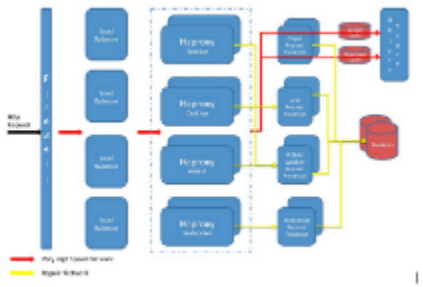
highscalability.com

Egnyte Architecture: Lessons Learned in Building and Scaling a Multi Petabyte Distributed System - High Scalability -

27-35 minutes



This is a guest post by Kalpesh Patel, an Architect, who works from home. He and his colleagues spends their productive hours scaling one of the largest distributed file-system out there. He works at Egnyte, an Enterprise File Synchronization Sharing and Analytics startup and you can reach him at @kpatelwork.

Your Laptop has a filesystem used by hundreds of processes, it is limited by the disk space, it can't expand storage elastically, it chokes if you run few I/O

intensive processes or try sharing it with 100 other users. Now take this problem and magnify it to a file-system used by millions of paid users spread across world and you get a roller coaster ride scaling the system to meet monthly growth needs and meeting SLA requirements.

Egnyte is an Enterprise File Synchronization and Sharing startup founded in 2007, when Google drive wasn't born and AWS S3 was cost prohibitive. Our only option was to roll our sleeves and build an object store ourselves, overtime costs for S3 and GCS became reasonable and because our storage layer was based on a plugin architecture, we can now plug-in any storage backend that is cheaper. We have re-architected many of the core components multiple times and in this article I will try to share what is the current architecture and what are the lessons we learned scaling it and what are the things we can still improve upon.

The Platform

- Tomcat
- MySQL

- HAProxy
- Nginx
- Memcached
- Apache
- Elasticsearch
- Redis
- RabbitMQ
- CentOS
- Puppet
- New Relic
- AppDynamics
- ZooKeeper
- LDAP
- Nagios
- Graphite
- Cacti
- Apache FTP server
- OpenTSDB
- Google BigQuery
- Google BigTable

- Google analytics
- MixPanel
- Tableau
- ReactJS/Backbone/Marionette/JQuery/npm/nighwatch
- Rsync
- PowerDNS
- Mashery
- SOA architecture based on REST apis.
- Java used to power core file system code
- Python used to power mostly client side code, migration scripts, internal scripts. Some python code still resides on server but most of it is migrated to Java as we had more server developers with java familiarity and scaling experience.
- Native Android/iOS apps
- Desktop and server sync client for various platforms
- GCE
- GCS/S3/Azure/….

The Stats

- 3 primary data centers including one in Europe (due to

safe harbor rules)

- 200+ Tomcat service nodes
- 300+ Storage nodes powered by Tomcat/nginx
- 100+ MySQL nodes
- 50+ Elasticsearch nodes
- 20+ HAProxy nodes
- and many other types of service nodes
- Multiple petabytes of data stored in our servers and GCS
- Multiple petabytes of data content indexed in elasticsearch
- Lots of desktop clients syncing files with cloud

Getting To Know You

What is the name of your system and where can we find out more about it?

Egnyte is the name of the startup and the core system has many main parts like CFS(cloud file server), EOS (Egnyte object store), Event Sync and ... You can find more about these in the For The Techies section at our blog.

What is your system is for?

It is powering sync and share needs of thousands of businesses and cloud enable their existing file systems to be used by a variety of endpoints like FTP, webdav, mobile, public api, Web UI and ....

Why did you decide to build this system?

In 2007 businesses/workforce had started to become more distributed, customers were using multiple devices to access the same files and there was a need to make this experience as smooth as possible

How is your project financed?

It was a bootstrapped company and later we went on an raised $62.5 million over last 8 years across 5 rounds.

What is your revenue model?

We don't have any free users, we do offer a 15 day free trial, after that customer pay by no of seats.

How do you market your product?

We started with SEM/SEO but overtime as we grew, we

used many channels to acquire customers like Social media, Biz dev, Trade shows, SEM, SEO, Inbound marketing and high touch sales for Enterprise customers.

How long have you been working on it?

It was founded in 2007, it's 8 years old currently.

How big is your system? Try to give a feel for how much work your system does.

