A simple object storage system for web applications



December 12, 2012

Motivation

Most web content is static and shared

Traditional NAS systems inefficient and costly for content distribution

Every interface to content is unique per application



Background circa 2006

Google file system

Cluster file systems

- Gluster
- Lustre
- IBrix

Scalable NAS

- Isilon
- Onstor

Parallel file systems

- pNFS
- Oceanstore



First attempt – IBrix

Commodity hardware

Scalable metadata

Scalable cluster

Good resilience

Problems

- Hierarchical metadata
- Weak metadata replication
- Client software required
- Client and server version mismatches



Second attempt – Object store

Purpose built

Commodity hardware

Open source software components

- Linux
- Tomcat
- JAVA
- MySQL

Simple external API

Manageability prioritized



Requirements

Shared nothing components

Scalable metadata

Separate metadata and data system components

Asymmetric components allowed

Multi-site capable

RESTful external API

- POST
- GET
- DELETE



Requirements

Multi-tenant

Strong data protection –

- Availability
- Durability
- Background checking and recovery

External security but internal access control

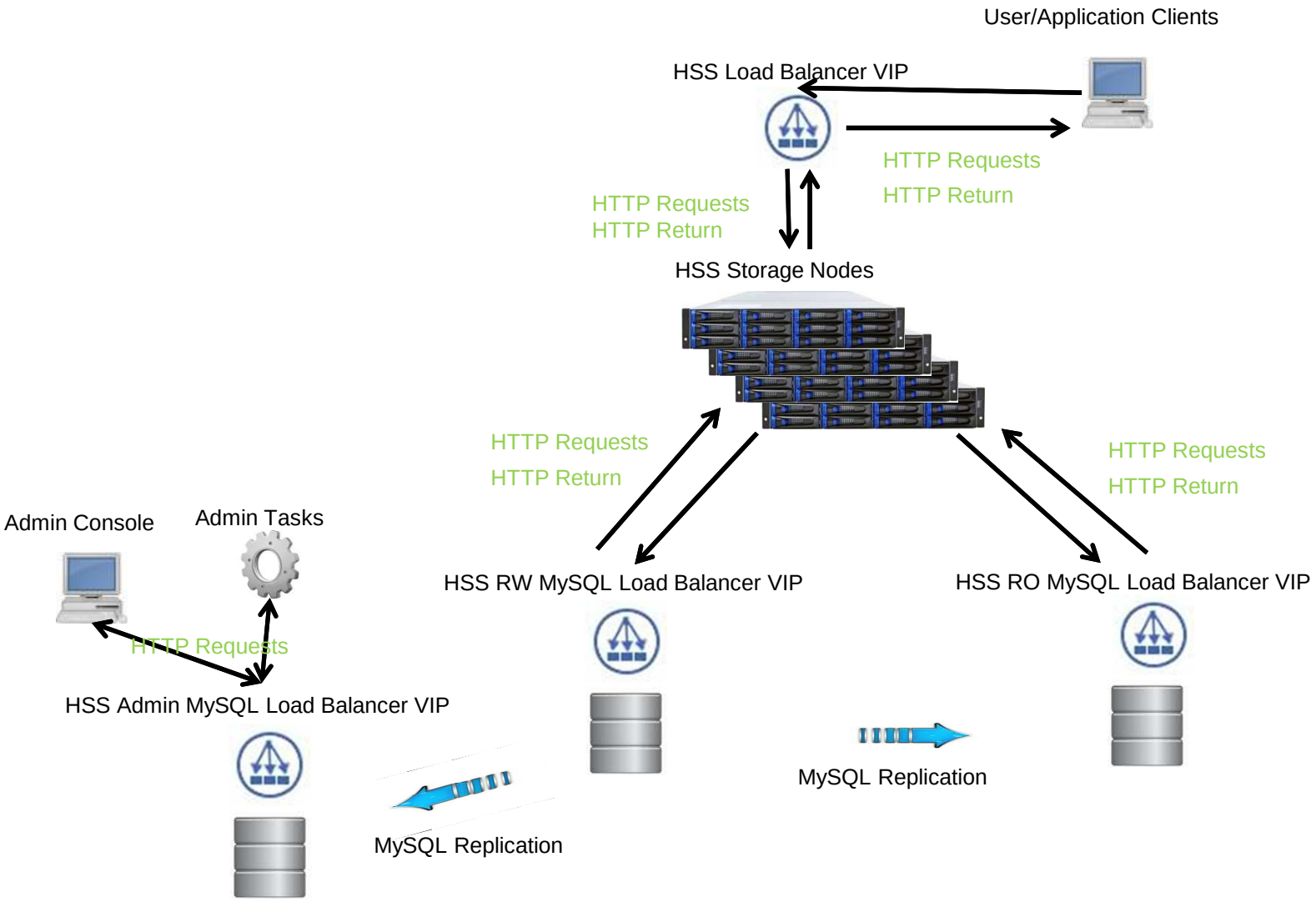
Extended object metadata

Modular

- Performance monitoring – external system
- Hardware monitoring – internal and external together



Implementation



Write example

POST request to VIP from client

Load balancer selects storage server

Calculate OID

Write file locally

Update DB with new OID and server owner

Create second replica copy

Update DB with OID and second server owner

Return OID to client

Set replication flag in DB to create third replica

Read example

GET request to VIP from client

Load balancer selects storage server

Storage server checks local cache for OID

- Cache miss causes OID lookup in DB

DB returns location of all replicas

Storage server retrieves one of the replicas

Storage server returns the file to the requestor

- If the file is above the redirect threshold send 302 redirect

Common failures

DB unavailable for write – 502 server error

Write failure of initial file – 500 server error

Write failure of second replica – retry

File not in DB – 404 not found

File retrieved corrupt or unavailable

- Use different replica
- Schedule replication to proper number of required replicas



