J. Parallel Distrib. Comput. 84 (2015) 76–86

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Scalable linear programming based resource allocation for makespan minimization in heterogeneous computing systems

Kyle M. Tarplee [a,*], Ryan Friese [a], Anthony A. Maciejewski [a], Howard Jay Siegel [a,b]

[a] Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, United States
[b] Department of Computer Science, Colorado State University, Fort Collins, CO 80523, United States

HIGHLIGHTS

- We present a novel scheduling algorithm for heterogeneous computing environments.
- Uses groupings of similar tasks and machines to reduce the computational complexity.
- Computes upper and lower bounds on the optimal makespan.
- Schedule approaches a lower bound on the makespan as the number of tasks increases.
- Scheduling algorithm run time scales linearly with the number of tasks.

ARTICLE INFO

Article history:
Received 12 September 2014
Received in revised form
15 March 2015
Accepted 7 July 2015
Available online 20 July 2015

Keywords:
High performance computing
Scheduling
Resource management
Bag-of-tasks
Heterogeneous computing
Linear programming

ABSTRACT

Resource management for large-scale high performance computing systems poses difficult challenges to system administrators. The extreme scale of these modern systems require task scheduling algorithms that are capable of handling at least millions of tasks and thousands of machines. Highly scalable algorithms are necessary to efficiently schedule tasks to maintain the highest level of performance from the system. In this study, we design a novel linear programming based resource allocation algorithm for heterogeneous computing systems to efficiently compute high quality solutions for minimizing makespan. The novel algorithm tightly bounds the optimal makespan from below with an infeasible schedule and from above with a fully feasible schedule. The new algorithms are highly scalable in terms of solution quality and computation time as the problem size increases because they leverage similarity in tasks and machines. This novel algorithm is compared to existing algorithms via simulation on a few example systems.

1. Introduction

Today's high performance computing (HPC) systems often have hundreds of thousands of machines. The need for these extremely large HPC systems is driven by increasingly larger HPC workloads comprising potentially millions of tasks. The increase in computational capability of HPC environments can only be maintained if the tasks can be intelligently assigned to machines quickly. Therefore, there is a growing need for efficiently scheduling tasks to machines in such large-scale environments.

Our work considers a common scheduling model where users submit a set of independent tasks known as a *bag-of-tasks* [7]. We assume that the full bag-of-tasks is known a priori [7] (i.e., *static scheduling*), a task can be scheduled to execute on only one machine, and machines may only process one task at a time. The HPC environments of primary interest have highly heterogeneous tasks and machines and are known as heterogeneous computing (HC) systems [17].

HC systems often have some special-purpose machines that can perform specific tasks quickly, while other tasks might not be able to run on them. Another cause of heterogeneity is differing computational requirements, input/output bottlenecks, or memory limitations. For instance, a task that runs on a GPU might execute much faster than the same task run on a general-purpose machine. The heterogeneity in execution time of the tasks provides the scheduler with degrees of freedom to greatly decrease the maximum of all the task finishing times, known as the *makespan*, compared to

* Corresponding author.
E-mail addresses: kyle.tarplee@colostate.edu (K.M. Tarplee), ryan.friese@colostate.edu (R. Friese), aam@colostate.edu (A.A. Maciejewski), hj@colostate.edu (H.J. Siegel).

http://dx.doi.org/10.1016/j.jpdc.2015.07.002
0743-7315/

a naïve scheduling algorithm. The makespan is a very common offline scheduling objective [14,27]. The algorithms in this work can be adapted to online batch mode scheduling algorithms where the makespan is minimized for each batch of tasks. When a new task arrives or a task is removed from the batch because it is now running on a machine, the schedule for the batch of tasks can be recomputed.

Finding the optimal schedule for this static scheduling problem is NP-Hard in general [13]. Therefore we seek to design algorithms that find near-optimal solutions relatively quickly.

In this study, a set of efficient and scalable algorithms are proposed that schedule heterogeneous tasks to a set of heterogeneous machines with the goal of minimizing makespan. These algorithms compute a lower bound using linear programming (LP) and then quickly compute the fully feasible schedule. The algorithms have very small run times, find schedules that have solutions closer to optimal as the problem size increases, and good asymptotic algorithmic complexity. This approach is therefore very well suited to large-scale HPC environments. Often large computing systems are composed of heterogeneous clusters of homogeneous machines. The proposed algorithms decompose naturally into a high level scheduler that determines which cluster should process the task followed by a lower level scheduler per cluster that assigns the task to a particular machine.