We store multibillion files and multiple petabytes of data. We observe more than 2K api request per second on average as per New Relic. We store data in 3 primary data centers due to safe harbor rules and location proximity. More on this is in the stats section.

What is your in/out bandwidth usage?

I dont have the exact numbers as it keeps growing but customers downloaded close to 1 Petabyte of compressed/uncompressed data last month.

How many documents, do you serve? How many images? How much data?

We store multibillion files and multiple petabytes of data. We store all kinds of files and our top 5 file extensions are pdf, doc/docx, xls/xlsx, jpeg and png.

What is your ratio of free to paying users?

All our users are paid users. We offer a free 15 day trial and after that they convert or we disable them.

How many accounts have been active in the past month?

All of our customers are paid accounts and almost everyone is active during the month . We power their file system needs, Who doesn't uses electricity at home?

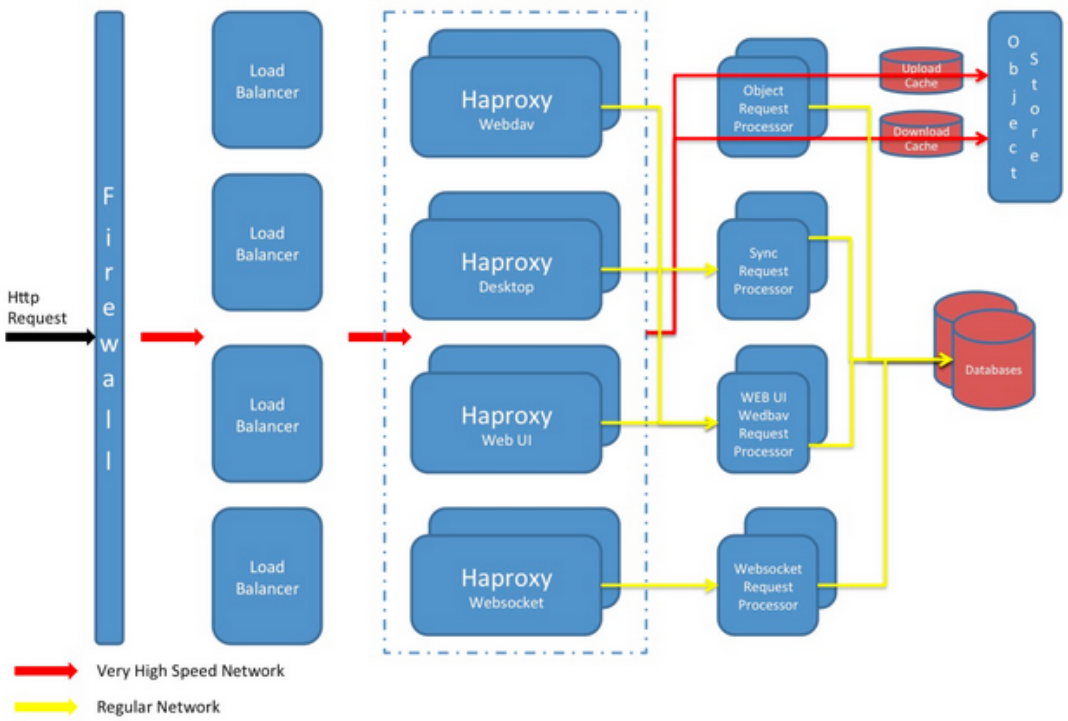
How Is Your System Architected?

What is the architecture of your system?

