Proposal: Software Auto-Tuning for MapReduce
PI: Jeremy Singer, University of Glasgow, http://www.dcs.gla.ac.uk/~jsinger
Google contacts: Diego Novillo, Peter Dickman, Rob Ennals

Abstract

MapReduce-based utility computing has become widespread. Performance tuning of MapReduce frameworks, such as Hadoop, is notoriously difficult since users lack expertise or access to critical configuration options. We propose an auto-tuning approach for MapReduce applications running on Hadoop clusters, based on supervised learning. With auto-tuning, we aim to provide performance that is significantly better than using the default Hadoop parameters, and within 5% of the best performance that can be obtained by parameter-space search techniques.

Research Problem: to use statistical machine learning to auto-tune MapReduce applications running on the Hadoop platform

1 Introduction

The Google MapReduce programming paradigm is extremely influential, forming the computational base for warehouse-scale computing and other important application domains, e.g. [6, 9, 19]. Hadoop is an open-source implementation of the MapReduce framework. It allows distributed Java applications to process massive data sets on clusters of compute nodes, in a scalable and fault-tolerant manner. Each individual node executes Hadoop worker tasks on instances of the Java virtual machine (JVM).

1.1 Performance Tuning is Required

Performance tuning is a major issue with Hadoop. There is often significant disparity between MapReduce application performance on a poorly configured cluster of nodes and a well configured cluster. For instance, Herodotou et al. show how varying a single parameter can give a 2x change in execution time for the standard MapReduce wordcount and terasort benchmarks.

Users may want to optimize execution of their MapReduce application, perhaps directly in terms of minimizing execution time, alternatively optimizing some function of compute nodes and execution time. In a utility computing context, they may want to minimise the cost of their computation which depends on the various market rates charged by the cloud computing provider. This function may be more complex than Amazon (among others) have led us to believe. Kambatla et al. show that computational elasticity for MapReduce applications is not entirely symmetric, i.e. 1 hour on 100 nodes may not accomplish the same work throughput as 100 hours on 1 node.

Cluster administrators and utility computing providers also want to optimize overall application execution, to provision computing services efficiently. Automatic reconfiguration cleanly handles short-term workload transience, and longer-term hardware upgrades.

In summary, there are strong scientific and economic drivers for improving the tuning of Hadoop clusters, and MapReduce-style utility computing platforms in general.

1.2 Manual Performance Tuning is Difficult

Manual performance tuning of Hadoop clusters, e.g. [12], is impractical for at least three reasons.

First, because there are parameters at different levels in the software system stack. It is not clear whether tuning should focus on the application, the MapReduce middleware or the underlying JVM instances running on each node. Indeed, it may be necessary to tune at all levels. However then we may find that parameters set at one level interact in non-obvious ways with settings at a higher level.

Second, because there are so many parameters at each level. Some parameters may depend on others, whereas some are completely orthogonal. This dependence relationship is not always apparent. An exhaustive search of the parameter space to find the optimal point is infeasible.

Third, because many parameters are inaccessible to users, either because they lack the expertise, or because they do not have permissions to change the default settings, e.g. if working in a utility computing context.

1.3 Automated Performance Tuning is a Promising Solution

Since performance tuning is necessary, but too difficult to do manually, we propose auto-tuning. This requires innovations in Hadoop performance profiling and parameter tuning prediction. The problem is an ideal fit for statistical machine learning in particular as a supervised learning task.

Figure illustrates our proposed approach. Given a MapReduce application $a$ with input data $i$, we first execute $(a, i)$ on default configuration Hadoop, to obtain reference profiling data. This becomes the feature vector $\tilde{x}$ that describes $(a, i)$. Next we do repeated runs of $(a, i)$, searching the range of configuration parameters to find a vector of parameter values $\tilde{p}$ that give optimal performance for $(a, i)$ on our cluster. (This search might be as simple as random, or systematic...
sampling of parameter space.) Now we have enough data to generate a single training example \((\vec{x}, \vec{p})\), which corresponds to one point in the co-ordinate space in Figure 1. This optimal parameter vector \(\vec{p}\) is what we want to be able to predict, for new programs for which we only have reference profile data \(\vec{x}\). Following this process for many applications and input data, we are able to generate a set of training examples.

Given the training set, we can produce classifier functions. A single classifier \(f_i\) may predict value \(p_i\) for a single configuration parameter, i.e. \(f_i(\vec{x}) = p_i\). If we make the simplifying assumption that the parameters are independent, we can have a separate classifier for each configuration parameter. Alternatively, we could account for relationships between configuration parameters by using more advanced machine learning techniques such as structured output prediction [15]. The end result is that we can apply the classifier(s) to previously unseen \((a, i)\) pairs to predict their optimal parameter settings \(\vec{p}\).

We note that many other systems optimization problems have been phrased as machine learning tasks, and proved to be amenable to auto-tuning. These include compiler optimization [8, 10] and application-specific garbage collection [14].

2 Outline of Project

Our plan of work splits into six phases. Figure 2 gives a Gantt chart for these phases, which are described below.

