Future Generation Computer Systems 27 (2011) 1083–1091

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A stochastic scheduling algorithm for precedence constrained tasks on Grid

Xiaoyong Tang [a,b,*], Kenli Li [b], Guiping Liao [a], Kui Fang [a], Fan Wu [b]

[a] Information Science and Technology College, Hunan Agricultural University, Changsha, China

[b] School of Information Science and Engineering, Hunan University, Changsha, China

ARTICLE INFO

Article history:
Received 30 September 2010
Received in revised form
28 March 2011
Accepted 11 April 2011
Available online 19 April 2011

Keywords:
Stochastic scheduling
Grid
Precedence constrained tasks
Makespan

ABSTRACT

This paper addresses the problems in scheduling a precedence constrained tasks of parallel application with random tasks processing time and edges communication time on Grid computing systems so as to minimize the makespan in stochastic environment. This is a difficult problem and few efforts have been reported on its solution in the literature. The problem is first formulated in a form of stochastic scheduling model on Grid systems. Then, a stochastic heterogeneous earliest finish time (SHEFT) scheduling algorithm is developed that incorporates the expected value and variance of stochastic processing time into scheduling. Our rigorous performance evaluation study, based on randomly generated stochastic parallel application DAG graphs, shows that our proposed SHEFT scheduling algorithm performs much better than the existing scheduling algorithms in terms of makespan, speedup, and makespan standard deviation.

1. Introduction

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components make it possible to construct large-scale high-performance Grid computing systems. These technical opportunities enable the sharing, selection, and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science, engineering, and commerce [1,2]. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. The scheduling problem deals with the coordination and allocation of resources so as to efficiently execute the users' applications. Stochastic Grid parallel applications are submitted by users and generally independent of each other, which request systems' services for their execution. The single parallel application is usually consisted of precedence constrained tasks that they generally require the use of different kinds of resources, e.g., computation, communication (network), storage resources, or specific instruments. A popular representation of a parallel application is the directed acyclic graph (DAG) in which the nodes represent application tasks and the directed arcs or edges represent inter-task dependencies, such as task's precedence. In some cases, weights can be added to nodes and edges to express computational costs and communicating costs, respectively. Scheduling aims at meeting user demands (e.g., in terms of makespan, sum of weighted completion times, maximum lateness, and number of tardy jobs) and the objectives represented by the resource providers (e.g., in terms of profit and resource utilization efficiency), while maintaining a good overall performance (throughput) for the Grid computing systems [3]. It is widely known that the problem of finding the optimal schedule is NP-complete [4] in the general case. Therefore, heuristics can be used to obtain a sub-optimal scheduling rather than parsing all possible schedules. These methods may obtain suboptimal results, but they are much computational cheaper.

List scheduling is a very popular method for precedence constrained task scheduling based on the DAG model. The basic idea of list scheduling is to assign priorities to the tasks of the DAG and place the tasks in a list arranged in descending order of priorities. A task with a higher priority is scheduled before a task with lower priority, and ties are broken using some method. An important issue in DAG scheduling is how to rank (or weigh) the tasks and edges (when communication delay is considered). The rank of a task is used as its priority in the scheduling. Once the tasks and edges are ranked, task-to-resource assignment can be found by considering the following two problems to minimize the makespan: how to parallelize those tasks having no precedence orders in the graph and how to make the time cost along with the critical path in the DAG as small as possible.

There are several effective Grid heuristics scheduling algorithms such as mapping heuristic (MH) [5], dynamic critical path(DCP) [6], levelized-min time (LMT) algorithm [7], dynamic-level scheduling (DLS) algorithms [8], critical-path-on-a-machine

* Corresponding author at: Information Science and Technology College, Hunan Agricultural University, Changsha, China.
E-mail addresses: tang_313@163.com (X. Tang), lkl510@263.net (K. Li), lgpxf@hunau.net (G. Liao), fk@hunau.net (K. Fang), wufan_whu@163.com (F. Wu).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.04.007

(CPOP) algorithms and heterogeneous earliest-finish-time (HEFT) algorithm [9–12]. The HEFT algorithm selects the task with the highest upward rank (an upward rank is defined as the maximum distance from the current node to the exiting node, including the computational cost and communication cost) at each step. The selected task is then assigned to the processor which minimizes its earliest finish time with an insertion-based approach which considers the possible insertion of a task in an earliest idle time slot between two already-scheduled tasks on the same resource. The time complexity of HEFT is $O(m \times v^2)$, where m is the number of processors and v is the number of tasks. HEFT algorithm significantly outperforms DLS, DCP, LMT, MH, and CPOP algorithms in terms of average makespan, speedup, *etc.,* [9].

Most of the above researches assume that parameters such as task processing time and communication time between precedence constrained tasks are fixed and deterministic which are known advance. However, in real-world problems, it usually does not suffice to find good schedules for fixed deterministic processing time, since tasks usually contain conditional instructions and/or operations that could have different execution times for different inputs [13–16]. A natural step to tackle this problem is to consider stochastic scheduling, that is, to interpret processing time and communication time as random variables and to measure the performance of an algorithm by its expected objective value. In this paper, let $V\{v_1, \ldots, v_n\}$ be a set of tasks of a parallel application that have to be scheduled on heterogeneous Grid systems with m heterogeneous machines (or unrelated parallel machines) so as to minimize the schedule length (makespan). Any machine can process at most one job at a time, and every job has to be processed on one of the m machines. In contrast to the deterministic version, the crucial assumption is that the task processing time $w(v_i)$ and communication time $w(e_{i,j})$ are not known in advance. Instead, one assumes that the task processing time and communication time are random variables from which we are just given their distribution function. Throughout the paper, task durations are supposed to be stochastically independent.

