Galaxy as an Extensible Job Execution Platform

John Chilton and the Galaxy Team
jmchilton@bx.psu.edu

Project: http://galaxyproject.org/ Code: https://bitbucket.org/galaxy/galaxy-central

Galaxy is a popular, open source platform enabling data intensive biomedical research with a vibrant development community. The core Galaxy project is maintained by many developers and researchers and includes diverse components for visual analytics, data management, and app store-like functionality… the core Galaxy concepts of tools and jobs underpin much this functionality. A Galaxy tool typically corresponds to a command-line driven application to perform some analysis and describes both the user interface presented to Galaxy users as well as how to transform the users inputs into a command-line for execution. Galaxy execution of this command-line is called a job. Despite the simplicity of this concept, Galaxy jobs running architecture is constantly growing richer and more extensible.

Galaxy deployers can configure any number of static job destinations representing various clusters or servers and submission parameters for jobs - this will be demonstrated as well as easy plugins (simple Python functions) termed dynamic job destinations that can be used to route jobs to these destinations, modify existing destinations for an existing job, or create entirely new destinations all based on the user submitting the job, the job inputs, or runtime conditions.

These destinations can be the local server hosting Galaxy, a variety of cluster resource managers, or remote servers running a highly configurable, light-weight application (which itself can target the local machine or resource managers). Communication with these remote job execution components can be securely configured over HTTP(S) or via message queue, and provides a myriad of easy to configure options for job staging.

Continuing this theme, Galaxy tool developers can describe abstract packages that a tools depends on. This talk will cover building plugin for resolving these including a discussion of the existing plugins for Galaxy's custom environment files, Galaxy tool shed packages, and standard Unix environment Modules.

Finally, Galaxy jobs can be instrumented for metric collection - the talk will discuss developing these plugins the stock ones - including Galaxy's collectl plugin providing detailed resource usage statistics for jobs (CPU, memory, etc…) in a resource manager agnostic fashion.

The extensibility and ease of configuration described above may answer the question of why application developer should want to build on Galaxy as a platform. This talk will discuss the how as well - including embedding applications into Galaxy via tools, building applications on top of Galaxy via the API, and simply running jobs like Galaxy using the remote job execution component to leverage these plugins and extensibility without needing to run Galaxy itself.