CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: Near real-time portal for HPC cluster status	
Sponsor Information:	**Christopher Coffey,** HPC Research System Administrator HPC (High Performance Computing) Information Technology Services, NAU chris.coffey@nau.edu 928-523-1167

Project Overview:

Researchers are always looking for easier ways to exploit HPC (high performance computing) cluster resources, and who can blame them. Traditionally, use of HPC resources requires interacting with a cluster via a console. For many users, typing and interpreting results in this fashion can be a source of frustration…and can even prevent many non-technical researchers from taking advantage of these resources.

With this in mind, our team in the HPC group of NAU Information Technology Services are gradually adding new easy-to-use tools and interfaces that allow researchers to utilize HPC resources. One tool that would be very helpful is a web app that can visualize and represent the cluster queue and utilization state in a way that is easy to view and understand, whether it's via the researcher's desktop browser, or their phone.

Specifically, the aim of this project is to create an open-source web portal tool that increases clarity of HPC queue status, and resource utilization, which in the end will be a step in the direction of making life easier for the end-user researcher. This project involves creating a secure web portal where HPC job statuses, and cluster utilization are shown in near real-time for *n* number of clusters.

- Should allow a variety of inspection and visualizations centered around showing the status of a particular cluster. Some specific functionalities will include:
    - Show # of jobs in running state, and pending state, cluster utilization, as well as current average wait time
    - Allow drill down into multiple views of job state in category: pending, and running while:
        - Listing all jobs, along with various information such as: jobid, jobname, user, account, requested cores, requested nodes, requested memory, submitted time, queued time, expected start time
    - Allow drill down into fields: user, account, and nodes, which results in list of jobs from that selection
    - Allow drill down to Individual job (we may end up needing to require authentication)
- Portal should be flexible allowing use of different back-end scheduling software: Slurm (http://slurm.schedmd.com/) at minimum, and Torque, LSF, and others.
- Portal should either have the ability, or be easily changed at a later date to connect high level display of utilization to ganglia (http://ganglia.info/) through drill down.

- Portal should be compatible with major browsers, including mobile. Ideally, design will allow for automated adjustment of GUI to accommodate mobile viewing.
- Other useful tools and features, as may emerge during the initial design and requirements phase of the project.
- Software should have friendly open source license such as GPL2, and be specifically designed and delivered for easy future development.
- As much as possible, project is modular, and can be easily adapted, customized

Knowledge, skills, and expertise required for this project:

The project generally draws on typical skills learned in the CS curriculum, with additional emphasis on:

- Linux operating system expertise
- Knowledge webserver (apache) function and deployment
- Database expertise, insofar as status and other data must be archived for use in the interface or for future data analysis needs.
- Web programming/design, with particular emphasis on modern Web2.0 portal implementation.
- Effective communication. Team must be able to learn HPC concepts and communicate design ideas clearly.

Equipment Requirements:

No special equipment should be required beyond a standard development platform (your laptop), as well as freely available environments and software tools.

Software and other Deliverables:

1. Complete secure Web2.0 portal implementing the a professional graphical interface to the functionalities outlined above.
2. Extensive end-user testing with non-technical scientist users is expected, with a strong beta-level deliverable expected.
3. Professionally documented codebase, delivered both on USB stick and via access to an online repository (e.g. Github).
4. A strong as-built document that details the design and implementation of the site. This must be robust enough to allow a future development team to easily pick up where you left off.
5. A straightforward user manual aimed a non-scientist end-users, introducing the main features and functions of the tool.