CS270 Project Proposal: Resource Management in Multi-* Clusters

Mosharaf Chowdhury
mosharaf@cs.berkeley.edu

April 6, 2010

Motivation

With the advent of MapReduce [2], Dryad [5] and similar frameworks as well as of cloud services like Amazon's EC2[1], cluster computing has become mainstream in recent years. Consequently, the complexity of resource allocation and scheduling in such multi-tenant, multi-framework, and multi-user environments have increased considerably. Resource management problems in multi-* cluster environments can now be classified into three large categories.

In the beginning of the cloud-age, resource management problem was all about scheduling online homogeneous jobs (e.g., MapReduce jobs) with heterogeneous requirements from different users in privately owned clusters [1, 4, 6]. Next, cloud providers started to offer services to create on-demand clusters (e.g., EC2). This added a new dimension where cloud providers first need to allocate resources from a large pool, and then their clients get to implement their own scheduling mechanisms in the leased resources (sometimes cloud providers offer the scheduling service as well). Finally, in order to support heterogeneous frameworks (e.g. MapReduce and MPI) simultaneously, research on cluster computing operating systems (OSes) (e.g., Nexus [3]) is gaining momentum.

Putting everything together, the resource management process in multi-* cluster environments roughly consists of (i) cloud providers delivering raw clusters based on resource requirements of their customers, (ii) customers running cluster OSes to manage those resources to manage and schedule jobs from multiple frameworks, and (iii) frameworks scheduling tasks[2] (with or without assistance from the cluster OS) to get the job done.

Existing work in this space ranges from fair and efficient schedulers for scheduling tasks [4, 6], heuristics for decentralized scheduling to handle multiple frameworks [3], to resource allocation mechanisms for raw clusters. However, barring rare exceptions, most mechanisms are straightforward heuristics based on empirical evidences without formal definition or study of their performance guarantees.

Project Overview

For this project, we are currently considering two possible directions in this problem space. We will pick one as we move forward.

[1] http://aws.amazon.com/ec2/
[2] A job consists of multiple smaller tasks.

1. **Resource allocation algorithm for cloud providers**: We want to formally model the online resource allocation problem by identifying resource requirements, constraints, involved parties, and their contrasting objectives. Next, we want to design one or more algorithms to maximize the revenue from the cloud provider's perspective. Finally, time permitting, we would like to perform complexity analysis of the proposed algorithm(s) and simulate them to compare against baseline counterparts.

2. **Job scheduling algorithm for cluster OS:** We want to formally define the cluster OS scheduling problem and design a centralized algorithm as opposed to the existing decentralized heuristic[3] [3]. Our objective, in this case, can be minimizing average makespans of jobs from different frameworks or implementing prioritized scheduling etc. Finally, time permitting, we will compare the proposed algorithm against its existing counterpart through simulation.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A view of cloud computing. Technical report, 2010.

[2] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[3] Benjamin Hindman, Andrew Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Scott Shenker, and Ion Stoica. Nexus: A common substrate for cluster computing. Technical Report UCB/EECS-2009-158, EECS Department, University of California, Berkeley, 2009.

[4] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg. Quincy: Fair scheduling for distributed computing clusters. In *SOSP*, 2009.

[5] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *EuroSys*, pages 59–72, 2007.

[6] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys*, 2010.

[3]Formally analyzing the complexity and performance of the decentralized mechanism can also be an interesting challenge.