Grant Proposal

Andrew Rosen

Abstract

Distributed Hash Tables (DHTs) are protocols and frameworks used by peer-to-peer (P2P) systems. They are used as the organizational backbone for many P2P file-sharing systems due to their scalability, fault-tolerance, and load-balancing properties. These same properties are highly desirable in a distributed computing environment, especially one that wants to use heterogeneous components.

We show that DHTs can be used not only as the framework to build a P2P file-sharing service, but as a P2P distributed computing platform. We propose creating a P2P distributed computing framework using distributed hash tables, based on our prototype system ChordReduce. This framework would make it simple and efficient for developers to create their own distributed computing applications. Unlike Hadoop and similar MapReduce frameworks, our framework can be used both in both the context of a datacenter or as part of a P2P computing platform. This opens up new possibilities for building platforms to distributed computing problems.

One advantage our system will have is an autonomous load-balancing mechanism. Nodes will be able to independently acquire work from other nodes in the network, rather than sitting idle. More powerful nodes in the network will be able use the mechanism to acquire more work, exploiting the heterogeneity of the network.

We plan on running large scale experiments using both cloud-computing platforms and small, consumer-grade hardware. This variety of computing device will establish the flexibility of our framework and its ability to use the heterogeneity of the network to its advantage.

By utilizing the load-balancing algorithm, a datacenter could easily leverage additional P2P resources at runtime on an as needed basis. Our framework will allow MapReduce-like or distributed machine learning platforms to be easily deployed in a greater variety of contexts. These would include any computationally intensive problems that occur in a wide variety of scientific disciplines, such as biology, chemistry, astronomy and physics.

1 Summary

What are DHTs

My dissertation focuses on creating a generalized framework for distributed computing based on Distributed Hash Tables (DHTs). Distributed Hash Tables are a tool which allows a computers (nodes) to self-organize into a decentralized network for storing and retrieving data. They have been used extensively to build peer-to-peer (P2P) file-sharing applications [3] as well as distributed storage systems [4].

DHTs are primarily used in P2P applications, but other applications, such as botnets, use DHTs for their decentralization. Our goal is to use DHTs primarily for their intuitive way of organizing a distributed system and to further extend the use of DHTs. In previous work [7], we showed that a DHT can be used to create a distributed computing framework. We used the same mechanism used in P2P applications that assigns nodes their location in the network to evenly distribute work among members of a DHT. The most direct application of a DHT distributed computing framework is a quick and intuitive way to solve embarrassingly parallel problems, such as:

- Brute force cryptography.
- Genetic algorithms.
- Markov chain Monte Carlo methods.
- Random forests.
- Any problem that could be phrased as a MapReduce [5] problem.

Unlike the current distributed applications that utilize DHTs, we want to create a complete framework that can be used to build decentralized applications. We have found no existing projects that provide a means of building your own DHT or DHT based applications.

Completed Work

There are many different types of DHTs, but all DHTs share very important qualities. They are *scalable*, which means that each additional node in the network minimally impacts the cost of keeping the network organized. DHTs are also highly *fault-tolerant*. Unlike many other systems, DHTs assume that nodes will be continuously entering and leaving the network. Because of the way DHTs are organized, they can handle large scale failures, such as a power outage affecting an entire city. The last quality of DHTs is that they are *load-balancing*. This means the data stored in a DHT is evenly distributed among nodes in the network.

All of these qualities are highly desirable in distributed computing, so it was a natural step for us to build distributed computing framework based on DHTs. We have shown in previous work that DHTs can be used to distribute work among nodes the same way data is distributed [7].

ChordReduce

ChordReduce [7] was a proof of concept for using distributed hash tables for distributed computing. It used Chord [9], a well studied DHT, and exploited its various features to perform MapReduce [5] tasks, a very popular method for framing distributed computing problems.

Unlike other distributed computing frameworks, we made no assumptions as to the context in which ChordReduce would be used. ChordReduce could be used in either a large datacenter or in completely heterogeneous, peer-to-peer context. In a heterogeneous network, we assume all the nodes represent different pieces of hardware, with differing levels of computational power and reliability. As a result, we needed to rigorously test the fault-tolerance of the system and made an exciting discovery while doing so.

To test fault tolerance, each node essentially flipped a coin weighted in its favor, and when it lost the flip, left the network and reentered as a new node. This simulated a P2P concept called *churn*, turbulence in the network caused by the continuous entering and leaving the network. Churn is normally a chaotic influence on the network, but we found that at high levels of churn, nodes advantageously redistributed work in such a way that tests with high levels of churn performed better than tests with low levels of churn.

