Computers and Electrical Engineering 35 (2009) 966–979

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Grid resource management policies for load-balancing and energy-saving by vacation queuing theory

Yin Fei [*], Jiang Changjun, Deng Rong, Yuan Jianjun

Department of Computer Science and Engineering, Tongji University, 4800 Caoan Road, Shanghai 201804, China
The Key Laboratory of "Embedded System and Service Computing", Ministry of Education, 4800 Caoan Road, Shanghai 201804, China

ARTICLE INFO

Article history:
Available online 10 March 2009

Keywords:
Grid
Resource management
Load-balancing
Energy-saving
M/G/1 vacation queue with closedown, startup

ABSTRACT

The resource management is the central component of grid system. The analysis of the workload log file of LCG including the job arrival and the resource utilization daily cycle shows that the idle sites in the Grid are the source of load imbalance and energy waste. Here we focus on these two issues: balancing the workload by transferring jobs to idle sites at prime time to minimize the response time and maximize the resource utilization; power management by switch the idle sites to sleeping mode at non-prime time to minimize the energy consume. We form the M/G/1 queue model with server vacations, startup and closedown to analysis the performance metrics to instruct the design of load-balancing and energy-saving policies. We provide our Adaptive Receiver Initiated (ARI) load-balancing strategy and power-management policy for energy-saving. The simulation experiments prove the accuracy of our analysis and the comparisons results indicate our policies are largely suitable for large-scale heterogeneous grid environment.

1. Introduction

A Grid [1] is a very large scale, generalized distributed network computing system that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains. The emergence of a variety of new applications demands that Grids support efficient data and resource management mechanisms. Resource Management [2] is central component of a grid system. Its basic responsibility is to accept requests from users, match user requests to available resources for which the user has access and schedule the matched resources. Applications may request resources from the Grid. Such resource requests are considered as jobs by the Grid. Depending on the application, the job may specify quality of service (QoS) requirements. The resource management is required to perform resource management decisions while maximizing the QoS metrics delivered to the clients [3].

With the Grid becoming a viable high-performance alternative to the traditional supercomputing environment, various aspects of effective Grid resource utilization are gaining significance. With its multitude of heterogeneous resources, a proper scheduling and efficient load-balancing across the Grid is required for improving the performance of the system. Due to uneven job arrival patterns and unequal processing capacities and capabilities, the processors in one grid site may be overloaded while others in a different grid site may be under-utilized. It is therefore desirable to dispatch jobs to idle or lightly loaded site in the grid environment to achieve better resource utilization and reduce the average job response time. This is a natural extension of the existing work on load-balancing in a traditional distributed system.

[*] Corresponding author. Address: Department of Computer Science and Technology, No. 13 Dormitory Caoan Road 4800, Tongji University Jiading Campus, Shanghai 201804, China. Tel.: +86 13601747205; fax: +86 21 69589864.
E-mail addresses: yinfei-mailbox@163.com, hhaafy@hotmail.com (F. Yin).

0045-7906/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2008.09.008

Grid computing tends to push high performance. Unfortunately, the "last drop" of performance tends to be the most expensive; that is, the last 10% increase in performance requires a disproportionally large amount of resources. The Earth Simulator, one of the world's fastest supercomputers with 640 computing sites, consumes 7 MW of power [4]. In particular, energy consumption – and the resultant heat dissipation – is becoming an important limiting factor; reducing energy saves money and increases reliability, among other things.

In this paper, we focus on the two aspects of resource management of grid environment: load-balancing and energy-saving. By analyzing the Large Hadron Collider Computing Grid (LCG) log file [5], we learn the statistical properties of the job arrival daily cycle which shows that there is an activity peak around midday. Even at that period, there are more than 10% sites remain idle which is source of load imbalancing. Meanwhile, few jobs arrive at non-prime time and at least 40% sites are idle which causes the energy consumption becoming critical concerns. Queuing systems in which the server works on primary and secondary customers arise naturally. As far as the primary customers are concerned, the server working on the secondary customers is equivalent to the server taking a vacation [6]. The vacation can be explained as idle sites receiving load for load-balancing or switching the idle sites to sleeping mode for energy-saving. We form the M/G/1 queue model with server vacations, startup and closedown to analysis the performance metric. We present our policies based on the analysis. Adaptive Receiver Initiated (ARI) load-balancing strategy for grid environment considers the job migration cost, resource heterogeneity and network dynamics when load-balancing is considered. Power management is energy-saving policy which explore the tradeoff between the site energy consume and QoS request satisfy. The simulation experiment proves the performance of our policies.

