論文の内容の要旨

Heterogeneity-transparent Operating System Structures for High Performance Computer Clusters （高性能計算機クラスタのための 異機種透明なオペレーティングシステム構成法）

清水 正明

This thesis describes heterogeneity-transparent operating system structures for high performance computer clusters.

The performance required of high-performance computing systems is increasing by a factor of 1.8 annually. To increase speed and lower cost, there has been a transition from the previous specially-developed vector processors or massively parallel processor systems to cluster systems of inexpensive yet high-performance general-purpose processors, which has now become the mainstream architecture.

To achieve even higher performance and a higher performance to power efficiency in the future, we can expect the processors that have improved numerical computation performance, such as many-core or Cell/B.E. processors, will be adopted. However, processors that are specialized for numerical processing are relatively inefficient for scalar operations such as file I/O that are common in operating system functions. Another problem is that cluster architectures cannot execute software assets that were developed for the distributed OS of the massively parallel processor system era. In cluster systems, an independent general-purpose OS executes on each processor node; a distributed OS, on the other hand, is shared by all nodes as a single OS. Because the OS architectures are different, the porting of distributed file systems and other software assets that were developed for the distributed OS is very expensive.

One objective of this thesis is to achieve both the computational performance of new architectures that use specialized processors and the OS processing performance of conventional general-purpose processors in a high-performance cluster. Another objective is to allow low-cost reuse of existing distributed services that were developed for systems that execute distributed OS. To attain these objectives, we adopt the two approaches described below for structuring heterogeneity-transparent operating systems. One approach is to add a generic processor node that provides high performance generic OS functions to a cluster of compute processor nodes. The generic OS functions are separated between the compute nodes and the generic nodes, and coordinated to provide a single OS environment to users. In large-scale systems, clustering is done in sets of “a generic node and compute nodes.” The second approach is to implement a distributed OS environment on a cluster of generic OS nodes so that it enable to use existing distributed OS services at

low cost.

The first approach involves reducing the functions of the OS of compute nodes, which have specialized processors for numerical computation, to focus on the high performance execution of application programs. The generic node, which has general-purpose processors that perform scalar operations efficiently, provides the generic OS functions. In this way, we can achieve both high performance execution of applications and high performance execution of operating system functions. The OS of the generic node manages the parallel processes of the compute nodes, performs central management of user files, and performs proxy processing of file I/O requests from the compute nodes. To make the different operating systems transparent to users and application programs so that the system can be used as a single OS, a heterogeneity-aware binary loader and a remote system call mechanism are provided. We have implemented this approach in the ***Harmonix*** OS. The Harmonix was evaluated using a heterogeneous cluster in which four x86_64 generic nodes were added to 16 compute nodes that used Cell/B.E. The evaluation results show that parallel program invocation was faster by a factor of 3.2 compared to the system of compute nodes alone. Furthermore, parallel I/O with the Lustre file system was up to 2.6 times as fast, confirming the improvement in OS performance. We also confirmed that adding generic nodes can maintain scalability when increasing the number of compute nodes results in the generic node being the bottleneck. To make the system appear to the user as a single OS, the heterogeneity-aware binary loader selects nodes according to differences in the executable binaries and executes the programs on the appropriate type of node. In that way, the user can execute compute node applications transparently from the generic node. The remote system call mechanism allows applications executing on compute nodes to transparently use the Lustre file system functions that are provided by the generic node.

For the second approach, we designed and implemented a system in which a microkernel interface is implemented using a generic OS as host and existing distributed OS services are executed on that microkernel. The microkernel interface can be implemented at low cost by using the functions of the host generic OS. Also, existing distributed OS services that depend on the functions of the microkernel can be easily ported. The proposed architecture was implemented as a Mach microkernel on AIX. We also ported the HSFS distributed parallel file system, which is a distributed OS service, to that environment. Evaluation results show that the cost of porting the HSFS was greatly reduced to one third the cost of re-implementing the service for the cluster architecture. Furthermore, the results of a performance evaluation performed on an eight node Hitachi SR11000 show that the local node performance, which is bound by the performance of the disk array, and eight-node performance scalability were achieved. The result of this approach has been applied in product form for the Hitachi SR11000, SR16000 and HA8000-tc/RS425 systems.

Above, we have presented methods for structuring heterogeneity-transparent operating systems that absorb differences in processor and OS architecture. This approach improves system performance in a manner transparent to the users and application programs. It also

reduces the cost of porting software assets between OS architectures.