CSE490H / Winter 2009

Version 5: January 13, 2009

Scalable systems infrastructure projects:

Implement the Chord distributed hash table system. Probably too easy, but a way to get started; to flesh it out, augment with Paxos each node in Chord is a cluster of nodes that run Paxos for fault tolerance.

Implement the latest, greatest Byzantine Fault Tolerance protocol (probably Zyzzyva), and figure out how to partition a keyspace for it to scale. Extra credit: use it to build some interesting replicated service (NFS server? a BFT version of Chubby?).

Scalable systems projects:

Design and build a scalable Internet anomaly detector over a very high throughput event stream -- perhaps layer on dryad/linq, but the goal would be low-latency as well as high throughput. Could be used for all sorts of things: intrusion detection, Eytan Adar-style "something interesting is happening" detection over news events, etc.

Build a large-scale, content-addressable archival store (e.g., Venti from FAST '02). Prototype it on EC2/S3/something and expose as a service, and maybe layer an interesting file system on top of it.

Implement a continuous, large scale version of spycrawler on Amazon's platform.

[quarter baked idea] marry some cool web site structure visualizer with hadoop to generate visualizations at scale, continuously crawl CSE department pages, and use it to show something interesting -- e.g., build a "what's new" meta-site for CSE that highlights changes to CSE web pages. Or, instead of a visualizer, maybe implement a "dead hyperlink checker + weblint + something else" hadoop job that scrubs the CSE pages and generates reports. Or, find and prototype one of the cross-site scripting vulnerability checking approaches, and scrub pages with that.

A user-level file system using p2p storage as the back end, e.g., so you can spread overload files out over the network.

Build a scalable server for an inverted index of some large corpus.

Port a distributed programming language like Erlang to run on AWS.

Solve the reduce straggler problem of MapReduce.

Astrophysics application projects:

We have a specific astrophysics application working on Hadoop (right, Slava?). We'd like to get it ported to Microsoft's Dryad. (We have Dryad systems in the department.) This is interesting because Dryad and Dryad/LINQ have somewhat different characteristics than Hadoop; worth learning about; although this particular application is unlikely to take particular advantage of those characteristics. [Update: We think it would be a pain in the butt to move this to Dryad, because the astrophysics core of the application has a lot of Unix dependencies in it. In other words, just getting it to run sequentially in a Windows environment would take a fair amount of effort.]

We have several other astrophysics projects that a great young prof in Astronomy would like CSE students to take a look at. These could be done on Hadoop or Dryad. A short description is below; a longer description that provides plenty of background is at http://lazowska.cs.washington.edu/ac.pdf.

Astrophysics is addressing many fundamental questions about the nature of the universe through a series of ambitious wide-field optical and infrared imaging surveys (e.g. from the properties of dark matter to the nature of dark energy). These surveys pose many fundamental challenges: how do we analyze and interact with data that is being taken at a rate 1000 times greater than existing experiments; how do we determine the interdependencies between the observable properties of stars and galaxies in order to better understand the physics of the formation of the universe when the data contain 1000s of dimensions and tens of billions of sources; how do we combine disjoint images, by collating data from several distinct surveys with different wavelengths, scales, resolutions and times in order to identify moving or variable sources.

What limits astronomy today is no longer obtaining or storing data; it is the question of how do we interact with and analyze Petabyte scale data repositories. Serial access to data sets via SQL queries will not scale to the size of the science questions we wish to address. We are reaching a stage where the data are much richer than the analyses we apply to them. Astronomy is, therefore, in need of a new model for knowledge discovery with massive data sets; one where the analysis moves to the data rather than from the database to the user, where the application scales elastically to the size of the problem in hand and where the figure of merit is not just the efficiency of the algorithm but the time from asking the question of the data and receiving the answer.

In the proposed research we will explore how the emerging map-reduce paradigm for cloud computing might revolutionize the way that the sciences approach access to and analysis of a new generation of massive data streams.

Projects

1. Create a spill-tree for indexing massive astronomical image archives so that images observed at different wavelengths can be cross- referenced to provide a multi-spectral image covering the X-ray, optical and infra-red wavelengths (a panchromatic view of the

sky). Spill-trees are a simple tree structure used to find k-nearest neighbors. The “spill tree" (Liu et al 2004) is distributed across multiple nodes and where some fraction of an adjacent node's domain is replicated in a local node's memory (i.e. part of the tree is duplicated on adjacent nodes so that neighbors that lie across boundaries between nodes do not need to communicate with the adjacent node). This is an efficient way to search trees without requiring low latency compute clusters.

2. Develop an image processing framework that takes overlapping images, warps these images to a common reference frame and detects sources within the multispectral stack. This will use the Hadoop map and shuffle operations to warp and detect sources (map) and then merge the source lists (shuffle). The data will be 30TB of images from the SDSS and 2MASS imaging surveys.

3. Search for Mars-like planets in the outer solar system. Planets could reside beyond the orbit of Pluto but we have yet to detect them (it is 1 in a million that they will be there but it is front page news if we found one). We can search for them by taking a time series of images and assuming a trajectory (or orbit) for the planet and looking for sources along these trajectories. This is data and cpu intensive as it requires searching many trajectories for all pixels in all images. The project is to take a series of 40TB of images from the SDSS which were taken over a 10 year period and search for candidate planets by distributing the search algorithm across multiple processors.

Other application projects:

The digital animation capstone course does its rendering using Maya on a bunch of PCs. Assuming that the terms of our license permit it, doing a smooth job of making this run on AWS would be an extremely useful and fun project – the speed of the rendering would be limited by Hank Levy’s ProCard rather than by the number of department PCs we can scam at the last minute. (I’m trying to deal with the licensing issues related to this; the project could be done regardless of this, but the results might not be usable, which would be a downer.)

Last quarter’s map project (Assignments 3 and 4) could be extended in various interesting ways that would constitute a fine project.

Implement Map/Reduce for operation on a graphics card. There actually has already been a paper written about this: http://web.mit.edu/rabbah/www/conferences/08/stmcs/papers/catanzaro-stmcs08.pdf. So the project would be to read this work, familiarize yourselves with GPU programming (most likely CUDA on NVidia hardware) and then implement M/R. Ideally, then run a comparison of M/R on Hadoop vs. M/R on graphics hardware and analyze it in terms of performance, power, cost, etc.

Create a traffic predictor, using historical traffic data (Google can help in obtaining data). Depending on the team size and time, interesting things to add to the predictor to make it more accurate/interesting would include things such as weather and special events.

Word bigrams, compute frequencies for word1 word2 pairs - could create a concordance index too.

City "bigram" distances - build a table of distances from city to city using geodata from previous exercises

Compute graphic tiles of the night sky from astronomical data

Implement a clustering algorithm on top of M/R (like EM inference, where it is map, reduce, reduce, reduce, reduce, ... until convergence)

Implement a join, matching one data source (city locations & names) with another data source (city costs, pollution, etc.) to produce another data set for location of (cost, pollution, etc).

Genetic Algorithms: It should be easy to develop a general framework. There is a wide range of problems to apply it to, from simple TSP to Genetic Programming for the brave.

Use the ENSEMBL genetic data already on AWS and create a phylogenetic tree.

Implement some basic statistical adaptation techniques such as HMM training (the forward-backward algorithm) and SCFG training (the inside-outside algorithm). These are interesting because even though each iteration is easy to do, the setup time for each iteration is huge, and so the real challenge lies in figuring out how to run many iterations.