An Optimal Task Assignment Policy and Performance Diagnosis Strategy for Heterogeneous Hadoop Cluster

Shekhar Gupta
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

Abstract

The goal of the proposed research is to improve the performance of Hadoop-based software running on a heterogeneous cluster. My approach lies in the intersection of machine learning, scheduling and diagnosis. We mainly focus on heterogeneous Hadoop clusters and try to improve the performance by implementing a more efficient scheduler for this class of cluster.

Introduction

Hadoop is a popular and extremely successful framework for horizontally scalable distributed processing of large data sets. It is an open-source implementation of the Google filesystem (Ghemawat, Gobioff, and Leung 2003), called HDFS, and Google's MapReduce framework (Dean and Ghemawat 2008). HDFS is able to store tremendously large files across several machines and using MapReduce, such files can be processed in a distributed fashion, moving the computation to the data, rather than the other way around. Taking care of data locality is one of the key features of Hadoop, which reduces unnecessary data traffic among nodes. Therefore, while optimizing the Hadoop scheduler we need to satisfy the data locality constraints.

Hadoop clusters typically execute a multitude of jobs in parallel. These jobs are subdivided into several *tasks*, often numbering in the thousands, which are executed in parallel on different nodes. While tasks belonging to the same job are very similar to each other in terms of their individual resource profiles, tasks belonging to different jobs can have different profiles in terms of their resource requirements. The resource requirement for a job is defined as utilization of CPU, memory, disk I/O or network I/O. Hadoop has a built-in scheduler that assigns tasks to different nodes while satisfying the *data locality* constraints.

One major shortcoming of Hadoop's existing schedulers is that they all implicitly assume a *homogeneous* cluster, i.e., they assume that all compute nodes in the cluster have similar hardware and software configuration. However, in reality clusters often feature specialized servers for various computations, also the size of cluster grows over time as new machines are added with different profiles. Therefore, the cluster need to be assumed rather heterogeneous. The scheduler's assumption about homogeneity prevents it from making optimal use of available resources in heterogeneous clusters, since they are not able to match jobs to their best compute nodes, consequently compromising global metrics such as throughput or maximum delay. Furthermore, earlier research has shown the benefits of heterogeneous clusters compared to homogeneous ones (Balakrishnan et al. 2005; Ghiasi, Keller, and Rawson 2005; Kumar et al. 2005). Intuitively, more specialized hardware can better suit a variety of differing job resource profiles.

The absence of a monitoring component which determines the dynamic state of every node is another shortcoming of the scheduler. In a cluster it is natural to assume that the performance of nodes degrades with time due to soft/hard faults. The Hadoop scheduler's ping based protocol efficiently detects hard faults (dead node or network failure), however it fails to identify soft faults (slowdowns). Therefore, the scheduler is unaware of nodes that are underperforming. Due to lack of knowledge about the state of the node, the scheduler might assign tasks to a node in a regular fashion. Such assignment policy can lower the throughput (the number of jobs processed in a particular time window) of the cluster.

Problem

The main goal of my thesis is to improve the performance of heterogeneous Hadoop clusters by proposing an efficient task assignment policy. Therefore, we aim to implement a task assignment policy for scheduling computational tasks onto compute nodes in a heterogeneous parallel and distributed system, such that the job throughput can be maximized. We plan to achieve this goal by matching job requirements with available resources. While making the scheduling decision the data locality constraints should always be satisfied. Additionally a diagnosis module is needed to identify what nodes are underperforming and why. We will incorporate this fault information into the scheduling policy so that better resource matching can be used.

Approach

We plan to reach the above goals by using an active machine learning approach. We divide the overall approach into two phases, explore and exploit. In the explore phase we actively try to learn nodes' resource capabilities and jobs' resource requirements. Later in the exploit phase we use the learnt information to implement an efficient task assignment policy and to device the diagnosis module. Such an approach is similar to the Pervasive Diagnosis approach (Kuhn et al. 2008), which diagnoses a system without stopping production. The explore and exploit phases are further subdivided as:

Explore

Learn Task Model In our approach, we need to model tasks completion time in terms of their resource requirements and nodes' resource capabilities on which the tasks will be executed. If the resource requirements are denoted as r and the machine capabilities as κ, then the model of the task is given as

$$F(r, \kappa) = T_{r,\kappa} \tag{1}$$

Here $T_{r,\kappa}$ is the task completion time. In our case we assume that time samples are the only observation we have. We aim to learn r and κ using above relation. The model can be linear or nonlinear. Therefore, the research challenge is to implement a model selection approach to build the model with a certain accuracy. In the beginning, we plan to assume a linear model and later a more complex non-linear model will be considered.