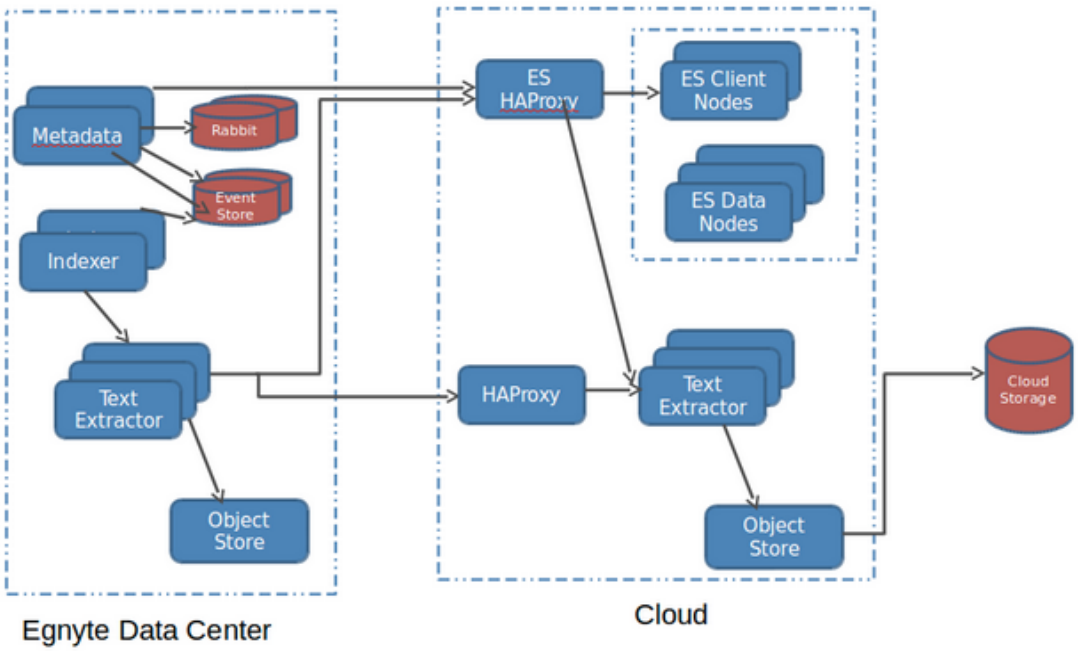
We use a service oriented architecture based on REST and it allow us to scale each service independently. This also allows us to move some of the backend services to be hosted in public cloud. All services are stateless and use database or our own object store for storage. As there are many services it's difficult to draw

all of them them in one diagram.

A 10000ft overview of typical request flow looks like below



A 10000ft overview of Search architecture looks like below



What particular design/architecture/implementation challenges does your system have?

Some of the biggest architecture challenges were: -

1. Scaling the file storage frugally
2. Scaling the metadata access
3. Realtime sync of files to desktop clients

What did you do to meet these challenges?

1. For storage we wrote our own and now we use a pluggable storage architecture to store to any public cloud like S3, GCS, Azure....
2. To scale metadata we moved to Mysql and started

using sharding. At some point we were throwing more hardware temporarily to get some breathing room in order to peel the layers of scaling onion one by one.

3. For real time sync we had to change our sync algorithm to work more like Svn where the client receives incremental events and try to do an eventual consistent sync with cloud state.
4. Monitor, Monitor and Monitor. You can't optimize what you can't measure. At some point we were monitoring too much that we can't focus on all metrics, we had to rely on anomaly detection tools like New Relic, AppDynamics, OpenTSDB and custom reports to allow us to focus on problems that are becoming from green->yellow->red. The intent is to catch them when they are yellow and before customer notices.

How does your system evolve to meet new scaling challenges?

We have re-architected many layers many times. I can't tell all in this article but I will try to list the few iterations of core metadata, storage,search layers over last 7 years.

1. Version 1: files metadata in lucene, files stored in

DRBD Filers mounted via NFS, search in lucene.
Choke point : Lucene updates were not real-time and it had to be replaced.

2. Version 2 : files metadata in Berkeley db, files stored in DRBD Filers mounted via NFS, search in lucene. Choke point : We broke the limits of NFS and it was choking left and right and it had to be replaced with http.
3. Version3 : files metadata in Berkeley db, files stored in EOS Filers served via HTTP, search in lucene. Choke point :Even sharded Berkeley DB was choking under the stress and there was a database crash with recovery taking hours, it had to be replaced.
4. Version4: files metadata in MySQL, files stored in EOS Filers served via HTTP, search in lucene. Choke point : Public cloud started becoming cheaper.
5. Version5: files metadata in MySQL, files stored in EOS/GCS/S3/Azure and served via HTTP, search in lucene. Choke point : Search started choking and had to be replaced.
6. Version6: files metadata in MySQL, files stored in EOS/GCS/S3/Azure served via HTTP, search in Elasticsearch. This is current architecture and soon one

of this may require another reincarnation :).

Do you use any particularly cool technologies or algorithms?

