論 文 の 内 容 の 要 旨
Abstract

論文題目　　Techniques for Enabling Highly Efficient Message Passing on Many-Core Architectures
（メニーコア型大規模並列計算機向けの高性能メッセージパッシング型通信技術）

氏　　名　　思　敏

Since multicore processors have become the most common processor architectures today, the next grade promotion for high-end processors is expected to be achieved by improving both thread- and instruction-level parallelism. There are two kinds of architectures dominating the high performance market today, the GPU accelerators and the General Purpose (GP) many-core architectures. In this dissertation, we focus on the latter. Many-core architecture, such as Intel Xeon Phi and IBM Blue Gene/Q, provides us a massively parallel environment containing dozens of cores and hundreds of hardware threads with powerful wide SIMD units. More and more scientific application developers have begun investigating ways to utilize such architecture for scaling application performance. However, the performance may be restricted in various ways. Unlike traditional CPUs, the performance capability of many-core architectures comes from massive low- frequency cores for better performance-to-energy ratio; thus sequential execution on such hardware could result in performance degradation. Furthermore, the other on-chip re- sources (e.g., memory) are not growing at the same rate as number of cores, potentially resulting in scalability issue.

Not only hardware architectures, the scientific applications are also moving toward complex hybrid and irregular models. In traditional regular applications (e.g., Fast Fourier transform), more and more applications start focusing on hybrid programming models comprising a mixture of processes and threads, that allow resources on a node to be shared between the different threads of a process, especially benefiting the execution on many-core architectures. The most prominent of the hybrid models used in scientific computing today is

MPI+OpenMP, where multiple OpenMP threads parallelize the computation, while one or more threads utilize MPI for their data communication. On the other hand, despite the well studied regular applications, a number of applications are becoming extremely dynamic and irregular especially in chemistry and bioinformatics domains. MPI-2 and MPI-3 introduced one-sided communication mode, which is more suitable for supporting the data movements in such irregular model rather than the MPI two-sided or group communication modes.

With growing complexity in both computing hardware and scientific applications, various critical communication issues raise up and resulting in severe degradation in application performance. This dissertation focuses on exploiting the capabilities of advanced many-core architectures on widely used message passing model, in order to address the communication problems existing in the popular hybrid programming model and the irregular one-sided mode and consequently contribute efficient communication approaches for various kinds of applications.

Firstly, in hybrid MPI+threads applications, a common mode of operation for such applications involves using multiple threads to parallelize the computation, while one of the threads issues MPI operations. Although such mode extremely improves floating point performance for computation of applications by massive parallelism, it also means that most of the threads are idle during MPI calls, which translate to underutilized hardware cores. Furthermore, since only single low-frequency core is contributing to communication, it may result in even performance degradation. To address the core idleness issue and improve the performance of communication, we propose an internally multithreaded MPI as the first contribution of this dissertation, that transparently coordinates with the threading runtime system to share idle threads with the application in order to fully utilize the computing resources as well as parallelizing MPI internal processing such as derived datatype communication, shared-memory communication, and network I/O operations for better performance.

Secondly, with regard to the irregular one-sided communication, however, the MPI standard does not guarantee that such communication is truly asynchronous. Most MPI implementations still require the remote target to make MPI calls to ensure progress on such operations, consequently the operation cannot complete at the target without explicit processing in software and thus may cause arbitrarily long delays if the target process is busy computing outside the MPI stack. Traditional implementations to ensure asynchronous completion of operations have relied on thread-based or interrupt-based models. Each of these models has several drawbacks, however, such as the inefficient core deployment in the thread model and the expensive overheads caused by multithreading safety in the thread model and by frequent per-message interrupts in the interrupt model. To address these drawbacks, we propose Casper, a process-based asynchronous progress model for MPI one-sided communication on multicore and many-core architectures as the second contribution of this dissertation. The central idea of Casper is to keep aside a small, user-specified number of cores on a multicore or many-core environment as "ghost processes," which are dedicated to help asynchronous progress for user processes through appropriate memory mapping from those user processes. Whenever user application issues an RMA operation to a user process Casper then transparently redirects such operation to the ghost process thus ensuring asynchronous completion. This approach has successfully resolved the communication bottleneck in the widely used NWChem quantum chemistry application by achieving up to 30 % performance improvement in the "gold standard" CCSD(T) simulation.

Although Casper provides simple but efficient asynchronous progress for irregular one-sided communication, the performance might not be optimal in a number of applications that always consist of multiple phases with varying proportion of communication and computation. Inefficient usage of asynchronous progress may even result in performance degradation. That is, the computation-intensive phase heavily relies on asynchronous progress, however, the communication-intensive phase does not have strong needs of asynchronous progress but more focuses on the load balance for large amount of RMA operations, which might not hold in Casper since the operations are consistently redirected to a few ghost processes. As the third contribution of this dissertation, we propose a dynamic adaptation mechanism embedded in Casper that transparently adapt the configuration of asynchronous progress for multi-phases applications.

Finally, apart from the lack of asynchronous progress, many irregular applications also suffer from loss of performance in a number of ways. For example, it is usual in imbalanced communication that an MPI process takes long time to wait for a message to arrive, the core on which it is scheduled is idle and underutilized. To comprehensively address these issues, we plan to investigate the concept of user-level processes, a way to provide multiple co-scheduled “OS processes” on a single core as the MPI processes, with exploiting the potential optimization in MPI communication runtime, such as better load balancing and light-weight checkpoint migration, as the future work of this doctoral research.