Raven Quarterly Report

December, 2008

John H. Hartman
Scott Baker

After much negotiation the contract was finally signed at the end of October. For the roughly two months since then we have made significant progress towards our upcoming milestones. We have been working very closely with the PlanetLab and Gush projects as part of this effort.

Port Stork to GENI Environment

Much of our effort on this milestone has been working closely with the PlanetLab project to define the underlying GENI interfaces to which Stork must be ported. Scott in particular has done a lot of work on this and developed the 'geniwrapper' software that provides GENI interfaces on PlanetLab. Stork currently uses the native PlanetLab interfaces and we are now beginning to port it to the GENI interfaces, but this effort should be significantly simplified by our participation in the geniwrapper project.

One modification to Stork we have already made is development of a Stork 'nest' proxy slice. This slice runs on all GENI components and allows the Stork software that is embedded in client slices to download packages and metadata efficiently. The proxy caches these files, so that each file is only downloaded once to each GENI component independent of how many slices on that component use the file. The nest proxy is very similar to an HTTP proxy, except that it allows files to be identified by the SHA-1 hash of their contents, rather than URL. This avoids naming conflicts and fits better with Stork's security model than does URLs. We modified the embedded Stork software so that a client slice first attempts to access files via the proxy, and if that fails, from the Stork repository directly.

Second, we have begun integration of Stork with Gush. This work is nearing completion and should be deployed shortly. Specifically, we have integrated Gush into Stork in two ways. First, we have added Gush support to the Stork developer tools, so that a Stork user can develop a package on his/her workstation and use Gush to push the package and relevant metadata directly to his/her slices. Second, we have added Gush support to the Stork repository so that the repository periodically uses Gush to push repository metadata to the Stork nest slivers. This allows the nest slice to provide up-to-date repository metadata to its clients without resorting to expensive polling of the repository.

Support for GENI Slice Management

We have been working with the PlanetLab project to define the proper interfaces for resource management (including slice management) for the geniwrapper. This work is in its early stages, but again, our participation both helps us understand what those interfaces are likely to look like once complete, and help us ensure that the interfaces provide the functionality Raven will need. Stork currently contains functionality for performing package management operations on groups of PlanetLab slices and nodes. Our next task is to change this to support groups of GENI slices and components, and to interface with the geniwrapper to instantiate slices on the proper components as specified by the group information. Raven may do via the geniwrapper interfaces directly, or may make use of the the slice manager tool the PlanetLab project is currently developing, depending on which best meets our needs.

We are also planning on integrating with Gush's “experiment” support and have an initial proof-of-concept that we can generate the XML files that Gush uses to define an experiment from Stork's group information. This, for example, allows a Raven user to use Gush to push out packages and metadata to the proper GENI components based on the Stork group information.

Release v1.0

The above milestones form the core of Raven v1.0. In support of the release we are in the process of setting up a Raven Trac site (raven.cs.arizona.edu) that will contain all of the Raven documentation, source code, tickets, internal milestones, published papers, etc. The site is currently under construction but we anticipate moving the Stork code base to the site within the next month.

As an organizational note, we have decided to split the original Stork project into a suite of tools in the Raven project. The Stork project contained several tools for deploying packages and managing slices, one of which is named 'stork'. This causes some amount of confusion, especially because those of us in the project alternately use the term 'Stork' to refer to the overall project and to the particular tool. Going forward we will refer to the package management tool as 'stork', and the package distribution tool as 'tempest', while the overall project is called 'Raven'. This will likely cause some initial confusion, but should be clearer in the long run than the current scheme in which there is a 'stork' tool that is part of a larger 'Stork' project.