- We use exponential backoffs when calling between core services and services have circuit breakers to avoid thundering herd.
- We use fairshare allocation on core service node resources to incoming requests. Each incoming request on core service node is tagged and classified into various groups. Each group has a dedicated capacity and if one customer is making 1000 request per second and other is making 10 request then this system would ensure that the second customer would not face noisy neighbor issue. The trick is both customer may lands on same group on some node by coincidence due to hashing but they won't land on same group on all nodes, we add nodename as a salt to the hash.
- Some of the core services with SLA are isolated in PODs and this ensure that one bad customer won't choke the entire data center.
- We use event based sync in our desktop sync client code, as server events are happening they get pushed

to client from server and client replays them locally.

What do you do that is unique and different that people could best learn from?

Focus on the core capability of your startup and for it if you have to build something custom then go for it. There are many unique things but the storage layer, event based sync is definitely worth learning, here are more details on it Egnyte object store and Egnyte Canonical File System.

What lessons have you learned?

- You can't optimize what you can't measure: Measure everything possible and then optimize parts of system that are used 80% of the time first.
- When you are small, introduce technologies slowly, don't try to find the perfect tool out there for the problem you have in hand. Coding is the easiest part in lifecycle but its the maintenance like deployment/operations/learning curve will be hard if you have too many technologies initially. When you are big you would have enough fat to divide into services and have each service use its own technology and maintain it.

- When you are a startup sometimes you have to move fast, introduce the solution that you can do best right now and re-architect it overtime if you see traction.
- Look for single point of failures and hunt them down relentlessly. Put an extra effort to fix problems that keep you up at night and go from defensive to offensive mode as soon as possible.
- In SOA build circuit breakers to start sending 503s if your service is choked. Instead of penalizing everyone, see if you can do fair share allocation of resources and penalize only the abusive users.
- Add auto heal capability in service consumers, a service can choke and the consumers like desktop client or other services can do exponential backoff to release pressure on server and auto heal when the service is functional again.
- Always be available: Have a service level circuit breaker and a circuit breaker by customer. For e.g. if accessing file system over webdav or FTP has a performance issues, and it will take 4 hours to fix, then for those 4 hours, you can just kill FTP/webdav at firewall and ask customers to use web ui or other mechanism to work. Similarly if one customer is

causing an anomaly that is choking the system then temporarily disable that customer or service for that customer and reenable it when issue is fixed. We use feature flags and circuit breakers for this.

- Keep it simple: New engineers join every month so goal is to have them productive from week one, a simple architecture ensures easy induction.

Why have you succeeded?

Traction trumps everything. We reached product/market fit when the EFSS market was just exploding. The timing with good execution, financial discipline by management team lead to the success. A lot of competitors went to freemium model and raised a boat load of money but we were charging from day one and this allowed us to focus on growing the solution/team as demanded by market. Being focused on paid customers allowed us to deliver an enterprise class solution without paying the freemium penalty.

What do you wish you would have done differently?

I wish public cloud was not cost prohibitive when we started. I also wish we were on SOA from day one, it

took us some time to reach there but we are there now.

What wouldn't you change?

At this moment I won't replace MySQL / EOS as it allowed us to go from defensive to offensive positioning. I can't comment 2 years from now I would have same thoughts, we may change or augment some of them. The architecture changes as you encounter the next growth spurt.

How much up front design should you do?

Excellent question. The answer is “it depends”,

- If you are designing something like core storage layer or core metadata layer then adding 2 more weeks to your design won’t hurt much. When we were migrating from Berkeley DB to MySQL on our core metadata layer, I was under pressure and I had thought of taking a shortcut, when I ran it through our CTO he advised on taking a bit more time and “Doing the right thing” and as a retrospective that was an excellent decision.
- For a public API it's good to do a decent front design as you won’t get second chance to change it and you will have to maintain it for next 4-5 years.

- However if you are designing something for an internal service and migrating it to a new architecture won't be an year long then I advise doing very minimal front design and just build the version quickly and iterate on it as the usage grows.

How are you thinking of changing your architecture in the future?

- Do deployment in the middle of week (we are almost there)
- Use more of public cloud for any new services and migrate more services to public cloud.
- Move remaining of our source code from svn to git
- Make current development environment as close to production as possible (may be use docker or …).
- Automate schema management
- Add more performance benchmarking
- Build continuous delivery pipeline so we can increase the deployment to be weekly or daily instead of biweekly.
- Remove joins from some of the fastest growing tables by rearchitecting.

