ACM Computing Surveys*, vol. 33, no. 4, pp. 427-469, 2001.

[Chockler07] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication," in *DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems*. New York, NY, USA: ACM, 2007, pp. 14-25.

[Eugster03] P. TH. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114-131, June 2003.

[Ferreira08] Kurt Ferreira, Ron Brightwell, Patrick Bridges**.** Characterizing Application Sensitivity to OS Interference Using Kernel-Level Noise Injection. *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'08),* Austin, TX, November 2008.

[Ganesh03] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), February 2003.

[Hoefler10] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. In International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10*)*, Nov. 2010.

[Howland04] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(8):995–1006, 2004.

[Lakshman10] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35-40, Apr. 2010.

[Lee99] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, 1999.

[Nataraj07] A. Nataraj, A. Morris, A. D. Malony, M. Sottile, and P. Beckman. The ghost in the machine: Observing the effects of kernel operation on parallel application performance. *In Proceedings of SC'07*, 2007.

[Oliker07] L. Oliker, A. Canning, J. Carter et al., "Scientific Application Performance on Candidate PetaScale Platforms," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*:1-12, 2007.

[Renesse98] R. V. Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proc. Middleware 98*, 1998.

[Renesse03] R. Van Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM Trans. Comput. Syst.*, vol. 21, no. 2, pp. 164-206, May 2003.

[Sottile04] M. Sottile and R. Minnich. Analysis of microbenchmarks for performance tuning of clusters. In *Proceedings of IEEE Cluster2004 International Conference on Cluster Computing*, pages 371–377, 2004.

[Stoica01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149-160, October 2001.

[Tock10a] Yoav Tock, Benjamin Mandler, "SpiderCast: Distributed Membership and Messaging for HPC Platforms: An Architectural Overview and High Level Design". *Colony-II technical report*, January 2010.

[Tock10b] Yoav Tock, Benjamin Mandler, Gennady Laventman, "SpiderCast: Distributed Membership and Messaging for HPC Platforms: Publish-Subscribe and DHT Services High Level Design". *Colony-II technical report*, May 2010.

[Varma06] J. Varma, C. Wang, F. Mueller, C. Engelmann, and S. L. Scott, "Scalable, fault tolerant membership for mpi tasks on hpc systems," in *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*. New York, NY, USA: ACM, 2006, pp. 219-228.