Distributed Processor Allocation in Large PC Clusters

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

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

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

Abstract

Current processor allocation techniques for highly parallel systems are based on centralized front-end based algorithms. As a result, the applied strategies are restricted to static allocation, low parallelism and weak fault tolerance. To lift these restrictions we are investigating a distributed approach to the processor allocation problem in large distributed memory machines. A contiguous and a noncontiguous version of a distributed dynamic processor allocation strategy are proposed and studied in this paper. Simulations compare the performance of the proposed strategies with that of well-known centralized algorithms. We also present the results of experiments on a Simens hpcLine Primergy Server with 96 nodes that show 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 used by the majority of commercial parallel machines, as well as the research community, 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).

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. The communication between host and mesh is done through a direct channel to a boundary node. This node is called an *entry point* and due to the distributed environment there is no restriction concerning the number of entry points.

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.

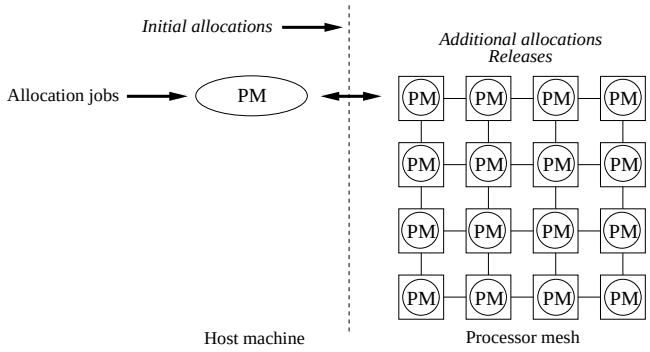

Figure 1. Distributed allocation

2.1. Distributed allocation algorithm

The implemented PM uses an enhanced version of the Leak algorithm. 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. An important factor is that the leaking water exhibits cohesion, which keeps the diameter of the resulting puddle as small as possible. 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 are no longer restricted to rectangles, but may have an arbitrary shape. This gives the processor management more flexibility to find a partition of suitable size, 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.

2.2. Noncontiguous allocation

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.

2.3. Rectangular vs. form-free allocation

Table 1 compares the obtained fragmentation for rectangular and form-free allocation under three load classess in a 32×32 mesh. The greater flexibility of the distributed Leak algorithm in finding free processors areas in the mesh (form- free search), reduces significantly the external fragmentation and eliminates the internal one. This means a greater efficiency in serving requests, reduction in the denied requests, and increase of machine utilization (around 75%). The noncontiguous version of the distributed Leak algorithm achieved 97% machine utilization in this simulation.

Algorithm	F_i	F_e	$Mutil$
Frame Slide	60.37%	7%	32.63%
Contiguous Leak	0%	24.50%	75.50%
Noncontiguous Leak	0%	2.97%	97.03%

Table 1. Fragmentation comparison

3. Conclusions

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, multiuser 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.