An Architecture for Real-Time Job Characterization

Abel Souza
Department of Computing Science
Umeå University, Sweden
abel@cs.umu.se

Abstract. Estimating the quality of a given statistical model is of great importance because they are based on a finite dataset from where all the learning comes. This dataset is usually very large, making it infeasible to continuously process huge amounts of data by statistical models. The Bags of Little Bootstraps (BLB) is a novel method for estimating the quality of a statistical model with a more favorable computational profile than the traditional bootstrap, because it estimates a good model from a limited amount of data. Herein, we suggest an architecture that combines BLB with real time sampling of server metrics in distributed applications. By making use of an Insight Engine (IE), this architecture could be used to efficiently allow applications to share resources and to create a statistical model for predicting their resource usage, enabling schedulers to allocate resources optimally.

1 Introduction

In recent times, especially with the high increase of analysis on massive amounts of data, scientists have been more often running Data Intensive (DI) applications inside High Performance Computing (HPC) clusters. However, in order to be properly placed in a HPC cluster, long time running jobs, like DI applications, would require a higher degree of integration with the system-level scheduler [3] than HPC ones.

DI applications are characterized for executing many repetitive simple instructions for a long amount of time (long running jobs) [3]. It usually have low data dependencies, where its decomposition in multiple tasks increase performance and time to finish. On the other hand, HPC applications often execute complex operations, with very strong inter-dependencies between its tasks. Besides that, HPC applications are very sensitive to disturbances in resource usage, such as Memory, Network and/or Disk Input/Output (I/O). Due to their dynamic (and sometimes random) nature, running DI applications in HPC clusters either wastes resources or limits their execution, as they are not always deadline bound.

Traditional HPC schedulers are commonly used for coarse-grained resource allocation [3] and usually require both a job geometry (e.g., number of nodes, cores and memory) and a deadline. HPC users tend to make bad estimations
about both these parameters, but DI jobs cannot be defined by such parameters due to their data characteristics, which may increase its execution time. Also, a majority of DI jobs in cloud data centers are long running jobs [1]. This is even more visible for jobs in the scientific community that tend to have very repeating patterns as researchers tend to focus on same problems for long periods of time.

Finally, in HPC Datacenters, just in a single server, there are thousands of hardware and software metrics (so called KPIs, Key Performance Indicators) that individually and aggregated can give insight in the performance, robustness, and efficiency of the server itself and the applications it provisions. At the datacenter level, the number of KPIs is even larger. Data such as nodes utilization traces, application logs and reliability measurements can provide relationships which cannot be seen without using data analytics techniques. They can be used to understand the state of a cluster as a whole and also to make better informed decisions for resource management or usage reporting. Once derived, these relationships can be used in order to better use the resources available in a cluster of computers, by sharing them. Sharing of cluster resources, especially fine grained as in HPC, causes to have many jobs running in a reasonably large or medium sized cluster. These jobs can have many tasks which eventually generate large amount of traces, especially if these tasks run for long amounts of time.

1.1 Motivation

Estimating the quality of a given statistical model is of great importance because they are based on a finite dataset from where all the learning comes. The Bags of Little Bootstraps (BLB) [4] is a novel method for estimating the quality of a statistical model. BLB has a more favorable computational profile than the traditional bootstrap, as it distributively gathers smaller quantities of data than a original potentially big dataset. Thus, BLB best fits to implementations on modern distributed and parallel computing architectures, such as DI and HPC Applications.

Here we suggest the use of a mechanism that combines BLB with real time sampling of server metrics in distributed applications. We propose an Insight Engine (IE) that could be used to allow long running jobs to share resources. This IE is used to create a statistical model that predicts resource usage for applications based on processor metrics. We advocate its feasibility for real time decisions on stream of traces coming from the tasks of a DI application running in the machines in a cluster. It works by comparing real resource usages from two representative benchmarks with the ones predicted by our real time model for similar executions in the benchmark. By doing so it creates an insight that can be helpful for better isolation, placement and assignment of resources to the running tasks.

2 Background

HPC clusters usually are managed by static resource managers like Slurm [8], or Torque [6], which commonly asks users to provide a deadline for launching jobs.
The jobs can finish sooner than the specified deadlines, and more importantly, can use less resources than what was required by the users. Usually users cannot use more resources than what has already been allocated to them in regard to the project they are member of and submitting their request as. Some HPC resource providers\(^1\) publish their internal data about projects and users we have more insight about them. By looking at various available traces from HPC and Cloud computing providers we see some outliers like long running or big jobs [5], and repetitive running of same applications with similar data as input. Unlike the general clouds, scientific communities, compute clusters have common patterns, as researchers tend to re-run the same applications over long periods of time. These patterns can help an administrator to automatize the application placement within a datacenter.