ChordReduce established three important ramifications. First, we experimentally demonstrated that DHTs are capable of being used as a framework for distributed computing. Second, DHT based distributed computing can have nodes enter the network at any time, even when a job is running, and be immediately put to work. Finally, we discovered that while churn is normally a disruptive force, at high levels it can be helpful to the network.

UrDHT

We built UrDHT [8], a generalized framework for building distributed hash tables, using the concepts we discovered creating DGVH [2]. UrDHT essentially acts as a "fill-in-the-blanks" for creating distributed hash tables. The novelty of UrDHT is that it makes it easy for scientists in any field to construct a DHT based application. For computer scientists, UrDHT provides a means of rigorously comparing different architectures.

Remaining Work

Now that we have UrDHT, we want to use it to create a platform for distributed computing. This differs from our previous work in ChordReduce in a few very important ways.

As we mentioned earlier, ChordReduce tested distributed computing on a single type of DHT - Chord. UrDHT is much more flexible and can be used to implement many different DHTs. We also plan on doing large scale tests with UrDHT, something we weren't able to do with ChordReduce due to limitations of resources. Finally, we will tackle more distributed problems than just MapReduce problems, such as Monte-Carlo approximations.

The most exciting feature we plan on adding to our distributed computing platform is a strategy we are calling *autonomous load-balancing*. We plan on exploiting the discovery we made about churn during our ChordReduce experiments. We want to create strategies that would allow nodes to reassign themselves in the network. This would allow nodes to automatically redeploy themselves to assist with completing the tasks at hand.

We also plan on creating similar strategies to account for heterogeneity in the network. Rather than having stronger nodes reassign themselves, they can represent themselves multiple times in the network, pretending to be multiple nodes. The stronger the hardware, the more nodes it can pretend to be. This is a reversal of the concept of an attack on a DHT, where an attack attempts to control a DHT by claiming to be multiple nodes in the network [6]. However, instead of using this concept as an attack on the network's security, we are using it to benefit the network.

Ramifications

Our research would have many applications within the field of Computer Science. Traditional MapReduce frameworks and other distributed computing solutions are limited to being deployed solely in a datacenter [1]. Distributed computing frameworks built with UrDHT would be able to be deployed any context, be it a data center, a P2P network, or a volunteer computing pool. Furthermore, the autonomous load-balancing feature would allow frameworks to automatically give nodes with more power more work.

Our framework would not only be of interest to computer scientists. The "plug-and-play" nature we envision would make it easy for experts in other fields to use our framework to solve computationally intensive problems. Some example problems are Monte-Carlo Methods (of interest to mathematicians and economists) and machine learning (of interest to biologists).

This project would allow both organizations with large amounts of computing power and average developers with fewer resources to spend less time setting up and configuring their hardware to work together. The goal is for our software to make distributed computing more of matter of "plug-and-play," allowing researchers to spend less time setting up and maintaining computing platforms.

Methodology

The method to validate our hypotheses and theoretical models is through large-scale real world experiments using variety of devices across the world. An effective approach to his is purchasing cloud computing resources from Amazon and Google.

While this would test the network by simulating either a P2P environment or centralized datacenter, most of the machines would be relatively the same strength. In order to fully test how well our strategies perform in heterogeneous networks, we need lower powered devices, which are individually inexpensive and easily deployable.

We have successfully developed our software to achieve this, but need funding to do real world network tests. These tests would allow us to experimentally show that our software conclusively works on a large scale and on heterogeneous networks.

References

[1] Virtual hadoop. http://wiki.apache.org/hadoop/Virtual

[2] Brendan Benshoof, Andrew Rosen, Anu G. Bourgeois, and Robert W Harrison. A distributed greedy heuristic for computing voronoi tessellations with applications towards peer-to-peer networks. In *Dependable Parallel, Distributed and Network-Centric Systems, 20th IEEE Workshop on*.

[3] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[4] Frank Dabek, M Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-Area Cooperative Storage with CFS. *ACM SIGOPS Operating Systems Review*, 35(5):202–215, 2001.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[7] Andrew Rosen, Brendan Benshoof, Robert W Harrison, and Anu G. Bourgeois. Mapreduce on a chord distributed hash table. In *2nd International IBM Cloud Academy Conference*.

[8] Andrew Rosen, Brendan Benshoof, Robert W Harrison, and Anu G. Bourgeois. Urdht. `https://github.com/UrDHT/`.

[9] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *SIGCOMM Comput. Commun. Rev.*, 31:149–160, August 2001.

