Decentralized Distributed Processing

Michael Tao
Thomas Jefferson High School for Science and Technology
Alexandria, Virginia

June 13, 2007

Abstract

With the large quantities of data being collected every day, a single computer's CPU's computational ability to analyze the data and to utilize meaning behind the data is less than satisfactory. In order to mine through the data within certain time constraints, a collection of computers is needed. The purpose of this project is to produce a medium for distributing the load of tasks to networked peers with varying computing power in an efficient manner. This will distribute the work load from one computer to other computers within a network of peer computers by sending portions of the data and the proper analytical tools to all of the specified peers while also computing various peer's tasks. Peers can be running on multiple computer platforms such as Windows and Linux.

Keywords: High performance, Data analysis, decentralized, distributed processing

1 Introduction and Background

Though distributed servers and clusters have existed for a while, there is a lack of sharing, most distributing acts rely on a single task giver, and the peers being enslaved to the server, with little / no reciprocation. As the quantity of data and complexity of analysis from individual groups becomes greater, the efficiency current distributed processing units will certainly become less than satisfactory.

Though distributed servers and clusters have existed for a while, there is a lack of sharing, most distributing acts rely on a single taskgiver, and the peers being enslaved to the server, with little / no reciprocation. As the quantity of data and complexity of analysis from individual groups becomes greater, the efficiency current distributed processing units will certainly become less than satisfactory. Some projects such as Boinc have been created to share the resources of many nodes on a certain set of registered tasks, but the individuals controlling those nodes are unable to input any tasks that they themselves want accomplished into this network. Currently nearly all of the distributed analysis systems are designed only for one specific application in order to specialize and do them efficiently. In the larger picture, of all of the data analyses needed to be done in the world, only a select few are being

done efficiently, and the medium for those which are being done with some efficiency are unusable by all of the processing base.

Each particular task has it's own unique characteristics which make general optimizations nearly impossible. In order to enable a more optimal system the best idea would be to allow the ones who understand the applications flowing through a distributed system to decide how to distribute tasks themselves. Combined with the ideal of removing a centralized source for distribution, each task's means for solution gains a nearly organic existance, splitting into a network of nodes through means which are nearly unpredictable, and yet still able to accomplish the task itself and in an optimal time period as well.

2 Development

Part of the point of this project was to give me a means to learn more about threads and networking in Java while also working on a topic that interested me. The first iterations involved learning the nature of threads of threads and networking within java through various "hello world" applications which eventually were expanded into usable methods within the project, while the second large iteration involved making threads which are spanned in an organization similar to what I will probably use in the final iteration for this project.

The next few iterations involved it is able to recognize tasks being handed to it and will then start doing each task one at a time. Each instance of the application would initialize each task-doing client upon necessity and could initialize multiple clients to do the task as necessary, decided by a heuristic located within the file the task is contained in. The distribution process was orriginally designed for only one level of distribution, but later on it was decided that having a each node redistribute until the heurustic fell below a certain level would be a more efficient process.

In the final iteration, both previous portions were combined to create an application which can recieve the input of a task to be done, check on how it is to be distributed based off of how the task itself describes, distribute itself over the group of peers available, and wait for each of the peers it distributes as the many times as the task's distribution heuristic decides appropriate. Eventually all of hte nodes in the tree that the distribution process creates eventually to return data to their parent nodes, which further.

3 Results and Discussion

The application is for the most part completed. There are is a lack of documentation for the development of task applications, but there are examples from which users can use as examples for development and the end product is usable. The core of the distribution system's architecture was changed around a few times, but in the end it was decided to allow for peers to distribute between all available peers at their liesure. That means that a peer can be sent it's own task several times, if it is not busy.

There are large requirements for users of the application to know what they want to do and be able to use their own means possible. That is where this sort of highly free decentralized application comes into competition with the other centralized applications. Those applications either are completely optimized for their own application, and therefore become immobile to other people's possible usages or use generic means for processing tasks, and therefore waste much more time on overhead necessary. This decentralized method allows for users to have great optimizations, but at the cost of the requirement that the users of this application write out how to implement the distribution process, which isn't the easiest thing to do. The easiest cure to this ailment, which was implemented in this project, was to have generic default distribution process.

References

[1] M. Wang, T. Madhyastha, N.H. Chan, S. Papadimitriou, C. Faloutsos, "Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic"

[2] Khalil Amiri, David Petrou, Gred Ganger, and Garth Gibson, "Easing the Management of Data-parallel Systems via Adaptation", *Procedings of 9th ACM SIGOPS European Workshops & CMU-CS-99-140*, September 2000.

[3] Khalil Amiri, David Petrou, Gred Ganger, and Garth Gibson, "Dynamic Function Placement for Data-Intensive Cluster Computing", *Supercedes Carnegie Mellon University School of Computer Technical Report & CMU-CS-99-140*, June 2000.

[4] Michael Mesnier, Eno Thereska, Daniel Ellard, Gregory R. Ganger, Margo Seltzer, "File Classification in Self-* Storage Systems", *Supercedes Carnegie Mellon Univerity Parallel Data Lab Technical Report*, January 2004.

[5] Joao Pedro Sousa, David Garlan, "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments", *Procedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, August 2002.