Features

Automatic file expiration configurable by application

OID can be specified for application flexibility

Frequently accessed files are cached on all servers

Usage accounting



Some statistics

99.5% of all requests take less than 100ms

99.9% of all requests take less than 500ms

Over 200M requests in a single day

Over 400M objects managed

165TB of objects served per month

40+ applications storing files



Future enhancements

Containers for objects – improve performance and reliability

Better geographic awareness – location affinity and latency improvements

Storage tiers – better resource allocation and performance

Improved modularity – different storage and metadata backends



Demo

Store a file through basic web UI

See where it is stored

Retrieve the copies

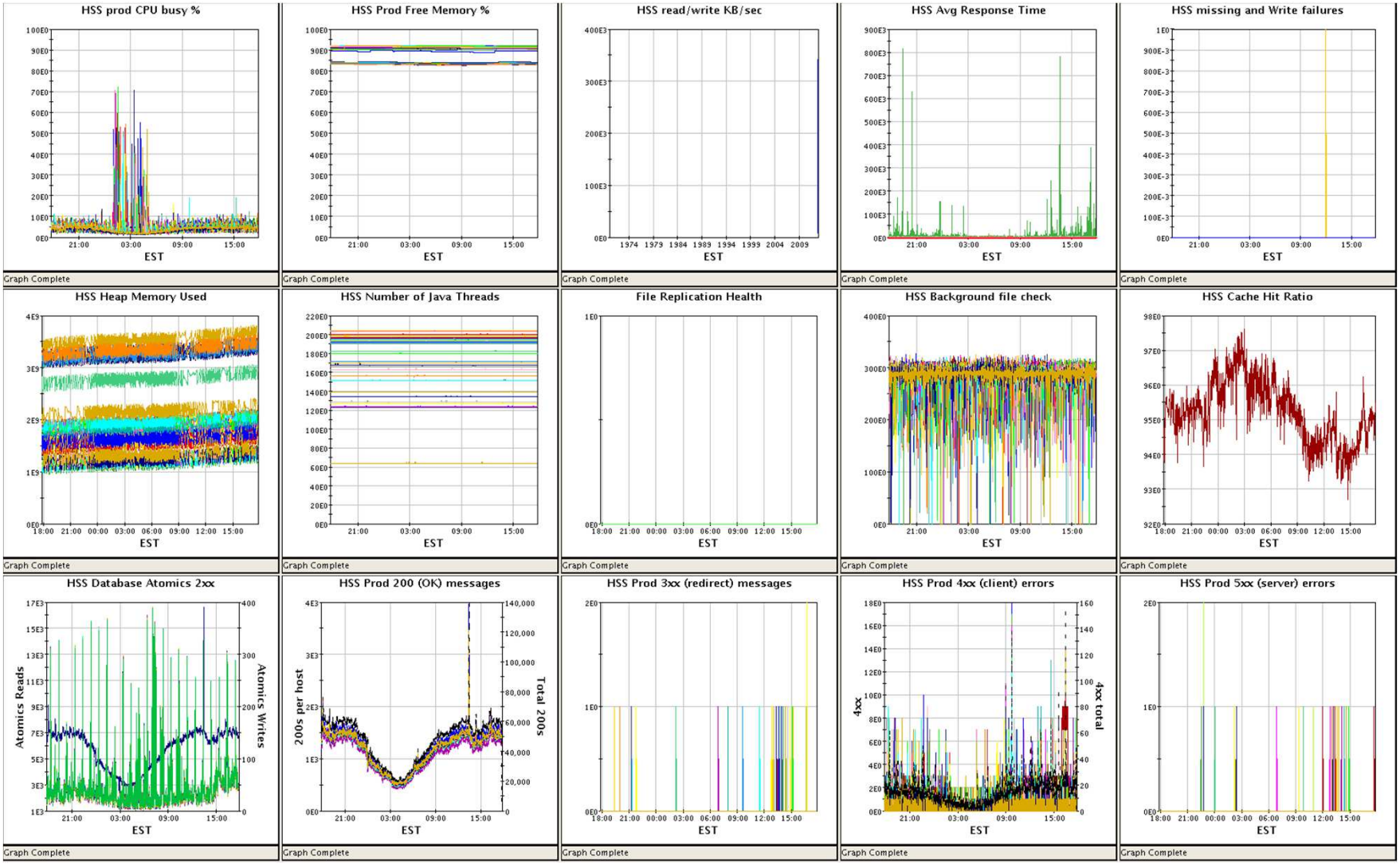
Delete the file

Fail to retrieve the deleted file

Look at some of the admin UI



Questions?


HSS prod CPU busy %
HSS Prod Free Memory %
HSS read/write KB/sec
HSS Avg Response Time
HSS missing and Write failures
HSS Heap Memory Used
HSS Number of Java Threads
File Replication Health
HSS Background file check
HSS Cache Hit Ratio
HSS Database Atomics 2xx
Atomics Reads
Atomics Writes
HSS Prod 200 (OK) messages
200s per host
Total 200s
HSS Prod 3xx (redirect) messages
HSS Prod 4xx (client) errors
4xx
4xx total
HSS Prod 5xx (server) errors
EST
Graph Complete