Scalability and Potential for Optimization in Dynamic Load Balancing - Centralized and Distributed Structures

Wolfgang Becker, Jörg Zedelmayr
Institute of Parallel and Distributed High-Performance Systems (IPVR)
University of Stuttgart, Germany

Phone: +49 711 7816 433
Fax: +49 711 7816 424
Email: wbecker@informatik.uni-stuttgart.de, zedelmay@informatik.uni-stuttgart.de

Abstract

This paper examines structural design alternatives for dynamic load balancing. Load balancing collects system state information and assigns or redistributes the application tasks among the processors of a parallel computing system in order to maximize overall throughput and stabilize response times. This can be performed either by a central component supervising the entire system, by cooperating per-processor load balancing agents or by cooperation of load balancing agents, each of them controlling a part of the processing system. We focus on load balancing techniques for cooperating agents which control system partitions in centralized fashion and propose an explicit and an implicit approach of interaction between the agents. Besides general considerations of performance improvement potential and scalability we derive results from a dynamic load balancing environment running different parallel applications on workstation clusters. We show that the approach of centralized agents per system partition is feasible and compare the explicit and implicit cooperation between partitions.

Keywords: dynamic load balancing, run time systems, scalability

1 Introduction and Motivation

Parallel computers and networks of workstations will be the system architectures to solve most of the big computational problems in the near future. User programs are being parallelized and there is an obvious need for application independent automatic load balancing. It is very efficient to distribute an application especially for certain system configurations, but if it is ported to other platforms, if several applications are using the system concurrently or if the actual load distribution is not completely predictable in advance this does not work any longer. Dynamic load balancing is required if the resource requirements are not known exactly in advance and if uncorrelated, unexpected tasks are entering the system.

The goal of dynamic load balancing is to optimize the overall system throughput and to stabilize response times. To achieve this, load balancing has to collect system state and application progress information and possibly exploit task load predictions. Based on this information it has to commit task assignment and task migration decisions that yield equal load on the processing nodes and minimize synchronization and data communication delays between related tasks.

Application independent load balancing is an operating system service. It can be organized in different degree between monolithic and distributed structure like operating systems as a whole. In this paper we focus on pros and cons of centralized, distributed and mixed structures to implement dynamic load balancing. For a general overview of load balancing we refer to [3].

1.1 Centralized Load Balancing

Centralized (often called ‘global’) load balancing employs a single agent for collection and management of state information and for making task assignment or task migration decisions (see figure 1). From a logical point of view this structure is optimal for several reasons:

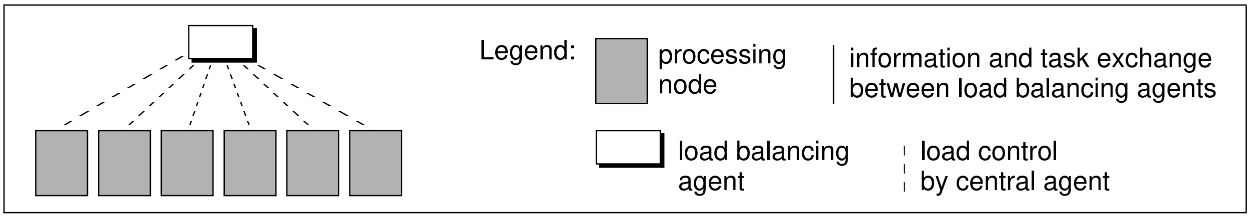

Figure 1: Centralized (global) load balancing structure.

published in Mitteilungen GI, Parallel Algorithmen und Rechnerstrukturen, GI/ITG Workshop Potsdam, 1994

- System and application state information, obtained by measurement or applications' predictions, are coherent at the central agent and need not be replicated among the system causing message traffic as well as differing age and accuracy of information.
- Load balancing can exploit global knowledge about the load state of the entire parallel system as well as about the progress and interrelationships of the applications' tasks in the system. So cooperative load balancing can be performed without contra-productive decisions that would arise from different partial information.
- Central assignment will not show any extreme load skews from the beginning. The central agent distributes tasks according to his best knowledge over the system as they arrive. In contrast to decentralized schemes, where load first arises where it was generated, in central structures load imbalance will occur only due to unpredictability of the applications' demands or due to suboptimal decision algorithms.

