

Replication and Load Balancing Strategy of STAR's RDBM



"So divinely is the world organized that every one of us, in our place and time, is in balance with everything else."

--Johann Wolfgang von Goethe

Michael DePhillips,
Mikhail Kopytine*, Jerome Lauret

*Kent State University

Problem Statement

- Our original plan laid out a simple (naive), but effective, configuration...
 - Policies were minimal (almost ad-hoc)
 - No "real" enforcement
- As usage grew, availability and QOS was being jeopardized
 - Improvements on an as-needed basis
 - With no definitive path forward
- Some resources were underused

Project Motivation

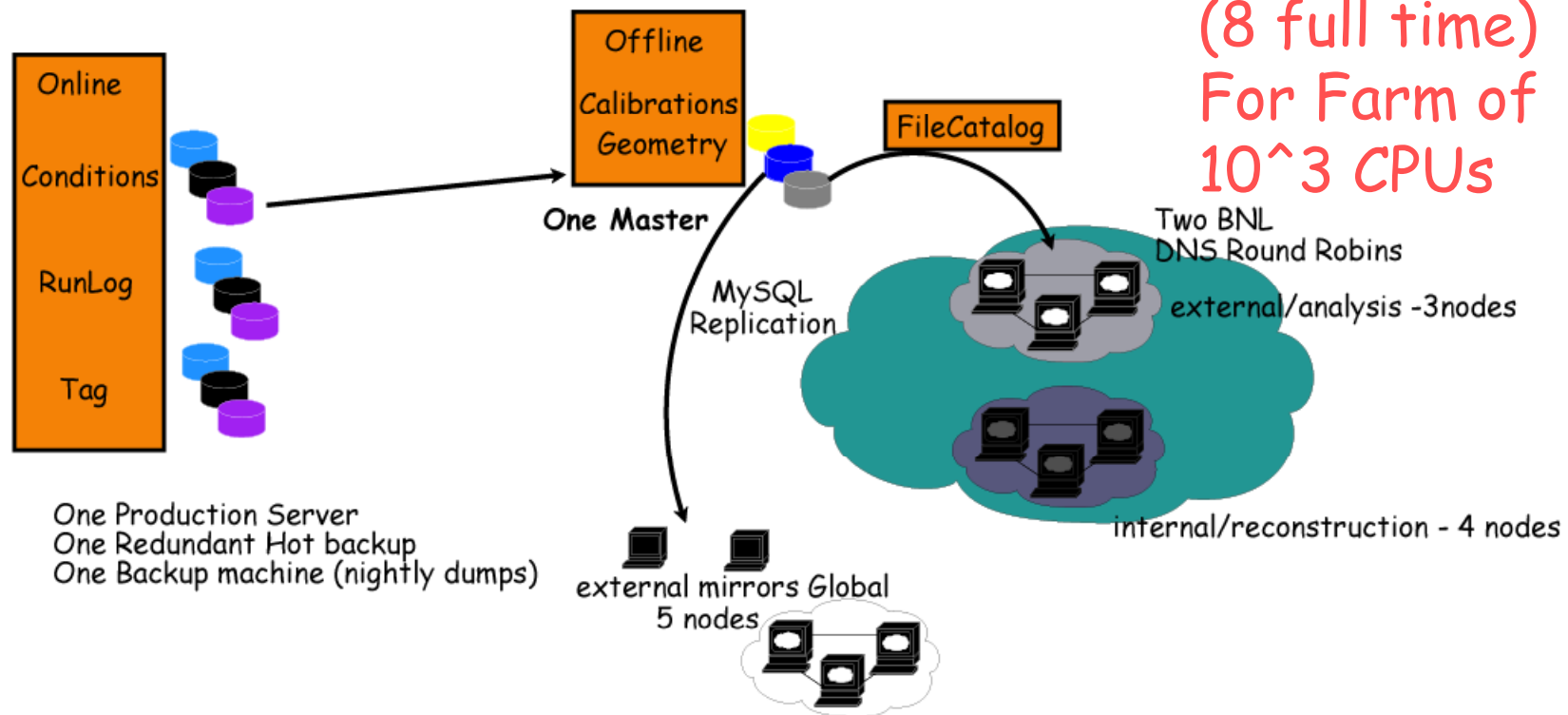
- To optimize the usage and maximize the throughput of STAR's database servers.
- Establish connection policies with a flexible configuration
 - Configurations need to be altered quickly (minutes)
- Be able leverage a global distribution of dbs
 - Improve db services to areas without database resources
 - Include, into the load balancing, sites/areas that do have databases
- Leverage different grades of hardware

