Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 558–568
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Efficient scheduling algorithm for component-based networks

Seokcheon Lee[a,*], Soundar Kumara[b], Natarajan Gautam[c]

[a] School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA
[b] Department of Industrial Engineering, The Pennsylvania State University, University Park, PA 16802, USA
[c] Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843, USA

Received 25 February 2006; received in revised form 28 June 2006; accepted 16 September 2006
Available online 27 October 2006

Abstract

In a grid computing environment, it is important to be capable of agilely quantifying the quality of service achievable by each alternative composition of resources. This capability is an essential driver to not only efficiently utilizing the resources but also promoting the virtual economy. In this paper, we design an efficient scheduling algorithm of minimizing completion time for component-based networks. The performance of the network is a function of resource assignment and resource allocation. Resource assignment assigns components to available machines and resource allocation allocates the resources of each machine to the residing components. Though similar problems can be found in the multiprocessor scheduling literature, our problem is different especially because the components in our networks process multiple tasks in parallel with their successor or predecessor components. The designed algorithm is simple but effective since it incorporates the fact that the components in a network can be considered independent under a certain resource allocation policy.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Multiprocessor scheduling; Grid computing; Component-based network; Completion time

1. Introduction

Individual systems are becoming interoperable by virtue of the *Grid technology* which provides inexpensive access to large computational resources across institutional boundaries [1–3]. Cost and quality considerations may force a large number of customers to look for grid resources to deal with their own computing problems. In a grid environment, a problem is processed by allocating multiple resources. Since there can be several alternative compositions of resources for a given problem, virtual markets will play a critical role in coordinating huge amount of economic entities such as customers and resource providers. There are various market mechanisms such as OCEAN [4], Compute Power Market [5], and Nimrod/G [6], proposed for the large-scale virtual economy. However, one essential enabler of such markets is the ability to agilely quantify the quality of service (QoS) achievable by each alternative composition of resources. Without such a capability, the alternatives cannot be evaluated in a timely manner and the virtual economy will fail to efficiently utilize the resources.

There can be various ways of defining QoS depending on the nature of the problems. We consider a class of problems whose QoS is determined by *completion time* for generating a solution. The completion time (also called makespan) is one of the most widely studied objectives in the multiprocessor scheduling literature. Regarding the problem solving structure we adopt a *component-based* architecture as a general framework. A component is a reusable program element. Component technology utilizes the components so that developers can build systems needed by simply defining their specific roles and wiring them together [7,8]. In networks with component-based architecture, each component is highly specialized for specific tasks.

In this paper, we design an efficient scheduling algorithm of minimizing the completion time for the component-based networks. A problem given to the network is decomposed in terms of *root tasks* for some components and those tasks are propagated through a task flow structure to other components. Since a problem can be decomposed with respect to space, time, or both, a component can have multiple root tasks that can be

* Corresponding address: 3368 Peppermill Dr., West Lafayette, IN 47906, USA. Tel.: +1 765 494 5419; fax: +1 765 494 1299.
E-mail addresses: lee46@purdue.edu, stonesky@gmail.com (S. Lee), skumara@psu.edu (S. Kumara), gautam@tamu.edu (N. Gautam).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.09.002

considered independent and identical in their nature. For a given set of resources the performance can vary depending on the way of utilizing the resources. *Resource assignment* assigns components to available machines with a set of constraints. Some components may not be separable to different machines and may only be allowed on to specific machines. Given a resource assignment, there can be multiple components in a machine sharing the machine's resources. So, *resource allocation* can play an important role in controlling the performance of a network. These two facilities determine the performance of a network and the minimal completion time represents the QoS achievable by an alternative resource composition.

UltraLog (http://www.ultralog.net) networks [9–13], implemented in Cougaar (Cognitive Agent Architecture: http://www.cougaar.org) developed by DARPA (Defense Advanced Research Project Agency), are the instances. In Cougaar a software system comprises of clusters and a cluster of components (called plugins). The task flow structure in those systems is that of components as a combination of intra-cluster and inter-cluster task flows. UltraLog networks are the next generation military logistics information systems. Each cluster in these networks represents an organization of military supply chain and has a set of components specialized for each functionality (allocation, expansion, inventory management, etc.) and class (ammunition, water, fuel, etc.).

