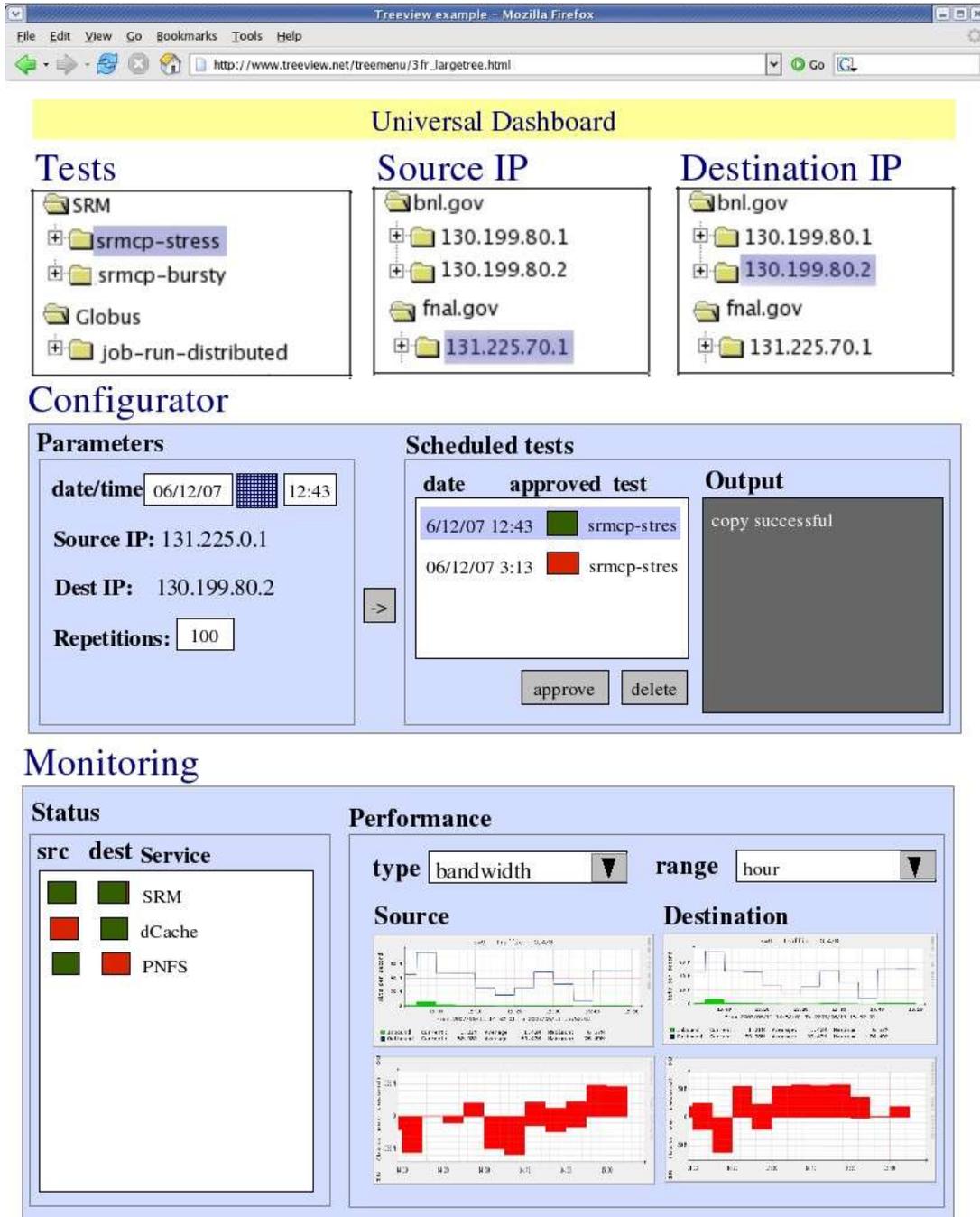


Atlas Tier 1/2 Load Testing and Monitoring

Jay Packard – June 11 2007

Shown below is a mockup of a web application named “Universal Dashboard” for supporting Atlas Tier 1-2 Load Testing/Monitoring by centralizing status monitoring (e.g. Nagios), performance monitoring (e.g. Ganglia, Cacti), and command execution:



