Master 1 MoSIG Research Project Report
Learning Job Runtimes in Homogenous HPC Systems

Valentin Reis
Supervised by: Denis Trystram & Eric Gaussier.

I understand what plagiarism entails and I declare that this report is my own, original work.
Name, date and signature:

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

1 Introduction

High Performance Computing (HPC) systems are complex machinery at the frontier between research in scheduling and systems engineering. We outline the two main difficulties that resource management software in this field have to face.

First, the ephemeral nature, and broad range of existing architectures of such systems make the development and application of theoretical results difficult. New schemes for distributing resources (e.g. memory caches, hard drives, processing units, RAM memory) are ubiquitous and their rate of occurence is high. Moreover, the topology of a HPC system can change on a monthly or weekly basis, with the addition of new hardware. Finding scheduling, and resource management strategies which can adapt to those changes is a current research problem.

In addition, the input data these systems have to work with presents many peculiarities. The nature of the information which users provide is very often loose (e.g. with only upper and/or lower bounds provided). For instance, and this will be the focus of this paper, the run time of a given job on a specific system is seldom known in advance, however an upper bound may be given by the user.

As a consequence of these difficulties most free, open-source and commercial resource management software use simple heuristics, which at best provide bounds on their performance, and at worst guarantee a few functional properties. An example of such a strategy is the First Come First Serve (FCFS) strategy to schedule parallel jobs on a homogenous cluster of machines. Among other properties (such as robustness to weak information about the amount of time a job will run on the system), this strategy guarantees the avoidance of starvation.

1.1 Research Direction

The general direction we are headed in, in the course of this research project, is to deal with the input data of the resource management systems. Accomodating for this data seems separable enough from the actual scheduling problems for work towards this objective to be rewarding. No innovative ways to query the data from the users will be studied : we will mainly rely on existing logs from HPC systems. Instead, we seek to apply Machine Learning techniques in order to reduce uncertainty of, and extract information and/or structure from, the input data of the HPC systems. We will be working with the problem of presenting input data in the most valuable way possible to a scheduling algorithm. How to use this data to the fullest will not be discussed. When assessing the relevance of the specific the information we choose to extract, references from the HPC scheduling litterature will provide ground to stand on.

1.2 Job runtimes

Most HPC resource management software (including the SLURM, OpenPBS, OpenLava and OAR software) do ask information about jobs to users, such as topological requests in terms of processing units and memory, the name of the executable, miscellaneous functional requirements and, last but not least, the expected run time of the job. This user-provided estimate of the run time of a job on a specific system will be referred as **reqtime** in the rest of the paper. Most of these software use the **reqtime** of a job as an upper bound on its run time, and kill it should **reqtime** be violated. As a consequence, this information is over-estimated by the users, if they choose to provide it. The following section will present a statistical analysis pertaining to this relationship.

The true run time of a job with respect to a given affected topology is of great interest, as the scheduling policies are highly dependent on this information to provide good solutions **??**. We will refer to this quantity as the **runtime** of a job. It must be clear that the **runtime** of a job is only defined with respect to a specific processing environment to which it might be affected. This can include and is not limited to, the network topology of the processing units, the availability of shared memory, message passing costs, and the operating system supporting the computations.

1.3 Problem Statement

We restrict the broad question of refining the data to a single relevant variable: the **runtime**. The problem statement we are dealing with is the following.

Given a specific homogenous HPC cluster with negligible communication costs, how to best predict the value of the **runtime** of a job? The choice of dealing with homogenous machines and without communication costs in a first approach has the interesting property of separing the data treatment step from the scheduling step. In this case, the **runtime** becomes an objective measure of a job that does not depend on external properties..

TODO:blabla

2 Motivation

2.1 Importance of runtime

TODO: to emphasise again... provide more references.

TODO: moreover, show existing solutions such as SLURM which use the walltime instead (leading to the next subsection)

2.2 reqtime vs runtime on a real system

TODO: show the curie log.

3 State of the art in predicting runtime

3.1 Nature of the prediction

TODO: -what to predict: value?, confidence interval?, distribution? whith which algorithms can we use those?

3.2 Predicting a value

TODO: -give the references and explain the historical methods, gibbons historical scheduler, the tsafir et al paper with mean of two last runtimes values userwise..

3.3 Predicting a distribution

TODO: -give the references and explain the probabilistic backfilling thing..

4 Our approach

4.1 Random Forests

TODO: -explain our approach, why it could lead to better results (external info+signal locality(ref hmm thesis for locality..))

4.2 Explainability

TODO: -explain one advantage of random forests: discussion about the trees after learning..

5 Preliminary Results

TODO:results..

6 Conclusions

TODO:conclure..

7 Acknowledgements

TODO:remercier..