Distributed Processor Allocation in Multicomputers.

Conference Paper · January 2000

Source: DBLP

CITATIONS
0

READS
45

3 authors, including:

Hans-Ulrich Heiß
Technische Universität Berlin
100 PUBLICATIONS **785** CITATIONS

SEE PROFILE

Philippe Olivier Alexandre. Navaux
Universidade Federal do Rio Grande do Sul
443 PUBLICATIONS **2,405** CITATIONS

SEE PROFILE

Distributed Processor Allocation in Multicomputers

César A. F. De Rose
Catholic University
of Rio Grande do Sul
Dept. of Computer Science
Porto Alegre, Brazil
derose@inf.pucrs.br

Hans-Ulrich Heiss
University of Paderborn
Dept. of Computer Science
Paderborn, Germany
heiss@uni-paderborn.de

Philippe A. O. Navaux
Federal University
of Rio Grande do Sul
Dept. of Computer Science
Porto Alegre, Brazil
navaux@inf.ufrgs.br

Abstract

Current processor allocation techniques for multicomputers are based on centralized front-end based algorithms. As a result, the applied strategies are usually restricted to static, contiguous, structure preserving allocation, and suffer from low parallelism and weak fault tolerance. To lift these restrictions we are investigating a distributed approach to the processor allocation problem for mesh interconnected distributed memory machines. We conducted several experiments, some simulated and some running over a Simens hpcLine Primergy Server with 96 nodes, which showed that distributed allocation is feasible with current technologies.

1. The processor allocation problem

Processor allocation involves the selection of a processor *partition* for a given parallel job, with the goal of maximizing throughput over a stream of many jobs. Because allocation operations have to be fast, allocation techniques usually restrict the feasible shapes of partitions to achieve some regularity, which facilitates their management. We call a partitioning scheme *structure preserving* if it generates partitions that are of the same topological graph family as the entire processor graph (subcube allocation in hypercubes and submesh allocation in meshes). In addition, many systems also require that the allocated processors are constrained to be physically adjacent (*contiguous* allocation) to reduce communication costs.

2. Distributed processor allocation

Several approaches to deal with the processor allocation problem can be found in the literature [3]. In spite of the fact that they apply different policies in the resource management, all the schemes have one in common: the control of allocated resources is done with a global data structure localized mostly in a host machine. The main problems of such centralized management are lack of scalability, the incompatibility with adaptive processor allocation schemes (dynamic allocation), and its weak fault tolerance.

Figure 1 shows a global view of the proposed distributed allocation [1] and the distributed *Processor Managers* involved in the allocation operation. The main differences to the centralized management are (i) the absence of a central data structure with information about the state of all processors, and (ii) the execution of allocation operations directly in the processor mesh in a distributed way, and not in a data structure localized in the host. The host machine is now only responsible for queuing the incoming requests and forwarding them to the processor mesh. Due to the distributed environment there is no restriction concerning the number of entry points in the mesh.

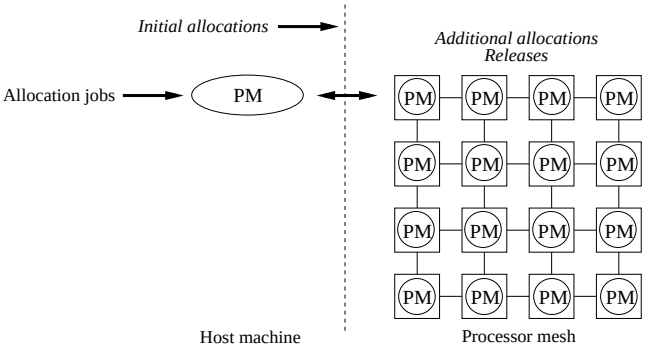

Figure 1. Distributed allocation

Each node in the mesh has a local Processor Manager (PM) responsible for the processor allocation. The PM's cooperate to solve the allocation problem in a distributed way.