The limits of central load balancing (there are no logical drawbacks) are the growing number of tasks and processing nodes in the system. For central load balancing is not scalable, it will cause processing time overhead and delays which grow at least linearly with the task arrival rate and with the number of processing nodes. Load information must be collected from all processing nodes and from all tasks (if available); For each assignment decision, in fact for each decision attempt, more information is to be reviewed and more alternatives exist as the number of nodes and number of tasks grow. So the load balancing agent himself will consume more processing time that otherwise could be used to perform application tasks; further the time between issuing a task and the time it gets assigned and starts working increases. Above some certain limit of system size centralized load balancing can no longer improve the overall throughput.

1.2 Distributed Load Balancing

Decentralized (also called distributed) load balancing divides the information management as well as the assignment and migration decisions among the system, usually employing one agent per processing node (see left of figure 2). Each load balancing agent keeps track of his load and the load of his surroundings. He updates the load information of the others by broadcasting significant state changes of his own load. He may request tasks from other nodes if he feels underload (receiver initiated methods), or he may push some tasks to others if he feels they are much less loaded (sender initiated). There is no location in the system where global state information is available, and decisions are made autonomous by the agents. Obviously, this approach is scalable, because increasing the number of processors does not impose more expenses or delays to the load balancing agents: each agent still has the same number of neighbors. The other main advantage of decentralized methods is, that if the load is fairly well distributed, load balancing may not do much. Especially in situations of overall heavy load, it is important that load balancing does not add unnecessary overhead. Centralized approaches usually become more busy as the task arrival rate grows.

Note, however, that a stable saturation by long running tasks does not yet cause trouble to centralized load balancing; and bursts of task arrivals usually cause load skew. While centralized load balancing avoids this skew from beginning, distributed methods start smoothing it by task transfers. The other disadvantages have been pointed out already in section 1.1. One further observation is worth mentioning: besides the problem, that load differences between distant processors are equalized slowly, task migrations via several agents cause significant delays opposed to immediate local execution.

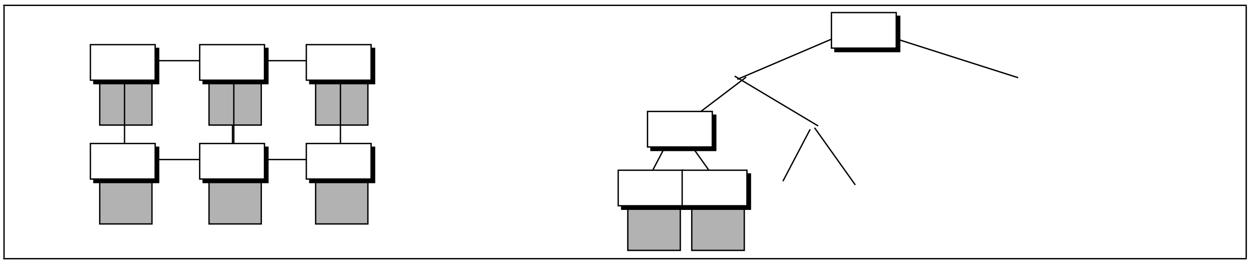

Figure 2: Decentralized (distributed) and hierarchical load balancing structure.

1.3 Mixed Load Balancing Structures

Completely decentralized (distributed) load balancing means to employ one autonomous load balancing agent per processing node. Besides the two extremes there is the approach of clustering the large parallel processing system and provide tight load balancing efforts within clusters, and less interaction and load equalization between clusters. This clustering may be continued hierarchically, or otherwise load balancing for huge systems must content itself with interaction between neighbored clusters only.

Hierarchical clustering tends to impose more information management and decisions on higher level agents (right of figure 2). So information has to be aggregated and decisions must be restricted to big tasks or groups of tasks. Usually