Learn Node Capabilities Once the above model has been derived the next step is to learn model parameters (r and κ) using time samples. The first model parameter that we aim to learn is node capabilities (κ). Our objective is to learn node capabilities in active learning manner by running some probe jobs on the cluster. Probe jobs are defined as jobs with known resource requirements. We assume that we have a set of probe jobs that can be used to learn node capabilities. It is very likely that probe jobs may not be actual production jobs, therefore, we assume that node capabilities will be learned in offline manner. We assume that observed time samples are not noise free and there is a random noise. An interesting research challenge can be addressed by determining the sequence of probe jobs to learn the most about node capabilities in the presence of noisy observations.

Learn Resource Requirement After learning the resource capabilities for every machine with a certain accuracy, we aim to learn job resource requirements or in other words, job parameters (r). At this step our objective is to learn resource requirements in an online manner. In other words, we want to run actual tasks on the cluster and learn their requirements without stopping the production. In order to learn during production phase, we aim to exploit time samples collected from actual production tasks to learn task requirements. To maximize the overall throughput of the explore and exploit phase, the duration of the explore phase should be minimal. Therefore, in this learning phase the incoming jobs must be scheduled in such a way that the information gain about task resource requirements can be maximized, given that we have a noisy observation model.

Exploit

Assignment Policy Once the job resource requirements and the cluster resource capabilities have been learnt, we enter into the exploit phase. Combining these two information according to the learned model (Eq. 1) the scheduler should be able to assign tasks belonging to jobs onto the compute resources that are most efficient to fulfill task resource requirements, in terms of software and hardware configurations. This scheduling policy only tries to improve overall throughput, however, we also need to maximize the utilization of the cluster as another objective function. Additionally the data locality constraints need to be satisfied.

Diagnosis The derived task assignment policy assumes a certain belief about performance of nodes in the cluster. This belief is derived in the initial stage of the explore phase where we learn nodes' capabilities. However, capabilities of nodes may change over time and the scheduler must update changed capabilities to generate more efficient task assignment policy. Therefore, we need a diagnosis module that learns about the dynamic state of the compute resources in the cluster and detects intermittent problems such as slowdowns. The preliminary idea to implement such monitoring module is to continuously learn node capabilities and infer problems in nodes by observing deviation in the capabilities from their initial value.

Current Status and Future Work

As of now we are working on the explore phase of the project, where we assume a linear model for task completion time(Eq. 1). We have developed a scheduling policy to learn job parameters, which selects nodes that reduces the entropy about the belief of job parameters most efficiently. However, at this stage we assume that node parameters are already known. In coming months we plan to implement the task assignment policy for the exploit phase. While the focus of this proposal is on Hadoop, our approach is generalizable to other parallel and distributed system frameworks.

References

Balakrishnan, S.; Rajwar, R.; Upton, M.; and Lai, K. 2005. The impact of performance asymmetry in emerging multicore architectures. In *In Proceedings of the 32nd Annual International Symposium on Computer Architecture*, 506–517.

Dean, J., and Ghemawat, S. 2008. Mapreduce: simplified data processing on large clusters. *Commun. ACM* 51(1):107–113.

Ghemawat, S.; Gobioff, H.; and Leung, S.-T. 2003. The google file system.

Ghiasi, S.; Keller, T.; and Rawson, F. 2005. Scheduling for heterogeneous processors in server systems. In *Proceedings of the 2nd conference on Computing frontiers*, CF '05, 199–210. New York, NY, USA: ACM.

Kuhn, L.; Price, B.; de Kleer, J.; Do, M. B.; and Zhou, R. 2008. Pervasive diagnosis: The integration of diagnostic goals into production plans. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, 1306–1312.

Kumar, R.; Tullsen, D. M.; Jouppi, N. P.; and Ranganathan, P. 2005. Heterogeneous chip multiprocessors. *Computer* 38(11):32–38.