The objective of an UltraLog network is to produce a logistics plan for a given military operation, which is an aggregate of individual schedules built by components. An operation is transformed into logistics requirements and the requirements are decomposed into root tasks (one task per day) for designated components. As a result, a component can have hundreds of root tasks depending on the horizon of an operation and thousands of tasks as the root tasks are propagated. As the scale of operation increases there can be thousands of clusters (tens of thousands of components) in hundreds of machines working together to generate a logistics plan. One of the important performance criteria of these networks is the (plan) completion time. This metric directly affects the performance of military operations.

Though similar problems exist in the multiprocessor scheduling literature [14–21], they have limitations in addressing this novel scheduling problem. They commonly consider the cases where each component only has to process one task after all of its predecessors complete their tasks. In contrast, a component in the networks under consideration processes multiple tasks in parallel with its successors or predecessors. The multiprocessor scheduling problems are NP-complete for most of its variants [22–24]. The scheduling problem we are addressing is even harder due to the parallelism. To address this complex scheduling problem there is a need to facilitate some simple but effective scheduling algorithms.

The organization of this paper is as follows. In Section 2 we formally define the scheduling problem in detail. Sections 3 and 4 design the scheduling algorithm. After describing the overall procedure of applying the algorithm in Section 5, we present empirical results in Section 6. Finally, we discuss implications and possible extensions of our work in Section 7.

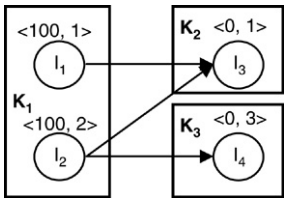

Fig. 1. An example network composed of four components in three machines. $\langle a, b\rangle$ denotes the number of root tasks and CPU time per task at the residing machine.

2. Problem statement

In this section we formally define the problem by detailing component-based network, resource assignment, and resource allocation. We focus on computational CPU resources assuming that the system is computation-bounded.

2.1. Component-based network

A component-based network consists of a set $\boldsymbol{I} = \{i : i \in \boldsymbol{I}\}$ of software components. Task flow structure of the network, which defines precedence relationship between components, is an arbitrary directed acyclic graph. A problem given to the network is decomposed in terms of *root tasks* for some components and those tasks are propagated through a task flow structure. Each component processes one of the tasks in its queue (which has root tasks as well as tasks from predecessor components) and then sends it to successor components. We denote the number of root tasks of component i as rt_i. Fig. 1 shows an example network comprised of four components residing in three machines. In the figure $\langle a, b\rangle$ represents rt_i and CPU time per task at the residing machine. Each of I_1 and I_2 has 100 root tasks. I_3 and I_4 have no root tasks but they have 200 and 100 tasks from the corresponding predecessors.

2.2. Resource assignment

There is a set $\boldsymbol{K} = \{k : k \in \boldsymbol{K}\}$ of available machines and P_i^k represents CPU time per task of component i at machine k reflecting computation speed differences between heterogeneous machines. Considering that some components may not be separable to different machines, we define a set $\boldsymbol{J} = \{j : j \in \boldsymbol{J}\}$ of clusters and denote the components of a cluster j as $\boldsymbol{M}_j$. Each component is a member of one of the clusters and the components in a cluster should be assigned to the same machine. Also, each cluster is allowed to specific machines and we denote the assignable machine set of cluster j as $\boldsymbol{N}_j$. We define resource assignment variable set $\boldsymbol{X} = \{x_{jk} : j \in \boldsymbol{J}, k \in \boldsymbol{K}\}$ in which x_{jk} is 1 if cluster j is assigned to machine k and 0 otherwise. The constraints of resource assignment variables are as in (1).

□ *Resource assignment constraints*

$$\begin{aligned} &\sum_{k\in\boldsymbol{N}_j} x_{jk} = 1 \quad \text{for all } j \in \boldsymbol{J} \\ &\sum_{k\notin\boldsymbol{N}_j} x_{jk} = 0 \quad \text{for all } j \in \boldsymbol{J} \\ &x_{jk} \in \{0, 1\} \quad \text{for all } j \in \boldsymbol{J} \quad \text{and } k \in \boldsymbol{K}. \end{aligned} \tag{1}$$

دریافت فوری ← متن کامل مقاله

ISIArticles
مرجع مقالات تخصصی ایران

✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
✓ امکان دانلود نسخه ترجمه شده مقالات
✓ پذیرش سفارش ترجمه تخصصی
✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
✓ دانلود فوری مقاله پس از پرداخت آنلاین
✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات