

Cluster Computing for Web-Scale Data Processing

Aaron Kimball Sierra Michels-Slettvet
Department of Computer Science and
Engineering
University of Washington
{ak, sierra}@cs.washington.edu

Christophe Bisciglia
Google, Inc.
christophe@google.com

ABSTRACT

In this paper we present the design of a modern course in cluster computing and large-scale data processing. The defining differences between this and previously published designs are its focus on processing very large data sets and its use of Hadoop, an open source Java-based implementation of MapReduce and the Google File System as the platform for programming exercises. Hadoop proved to be a key element for successfully implementing structured lab activities and independent design projects. Through this course, offered at the University of Washington in 2007, we imparted new skills on our students, improving their ability to design systems capable of solving web-scale problems.

Categories and Subject Descriptors

K.3.2 [Computer and Information Science Education]: Computer science education

General Terms

Design, Experimentation

Keywords

Education, Hadoop, MapReduce, Clusters, Distributed computing

1. INTRODUCTION

As the size and scope of problems faced by software engineers continue to grow, distributed and parallel computing skills become increasingly important. Web-scale problems involve ever-expanding data sets, currently ranging in size from gigabytes to several terabytes. This data must often be processed in a distributed fashion, which requires engineers to weigh many aspects of system design not casually considered even a few years ago.

Through discussions with industry, members of our department identified a gap in current undergraduate education. While our students demonstrate strength in traditional computer science subjects, they are not being adequately prepared to appreciate the magnitude of problems that industry tackles nor to propose and implement distributed solutions to them. With the rising prominence of web-scale

computing as well as the increasing focus on multi-core computing architectures, the time is right to provide more emphasis on parallel and distributed computing to our undergraduates. To explore possible solutions to this need, we collaborated with Google engineers, focusing on understanding and overcoming the challenges in teaching modern cluster computing techniques.

The result has been the development of a course that teaches undergraduates the skills to manipulate large-scale data sets on distributed platforms. In his ICER '06 paper [11], Sahami argues for the necessity of broadly-available coursework addressing scalability issues in computing. He notes that such courses provide a new opportunity for students—

“to work on building much more realistically scalable services dealing with large data repositories, large-scale computation, or both... designing algorithms that can be run on web-scale data by running as parallelized services on a cluster of machines.”

Our course directly meets this need.

What makes this different from other course offerings is its focus on changing the way students approach problems, motivated by processing large-scale data sets and the availability of Hadoop [8]: a low-overhead open-source distributed system based on Google's MapReduce [6] and Google File System [7] infrastructure. Hadoop runs on commodity Linux clusters and presents a straightforward programming interface suitable for use by students in the time frame of an academic quarter. By contrast, the use of existing platforms such as PVM in a classroom [5] introduces a high level of overhead [10]. Furthermore, courses based on PVM or MPI then typically focus on how to implement distributed systems algorithms for correctness and reliability on top of these bases; our course instead concentrates immediately on using an existing platform to complete real-world data processing tasks.

A pilot iteration of the class was developed by Google engineers and offered at the University of Washington in the winter of 2007. We examined feedback generated from this class and re-designed the course for the spring. This paper discusses the re-designed course curriculum and evaluates how well it served our students. We also describe tools and materials which can serve as a starting point for educators interested in building similar quarter or semester long courses at their own institutions.

The rest of the paper proceeds as follows. Section 2 lays out the educational objectives we had for the class, as well as the assumptions influencing its design. Section 3 discusses the software and hardware resources available as a platform for course projects. In Section 4 we give an overview of the curriculum and a description of the execution of the course in Spring 2007. Section 5 then provides an evaluation of how well it worked. Section 6 introduces some materials made available to encourage adoption of course-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

Copyright 2008 ACM 978-1-59593-947-0/08/0003 ...\$5.00.

work like the one described in this paper. Section 7 discusses related work. The paper concludes in Section 8.

2. OBJECTIVES AND ASSUMPTIONS

The primary purpose of the course was to aid undergraduates in developing large-scale problem solving skills. To that end, we identified a list of abilities for our students to acquire and refine:

- Thinking creatively about large-scale problems in a parallel fashion and designing parallel solutions
- Managing and working with large data sets under memory and bandwidth limitations
- Developing a solid foundation of algorithms and patterns frequently applicable to large-scale data
- Identifying and understanding the engineering trade-offs made in real production systems

When designing the course to meet these objectives, we made several assumptions about the students who would be taking the course and the most effective way to teach the above skills.

The students were primarily in their third or fourth year of study. We required them to have completed a data structures course, as well as a programming languages course which introduced them to functional programming concepts—though since this is not reinforced enough throughout our curriculum, some review was necessary. Students were not required to have taken any prior systems courses such as Operating Systems or Networking, and had minimal previous exposure to parallel computing constructs such as threads. Furthermore, we assumed no prior exposure to large data set processing.

Finally, it was our belief that the best way to impart the skills listed above was through hands-on training using both structured and free-form practical activities. A set of formal lab activities would provide students with initial training, which would then be internalized through a design project requiring significant independent thought and initiative.

3. NOVEL RESOURCES

Several software and hardware resources available to us represent technological advances over previous distributed computing courseware tools. In this section we describe the unique features of these components that are incorporated into our course offering. Later, in Section 5, we evaluate their effectiveness and value.

3.1 MapReduce and Hadoop

MapReduce is a Google-developed programming model for processing large data sets. Its power is derived from its simple yet flexible API, which frames parallel problems as a set of two functional operations: a many-to-many mapping of input data elements to intermediate key/value pairs, followed by a set of parallel many-to-one reduction steps from intermediate values with the same key to final results. Programmers specify “mapper” and “reducer” methods to perform their computation; the MapReduce engine then automates the distribution of these processing components over a cluster of machines, managing failures and rescheduling tasks as needed.

This design contrasts with most other distributed computing frameworks such as MPI or PVM, which offer programmers greater flexibility in arbitrarily structuring their programs in exchange for accepting a greater degree of system complexity—the underlying framework is used primarily for inter-process communication. MapReduce, on the other hand, factors out the common problems of repli-

cating services and managing reliability and process synchronization, allowing programmers to focus on the actual computation, while worrying less about the infrastructure. An example MapReduce program is given in Figure 3 in the Appendix.

Hadoop is an open source Java-based implementation of MapReduce and the Google File System (GFS). While MapReduce and GFS were originally designed at Google, external projects such as Hadoop have brought this computing paradigm to several other corporations and institutions across the computer science field. Our class ran Hadoop on top of commodity Linux machines, which comprised our computing cluster, described next.

3.2 Hardware Overview

Our computing base was a forty node Linux cluster. Each machine was powered by dual Xeon CPUs with 2 GB of RAM and 300 GB of hard drive space per processor, yielding a total of 80 processors, 160 GB of memory, and 24 TB of raw storage space. The machines were internally connected via gigabit Ethernet.

3.3 Eclipse Plugin

Students in the class used Eclipse¹ to develop Java applications. A plugin developed by IBM for interfacing with Hadoop clusters allowed students to interact with the distributed environment from within the IDE. This plugin is open source and has been contributed to the Hadoop project. A screen capture of the plugin in action is given in Figure 1.

4. CASE STUDY: SPRING QUARTER 2007

4.1 Course Overview

The curriculum for the course was divided into two major components: the systems-based background behind distributed computing, and the algorithms and application-level material built on top of it. These components are both described in more detail in the next two subsections.

We implemented the curriculum as a ten week offering aimed at upper-division students in the University of Washington Computer Science and Engineering department. For six weeks, students attended two hours of formal class (lecture and discussion) as well as two hours of semi-structured lab activity time. An optional design project completed the academic quarter; students who signed up for this received additional credit. Twenty-one juniors and seniors enrolled in the class, fourteen of whom also opted for and completed the design project.

4.2 Lectures: Engineering and Systems

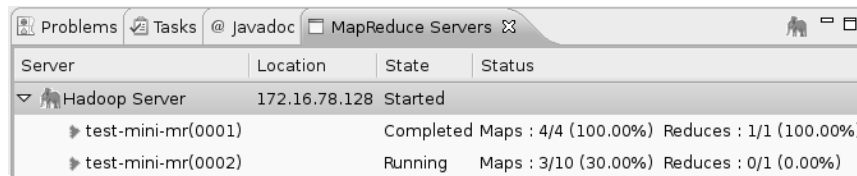
The lecture content [9] primarily focused on the design and analysis of distributed systems. We intended for the students to understand the decisions that went into successful system design, and develop an appreciation for trade-offs made during the engineering process.

Background material including functional programming principles, multithreaded programming and networking was reviewed. The students were then introduced to the specific distributed systems they would be working with: MapReduce and GFS.

To place these in the broader context of distributed computing, related systems such as BOINC [2], NFS and AFS [12] were introduced as well. Guest speakers provided more in-depth study of real systems by presenting the Nutch search engine [4] and the backend for Google Maps.² In discussions on these systems, students cov-

¹<http://www.eclipse.org>

²<http://maps.google.com>



Server	Location	State	Status
Hadoop Server	172.16.78.128	Started	
test-mini-mr(0001)		Completed	Maps : 4/4 (100.00%) Reduces : 1/1 (100.00%)
test-mini-mr(0002)		Running	Maps : 3/10 (30.00%) Reduces : 0/1 (0.00%)

Figure 1: The Hadoop plugin for Eclipse displaying the status of two MapReduce programs being executed on a cluster

- 1) Introduction to Distributed Computing and Parallelization
- 2) MapReduce: Theory and Implementation
- 3) Networks and Distributed Systems
- 4) Real-World Distributed Systems: Nutch and Google Maps
- 5) Distributed File Systems (NFS, AFS & GFS)
- 6) Other Distributed Systems (DNS, BOINC & OLPC)

Table 1: Lecture topics included in Spring 2007

ered such topics as expressibility of APIs, interoperability, scalability, reliability and performance. The full lecture syllabus is given in Table 1.

No course textbook was used; readings for each lecture were taken from academic papers about MapReduce or GFS, as well as other technical papers and articles (e.g., [2, 4]).

4.3 Labs: Algorithms and Applications

Students applied their understanding of distributed systems to the design of algorithms and applications in a distributed setting.

We deliberately did not focus on traditional distributed systems topics such as distributed synchronization, election algorithms, or transaction management. While these are necessary components of distributed systems, our projects focused on being clients of these elements rather than implementers.

The topics we wanted to cover were addressed by four lab programming assignments. The first lab walked students through setting up the Eclipse environment and interfacing with the cluster by adding and removing files, and running a simple MapReduce word count program. The second lab required performing a web crawl with Nutch, cleaning the corpus of markup, and creating a searchable inverted index. The third lab involved building a link graph representation of the Wikipedia corpus and implementing the PageRank [3] algorithm over it. The final lab required implementing canopy and k-means clustering over the Netflix Prize Data³ in order to create a naïve recommendation system.

4.4 Design Projects

The design project component was one of the driving motivations behind this course, as the projects would provide the primary mechanism for students to demonstrate their competency in distributed computing skills.

Each student wrote a project proposal, which identified a particular problem or question in any computation-based field. The proposal then listed sources of data that could be processed to evaluate the question and suggested an implementation plan. Students working in groups of 1–3 then had four weeks to implement the proposals. The eight strongest proposals were selected by students to carry through to completion. A few noteworthy projects are described in more detail in Section 5.1.

The students were encouraged to utilize the computing cluster’s

³<http://www.netflixprize.com>

available power as much as they could. Students loaded and processed data sets of up to a terabyte to solve a wide range of problems. Throughout the design project portion of the course, students met weekly with course staff to follow up on their progress. Course staff members offered suggestions when students were stuck, but the students primarily worked independently.

5. CASE STUDY: EVALUATION

The course offering was a success. Students were excited about the course and confident about the skills they learned, ranking the course highly on a standardized course evaluation survey. On a scale of 1–5, the course as a whole had a median score of 4.4, and the course content received a median score of 4.6. Ranked against all 400-level courses in the department with under thirty students, these values were in the 7th and 8th deciles respectively, and were in the 8th and 9th deciles respectively compared across the entire department. Our offering earned higher scores than several other design project-based courses offered in the department the same quarter.

While the size of our course offering was limited due to its experimental nature, our vision for this course is to be a first-class component of the upper-division undergraduate curriculum on par with established courses such as Operating Systems. In response to the positive feedback generated so far, our department plans to offer the course in an increased size as part of its regular time schedule beginning in the 2007-2008 academic year.

Furthermore, the Hadoop platform proved its merit as an educational tool. Students were able to understand the interface very quickly, which allowed the majority of the time to be spent on developing solutions rather than “teaching to the tools.” Also, because Hadoop is based on Java, students were able to use a language with which they were already familiar. Finally, Hadoop had low administrative needs; our student system administrator worked about three hours per week during the quarter after the initial setup which required two days.

5.1 Successful Projects and Skills Learned

As expected, the design project proved to be a strong course component. The success of the students at implementing their projects demonstrates the efficacy of our course at helping the students to acquire the skills listed in Section 2.

One group implemented a Bayesian filter to examine the Wikipedia RSS feed for spam. Another built an n-body simulator that successfully modeled an 80,000 body representation of the Milky Way and Andromeda galaxies colliding. An unsupervised synonym extractor for online documents by one student scaled to millions of pages.

One particularly impressive project was Geozette,⁴ a news web site that associates stories with geographic location. Its pipeline processed 200 RSS feeds, calculated the geographic location of each article in each feed, clustered them by subject and then pushed

⁴<http://www.geozette.com>



Figure 2: Geozette, a student project which clusters news articles and correlates clusters with geographic location

the stories to a web site frontend utilizing the Google Maps API. A screenshot of Geozette is given in Figure 2.

To produce their results, students acquired their own data sets, demonstrating their ability to evaluate the utility and manageability of available data. They then transformed these data sets into forms that could be used with Hadoop. The data sets they acquired were as large or larger than those used in labs, proving they could scale the techniques they learned to larger projects.

Furthermore, their implementations demonstrate a synthesis of skills going beyond what was taught in labs. Geozette used a continuous processing cycle to extract and incorporate fresh news articles in real time. The Bayesian Wikipedia filter processed a very large data set; these students encountered new bandwidth limitations which they overcame on their own. Both the synonym extractor and Geozette were based on algorithms taken from research papers; the students independently selected these algorithms and transformed them into parallelized implementations. Several projects employed publicly available web APIs. Finally, another project applied MapReduce to video processing—a completely different domain of problem.

There were a few factors contributing to the success of the independent design projects. The work that went into the labs gave the students a solid grounding in developing algorithms to run on Hadoop. Each team met with two teaching assistants each week to get support for data, algorithms, and help with Hadoop. The teaching assistants were already familiar with Hadoop and distributed systems programming before the beginning of the course. Finally, Hadoop itself proved to be a manageable system for students to grasp.

5.2 Lessons Learned

While the course in general was a success, there were particular areas identified for improvement in future offerings.

One lesson learned about the design projects was that while impressive results were generated by students this quarter, the four week time frame is a frustrating constraint on planning and completing valuable projects. The next version of this course will see a ten week offering focused exclusively on lectures and labs, fol-

lowed by a second ten week course for students who wish to pursue a design project. This will allow them to explore a serious cluster computing project with demands on par with other Capstone projects in our department.

Another area for improvement is the in-lecture content. Our 2007 course offering saw a disconnect between applications-level tasks (which were primarily handled in project labs) and systems background (which took the majority of the lecture time). Future iterations of the course should see a better integration of these topics and will focus more heavily on the development of distributed applications and further deemphasize the systems component.

Finally, more efficient management of our cluster would have enabled the use of fewer computing resources. Over one academic quarter, we observed that approximately 15,000 CPU-hours of computing time were consumed by our students. This represents less than 20% average load on the cluster, but was primarily compressed into particular peak usage periods. Optimizing peak resource demand by staggering project deadlines and improving the job scheduling algorithm in Hadoop would have allowed us to use approximately one processor per student.

Other universities looking to offer a similar course may not necessarily need to dedicate a cluster to the class; resources such as low-priority tasks on existing clusters or in computing labs could be leveraged to provide a sufficient computing base. Alternatively, commercially-available resources such as Amazon’s Elastic Compute Cloud could be employed. Based on our observations, we estimate that this resource would currently cost under \$150 per student, and we would expect this price to drop over time. Finally, students can run individual instances of Hadoop from within virtual machines on their own computers, providing a functionally equivalent but less powerful experience.

6. MATERIALS ONLINE

To encourage other universities to include Hadoop-based cluster computing components in their coursework, we have made a number of resources available through Google Code For Educators.⁵ This includes lectures and lab activities used to teach at the University of Washington [9] as well as additional teaching materials.

We were also interested in lowering the technical barriers to getting started with Hadoop. This summer we worked with Google engineers who have developed a ready-to-use VMWare virtual machine image that includes a preconfigured Hadoop node running within a Linux environment. This allows turn-key use of Hadoop for individuals who want to experiment with the platform without fully configuring a computing cluster themselves. The virtual machine image, the Hadoop plugin for Eclipse, as well as our educational materials are publicly available for download at the Google Code For Educators web site.

7. RELATED WORK

Undergraduate courses in distributed computing are not a new concept. For example, one is presented in [5]. The ACM Computing Curricula 2001 for Computer Science [1] provides two course outlines which cover material similar to our offering: “Architectures for Networking and Communication” (CS222), and “Net-centric Computing” (CS230). Our course design provides a new way to communicate these topics to students in a manner more suited to the evolving state of modern distributed systems.