The implemented PM uses an enhanced version of the

Leak algorithm [1]. This algorithm is based on the principle of leaking water. From an origin point, an amount of water leaks and flows to the directions where no resistance is encountered. In the case of a distributed processor allocation, the number of processors to be allocated corresponds to the amount of leaking water. The processors already allocated in the mesh are the resistance areas and the final area formed by the allocated processors is the resulting puddle. The essential feature of the algorithm is its form-free allocation strategy, i.e. partitions may have an arbitrary shape. This gives the processor management more flexibility to find a free partition, and results in less fragmentation. Due to the recursive nature of the algorithm and its distributed execution in the machine, it is also important to notice that different flowing directions allocate processors in parallel, resulting in a reduced allocation time.

Current communication technologies like wormhole routing enable us to consider noncontiguous allocation schemes, since the number of hops between nodes is not the dominant factor determining message latency [2]. The use of small partitions of free processors scattered in the machine to form larger non-contiguous partitions decreases the external fragmentation significantly. However, noncontiguous allocation introduces potential problems due to message contention because the messages occupy more links, yielding potential communication interference with other jobs. The idea is to try to serve a request with contiguous allocation, and to look for noncontiguous additions only on demand. This way the noncontiguous scheme should be seen as an addition, and not as an alternative to contiguous allocation.

3 Performance Analysis

In order to investigate the potential and the feasibility of the proposed allocation in large PC clusters we conducted message-passing contention experiments, intermittent service requests experiments, performance experiments and fragmentation experiments. For the first three experiments we used the Siemens hpcLine Primergy High Scalable Compute Sever at the Paderborn Center for Parallel Computing (PC^2). The Primergy Server is a distributed memory multicomputers with 96 compute nodes (dual Intel Pentium II) connected by a 400 megabyte per second unidirectional two dimensional SCI mesh (IEEE 1596), with wormhole, XY routing. Programs were written in a special MPI version for the SCI hardware that runs over the Solaris operating system. Our discrete event simulator [1] is a multi-computer simulator supporting experimentation with distributed allocation strategies on architectures with mesh-connected network topologies. The simulator was used in the fragmentation experiments to evaluate how the proposed strategies behave in bigger machines (up to 1024 nodes).

4. Conclusions

As a first step in evaluating the feasibility of the distributed allocation on real parallel machines, we measured worst-case contention in the Primergy Server. Contention plays an important role in the proposed allocation since we will have allocation messages stealing bandwidth from the ongoing applications and since we are also proposing a noncontiguous strategy. Our tests showed virtually no contention is noticeable for messages smaller than 4 kilobytes. For larger messages contention begins to slow message-passing performance, but only for more than 6 pairs of communicating nodes. This empirical data is encouraging, and indicates that the small messages generated by the distributed allocation (100 bytes mean size) will not affect the performance of the ongoing computations. It also indicates that the proposed noncontiguous allocation that may separate communicating pairs in a noncontiguous partition, is feasible in this machine. In the proposed distributed allocation, nodes which participate in ongoing computations on the machine will also have intermittent service requests to their local processor manager. We were unable to measure any performance degradation on real applications running simultaneously with the distributed management compared with the same applications running in a dedicated system. A possible explanation for this behavior is that in our SCI-based machine no interrupts are used.

Our study shows that the distributed approach is feasible for large cluster machines with current communication technologies and permitted a greater parallelization of the allocation operations, eliminated the bottlenecks of the centralized model, and achieved a better scalability of the allocation algorithms. This enables us to lift several restrictions of the centralized strategies and to experiment with adaptive, form-free, noncontiguous processor allocation schemes. As a result, system utilization for the noncontiguous version of our algorithm reaches as high as 97 percent. We concluded that distributed allocation provides a new approach that will help highly parallel systems to achieve better price/performance ratios in high demand, multi-user environments.

References

[1] C. A. F. De Rose. *Distributed Processor Management in Multicomputers*. University of Karlsruhe, Germany, Phd. Thesis, 1998.

[2] V. Loet al. Noncontiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 8(7), July 1997.

[3] Y. Zhu. Fast processor allocation and dynamic scheduling for mesh multicomputers. *International Journal of Computer Systems Science and Engineering*, 2(11):99–107, 1996.