Since there are a number of scheduling models with different conditions considered in stochastic scheduling [17–19,13,20,14, 21–24], it is convenient to refer to them in the notation of Graham et al. [23] and Allahverdi et al. [24]. Each problem is denoted by the standard three-field notation $\alpha|\beta|\gamma$ with the following intended meaning.

- The field α specifies the machine environment. For instance, $\alpha = P$ denotes the model with identical parallel machines, $\alpha = Q$ denotes the problem where machines have different speeds s_k, and the processing time of task v_i on machine k is $w(v_i)/s_k$, and $\alpha = 1$ is used for problems with a single machine.
- The field β contains the task characteristics. It can be empty, which implies the default of non-preemptive, precedence constrained tasks. Possible entries are, among many others, *prec* for precedence constrained tasks, or *pmtn* for preemptive tasks.
- The field γ denotes the objective function. It is generally a function of the completion times of the tasks. For the total weighted completion time, we write $\gamma = \Sigma w_i C_i$. For the makespan $\gamma = C_{\max}$.

As an example, $Q|v_i \sim stoch, prec|E[C_{\max}]$ is the stochastic scheduling problem to minimize the makespan of precedence constrained tasks on Grid heterogeneous parallel machines.

Precedence constraints between tasks play a particularly important role in most real-world parallel applications. Therefore, it would be both of theoretical and of practical interest to incorporate those constraints into stochastic scheduling. Considering stochastic tasks with precedence constraints, such as the stochastic DAG model, Skutella and Uetz obtained a performance guarantee of $(1+\varepsilon)\left(1+\frac{m-1}{m\varepsilon}+\max\left\{1, \frac{m-1}{m}\Delta\right\}\right)$ for $P|v_i \sim stoch, prec|E[w_iC_i]$ (here, ε is an arbitrary constant) [21], where they assume that the squared coefficients of variation of all processing time $w(v)$ are bounded by some constant Δ that is expressed as $\mathrm{Var}(w(v))/E[w(v)]^2 \le \Delta$, for all $v \in V$. For $\Delta \le 1$, the performance guarantees can be simplified to $3+2\sqrt{2}-(1+\sqrt{2})/m$. In most case, the schedule length (makespan) is the key objective of stochastic scheduling. If all tasks are identically and exponentially distributed, Chandy and Reynolds show that a *highest level first* policy minimizes the expected makespan $E[C_{\max}]$ for the special case of two machines and in-tree precedence constraints [22]. Here, the level of a task simply denotes the number of successors in the precedence graph. They also give counterexamples for the case of more than two machines. Bruno extends this result by proving that highest level first even minimizes the makespan stochastically [25]. Pinedo and Weiss extend Chandy and Reynolds result to the case where the exponential processing time distributions of tasks of different levels may be different [26]. For the case of more than two machines, Papadimitriou and Tsitsiklis prove asymptotic optimality of a highest level first policy [27]. Their assumptions are that all processing times are independent and identically distributed, and this distribution has finite moments of any order. This is particularly true for the exponential distribution. Although somewhat counter-intuitive, results for out-trees (outforests) seem to be much more difficult to obtain: Coffman and Liu show that a preemptive highest level first policy is optimal to minimize the expected makespan for some rather restricted scenarios [28]. These results, however, are restricted to somewhat artificial precedence constrained tasks (such as in-tree precedence constraints) and execution on identical machines.

To the best of our knowledge, no results of scheduling algorithm were previously known for $Q|v_i \sim stoch, prec|E[C_{\max}]$ on Grid with m heterogeneous machines. In recognition of this, we drive a stochastic scheduling model for this problem. In order to effectively scheduling precedence constrained stochastic tasks, we propose a stochastic heterogeneous earliest finish time (SHEFT) scheduling algorithm, which incorporate the stochastic attribute, such as expected value and variance, of task processing time and edge communication time into scheduling. At last, we establish a stochastic scheduling simulation experiment platform and compare SHEFT with two well-known scheduling algorithms HEFT and DCP.

The rest of the paper is organized as follows: In Section 2, we describe the definitions and background of stochastic scheduling problem on Grid computing systems. In Section 3, we propose a SHEFT scheduling algorithm. In Section 4, we verify the performance of the proposed algorithm by comparing the results obtained from performance evaluation and conclude the paper in Section 5.

2. Definitions and background

2.1. The Grid scheduling architecture

A Grid is a system of high diversity, which geographically distributed sites interconnect through WAN. We define a site as a location that contains many computing resources of different processing capabilities. Heterogeneity and dynamicity cause resources in grids to be distributed or in clusters rather than uniformly. We can also generalize a scheduling process in the Grid into two stages: resource discovering and filtering and resource selecting and scheduling according to certain objectives. As a study of scheduling algorithms is our primary concern here, we focus on the second step. Based on these observations, Fig. 1 depicts a model of Grid scheduling systems.

As Grid resources are under the control of local schedulers, which provide access point to the resources, the central Grid scheduler must be built on top of the existing local schedulers. Basically, a central Grid scheduler (GS) receives applications