- Add an auto balancer for Mysql shards so I don't need to worry about occasional hotspots.
- Thin out some of the fat services into granular ones
- Use memcached proxy

How Is Your Team Setup?

How many people are in your team?

Around 100 Engineers(devops/ops/qa/developers/…), rest are sales,marketing,support, HR.

Where are they located?

Fairly distributed engineering team in the start but now gravitating mostly in Mountain View, Poznan , Mumbai. Some remote employees like myself and a handful others work from home.

Who performs what roles?

It's a big team, we have Product managers, UX team, devops, scrum teams, architects, engineers performing various roles. Initially at the start engineering team was flat and everyone would report to VP of engineering but now we have added a layer of management in

between.

Do you have a particular management philosophy?

If you develop something then you own the lifecycle of that product, which means you would work with QA,devops to ensure its tested/deployed. When it goes to production you would monitor it using the various internal tools like New Relic/Nagios and if there is a regression you would fix it.

If you have a distributed team how do you make that work?

Autonomy, 1-1 communication, hackathons, give them challenging work and they would be motivated.

What is your development environment?

- Ubuntu for server teams
- UI team uses Windows/mac and connect to local Ubuntu VM for REST API server or connect to shared QA instance
- Eclipse/Idea
- AWS for builds
- Maven

- Git/SVN
- Jenkins
- ReviewBoard/Sonar
- JIRA
- Google docs
- Jabber/Slack/Hangouts/Skype

What is your development process?

We use Scrum in the server team and have bi weekly releases. Developers/Teams working on long term features work on a sandbox and when done they get it tested by unit tests/selenium/manual QA and then merge to trunk to catch the 2 week release pipeline. We eat our own dogfood and the code goes to UAT (egnyte.egnyte.com used by all employees) 1 week before release, we catch any surprises not detected by automated tests. We do a production deploy every Friday night and monitor new relic, exception reports daily for any anomalies. We are changing deployment to be done in middle of week soon.

Is there anything that you would do different or that you have found surprising?

Many engineers work from home and it's surprising to see given autonomy, many remote employees are as productive and motivated as the HQ employees.

What Infrastructure Do You Use?

Which languages do you use to develop your system?

Java/Python mostly

How many servers do you have?

Last I know we have 700+. 100+ MySQL servers, 50+Elasticsearch, 200+tomcat services nodes, 300+ local storage nodes, 20+ HAProxy nodes and cache filers, nginx, python servers and various other nodes.

How is functionality allocated to the servers?

We use a service oriented architecture and servers are allocated based on the type of service. Some of the top level services are :

- Metadata
- Storage
- Object service

- Web UI
- Indexing
- EventSync
- Search
- Audit
- Snapshot/Data monitor
- Content Intelligence
- Real Time event delivery
- Text extraction
- Integrations
- Thumbnail generation
- Antivirus
- Spam
- Preview
- rsync
- API gateway
- Billing
- Support
- Sales
- and many more ….

How are the servers provisioned?

Most of the services are puppetized and run on VM, we run physical for only few of the things like MySQL, memcached, Metadata servers, indexing but most of this would get converted to VMs except database/cache/storage nodes. We use a third-party that provisions the servers based on a template and put it in data center and make it available for use to use.

What operating systems do you use?

CentOS7

Which web server do you use?

Nginx, Apache. Apache is used in some old flows and will get deprecated over time.

Which database do you use?

MySQL. We had used other databases like Berkeley DB, Lucene, Cassandra in past but we migrated overtime all of them to MySQL because of its developer/ops familiarity and scalability. More on this can be found at MySQL at Egnyte.

Do you use a reverse proxy?

Yes Nginx and HAProxy

Do you collocate, use a grid service, use a hosting service, etc?

We collocate.

What is your storage strategy?

We started by creating our own servers and packing as many hard drives as possible in a machine, we used to call them as DRBD Filers. We did this as AWS was cost prohibitive. We had evaluated GlusterFS but it wasn't scaling to meet our needs at that time so we built our own. Overtime S3 became cheap and GCS/Azure were born and we had architected the storage layer to be pluggable so now customers can decide which storage engine they want to use (Egnyte, S3, GCS, Azure, ….).

How do you grow capacity?

