ACI-1261715: Long Term Access to Large Scientific Data Sets: The SkyServer and Beyond

PI: Alexander S. Szalay, JHU

1. Reacting to Changing Technologies

iPython Scripting

It is clear that Data Intensive Science is different today than it was 3 years ago. Our users increasingly embraced Python as their main scripting platform. With the emergence of Jupyter/iPython we felt it was necessary to change our development plans and use it as the main scripting platform for the SciServer.

Data Containers

With the emergence of Docker containers it became much easier to package tools and large applications. With our scripting applications under way our users wanted to link the database queries to massive amounts of raw data. As a solution we came up with large data containers, like the one adopted for SDSS, providing access to over 150TB of raw images and spectra. We are now adopting this for other areas.

High-Speed Data Movement

Over the last year we moved several petabytes of data to the SciServer, from various domains. While we have a 100G connection to the outside world, we had to spend quite a lot of time to fine-tune our DTN settings, aggregate multiple Globus nodes, and work with our remote data sources to configure their gateways to enable fast transfers. While in the end the speed of the network transfers was quite fast, it took a lot of case-by-case tuning and configuration, consuming a disproportionate amount of time.

2. Scalability and a Robust, Secure Environment

Consolidating our Security Framework

Our different system components each had historically a different security solution. In order to integrate these into a homogeneous platform, we had to switch to a single key sign-on system based on Keystone. This work was very tedious, but necessary, and by now we have accomplished this task successfully.

Maintaining a Robust Monitoring Environment

We have about 7,000 disk drives spinning in the SciServer system, serving a wide set of large scientific projects, with on-line databases and data services. Our previous monitoring system that checked 'heartbeats' of the different system calls did not provide enough details. Detection and replacement of failing disk drives often happened after the fact. It became clear that we had to implement a two-tier strategy: on one hand we have to monitor every disk drive's SMART counters on a one-minute granularity, and integrate these into a sophisticated alert service. We have used scollector/bosun as the underlying framework and are now collecting about 50 different counters of every drive every 30 seconds. The data collected is stored in a database, and hopefully we will be able to start correlating these logs with actual failures.

Maintaining a Robust Operating Environment

As we provide a lot of 24/7 database services, over many different data sets, spanning more than a petabyte of data, frequent disk failures are inevitable. We had to replace our previous, somewhat ad-hoc procedures with a much more formal environment, where most databases have at least two 'hot' copies, heavily partitioned into typically 100-400GB shards, so any missing partition can be quickly replaced. We also have multiple backups on an optimized 2.8PB backup server, that can deliver data for recovery saturating a 40G link. These procedures have dramatically increased the robustness of our database services and decreased the human manpower needed to recover from the failures. We are not yet in a fully automated recovery mode yet, but we can start to see the path leading there.