Information Sciences 319 (2015) 71–91

Contents lists available at ScienceDirect

Information Sciences

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

Asymptotic scheduling for many task computing in Big Data platforms

Andrei Sfrent, Florin Pop *

Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Romania

ARTICLE INFO

Article history:
Received 9 July 2014
Received in revised form 16 March 2015
Accepted 20 March 2015
Available online 27 March 2015

Keywords:
Asymptotic scheduling
Many-task computing
Cloud computing
Big Data platforms
Simulation

ABSTRACT

Due to the advancement of technology the datasets that are being processed nowadays in modern computer clusters extend beyond the petabyte scale – the 4 detectors of the Large Hadron Collider at CERN produced several petabytes of data in 2011. Large scale computing solutions are increasingly used for genome sequencing tasks in the Human Genome Project. In the context of Big Data platforms, efficient scheduling algorithms play an essential role. This paper deals with the problem of scheduling a set of jobs across a set of machines and specifically analyzes the behavior of the system at very high loads, which is specific to Big Data processing. We show that under certain conditions we can easily discover the best scheduling algorithm, prove its optimality and compute its asymptotic throughput. We present a simulation infrastructure designed especially for building/analyzing different types of scenarios. This allows to extract scheduling metrics for three different algorithms (the asymptotically optimal one, FCFS and a traditional GA-based algorithm) in order to compare their performance. We focus on the transition period from low incoming job rates load to the very high load and back. Interestingly, all three algorithms experience a poor performance over the transition periods. Since the Asymptotically Optimal algorithm makes the assumption of an infinite number of jobs it can be used after the transition, when the job buffers are saturated. As the other scheduling algorithms do a better job under reduced load, we will combine them into a single hybrid algorithm and empirically determine what is the best switch point, offering in this way an asymptotic scheduling mechanism for many task computing used in Big Data processing platforms.

1. Introduction

The motivation behind solving scheduling problems is the current needs for efficiently processing very large data sets. At CERN, for instance, ATLAS and the three other main detectors at the LHC produced 13 petabytes of data in 2011 [5]. At this scale, even minor optimizations in the scheduling algorithms will result in major improvements in terms of processed data. In the field of meteorology we also find data sets ranging from terabytes to petabytes [25]. Some of the world's supercomputers are working solely with the purpose of predicting life threatening events such as hurricanes, storms and earthquakes. Examples of applications addressing the problem of Big Data processing are: clustering and data mining [15,19], prediction and analytics [26,27], geographic and satellite data processing [11], etc. There is, indeed, a lot of focus on optimization at all

* Corresponding author.
E-mail addresses: andrei.sfrent@hpc.pub.ro (A. Sfrent), florin.pop@cs.pub.ro (F. Pop).

http://dx.doi.org/10.1016/j.ins.2015.03.053
0020-0255/

possible levels: from the hardware platforms, scheduling software to the algorithms that are used to actually process the data [1].

From the economical point of view, good scheduling techniques help production costs to decrease by efficiently taking advantage of the available resources. On certain occasions providing a reliable scheduling framework that is able to deal with a very high volume of jobs per time unit and maximizing function of the executed jobs is directly linked to income. Even major companies have trouble from time to time when users generate too many requests that have to be processed in a short amount of time.

The evolution of modern scheduling algorithms has been closely related to discoveries in the area of artificial intelligence, as genetic algorithms, for instance, are largely used nowadays to solve this type of problem. Moreover, GA are often combined with heuristics that were empirically proved to yield better results. As simplicity is a prerequisite for reliability (Edsger W. Dijkstra), our goal is to design a simple algorithm that performs optimally under certain conditions. Such algorithms may be combined into powerful hybrids that are able to solve the exact instance of the scheduling problem that occurs in our system.

The general scheduling problem, known as the *Job Shop Problem* (or JSP), is a problem in discrete or combinatorial optimization and is notoriously hard to solve and has been discussed may times in the literature [3,43]. The problem may be stated as follows: we are given a set of jobs, each one of them consists of multiple operations and each operation needs to be executed on a specific machine, and a set of machines for these jobs to run on; find a planning of the jobs on machines so that all of them complete in the minimum possible amount of time. This problem is, in fact, a generalization of the TSP (Traveling Salesman Problem) and it is known to be in the NP Hard class.

This paper addresses the following scheduling problem: a finite set of jobs is given and each job consists of the execution of an undetermined number of tasks. A task is a sequence of specific operations and its length depends on the machine on which it is scheduled to run. We have a cluster consisting of a limited number of heterogeneous machines, one machine can process a specified numbers of operations at a time and preemption is not allowed. Our goal is to maximize the total number of jobs that are processed at up to a certain point. Our model is a version of the JSP – there are always only three stages for each job (Read, Process, Write) and all of them must be scheduled on the same machine. The scheduling problem consists of pairing jobs with machines in such a way that the total value is maximized. We are confronting two separate issues here: the predictability of the system (the capacity of knowing the job rates ahead) and the 0/1 Knapsack problem.

The predictability of the system has been discussed multiple times in the literature and several techniques have been developed in order to accurately predict the load of the system or number of tasks that will come at a future point in time. Other approaches include ANN (artificial neural network) [14] approaches and advanced statistical models. In our approach, we remove the prediction issue by assuming well known task rates for each job (we can also see it as a good prediction model that is unspecified).

This paper describes the building of a hybrid algorithm to solve the scheduling problem with respect to our model. In Section 2 an overview on related work is presented. In Section 3 we introduce our model and some of the restrictions we impose on the system. The asymptotically optimal algorithm is presented in Section 4 together with a mathematical demonstration of its optimality under heavy load circumstances. In Section 5 we approach the problem of building a hybrid algorithm and investigate the optimal switch points. The simulation framework we built in order to assess the performance of our schedulers is described in detail in Section 6. Lastly, Section 7 presents the results of our experiments and a brief comparison between several scheduling algorithms. We also included, in Section 8, conclusions derived from our work and next research directions.

2. Related work

The Job Shop Problem is one of the fundamental problems in the field of job scheduling. Because even for small instances of this problem it is hard to provide exact solutions, most of the research is focused on developing approximation algorithms and heuristics. Some authors have successfully used genetics-based learning systems and even hybrids that combine the advantages of both immune and genetic algorithms [16,35]. Other papers include other approaches such as incomplete search algorithms [4] and ANN (artificial neural networks) [42]. In [42] the author maps the problem to a Hopfield Network that is built in such a way that the energy function minimization will lead to a good solution to the scheduling problem. Another scheduling approach was proposed in [37] for hypergraph-based data link layer scheduling for reliable packet delivery in wireless sensing and control networks with end-to-end delay constraints.

Easier problems in the NP set (such as Flow Shop Problem) have been solved by adapting existing approximation algorithms for the Traveling Salesman Problem. Garey showed back in one of his research papers dating from 1976 that Flow Shop Problem is NP-Complete for $m \geqslant 3$ (where m is the number of machines) [10]. This means there is a known polynomial reduction from FSP to TSP and any approximation algorithm for TSP can be used to solve FSP as well. As a side note, the $m < 3$ condition makes the P algorithm unusable in modern computer clusters.

Let's consider the case of only one machine. Let's also consider a fraction of our scheduling time T as being a knapsack with a total weight capacity of T units. Each **job** represents an object type and the actual objects are called **tasks**. Each job (*object*) takes a certain amount of time T_j to execute on the specified machine (*object's weight*). Every task that is completely executed within the time T has an assigned value (*object's value*). From the 0/1 Knapsack we learn that we are dealing