We contend that traditional platforms for distributed systems programming such as MPI or PVM (as Cunha and Lourenço used) are

⁵<http://code.google.com/edu/>

unnecessarily complicated for a first introduction to distributed systems programming, a view corroborated by Pheatt in [10]. Pheatt specifically notes that PVM, MPI and BOINC-based clusters require high amounts of overhead. While he then proposes his own distributed framework, we believe that there is value to using existing high-performance, low-overhead open-source systems like Hadoop. The MapReduce-based environment allows students to focus heavily on creating solutions to real problems and eliminates the high learning curve associated with other tools while still being a viable real-world system already in use by several industry leaders.

8. CONCLUSIONS

We described the design of a course that uses distributed computing to motivate student projects in large-scale data processing. Students learned new techniques for managing web-scale data sets and processing them in an efficient manner by working with a real-world, open source distributed computing platform. Our students focused on application-level algorithms and completed design projects demonstrating their mastery of distributed data processing skills. Our student feedback metrics as well as our own evaluation of student performance demonstrated the success of this method.

Our teaching materials and additional resources are available through Google Code For Educators to others in the academic community who wish to explore Hadoop-based course initiatives.

Acknowledgments

The development and execution of this course and related materials would not have been possible without the dedicated help of several individuals. Our many thanks are extended to Taj Campbell, Jack Hebert, Hannah Tang, Christophe Taton and Albert Wong. We would also like to thank the several individuals who reviewed this paper and provided many insightful comments to improve it.

9. REFERENCES

- [1] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 2001. IEEE Computer Society and Association for Computing Machinery.*, 2001.
- [2] D. P. Anderson, C. Christensen, and B. Allen. Designing a runtime system for volunteer computing. *Proceedings of the 2006 IEEE/ACM SC06 Conference*, Nov. 2006.
- [3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.
- [4] M. Cafarella and D. Cutting. Building Nutch: Open source search. *ACM Queue*, Apr. 2004.
- [5] J. C. Cunha and J. Lourenço. An integrated course on parallel and distributed processing. In *SIGCSE '98: Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, pages 217–221, New York, NY, USA, 1998. ACM Press.
- [6] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, December 2004.
- [7] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. *SIGOPS Operating Systems Review*, 37(5):29–43, 2003.
- [8] Hadoop. <http://lucene.apache.org/hadoop/>.
- [9] A. Kimball and S. Michels-Slettvet. CSE 490H lecture notes: Problem solving on large scale clusters. http://code.google.com/edu/content/submissions/uwspr2007_clustercourse/listing.html, 2007.
- [10] C. Pheatt. An easy to use distributed computing framework. *SIGCSE '07: Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education*, pages 571–575, 2007.
- [11] M. Sahami. Scaling computer science education to education on scaling in computer science. *Workshop on Integrative Computing*

Education & Research (ICER): Preparing IT Graduates for 2010 and Beyond, Jan. 2006.

- [12] M. Satyanarayanan, J. Howard, D. Nichols, R. Sidebotham, A. Spector, and M. West. The ITC distributed file system: Principles and design. In *Proceedings of the 10th ACM Symposium on Operating System Principles*, pages 35–50, New York, NY, USA, Dec. 1985. ACM Press.

APPENDIX

A. EXAMPLE MAPREDUCE PROGRAM

```
public static class LineIndexerMapper
    extends MapReduceBase implements Mapper {

    private final static Text word = new Text();
    private final static Text summary = new Text();

    public void map(WritableComparable key,
        Writable val, OutputCollector output,
        Reporter reporter) throws IOException {

        String line = val.toString();
        //intermediate value is "filename:linenum"
        summary.set(key.toString() + ":" + line);
        StringTokenizer itr =
            new StringTokenizer(line.toLowerCase());
        while (itr.hasMoreTokens()) {
            //for each word, emit (word, "file:line")
            word.set(itr.nextToken());
            output.collect(word, summary);
        }
    }

    public static class LineIndexerReducer
        extends MapReduceBase implements Reducer {

        public void reduce(WritableComparable key,
            Iterator values, OutputCollector output,
            Reporter reporter) throws IOException {

            //aggregate "file:line" strings from a given
            //word into one string.
            boolean first = true;
            StringBuilder toReturn = new StringBuilder();
            while (values.hasNext()) {
                if (!first)
                    toReturn.append('^');
                first=false;
                toReturn.append(values.next().toString());
            }
            output.collect(key,
                new Text(toReturn.toString()));
        }
    }
}
```

Figure 3: An example MapReduce program written for Hadoop

Figure 3 gives an example MapReduce program written for Hadoop. Some initialization code is omitted. This program generates an index over a set of documents mapping words to files and lines. The key feature of this program is its reliance on Hadoop to perform fundamental operations such as replication, data distribution, and output aggregation; the programmer simply implements the *Mapper* and *Reducer* interfaces to process the data.