J. Parallel Distrib. Comput. 109 (2017) 155–177

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Prior node selection for scheduling workflows in a heterogeneous system

Hidehiro Kanemitsu [a,*], Masaki Hanada [b], Hidenori Nakazato [c]

[a] Global Education Center, Waseda University, 1-6-1, Nishiwaseda, Shinjuku, Tokyo 169-8050, Japan
[b] Department of Information Systems, Tokyo University of Information Sciences, 4-1, Onaridai, Wakaba-ku, Chiba 265-8501, Japan
[c] Department of Communications and Computer Engineering, Waseda University, 3-14-9 Okubo, Shinjuku-ku 169-0072, Japan

HIGHLIGHTS

- Prior node selection algorithm for scheduling workflows, called LBCNS is proposed.
- Objective of LBCNS is to minimize the schedule length while fairly scheduling each job.
- LBCNS can be applied for any task scheduling algorithms including list scheduling algorithms.
- Experimental results show that the schedule length, efficiency, and fairness are improved.

ARTICLE INFO

Article history:
Received 26 November 2016
Received in revised form 24 May 2017
Accepted 7 June 2017
Available online 19 June 2017

Keywords:
DAG
Heterogeneous system
Processor grouping
Node grouping
Task scheduling
Workflow scheduling

ABSTRACT

Many workflow scheduling algorithms for heterogeneous systems have been developed to satisfy multiple requirements such as minimizing schedule length while maximizing throughput. In particular, in list-based scheduling approaches, the schedule length depends on the given nodes as well as the task allocation and ordering policies. This is because the scheduling priority is derived by averaging the execution time and communication time of the given nodes. If the set of nodes can be adjusted before the scheduling tasks, a small schedule length can be achieved. In this paper, we propose a prior node selection algorithm, called lower bound based candidate node selection (LBCNS) to select a subset of given nodes to minimize the schedule length while fairly scheduling each job. Our proposal has two approaches: (i) LBCNS_DEFAULT, which considers the job characteristics and each node's performance, and (ii) priority-based LBCNS, which additionally takes each scheduling priority into account for a dedicated task scheduling algorithm.

The experimental results of extensive simulations show that LBCNS_DEFAULT has the best fairness for scheduling multiple workflow jobs, while priority-based LBCNS achieves the minimum schedule length with the highest efficiency for a single workflow job and multiple workflow jobs.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Recent distributed processing schemes, e.g., grid and cloud systems, enable submitted parallelizable jobs to be executed by idle computational resources in order to distribute the workloads imposed by those jobs. For example, resource provisioning in a cloud system and a node[1] grouping strategy in a grid are needed to minimize the response time (i.e., the schedule length), maximize the throughput, or meet other requirements. In particular, the virtual machine (VM) selection policy in a resource provisioning policy has an impact on the response time. In light of the independent task scheduling used in a heterogeneous system, a node grouping method takes the performance of each node into account, i.e., each node in a group has similar processing speed or every group has a similar average processing speed.

The above examples are based on the question of how the subset of nodes should be determined from the given idle nodes to satisfy the predefined objective functions. However, no theoretical approach to this problem has yet been established. In a MapReduce architecture, although how to derive the optimal number of map tasks or nodes to minimize the response time is a challenging issue, the actual approach samples the historical execution information

* Corresponding author.
E-mail addresses: kanemih@ruri.waseda.jp (H. Kanemitsu), mhanada@rsch.tuis.ac.jp (M. Hanada), nakazato@waseda.jp (H. Nakazato).

[1] In this paper, the term "node" refers to a device with one execution mechanism and one way to communicate with other devices. In the field of task scheduling, typically the term "processor" is used for such a device. However, in this paper, we use "node" in order to emphasize that each device or processor can communicate over the network.

http://dx.doi.org/10.1016/j.jpdc.2017.06.005
0743-7315/© 2017 Elsevier Inc. All rights reserved.

of the system [22]; that is, the approach investigates the correlation between the number of map tasks and actual execution time in advance. Such an approach is time consuming. Furthermore, the workflow or directed acyclic graph (DAG) scheduling problem is *NP*-complete [9]. This problem must take into account the processing speed and communication bandwidth of each node as well as the workflow characteristics such as precedence constraints among tasks. Furthermore, whether the job is data or computationally intensive must be considered to select the subset of given nodes. However, to our knowledge, no subset selection approach for workflow scheduling exists.

As cost-effective task scheduling algorithms in a workflow for heterogeneous systems, list-based scheduling algorithms are well-known [5,16,28,29]. In these algorithms, the scheduling priority assigned to a task is derived from the processing speeds of all nodes and all communication bandwidths in the given set of nodes. It follows from this that the response time (hereafter, the "schedule length") depends on not only the performance of the assigned nodes but also that of the unassigned nodes. This characteristic indicates that the schedule length has space for further improvement depending on the policy to determine the subset of the given nodes.

In practice, multiple jobs can be submitted to a scheduling system. In such a case, if an appropriate subset of the given nodes is determined for one job, the schedule length (i.e., the maximum response time for all jobs) can be prolonged by the inappropriate resource allocation for the other jobs caused by a shortage of appropriate nodes. Thus, it is necessary to fairly allocate resources among all jobs in order to reduce the maximum schedule length of the submitted jobs. Here, we consider the case in which one or more workflow jobs have been stored in the job pool in the system. They must be scheduled simultaneously to minimize the maximum schedule length among the jobs. In such a case, these multiple jobs can be integrated into a larger job, where dummy start and end tasks are added to the workflow. It can then be scheduled by conventional a scheduling algorithm, and therefore the schedule length, i.e., the maximum schedule length of those jobs is effectively reduced, while the slowdown, which is an index of fairness in terms of assignment resources, is sacrificed [31]; that is, such an integration approach can be further improved if each resource is fairly assigned.

