Midterm Report

Distributed Processing System with MapReduce Model on Browsers and Node.js

101-2 Special Topics on Cloud Computing, CSIE, NTU

Tzu-Hsien Gao, Tai-Lun Tseng

Email: {r01944033, r01922094}@csie.ntu.edu.tw

0.1 Preface

This report summarizes the collective results on the project so far. It includes the original project description, survey results on several open source projects and web-related techniques that are useful in the project, roadmaps for implementation, and some possible issues for the project.

While the report is aimed to become a milestone of works done in the midterm, most parts of this report will reappear in the final project report.

0.2 Motivation

MapReduce provides a distributed way for processing large-scale data. In traditional MapReduce implementations, a server node dispatches data and tasks to various client nodes and collect results after mapper and reducer functions are done by clients. Both server and client need efforts to configure connection and execution environment for MapReduce programs.

In order to get many clients in a more convenient way, we introduce a new kind of MapReduce client, i.e. browsers. In the initial scheme, researchers who need more computing resources do not necessary set up lots of clients; instead, they can use a HTTP server that establishes connection with clients visit the researchers website (server). Data, mappers and reducers can therefore dispatch to those browsers and execute. To support a research, a resource provider only need to keep a browser opened and connected to the server. And in the future, we can extend the functionality more by enabling other node.js applications

to become clients; thus researchers who are doubt for the speed of browser computing can set up client machines with node.js and libraries we provided.

0.3 Project Description

This library runs a program using MapReduce programming model on the server and dispatch mapper/reducer tasks to many registered client browsers program.

To register, clients visit a site hosted by the server and accept worker permissions. Javascripts of the site will create several HTML5 workers. They will be mappers or reducers (a browser client may has mapper and reducer workers concurrently), waiting for dispatching tasks and data from the server. Upon receiving works, the workers become active, execute the map/reduce works, return results back after finsihing execution repeatly.

Different from traditional MapReduce model, we cant communicate between browsers since the standard of communication between browsers is not popular in this time, so the the intermediate result done by map workers will be report to theserver, and then shuffle these intermediate results to the reduce workers.

Also there will be an API for node.js that provides adapters connecting server-side javascripts and node.js computing environment. Eventually the server can also dispatch tasks to node.js servers, i.e. treating node.js as a powerful MapReduce computing nodes.

0.4 Surveys on Related and Supported Works

Surveys

0.5 Architecture

Architecture

0.6 Issues

Possible issues

0.7 Reference

Reference