<<Position Paper Title Here>>

CS680 Distributed Software Development Position Paper (Spring 2010)

Sammie Stahlback

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

Sws28@drexel.edu

Jordan Osecki

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

Jmo34@drexel.edu

Michael Kim

Drexel University  
3141 Chestnut Street  
Philadelphia, PA 19104

Mk394@drexel.edu

# ABSTRACT

MapReduce distributed systems typically are restricted to a subset of computers in a close geographic location. These computers are also typically pre-assigned to be either a “master” or “worker”. The jobs that are accomplished are usually submitted by whoever is overseeing the operation. Therefore, the system has limited processing power by computers which cannot be easily reassigned and if there is a failure, there is no way to recover from it without human intervention.

This paper proposes a new type of MapReduce system. This system would be on a Peer-to-Peer (P2P) network and would be de-centralized. The system would employ self-configuration, setting up its topology of workers, master, and job client for each job. The system would employ self-healing by reacting if any of the three worker types fail during the middle of the process.

The result is a system which can perform MapReduce operations with access to unlimited processing power in the form of any nodes that wish to connect from anywhere. The roles of master, worker, and job client can dynamically change between jobs using the self-configuration properties, meaning that anyone who wants to submit jobs can and any node can also lead the process as master. If a failure does occur, it can use the self-healing properties to recover by replacing the failed node. If a worker fails, grab a free node or another worker and re-distribute the failed node’s work to them. If the master fails, have the nodes appoint a new one, which can take over at any point during a job.

The group behind this paper produced a simulation version of this software and tested it under regular MapReduce conditions and under failures. Data is presented showing the benefits of self-healing and self-configuring, as well as discussions about taking the simulation further and applying the concepts to a real world model. There are also discussions and results presented regarding if this system is more beneficial than a typical MapReduce system.

**Categories and Subject Descriptors**

C.2.4 [**Distributed Systems**]: Distributed applications.

**General Terms**

Algorithms, Management, Performance, Design, Reliability, Experimentation, Theory.

**Keywords**

Distributed systems, peer-to-peer network, de-centralized, MapReduce, self-healing, self-configuration, self-\*.

# TODO

TODO

TODO

# TODO

TODO

TODO

# TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

# TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

## TODO

TODO

TODO

# TODO

TODO

TODO

# TODO

TODO

TODO

# ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

The team would also like to thank Professor Valetto for helping to guide the project in the right direction and raise issues for the team to consider.

# BIBLIOGRAPHY

1. Cardona, K., Secretan, J., Georgiopoulos, M., and Anagnostopoulos, G. 2007. A Grid based system for data mining using MapReduce. <http://cygnus.fit.edu/amalthea/pubs/Cardona_Secretan_TR-2007-02_AMALTHEA.pdf>
2. Dean, J., and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM, Vol. 51, No. 1, January 2008,* pp. 107-113. <http://www.scribd.com/doc/240523/MapReduce-Simplified-Data-Processing-on-Large-Clusters>
3. Hadoop, <http://hadoop.apache.org/core/>
4. Marozzo, F.,Talia, D., and Trunfio, P. Adapting MapReduce for Dynamic Environments Using a Peer-to-Peer Model. Extended Abstract. <http://www.cca08.org/papers/Poster7-Domenico-Talia.pdf>
5. SETI@Home, <http://setiathome.berkeley.edu>
6. Skynet, <http://rubyforge.org/projects/skynet>
7. Yang, H., Dasdan, A., Hsiao, R., and Parker, D. Map-Reduce-Merge: Simplified Relational Data Processing on Large-Scale Clusters. *SIGMOD’07, June 12-14, 2007, pp. 1029-1040.*

# APPENDIX

TODO

TODO

De-centralized Peer-to-Peer MapReduce System

Omar Badran, William Shaya, and Jordan Osecki

**The following is the document describing precisely how this project’s work reflected the original proposal and factors that have turned out differently in any way:**

The project for the most part followed our proposal perfectly. The team stayed with the basic ideas, used the same research throughout, and found ourselves only clarifying certain things that were left purposely undefined in the proposal, pending further research. Examples of this include some of the algorithms that were going to be deployed at each step of the MapReduce process.

The simulation itself stayed relatively the same to our proposed model. The basic ideas stayed the same. The node structure with all of the classes and methods outlined did not change throughout our implementation. The self-\* algorithms and methods of self-healing and self-configuration chosen also stayed the same.

The largest change that occurred between our initial proposal and our finished product was that the team decided to follow the approach of a MapReduceMerge-type system, since this would allow the master to only have to assign jobs, the workers to only have to do their jobs, and the submitter node to only have to receive and merge the individual results. The team saw this as an opportunity to gain even more advantages over the traditional MapReduce system. By using this new approach, it significantly reduces the work of the master, making the system more balanced and less dependent on the master. With this less work, there is no longer any reason for the master to handle any of the data, so the job client submitting the job is responsible for holding on to it. This is beneficial because if a central repository or the master held on to the job and either failed, it would be catastrophic. But in this scheme, if the job client fails, it is losing the job that it owns, so it has incentives to stay up and if it loses its data, then the rest of the workers and master do not care because they had no stake in the submitted job.

Another change that occurred between the proposal and final implementation was that the team set one of its main goals to be to prove that the churn of a P2P network is more than offset by the available number of nodes. It was always a goal to focus on how the self-\* algorithms, de-centralization, and Peer-to-Peer network would be benefits over the traditional MapReduce closed networks, but it became apparent for the team early that Peer-to-Peer would bring some negatives along as well as the positives, so it would be vital to show that the positives outweighed both the negatives of it and negatives of the traditional system.

The final change between proposal and final implementation was that the team settled on algorithms for master assignment and worker assignment, which involved efficiency values. In the proposal, the team discussed efficiency values as future work. However, for the final implementation of the system, the team chose to incorporate random values in its algorithms which would represent efficiency values. In the future, these values will be some measure of a node’s efficiency to make the network even more efficient.