3 Insight Engine Architecture

Observations done in the previous sections exemplifies how to design an architecture that fundamentally improves the time to create insights, and also to even reduce the overhead and improve performance of the outcome. As HPC and DI applications tend to re-run often times, we claim that application resource usage metrics will have similar behaviours in respect to their outputs. This pattern assists a statistical model to accurately predict application behaviour and thus to help a scheduler to place applications in a datacenter.

Figure 1 shows this architecture where a DI job scheduler (Mesos [2]) runs on side of a HPC scheduler (SLURM [8]) to collocate HPC and DI jobs in same nodes. On top of that, there is an Insight Engine (IE) which collects various samples from the servers where the application is running on. The samples are derived from the servers usage, like processors counters which can be mean CPU usage rate, memory and page cache memory usage, and others related to Input/Output (I/O) operations like mean disk I/O time and mean local disk space used.

\[\text{Fig. 1. General Architecture for the Insight Engine}\]

\(\text{\[1\] E.g. SNIC High Performance Computing, http://www.snic.vr.se/projects/snic-hpc}\)
Figure 2 illustrates how the IE could be used in a hypothetical scenario where a cluster of computers are partitioned in order to support the sharing of HPC Applications (Firm Long Jobs) and more Dynamic ones (Dynamic Jobs), like the DI types. Firm Long Jobs, with strict requirements would be scheduled in the traditional way, as they are mostly done today. Dynamic jobs would in turn be scheduled by a more dynamic scheduler, like Mesos. Mesos can have a finer grained view of the cluster, and with the support of the IE, it could optimally collocate some of these jobs with the firm jobs that are already running in some of the available resources. In case a performance degradation is detected in relation to the firm jobs, the dynamic jobs can be killed as they usually have the support of the dynamic scheduler to restart in a different server.

Fig. 2. Worker Nodes producing traces to the Insight Engine (IE)

3.1 Techniques and Approach

The proposal technique starts by the selection of a subset of performance anomalies and the definition of key criterias to characterize and compare their state through modeling of applications. Techniques such as multi-variate linear models, feature selection, and classification can be used to evaluate and characterize different models for predictability of DI and HPC applications.

Then, hardware metrics such as CPU, memory, network utilization, as well as derived metrics such as FLoating-point Operations Per Second (FLOPs) are gathered. A training data is selected for different operational conditions, situations where everything is running smoothly as well as scenarios where the server is overloaded and the applications experience performance anomalies. In a final evaluation, it should be investigated whether our proposed system can detect anomalies in some application and study precision and recall (false positives and false negatives). This evaluation will be measured by applying the Bags of Little Bootstraps, a powerful new alternative for automatic, accurate assessment of
estimator quality that is well suited to large-scale data and modern parallel and distributed computing architectures.

4 Summary

Integrating two different resource managers in the same cluster, as they have different objectives and isolation must be guaranteed. Enforcing this isolation can be the limiting factor since HPC clusters are usually bigger and more scalable than normal DI cloud platforms. Also, the rate of additional failures that a DI scheduler will see compared to normals should be studied, as interferences may happen more frequently.

Two main problems may rise while sampling resource usage metrics: performance degradation and metrics accuracy. Firstly, application performance may degrade during sampling because in order to so you should basically pause the application during its counter readings. This should be ultimately minimized, by finding an optimal rate of sampling over time. Secondly, it is still not possible to properly read many simultaneous counters at once. This requires a multiplexer for reading the counters, changing their values over time and ultimately, decreasing the statistical model accuracy. This may be mitigated by adapting the model to these discrepancies.

For what is missing, two relevant benchmarks can be used in order to evaluate our proposal methodology. The first, the Unified European Application Benchmark Suite (UEABS)\(^2\) is a set of 12 application codes forming a single suite, with the objective of providing a set of scalable, currently relevant and publically available codes and datasets, of a size which can realistically be run on large systems, and maintained into the future. The second, BigDataBench is a benchmark suite for scale-out workloads [7]. It models five typical and important big data application domains: search engine, social networks, e-commerce, multimedia analytics, and bioinformatics. In total, it includes 14 real-world data sets, and 34 big data workloads.

5 Conclusion

Great amounts of data bring issues, including statistics in data analysis and visualization. Still, there is still the need to estimate the quality of statistical models. Distributed applications may generate large datasets that can be used to estimate application performance. The volume of the data can remain quite significant, as datasets can be high dimensional, be used to train complex models with many parameters, and can have many potential sources of bias. Even with enough data so to perform highly accurate estimations, the efficient computation of the data remains essential to allow efficient use of available resources by processing only as much data as is necessary to achieve a desired confidence. BLB has good statistical properties and bootstrap’s generic applicability, while

\(^2\) http://www.prace-ri.eu/ueabs/
performing better. By using BLB, our proposed architecture can efficiently determine the right amount of data to be collected from distributed applications run-time systems in order to provide means of an estimator to make usage of this data in order to predict performance metrics. This would be useful for resource allocation and placement of applications, besides better scheduling in large HPC datacenters.

References