The current challenges of node selection for workflow scheduling can be summarized as follows: (i) the scheduling priority for each task in a job is not accurate for minimizing the schedule length because it is derived from all given nodes, and (ii) whether selecting nodes before scheduling tasks can affect on the fairness in terms of slowdown or not is unknown. In the context of scheduling tasks in case of multiple workflows, minimizing the schedule length cannot always lead to a fair scheduling. However, before scheduling tasks, if we can select the nodes having similar performance each other and also each having a great effect on minimizing the schedule length, such a prior node selection can take an important role for resource provisioning.

In this paper, we propose a candidate node selection algorithm, called the lower bound based candidate node selection (LBCNS) for determining the subset of given idle nodes that achieves the minimum schedule length while fairly scheduling each job in a heterogeneous distributed system. The key concept behind LBCNS is to select a set of nodes that obtain good performance by considering both each workflow characteristic and the performance of the given idle nodes. As a result, only a set of nodes that can help minimize the schedule length are selected as assignment candidates. In particular, the running time without idleness of node p_i is defined as $\varepsilon(p_i)$. The subset of nodes with small $\varepsilon(p_i)$ is selected as the set of assignment candidates for scheduling tasks. Moreover, we propose dedicated node selection algorithms for state-of-the-art task scheduling algorithms that take each scheduling priority derivation policy into account to further improve the schedule length. Experimental results obtained via simulation show that a schedule length that is better than that of other node selection policies can be obtained by LBCNS for single and multiple workflow jobs. Furthermore, we show that the efficiency, which is defined as the speed-up ratio divided by the number of assigned nodes, can be reduced. Furthermore, unfairness, defined as the variance in terms of slowdown among jobs, can be reduced.

The remainder of this paper is organized as follows. Section 2 describes the assumed system and model. Related work in node grouping methods and task scheduling algorithms are reviewed in Section 3. We then present the proposed node selection algorithm LBCNS in Section 4. Extensions of LBCNS for specific task scheduling algorithms are then presented in Section 5. Experimental results are described in Section 6. Finally, we conclude this paper in Section 7.

2. Assumptions

2.1. Job model

We assume a job to be executed on the nodes is a DAG, or a workflow job. Let $G = (V, E)$ be the DAG or workflow, where V is the set of tasks, E is the set of edges, i.e., data communications among tasks, and V_{cls} is the set of assignment units, where each assignment unit contains one or more tasks. An ith task is denoted as n_i. Let $w(n_i)$ be the size of n_i, i.e., $w(n_i)$ is the sum of the time units needed to process the task by the reference node. We define the data dependency and direction of data transfer from n_i to n_j as $e_{i,j}$. Further, $c(e_{i,j})$ is the sum of time units taken to transfer data from n_i to n_j over the reference communication link.

One constraint imposed by a workflow is that a task cannot begin to execute until it receives all the data from its predecessor tasks. For instance, $e_{i,j}$ indicates that n_j cannot be started until the data from n_i arrives at the node that will execute n_j. Further, $pred(n_i)$ is the set of immediate predecessors of n_i, and $suc(n_i)$ is the set of immediate successors of n_i.

In the context of multiple job scheduling, each job is defined as G_i, where $1 \leq i \leq J$, i.e., there are J workflow jobs in a job pool and they must be scheduled simultaneously.

2.2. System model

For the system model, we assume that each node is completely connected to other nodes with non-identical processing speeds and communication bandwidths. The set of nodes is expressed as $P = \{p_1, p_2, \ldots, p_N\}$, and let the set of processing speeds be $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$. The execution time when n_k is processed at the speed of α_i is expressed as

$$t_p(n_k, \alpha_i) = \frac{w(n_k)}{\alpha_i}. \tag{1}$$

This assumes that the execution time is inversely proportional to the processing speed, i.e., the throughput of each processor in a node using a uniform execution model. Let the set of communication bandwidths be $\beta = \{\beta_1, \beta_2, \ldots \beta_N\}$ when the communication bandwidth of the reference communication link is set to 1. If $c(e_{i,j})$ is sent from p_k to p_l, the communication time is defined by the communication speed $L_{k,l}$, where $L_{k,l} = \min\{\beta_k, \beta_l\}$. This assumes that the communication time is dominated by the smallest communication bandwidth through every device between p_k and p_l, provided that the bandwidth on every router is supposed to be larger than $L_{k,l}$ for $\forall\{p_k, p_l\} \in P$. Thus, the communication time of $e_{i,j}$ from p_k to p_l is defined as

$$t_c(e_{i,j}, L_{k,l}) = O_k + \frac{c(e_{i,j})}{L_{k,l}}, \tag{2}$$

where O_k is the setup time for communication at p_k. Typically, O_k is negligible with respect to $c(e_{i,j})/L_{k,l}$.

Download English Version:

https://daneshyari.com/en/article/4951608

Download Persian Version:

https://daneshyari.com/article/4951608

Daneshyari.com