CS630 Project proposal

Ali Hassani

April 6, 2023

I intend to complete an infrastructure project primarily on creating minimal a job scheduling / workload management service for compute clusters, similar to SLURM [1]. The goal is to create an open source solution which provides a minimal simulation of modern compute clusters using such services for training purposes, and is able to run on minimal hardware without real compute servers required. This solution will use Docker containers to simulate compute nodes and their network infrastructure, and the minimal workload manager will provide an interface for monitoring and scheduling jobs.

1 Problem: Workload management

Compute clusters are often comprised of a set of networked computers, each with certain computational capabilities and resources, and they are set up in such a way that multiple users can request said resources for specific experiments or jobs. This is a distributed environment, where compute nodes collaborate to create a single unified system through which researchers, engineers, and scientists conduct computationally heavy experiments based on their requirements, and without having to consider per-node physical hardware requirements. An example of that is in machine learning research, where experiments, depending on their nature and size, require different levels of resources, particularly hardware accelerators, such as general-purpose graphical processing units (GPGPUs) and tensor processing units (TPUs). A standard practice in such scenarios is a cluster management software that is installed on all of the networked computers, or *compute nodes*, and one or more computers manage, organize, and control the rest via network communications made through the cluster manager, which are referred to as *head nodes*. Cluster management software relies on a workload management service or module, which provides an interface for users in order to allow them to request resources, and schedule their jobs accordingly.

2 State of the art solution

SLURM (Simple Linux Utility for Resource Management) [1] is one such service, and is among the most widely adopted among large-scale supercomputers and compute clusters. While the cluster management software keeps track of all nodes, their health status, and so on, it also passes their IP addresses to the workload manager. The workload manager, in this case SLURM, runs daemons, which are processes that start automatically after the operating system boots and continue to run in background, on head and compute nodes. On compute nodes, the daemon is responsible for reporting node resource availability, running jobs, and other relevant information to the head node. On the head node, controller daemons communicate with communicate with compute nodes, which are exposed via their local IP address.

Users then log into head nodes, and use the SLURM interface (commands such as ‘scontrol‘, ‘sinfo‘, ‘srun‘, ‘sbatch‘) to check available nodes and resources, their running jobs, and submit new jobs. The interface is responsible for communicating with the daemon to schedule jobs and report information.

3 Approach and environment

As mentioned, the purpose of this project is to create a minimal workload management and job scheduling service similar to SLURM. This can be broken into creating three key services:

1. **Control daemon:** will run on head node(s), monitor compute node(s), manage and schedule jobs submitted to it via the interface, and assign jobs to compute node(s).
2. **Client daemon:** will run on compute node(s), report resource usage statistics to control daemon, and run jobs assigned to it by control daemon.
3. **Interface:** will be exposed to users on head node, allowing them to monitor resource usage, job statistics, and submit jobs.

Deployment and testing can be done by simulating a compute cluster using Docker. Multiple containers, each with a specific amount of resources (memory, CPU cores, and possible GPUs), will simulate compute nodes, and a single container with just enough resources to handle the control daemon and interface can serve as the head node. Docker's network infrastructure will also assist in setting up communication, and a shared file system.

4 Timeline

The tentative timeline for completing this project is as follows:

1. Setting up environment / docker images: week 1
2. Creating the control daemon: week 3
3. Creating the client daemon: week 4
4. Creating interface: week 5
5. Writing job scheduler: week 6

References

[1] Andy B Yoo, Morris A Jette, and Mark Grondona. “Slurm: Simple linux utility for resource management”. In: *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9*. Springer. 2003, pp. 44–60.