The rest of the paper is organized as follows: In Section 2, we will present the related work on load-balancing and energy-saving in recent year. In Section 3, we analyze the LCG log file. In Section 4, we develop a novel queuing analytical model. In Section 5, we introduce our two polices and give the experiment results in Section 6. Finally, Section 7 concludes the paper.

2. Related work

In general, any load-balancing algorithm consists of four basic policies – transfer policy, selection policy, location policy, and information policy [7]. The transfer policy decides if there is a need to initiate load-balancing across the system. By using workload information, it determines when a site becomes eligible to act as a sender (transfer a job to another site) or as a receiver (retrieve a job from another site). Once the transfer policy decides that a site is a sender, a selection policy selects a task for transfer. Many studies (e.g. [8,9]) have shown that (1) migration of the executing job is often difficult in practice, (2) the operation is generally expensive in most systems, and (3) there are no significant benefits of such a mechanism. The location policy determines a suitably underloaded processor. In other words, it locates complementary sites to/from which a site can send/receive workload to improve the overall system performance. Location-based policies can be broadly classified as sender initiated, receiver initiated, or symmetrically initiated [7,10–13]. In Ref. [14], five dynamic load-balancing strategies designed to support highly parallel systems are presented and compared. They are the Sender (Receiver) Initiated Diffusion (SID/RID), the Hierarchical Balancing Method (HBM) which organizes the system into a hierarchy of subsystems, the Gradient Model (GM) which employs a gradient map to guide the migration of tasks, and the Dimension Exchange Method (DEM) which requires a synchronization phase prior to load-balancing. The results indicate that the RID approach performs well, and can most easily be scaled to support highly parallel systems. Further, while balancing the load, certain types of information such as the number of jobs waiting in queue, job arrival rate, CPU processing rate, and so forth at each processor, as well as at neighboring processors, may be exchanged among the processors for improving the overall performance. Based on the information that can be used, load-balancing algorithms are classified as static, dynamic, or adaptive [7,15–17]. Based on the degree of centralization, load-scheduling algorithms could be classified as centralized or decentralized [7,17]. In the centralized approach, the load scheduling is done by only one site in the distributed system which acts as the central controller. It has a global view of the load information in the system, and decides how to allocate jobs to each of the sites. The rest of the sites will execute the jobs assigned by the controller. The centralized approach is more beneficial when the communication cost is less significant, e.g., in the shared-memory multi-processor environment. Many authors argue that this approach is not scalable, because when the system size increases, the central controller may become a system bottleneck and the single point of failure. In the decentralized approach, all sites in the distributed system are involved in making the decision. It is commonly agreed that distributed algorithms are more scaleable and have better fault tolerance. Since the load-balancing decisions are distributed, it is costly to let each site obtain the dynamic state information of the whole system. However, decentralized algorithms have the problem of communication overheads incurred by frequent information exchange between processors. Most algorithms [13,18] only use partial information stored in the local site to make a sub-optimal decision.

As to the energy-saving issue, power management techniques have been studied extensively in the context of CPU, memory and disk management in the past [19,20]. Nowadays, a major research issue in distributed mobile computing is to design efficient mechanisms for minimizing energy consumption in the wireless terminals. A Markov model to analyze the sleeping/active dynamics in sensor nodes was developed in [21,22]. As to the high performance computing, the low-power high-performance clusters have been developed to stem the ever-increasing demand for energy. Ref. [23] investigates the tradeoff between energy and performance (execution time) for HPC applications on a real small-scale power-scalable cluster.