In summary the contributions of this paper are:

1. the formulation and evaluation of an algorithm that efficiently computes a tight lower bound on the makespan,
2. the design and evaluation of a recovery algorithm to take the lower bound solution and compute a near-optimal feasible schedule,
3. a comparison to other heuristic scheduling algorithms, and
4. an evaluation and analysis of the scaling properties of the proposed algorithms and algorithms from the literature.

The rest of this paper is organized as follows. First an algorithm for minimum makespan scheduling is presented in Section 2. Section 3 describes the nominal HC system and workload used for simulations and evaluation. Bounds on the solution quality are provided by the algorithm and are discussed in Section 4. In Section 5, we compare this algorithm to other heuristic algorithms. The applicability of the algorithm to very large-scale problems is shown in Section 6 along with simulation results for very large system configurations. We discuss related work in Section 7, and Section 8 concludes this study and presents some ideas for future work.

2. Algorithm design

2.1. Approach

The fundamental approach of this paper is to apply divisible load theory (DLT) [5,4] to ease the computational requirements of calculating a solution to the makespan scheduling problem. The technique operates in two steps to calculate the lower and upper bounds on makespan. The first step uses DLT, where we assume a single task is allowed to be divided and scheduled onto any number of machines, to calculate the lower-bound solution. After the lower-bound solution is computed, a two-phase algorithm is used to recover a feasible solution from the infeasible lower-bound solution. The feasible solution will be shown empirically to be a tight upper bound on the optimal makespan.

Heterogeneous computing (HC) systems often have groups of machines, typically purchased at the same time, that have identical or very similar performance characteristics. This allows one to group these similar machines (for the purposes of analysis) into a unique machine type. Machines belonging to a *machine type* have virtually indistinguishable performance properties with respect to the workload. Machines of the same type may differ vastly in feature sets so long as the performance of the tasks under consideration are not affected. Tasks often exhibit natural groupings as well. Tasks of the same *task type* are often submitted many times to perform statistical simulations and other repetitive jobs. Having groupings for tasks and for machines permits less profiling effort to estimate the run time for each task on each machine.

Traditionally the static scheduling problem is posed as assigning all tasks to all machines. The classic formulation is not well suited for recovering a high quality feasible solution from a relaxation of the problem. The decision variables in the classic formulation are binary valued (a task is assigned or not assigned to a machine), and rounding a real value from the lower bound to a binary value can change the objective significantly. Complicated rounding schemes are necessary to iteratively compute a suitable solution. Rather than addressing the problem of assigning all tasks to all machines, we pose the problem as determining the number of tasks of each type to assign to machines of each type. With this modification, decision variables will be large integers $\gg 1$, resulting in only a small error to the objective function when rounding to the nearest integer. This approximation is most accurate when the number of tasks assigned to each machine type is large. In addition to easing the recovery of the integer solution, another benefit of this formulation is that it is significantly less computationally intensive due to solving the higher level assignment of tasks types to machine types with DLT, before solving the fine-grain assignment of individual tasks to machines. As such, this approach can be thought of as a hierarchical solution to the static scheduling problem.

2.2. Lower bound

The lower bound on the makespan is given by the solution to an LP problem and is formulated as follows. Let there be T task types and M machine types. Let T_i be the number of tasks of type i and M_j be the number of machines of type j. Let μ_{ij} be the number of tasks of type i assigned to machine type j, where $\mu_{ij} \in \mathbb{R}$ is the primary decision variable in the optimization problem. Let **ETC** be a $T \times M$ matrix where ETC_{ij} is the *estimated time to compute* a task of type i on a machine of type j. The **ETC** matrix is frequently used in scheduling algorithms (e.g., [10,15,7,8,16]). **ETC** is generally obtained from historical data in real environments.

The lower bound on the finishing time of the machines of a given type is found by allowing tasks assigned to a machine type to be divided among all machines to ensure the minimal finishing time. With this conservative approximation, all machines of type j finish at the same time. The finishing time of all machines of type j for divisible tasks, denoted by F_j, is given by

$$F_j = \frac{1}{M_j} \sum_i \mu_{ij} ETC_{ij}. \tag{1}$$

Throughout this work, sums over i always go from 1 to T and sums over j always go from 1 to M, thus the ranges are omitted. Given that F_j is a lower bound on the finishing time for a machine type, the tightest lower bound on the makespan is

$$MS_{LB} = \max_j F_j. \tag{2}$$

The resulting optimization problem for the lower bound is:

$$\begin{aligned} \underset{\boldsymbol{\mu},\, MS_{LB}}{\text{minimize}} \quad & MS_{LB} \\ \text{subject to:} \quad & \forall i \quad \sum_j \mu_{ij} = T_i \\ & \forall j \quad F_j \le MS_{LB} \\ & \forall i, j \quad \mu_{ij} \ge 0. \end{aligned} \tag{3}$$