CS-647 Distributes Software Systems

Project Pre-proposal

Omar Badran, Jordan Osecki, and Bill Shaya

Our CS647 group would like to explore the MapReduce distributed software system for our term project. We are proposing the development of a Java application that will simulate a MapReduce system that will count the number of words in a file. Upon running the application, our software framework will read a configuration file and will spawn a pre-configured number of worker nodes to simulate a distributed computational environment. The configuration file will also contain settings that the simulator will use to simulate various scenarios such as faults, worker performance, etc.

Our group plans to incorporate self adaptation through self healing and self optimization. Self healing will be accomplished by monitoring the worker nodes. If a worker node fails due to loss of connectivity to the network, or some other fatal condition, the failed node’s computation will be redistributed to a healthy node. Therefore the overall computation can seamlessly complete despite the single failure. Our application framework will include a module to induce random failures throughout the simulated network in order to exercise self healing. Self optimization will be accomplished by evaluation of the performance of an individual worker node. Our application framework will also include a module to induce performance changes in a worker node. As computations are executed, performance will be evaluated, and if necessary, reallocation of computations will be performed in order to optimize computational speed. In order to evaluate the effects of self adaptation, timed metrics will be recorded and analyzed.

There are several notable MapReduce systems that exist such as Skynet and Hadoop. Skynet is an open source Ruby implementation of Google’s MapReduce framework, which is adaptive, fault tolerant, and has only worker nodes which can act as a master at any given time. Hadoop is a Java framework to implement MapReduce functionality, which is currently used in Yahoo web searches. We believe that these systems do not consider jobs already being processed when a node fails and the work is redistributed to other nodes. This may lead to jobs being completed out of order because a reducer may be waiting for a particular mapping operation to complete because of a failed node. The reducer may go ahead and start reducing for another job. From an efficiency point of view, when a node is slow, backup tasks are spawned and it seems that there would be a race to see which node finishes first. We believe it may be better to assign more tasks to the faster nodes and only use the slower nodes for smaller jobs.

We feel that our project has adequate scope for a team of three. Work breakdown components will include the master functionality, worker functionality, self adaptation incorporation, fault detection and handling, performing experiments/trials with the simulation, and documenting our progress and conclusions. Each component can be completed independently by a group member, and we do not anticipate any issues with completing the project by the end of the class term.

TODO: Address why this is a good idea, citing from the rubric the properties of novelty, relevance, and significance

TODO: Cite sources here?

TODO: Start to split into the sections of the proposal? Perhaps we should ask this. What we have is great already, but it is very informal and more a stream of consciousness than an organized proposal. Depending on what Peppo wants, we can stay for now or start to semi-convert this to proposal format (mostly just using the Headings to organize and then see what we need to expand on, not two columns, etc.) What do you think? I vote for starting to convert it and can definitely do this tomorrow night.