A user would first select a test, a source IP (or group of IPs such as a site or a subnet), and a destination IP (or group of IPs). For this combination of test, source IP, and destination IP, a template of parameters for this test would be shown, some of which are determined by the IP and others that are determined by the user such as date/time. When the parameters are chosen, she presses the -> button to have it scheduled. If this test involves a remote site, the administrator at that site will see the scheduled tests and decide whether to approve it or not. Also, for this combination of test, source IP, and destination IP, monitoring information would be shown including service status and graphs of the specified type (bandwidth, errors, files transferred, etc.) and range (hour, day, week, month, etc.). Besides the endpoint graphs, graphs at the various levels of the middle-ware stack would also be shown.

The services and IPs show a colored icon that indicate the status of the service or host (e.g. red for down or green for up), which behind the scenes would involve parsing Nagios logs or calling web services. The parent folder's status will summarize the status of it's children. The scheduled commands also show a colored icon that indicate whether the remote administrator approves the test. An administrator would see and decide whether to approve tests created by administrators from other sites if the test involves his site. Implementing this coordination and storing/scheduling the tests would be a large part of the development of this application.

This application would be configured using xml that looks something like:

```
<dashboard>
  <tests>
    <test name="srmcp" services="SRM, dCache, PNFS" >
      <test name="srmcp-stress">
        <parameters>
          <parameter name="source IP" type="srcip"/>
          <parameter name="dest IP" type="destip"/>
          <parameter name="repetitions" type="manual"/>
        </parameters>
      </test>
      <test name="srmcp-bursty">
        ...
      </test>
    </test>
    <test name="Globus" ...>
      ...
    </test>
  </tests>
```

```

<services>
  <service name="SRM"/>
  <service name="dCache"/>
  <service name="PNFS"/>
</services>

<ips>
  <ip name="bnl.gov">
    <graphs>
      <graph type="num files" service="SRM"
        uri="http://graphs.bnl.gov/graph?s=@START@"/>
      <graph type="bandwith" service="PNFS"
        uri="http://graphs.fnal.gov/graph2?hour"/>
    </graphs>
    <statusLogs>
      <statusLog type="nagios" service="SRM" uri="srmstatus.txt"/>
    </statusLogs>
  <ip name="fnal.gov">
    <graphs>
      ...
    </graphs>
  </ip>
  <ip name="130.199.80.2" services="PNFS"/>
</ip>
  <ip name="131.225.*" alias="fnal.gov" services="SRM, dCache, PNFS">
    <ip name="131.225.70.1" services="PNFS"/>
  </ip>
</ips>

</dashboard>

```

Basically in this configuration, tests along with the parameters required to run the test, services, and IPs along with the graphs and status logs that monitor this IP are all defined. The dashboard engine would figure out for the selected test and IP which graphs and status logs are appropriate to display. Another feature of this configuration would be that it would allow for arbitrary hierachy for tests, services, and IPs.

Details and have been left out in this proposal since this is a first iteration. If it seems to fit our needs, it can be extended. It may not meet our needs if centralization is

not too important or if it is too much effort to be worth it. The alternative is to bookmark the various monitoring and status URLs (e.g. http://www.opensciencegrid.org/Grid_Monitoring or <https://www.racf.bnl.gov/Facility/Monitor/dashboard.html>), have a command prompt available, and a method of communication with other administrators, and then to switch between these. It may be that the focus should not be so much on centralization, but on creating the key lower level testing scripts that are currently lacking.

To summarize, the Universal Dashboard is not extremely sophisticated by itself, but simply brings brings together scripts, status monitoring, and performance monitoring to hopefully make testing and monitoring a smoother experience. Because it is so general, it can accommodate a few services and hosts to begin with, and then more can be added over time. Also because it is general, multiple people can simultaneously work on various testing scripts to be used by this application without worrying about the whole.