Original Server Configuration

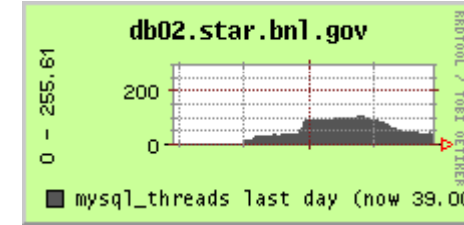
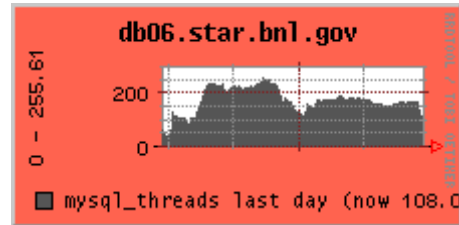
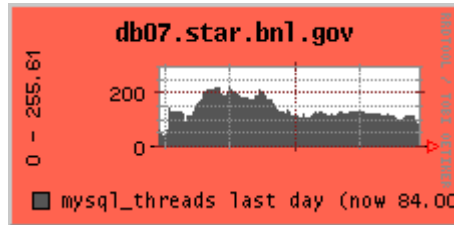
- MySQL Master/Slave Replication
 - Distribute the load to as many slaves as possible
- DNS Round Robin with two distinct / isolated pools... (analysis/production)
- Service administered via a distributed, small configuration file (XML -format)
- Scaling plan was to add nodes as needed

Server Topography

10 BNL Servers
(8 full time)
For Farm of
 10^3 CPUs






DNS results



Would be useful to switch/swap/mix in DNS land, however, that's at an institution level

- For Stress relief - Low Hanging Fruit -
 - DB Optimization methods
 - Hardware tweaking
 - Query optimization
 - API code optimizationwhen these were exhausted, and the usage increased frustration escalated

Alternatives/Solutions

- Buy some new nodes 
 - Scalable solution?
 - Ignores the lack of balance between the two pools
- Commercial Load Balancer 
 - STAR requirements are dynamic and less predictable
 - global slaves
 - heterogeneous hardware
 - Many different types of tasks
 - Feature limited
- Software 

Design Decisions

- Backward Compatible
- Configurations will use a more sophisticated XML which added libxml2 as a dependency
- Increase flexibility
 - Allow for the heterogeneity of the pools
 - Use XML configuration to define additional groupings and their attributes
 - Two stages - local then global

Programmed Flexibility / Features

- Time of day
 - Day/Night where Night = 11pm to 7am EST
- Day of week
- Weighting factor (machine "power")
- Connection limits
- Pools defined by different criteria
 - Type of usage (e.g., production)
 - Users (e.g., development)
 - Type of Access (e.g., read or write)

