Chapter 38
State–Carrying Code for Computation Mobility

Hai Jiang
Arkansas State University, USA

Yanqing Ji
Gonzaga University, USA

ABSTRACT
Computation mobility enables running programs to move around among machines and is the essence of performance gain, fault tolerance, and system throughput increase. State-carrying code (SCC) is a software mechanism to achieve such computation mobility by saving and retrieving computation states during normal program execution in heterogeneous multi-core/many-core clusters. This chapter analyzes different kinds of state saving/retrieving mechanisms for their pros and cons. To achieve a portable, flexible and scalable solution, SCC adopts the application-level thread migration approach. Major deployment features are explained and one example system, MigThread, is used to illustrate implementation details. Future trends are given to point out how SCC can evolve into a complete lightweight virtual machine. New high productivity languages might step in to raise SCC to language level. With SCC, thorough resource utilization is expected.

INTRODUCTION
The way in which scientific and engineering research is conducted has radically changed in the past two decades. Computers have been used widely for data processing, application simulation and performance analysis. As application programs’ complexity increases dramatically, powerful supercomputers are on demand. Due to the cost/performance ratio, computer clusters are commonly utilized and treated as virtual supercomputers. Such computing environments can be easily acquired for scientific and engineering applications.

For each individual computer node, multi-core/many-core architecture is becoming popular in the computer industry. In the near future, hundreds and thousands of cores might be placed inside of computer

DOI: 10.4018/978-1-60566-661-7.ch038
nodes on server-clusters. Multi-core clusters are promising high performance computing platforms where multiple processes can be generated and distributed across participating machines and the multithreading technique can be applied to take advantage of multi-core architecture on each node. This hybrid distributed/shared memory infrastructure fits the natural layout of computer clusters.

Since computer clusters for high performance computing can change their configurations dynamically, i.e., computing nodes can join or leave the systems at runtime, the ability of re-arranging running jobs is on demand to exploit the otherwise wasted resources. Such dynamic rescheduling can optimize the execution of applications, utilize system resource effectively, and improve the overall system throughput. Since computation mobility, i.e., the ability of moving computations around, is one of the essences to this dynamic scheduling, it has become indispensable to scalable computing for the following outstanding features:

- **Load Balancing**: Evenly distributing workloads over multiple cores/processors can improve the whole computation’s performance. For scientific applications, computations are partitioned into multiple tasks running on different processors/computers. In addition to variant computing powers, multiple users and programs share the computation resources in non-dedicated computing environments where load imbalance occurs frequently even though the workload was initially distributed evenly. Therefore, dynamically and periodically adjusting workload distribution is required to make sure that all running tasks at different locations finish their execution at the same time in order to minimize total idle time. Such load reconfiguration needs to transfer tasks from one location to another.

- **Load Sharing**: From the system’s point of view, load sharing typically increases the throughput of computer clusters. Studies have indicated that a large fraction of workstations could be unused for a large fraction of time. Scalable computing systems seek to exploit otherwise idle cores/processors and improve the overall system efficiency.

- **Data Locality**: Sharing resources includes two approaches: moving data to computation or moving computation to data. Current applications favor data migration as in FTP, web, and Distributed Shared Memory (DSM) systems. However, when computation sizes are much smaller than data sizes, the code or computation migration might be more efficient. Communication frequency and volume will be minimized by converting remote data accesses into local ones. In data intensive computing, when client-server and RPC (Remote Procedure Call) infrastructure are not available, computation migration is an effective approach to accessing massive remote data.

- **Fault Tolerance**: Before a computer system crashes, local computations/jobs should be transferred to other machines without losing most existing computing results. Computation migration and checkpointing are effective approaches.

Computation migration feature has existed in some batch schedulers and task brokers, such as Condor (Bricker, Litzkow, & Livny, 1992), LSF (Zhou, Zheng, Wang, & Delisle, 1993) and LoadLeveler (IBM Corporation, 1993). However, they can only work at a coarse granularity (process) level and in homogeneous environments. So far, there is no effective task/computation migration solution in heterogeneous environments. This has become the major bottleneck in dynamic schedulers and obstacle for scalable computing to achieve high performance and effective resource utilization.

**State-Carrying Code (SCC)** is a software mechanism that intends to provide the ability of moving computations around. It allows programs to contain both normal statements for application execution...
19 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:
www.igi-global.com/chapter/state-carrying-code-computation-computation-mobility/36438?camid=4v1

This title is available in InfoSci-Books, Business-Technology-Solution, InfoSci-Software Technologies, Science, Engineering, and Information Technology, InfoSci-Select, InfoSci-Computer Science and Information Technology. Recommend this product to your librarian:
www.igi-global.com/e-resources/library-recommendation/?id=1

Related Content

**Carrier-Grade Distributed Cloud Computing: Demands, Challenges, Designs, and Future Perspectives**
www.igi-global.com/chapter/carrier-grade-distributed-cloud-computing-/82541?camid=4v1a

**Distributed Adaptive Windowed Stream Join Processing**
www.igi-global.com/article/distributed-adaptive-windowed-stream-join/53852?camid=4v1a

**Architectural Resiliency in Distributed Computing**
www.igi-global.com/article/architectural-resiliency-distributed-computing/74167?camid=4v1a

**Mechanism for Privacy Preservation in VANETs**
www.igi-global.com/chapter/mechanism-privacy-preservation-vanets/61989?camid=4v1a