LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-222559

FY 2006 Accomplishment Colony - "Services and Interfaces to Support Large Numbers of Processors"

T. Jones, L. Kale, J. Moreira, C. Mendes, S. Chakravorty, A. Tauferner, T. Inglett

July 3, 2006

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Colony – *"Services and Interfaces to Support Large Numbers of Processors"*

Terry Jones[1], Lawrence Livermore National Laboratory
Laxmikant Kale[2], University of Illinois at Urbana-Champaign
Jose Moreira[3], International Business Machines
Celso Mendes, University of Illinois at Urbana-Champaign
Sayantan Chakravorty, University of Illinois at Urbana-Champaign
Andrew Tauferner, International Business Machines
Todd Inglett, International Business Machines

Summary

The Colony Project is developing operating system and runtime system technology to enable efficient general purpose environments on tens of thousands of processors. To accomplish this, we are investigating memory management techniques, fault management strategies, and parallel resource management schemes. Recent results show promising findings for scalable strategies based on processor virtualization, in-memory checkpointing, and parallel aware modifications to full featured operating systems.

The trend towards larger processor counts benefits application developers through more processing power. It also challenges application developers to harness ever-increasing numbers of processors for productive work. Much of the burden falls to operating systems and runtime systems that were originally designed for much smaller processor counts. Under the Colony project, we are researching and developing system software to enable general purpose operating and runtime systems for tens of thousands of processors. These technologies will be demonstrated on multiple platforms including IBM's Blue Gene class machines.

As machines become larger, system-wide management of important resources across the entire becomes increasingly important for improving performance and scalability of applications. Important examples include CPU time, communication time and memory. Typically, application programmers have to explicitly write code to balance the use of these resources. However, with larger processor counts and more complex applications, programmers are faced with an increasingly difficult task in balancing resource usage. Next generation parallel operating and runtime systems must provide automatic resource balancing. As a part of the Colony project, we are further developing infrastructure and strategies for automated resource management by delegating low-level management tasks to operating system and runtime software. We accomplish this through dividing the parallel application into much smaller migratable work units (objects) such as user-level threads and parallel objects. These relatively fine-grained units make it possible for the parallel operating system to manage resources automatically, and facilitate new fault-tolerance and job-management strategies. Our approach assigns the arduous task of mapping the work to processors to the runtime system, using less programmer time and effort to achieve highly efficient parallel programs.

---

[1] Terry Jones, LLNL Principal Investigator, Telephone Number 925-423-9834, E-mail address trj@llnl.gov
[2] Laxmikant Kale, UIUC Principal Investigator, Telephone Number 217-244-0094, E-mail address kale@cs.uiuc.edu
[3] Jose Moreira, IBM Principal Investigator, Telephone Number 914-945-1709, E-mail address moreira@us.ibm.com

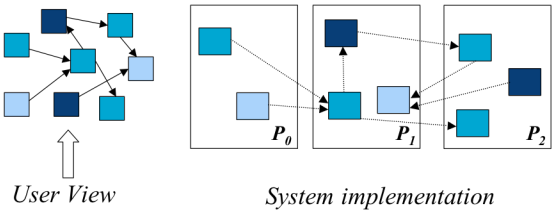

Figure 1: Charm++ runtime management permits easily going from a *user view* of small work-units to an efficient system implementation. Colony is extending Charm++ for extreme scale usage.

In the area of automated resource management, Charm++ offers a variety of load balancing schemes. These schemes can be generally partitioned into centralized and distributed approaches. Centralized balancers achieve the best possible balance at the cost of high memory and network usage while distributed balancers do not reach good balance quickly. Under the Colony project, we have started to develop a new, hybrid load balancing approach that divides processors into groups, and applies a centralized balancing scheme to each group. Our preliminary results from this approach on large systems show that we can achieve good balance in an acceptable amount of time and at a more controllable cost. We have also investigated new balancing schemes that take into account the topology of the system interconnect. Our recent findings for performance of a 2D-based application running on Blue Gene/L, which has a 3D-Torus interconnect shows this scheme takes much less time than random mappings.

The complexity of tomorrow's machines has focused concern on mean time to failure and fault tolerance becomes an important issue. During this past year, we developed a new mechanism for proactive handling of impending faults, based on object migration. Upon receiving a warning of an imminent fault on a given processor, our runtime system efficiently migrates execution away from that processor. More recently, we have been working on enhancing and extending another scheme to tolerate faults, based on message logging. This ambitious and long-term research aims at a scheme that does not penalize all processor's progress for the failure of one, and allows for a fast recovery from faults. A preliminary version of this scheme is under tests to assess its effectiveness and its underlying costs to the application.

To address questions on the possibilities and limitations of full Linux for extreme scale systems, the Colony project is developing an all-Linux solution for IBM's Blue Gene system. Blue Gene is notable for high performance and large numbers of processors (delivered systems of over 128,000 processors). In addition to extending the spectrum of applications for Blue Gene/L, this activity will deliver a platform for studying extreme scalability of Linux. Recent efforts have produced a design and prototype to allow Linux to run on BlueGene/L compute nodes. This provides additional flexibility for HPC applications versus the current custom kernel that runs on BlueGene/L compute nodes. This prototype Linux implementation for BlueGene/L has been successfully booted in IBM laboratories and form the basis for future work related to the HPC Colony Project.

The use of Linux as the operating system for Blue Gene compute nodes presents its own challenges such as the issue of asynchronous and nondeterministic behavior that is often observed in server-grade operating systems. Colony is devising strategies to manage the many sources of asynchronous events in Linux (TLB misses, process/thread scheduler, I/O events) so that they do not impede scalability to tens of thousands of processors. During the last year, these strategies have demonstrated success on machines in the 1000 to 8000 processor count range. Next steps for these promising techniques include scaling studies on Linux-based Blue Gene machines. Other Colony work underway focuses on the documentation and investigation of system management issues on large scale HPC systems like Blue Gene/L.

For further information on this subject contact:
Terry Jones, Coordinating PI
Lawrence Livermore National Laboratory
Email: trj@llnl.gov
Phone: 925-423-9834
Web: http://www.hpc-colony.org