Local XML-snip

```
<Server scope="Production" user = "recco" accessMode = "read">  
  <Host name="db2.star.bnl.gov" port="3333"/>  
  <Host name="db3.star.bnl.gov" port="3333"/>  
  <Host name="db4.star.bnl.gov" port="3333"/>  
  <Host name="db5.star.bnl.gov" port="3333"/>
```

```
</Server>
```

```
<Server scope="Analysis" accessMode = "read">  
  <Host name="db6.star.bnl.gov" port="3333"/>  
  <Host name="db7.star.bnl.gov" port="3333"/>  
  <Host name="db8.star.bnl.gov" port="3333"/>
```

```
</Server>
```

```
<Server scope="Analysis" whenActive="night" accessMode = "read">  
  <Host name="db1.star.bnl.gov" port="3333"/>
```

```
</Server>
```

```
<Server scope="Analysis" user="john,paul,george,ringo" accessMode = "read">  
  <Host name="db1.star.bnl.gov" port="3333"/>
```

```
</Server>
```

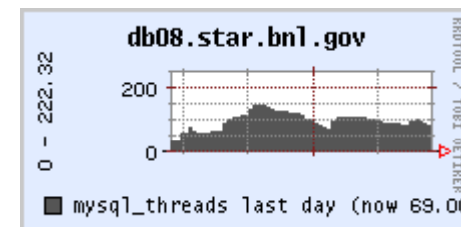
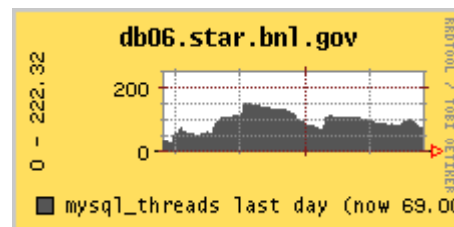
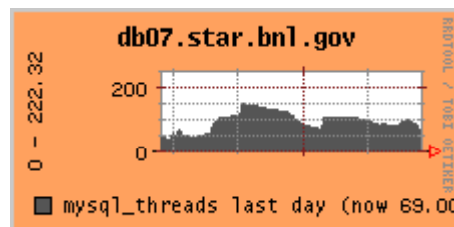
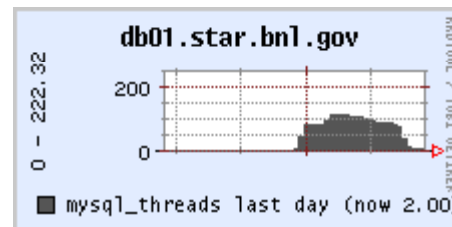
DB_CONFIG_LOCAL
Location is
defined on login env variable

Ad-hoc flexibility

XML+Load Balancer (c++)

- The API then parses makes a connection "show processlist", counts the number of threads, picks the lowest and makes the connection
- Results:

*At Most
0.020 seconds*



Phase 2 – Global db Access

- Two choices for Users with a “personal” copy of the STAR environment
 - Maintain a slave (e.g., MIT, YALE, ...)
 - Prior to the LB - access one of two db farms BNL/PDSF
 - Load Balance between these available nodes and also “entice” these smaller slave-owners to offer-up some db access

Global - Assessing Remote Sites

Second XML file

- GLOBAL_CONFIG (if local is not found and GLOBAL is defined)
 - list of available node/farms are remote

CAVEATS

- Rigid enforcement of policies become very important
 - Institutions may be more inclined offer up a node as a "good neighbor" with specific policy assurances
 - only available "at night"
 - maximum of N connections.
- Firewall - Needs an open port.
- Granularity is greater because it chooses between the open port configuration (i.e., gateway DNS name) - which then uses DNS-round robin

Next Version

- Fine tune Load Balancing decisions
 - Incorporate CPU and I/O
 - Leads to more automated DBA monitoring other API enhancements
 - Closing connections to remove sleeping MySQL threads
- Abstracting LB code away from database application
- Incorporate into our Online DB Nodes
 - Standalone
 - Online API (requests will be more specific)
- Weave it into a Monitoring Application

Summary

- We took a typical early approach to db distribution and created a policy based, centrally managed, load balanced system that can provide and maintain:
 - availability
 - complete usage of all available hardware
 - policies are feature rich
 - and a presents clear path to the future with regard to improvements, abstraction and scaling.
 - All with very low overhead