(Phase 1) We intend to take a set of representative MapReduce applications such as wordcount, terasort, pigmix [3], and programs from the hadoop.examples package for benchmark tests. We will show evidence of the tuning problem by varying system parameters and demonstrating the dramatic effects on application performance.

(Phase 2) Next we will identify a set of system parameters as candidates for our auto-tuning process. We will consider parameters from both the JVM and Hadoop levels in the runtime stack. We will use the parameter tuning case studies from phase 1 to assist us in determining which parameters we might include. To avoid application-specific dependences, we will not use application-level parameter tuning.

(Phase 3) We will identify a set of features that describe the runtime performance of a Hadoop application. These might be directly related to performance profiling (such as task throughput, relative times for map and reduce phases) or more low-level (such as per-node garbage collection activity).

(Phase 4) Now we will phrase the parameter tuning problem as a supervised learning task. As described in Section 1.3, we accumulate a set of training examples, which can be used to generate classifiers. In previous machine learning tasks, we have used non-parametric classifiers such as decision trees [14], k-nearest neighbours [8, 10], and support vector machines [17]. We expect to employ something similar for this problem, since we cannot make any assumptions about linear relationships between the profiling information (features) and the configuration parameters (classifier outputs). A testing stage, using a strategy like leave-one-out cross-validation, assesses the accuracy of the generated classifiers.

(Phase 5) We will use the classifiers constructed in the previous phase to carry out ahead-of-time parameter prediction for previously unseen Hadoop applications. This is off-line learning, and involves fixing system configuration parameters to application-specific values at the start of execution, based on features collected from a profiling run. We will judge this auto-tuning process to be successful if:

1. we can improve performance in relation to the ‘default’ Hadoop configuration, for all our benchmarks.
2. our ahead-of-time auto-tuning achieves within 5% of the performance of the best configuration we can find via manual tuning, i.e. parameter state-space exploration.

(Phase 6) In exploratory work we will move to consider further possibilities like dynamic learning and tuning of the Hadoop runtime system. This approach would be needed for long-running, adaptive workloads, for the most flexible platforms provided in a utility computing context. Online learning techniques are appropriate here.

3 Dissemination

We intend to release our tuning framework to the Hadoop user community as open-source code, together with appropriate documentation. We will use a public repository such as SourceForge or Google Code to host our project. Ideally we would like to engage the community in testing its functionality, providing benchmarks, etc. We will investigate this possibility with interaction through the Hadoop mailing lists.

Academic publications will target the usual programming language and parallel computing conferences, e.g. PLDI, OOPSLA, PPoPP. We also aim to make use of regional academic networks such as SICSA\[^{1}\] and HiPEAC\[^{2}\].

\[^{1}\] http://www.sicsa.ac.uk
\[^{2}\] http://www.hipeac.net

(Multicore computing strand)

(Expertise on machine learning for systems tuning resides in the Compilation research group)
We also intend to release our data sets to the machine learning community, via an open mechanism like the UCI repository.

4 Related Work

Babu’s position paper \cite{S. Babu 2007} identifies the problem of auto-tuning MapReduce applications in Hadoop. This work focuses only on Hadoop specific framework parameters, rather than the multiple levels we consider. Potential heavy-weight solutions include speculative concurrent execution of multiple instances of the same task, with each instance having different configuration parameters. Poorly performing configurations can be killed early.

The Starfish system \cite{H. Herodotou, 2011} also tunes only Hadoop specific runtime parameters. It uses a standard search strategy in combination with the What-if Engine, which is a MapReduce simulator based on performance models. These models are manually derived by experts, rather than being generated using machine learning techniques. However significant performance improvements are demonstrated on standard Hadoop benchmarks, by tuning only 3 or 4 critical platform parameters.

Wang et al \cite{G. Wang, 2009} describe a software simulator called MRPerf, that predicts how various parameters will impact on MapReduce application performance on a cluster of compute nodes. This work is similar in spirit to our proposal, in that they consider configuration parameters at different layers (cluster hardware, Hadoop framework and application configuration). They have a predictive model to show how the actual application will perform on a real system. They compare the predictive performance with actual performance on real clusters, and demonstrate accuracy in the map and reduce phase performances for standard Hadoop benchmarks within 3.42% (map) and 19.32% (reduce) of actual measured values. However, MRPerf only predicts performance given a set of parameters. It is unable to suggest optimal parameters for a given application on any cluster setup. Also the paper gives no details about the time taken to perform a simulation. With our proposed machine learning based approach, once the model has been sufficiently trained, it is extremely cheap to obtain further predictions.

Wieder et al \cite{Wieder et al 2010} use linear programming to optimize cloud application deployment decisions. However they are assuming application mobility across different providers, based on fluctuations in the cost models, rather than tuning performance on one particular platform.

To the best of our knowledge, there is no existing work that uses statistical machine learning to optimize the tuning of the multiple layers of a software stack running Hadoop applications on a cluster—which is the essence of our proposal.

References

[1] Hadoop: Open source implementation of mapreduce, \url{http://hadoop.apache.org}

\textit{http://www.ics.uci.edu/~mlearn/}