2 Vita

Education

Ph.D. in Computer Science, Georgia State University. May 2016 (Expected)
M.S. in Computer Science, Georgia State University. May 2014, 3.89 GPA
B.S. in Computer Science, Georgia Institute of Technology. May 2010, 3.00 GPA
Minor in Music, Georgia Institute of Technology. May 2010

Appointments

2CI Astroinformatics Fellow, Georgia State University, Aug 2012 - Present
Graduate Research Assistant, Georgia State University, Aug 2011 - Present
Graduate Lab Assistant, Georgia State University, May 2011 - 2013

Publications

1. Andrew Rosen, Brendan Benshoof, Robert W. Harrison, Anu G. Bourgeois "MapReduce on a Chord Distributed Hash Table" Presentation ICA CON 2014, Poster at IPDPS 2014 PhD Forum

2. Brendan Benshoof, Andrew Rosen, Anu G. Bourgeois, Robert W. Harrison "VHASH: Spatial DHT based on Voronoi Tessellation" ICA CON 2014

3. Erin-Elizabeth A. Durham, Andrew Rosen, Robert W. Harrison "A Model Architecture for Big Data applications using Relational Databases" 2014 IEEE BigData - C4BD2014 - Workshop on Complexity for Big Data

4. Chinua Umoja, J.T. Torrance, Erin-Elizabeth A. Durham, Andrew Rosen, Dr. Robert Harrison "A Novel Approach to Determine Docking Locations Using Fuzzy Logic and Shape Determination" 2014 IEEE BigData - Poster and Short Paper

5. Erin-Elizabeth A. Durham, Andrew Rosen, Robert W. Harrison "Optimization of Relational Database Usage Involving Big Data" IEEE SSCI 2014 - CIDM 2014 - The IEEE Symposium Series on Computational Intelligence and Data Mining

6. Brendan Benshoof, Andrew Rosen, Anu G. Bourgeois, Robert W. Harrison "A Distributed Greedy Heuristic for Computing Voronoi Tessellations With Applications Towards Peer-to-Peer Network" IEEE IPDPS 2015 - Workshop on Dependable Parallel, Distributed and Network-Centric Systems

7. Chaoyang Li, Andrew Rosen, and Anu G. Bourgeois "A Novel Approach to Efficiently Detect 3D Full-View Coverage for Camera Sensor Networks" Submitted to IPDPS 2016

8. Andrew Rosen, Brendan Benshoof, Robert W. Harrison, Anu G. Bourgeois "UrDHT: A Unified Model for Distributed Hash Tables" Submitted to IPDPS 2016

9. Andrew Rosen, Brendan Benshoof, Robert W. Harrison, Anu G. Bourgeois "The Sybil Attack on Peer-to-Peer Networks From the Attacker's Perspective" In preparation

Service

Vice Chair, Georgia State University Chapter of the Association for Computing Machinery (ACM), May 2012 - Present
Treasurer, Georgia State University Chapter of the Association for Computing Machinery (ACM), May 2012 - May 2014
New Graduate Student Orientation Panelist, Georgia State University, 2014-2015
Department Representative, Georgia State University Arts and Sciences Tech Fee Committee, 2013 - 2015

External Funding

TCPP Travel Grant for 28th IEEE International Parallel & Distributed Processing Symposium

Honors and Awards

Outstanding Graduate Student Teaching Award, Department of Computer Science, Georgia State University, 2014

Employment

Instructor, Georgia State University, Fall 2011 and 2012 (CSc 3410), Spring 2014 (CSc 2010), Spring 2015 (CSc 3410)
Developer, Georgia Tech Sonification Lab, Atlanta, GA May-Dec 2010
Undergraduate Researcher, Georgia Tech Sonification Lab, Atlanta, GA Fall 2007 -Dec 2009

3 Budget and Justification

Item	Amount requested
Cloud Computing Server Time	$1300
12x Raspberry PI 2	$500
Raspberry PI peripherals	$200
Total Amount requested	$2000

Cloud Computing Costs

The base cost for a cloud computing machine ranges between $0.008 and $0.075 per machine per hour. We need at least 100 machines for at least 100 hours of experimentation at the bare minimum. Buying cloud computing time from both Google and Amazon would allow us to simulate realistic networks that span continents and the globe.

Raspberry PI Costs

As we mentioned in the summary, we need more machines in order to test heterogeneity in the network. Raspberry PIs allow us to test the lower end of computing power. Raspberry PIs are credit-card sized computers which are powerful enough to test our software, but much weaker in computational power compared to what we can purchase from Amazon. They would allow us to our autonomous load-balancing and see how well our algorithms distribute work between stronger and weaker nodes. Furthermore, we could attach some to different network interfaces, making some wired and some wireless, which would allow us to further vary our network topology.

An individual Raspberry PI 2 Model B, the latest version, is approximately $40 on Amazon. These machines are bare bones and do not come with power adapters, wifi, or SD cards to serve as a hard drive.