Comp 4107: Project Proposal

Jordan Amos (100769796)
Ashley Moni (100954492)

Problem Statement

In the current incarnation of the web, a typical application will comprise of multiple processes. There's usually several services and APIs to make a modern web application work. There is also, usually multiple instances of the same process behind a *load balancer* in order to handle large quantities of simultaneous requests. This is more crucial in single threaded processes like the popular nodejs. The problem becomes how many instances of each process should be run, and how do we decide which process gets the request.

The most popular approach is a simple round-robin approach, that evenly distributes the requests among each process, based on who's next in line. Since the individual processes have the ability to cache information and specialize if repeating a task, there is potential for improvement over this simple algorithm.

Solution Approach

The proposed solution is to use a task allocation algorithm to perform the routing of individual requests to hopefully improve the response time over a simple round-robin approach. Since the problem is more evident in NodeJS, and there is a large open-source community, we propose to solve the problem using that toolset. Using the node-http-proxy open source project on Github should allow us to focus on the routing algorithm, and not get hung up on the implementation details of the underlying web stack, while making a fully functional load balancer. More established load-balancers like HAProxy, PM2 and Nginx are far too overbuilt, for academic purposes.

While the initial solution is for process management on a single computer, the solution could be extended to VM and process management for systems that use services like AWS, Google Compute or Heroku. This algorithm could possibly reduce the costs for services that charge for compute time by maximizing efficiency of individual processes and VMs.

Analysis Method

There are several tools used to benchmark HTTP performance. The one we would use is called wrk. We are trying to run at optimal efficiency for each process. We consider this to be the maximum cpu usage, before response time degrades. We will test our solution against a round-robin solution, and two solutions that don't use node-http-proxy. The built-in node clusters module and the popular internet heavyweight, nginx. We should be able to tell if our solution is better than a simple round robin approach, and if our solution is able to stand up to the overbuilt heavy weights. The benefit of testing the other solutions along with a custom round robin using our setup is we will be able to see the baseline difference between node-http-proxy, nginx, and node clusters.

References

http://goldfirestudios.com/blog/136/Horizontally-Scaling-Node.js-and-WebSockets-with-Redis
http://blog.keithcirkel.co.uk/load-balancing-node-js/
https://github.com/nodejitsu/node-http-proxy
https://github.com/wg/wrk