CPT Report

Sathyanarayanan Gunasekar {sathya@cs.wisc.edu}

August 8, 2011

1 Freight

This summer I worked on building Freight for facebook. Freight is a service for transferring large files across many machines on the facebook network. Several services at facebook have very large data sets which are generated in one location and must be syncronized efficently to hundreds of other nodes that utilize that data to serve production traffic. Freight was designed to faciliate these transfers. Users can submit jobs specifying the list of sources, destinations and options specific to the transfer such as pre or post processing hooks, limits specifying how fast the transfers can happen.

In the process of transferring these data sets Freight will ensure that the network switches particpating in the transfers do not become over loaded or under utilized. Freight will graph the network topology for all active transfers, find the optimal transfer rates and send commands as appropriate to run transfers fast/slow. Depending on the current load on the relevant network switches, Freight will throttle individual transfers up or down. To do this, Freight will start agents on each node involved in the transfer and use that agent to execute control commands for that host. This design allows for a fully distributed control structure to moderate transfers. The central server has a full view of all active transfers and will choose the appropiate throttle commands to optimize transfers across all Freight jobs.

Freight also includes an abstraction layer called PubSub. This layer is a mix of a message queue and a virtual filesystem. Freight clients can choose to subscribe to a Freight cargo and will be notified when a newer version of that cargo becomes available. Newer versions of the cargo are supplied by publishers. A client can act as both a publisher and subscriber. Once a subscriber gets notified of a newer version of cargo, he can pass in a binary answer indicating whether he wants the new version or not. On a positive reply, freight ships the newer version in an optimized way to the subscribers.

The main server is written in python and it forks off several helper processes which handle tasks such as launching remote agents called trains, collecting data from the running transfers, fetching current switch usage for active transfers and determining, using some heuristics, which throtting descisions to make to optimize transfer rates. All trains report back to the central server which in turn communicates to the trains through thrift calls. Thrift is an RPC framework written in house at facebook and open sourced.

We employed a number of transfer mechanisms depending on whether it is a one-many, many-many or one-one transfer. For the former we choose bit torrent and the later will fall back to rsync. Freight also can stream data from hadoop.

The two main requirements of the project were speed and stability. Making Freight fast was an interesting and unexpected challenge as the open source transfer mechanism we choose for one to many and many to many transfers did not scale well to the speeds the facebook network is capable of. Tuning bittorrent became a large and highly leveraged part of the project. Freight also had to be resilient as the transfers Freight was managing could take many hours or days.