We do capacity planning sessions and overtime we came up with some metrics, based on those we watch the key indicators in our monitoring reports and pre-

order some extra capacity. Some services are now cloud enabled and we just provision more of them with a click of button.

Do you use a storage service?

Yes Egnyte, S3, GCS, Azure,

How do you handle session management?

We rewrote our architecture many times and currently 99% of the services are stateless. Only the service serving web UI uses session, we use sticky sessions in tomcat backed by memcached-session-manager but eventually my plan is to make this also stateless.

How is your database architected? Master/slave? Shard? Other?

We use Master-Master replication for almost all the databases with automatic failover, but switchover on some of the heavily mutating databases are manually done, we had encountered some issues where automatic switch would cause application data inconsistency due to replication lags and we need to re-architect some of core filesystem logic to fix this, we would eventually get this done. More details at length

on database architecture are answered below in question about handling database upgrades.

How do you handle load balancing?

We geo balance customers based on the IP they are accessing the system using DNS and within a data centre they are routed to their corresponding POD using HAProxy and inside POD they are again routed using HAProxy

Which web framework/AJAX Library do you use?

We have changed UI many times and this is one thing that is always in flux. In past we had use ExtJS, YUI, JQuery and what not. The latest iteration is based on ReactJS/Backbone/Marionette.

Which real-time messaging frame works do you use?

We use Atmosphere but eventually we would replace it with NodeJS

Which distributed job management system do you use?

We use RabbitMQ and Java/python based consumer services for this.

Do you have a standard API to your website? If so, how do you implement it?

Our API is classified into 3 types:-

1. Public API: This is the api we expose to third party app developers and integrations team and our Mobile app. We deprecate/upgrade api signature following proper deprecation workflow and changes are always backward compatible. We use Mashery as a gateway, the API is documented at https://developers.egnyte.com/docs
2. API for our clients: This api is internal to our clients and we don't guarantee backward compatibility if someone other than us uses this.
3. Internal protected API between services : This is the API used internally within our data centers by services to talk to each other and this can’t be called from outside firewall.

What is your object and content caching strategy?

We store petabytes of data and we can’t cache all of it

but if a customer has 50 million files on a give 15 day period he might be using only 1 million of them. We have cache filers nodes based on tomcat/nginx/local file system and it acts in LRU fashion. We can elastically increase decrease the no of cache filer servers. One of our biggest problem is upload speeds, how do you upload data as fast as possible to Egnyte from any part of the world, for this we built special Network pops, if you are curious you can read more on it at Speeding Up Data Access for Egnyte Customers

Memcached is used for caching metadata, we use separate memcached pool for caching long lived static data and file system metadata. The core file system metadata is huge, won't fit in current memcached nodes and would evict the recently used other kinds of data. to prevent this we use 2 kinds of pools and application code decides where to look for what kind of data. We allow evictions in filesystem memcached cache and strive for zero evictions in other kinds of memcached pools.

What is your client side caching strategy?

For our web ui we use PageSpeed that does resource optimization, caching and we use requireJS and

various other ways to download only required modules. Our Mobile and Desktop clients are rich use local filesystem as a cache.

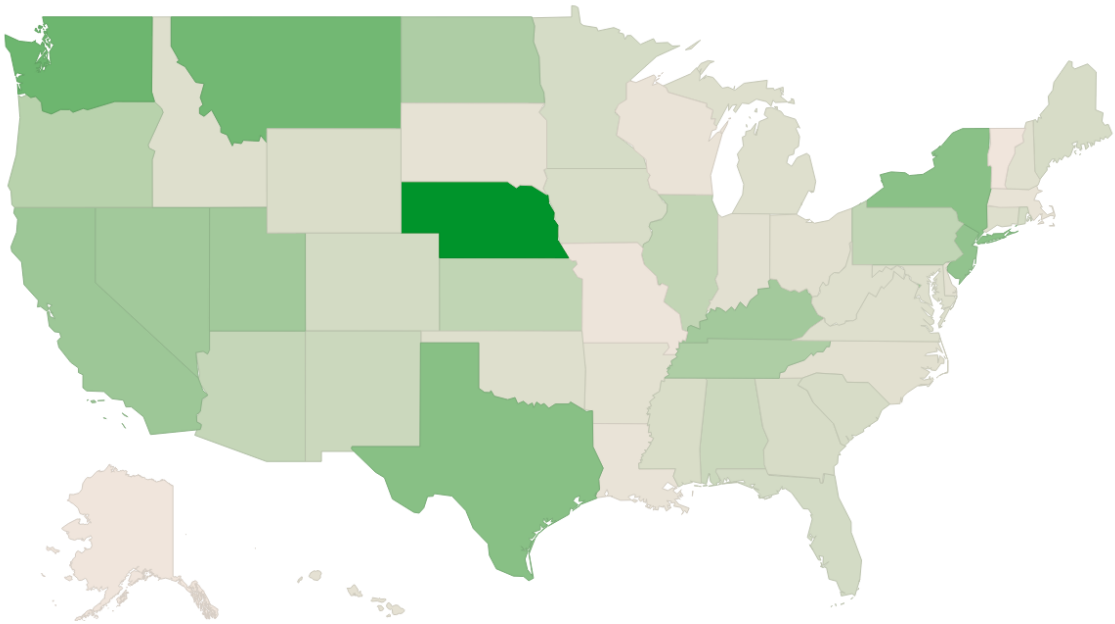
Which third party services do you use to help build your system?

Google BigQuery, New Relic, AppDynamics, MixPanel, Flurry, Tableau are some analytics services we use but most of the core components are build by us.

How Do You Manage Your System?

How do check global availability and simulate end-user performance?

We use nodes in different AWS regions to test bandwidth performance consistently. We also use internal haproxy reports to plot upload/download speeds observed by customer and proactively hunt them and use network pops and other strategies to accelerate packets.

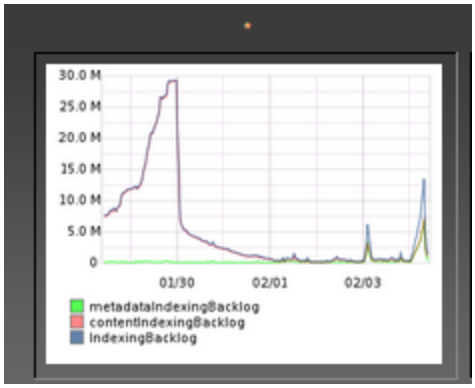


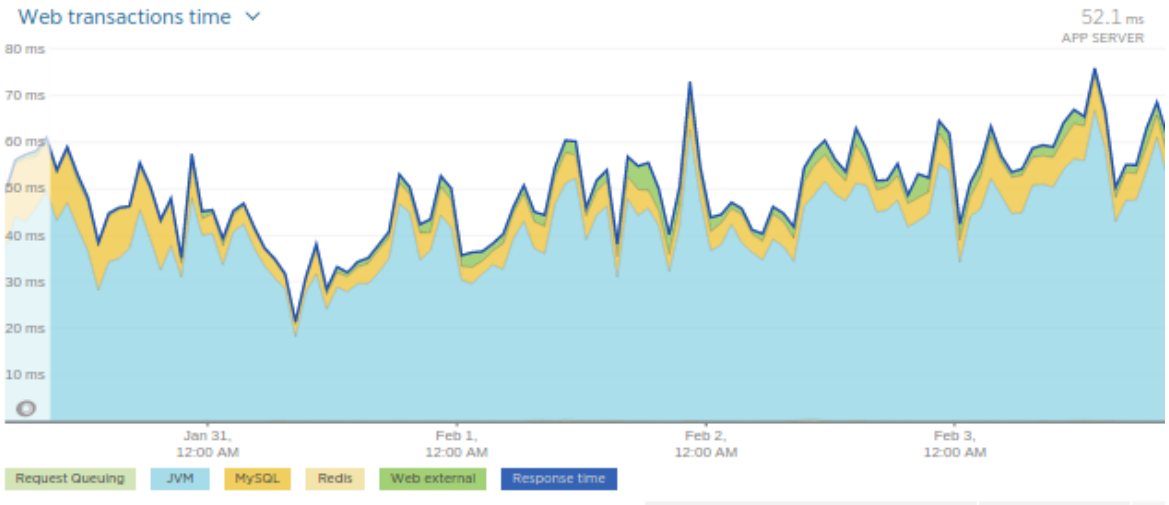
How do you health check your server and networks?

Nagios monitors and New Relic and some internal proactive exception analysis are used. More details on it are in this blog post

How you do graph network and server statistics and trends?

We use Graphite, cacti, Nagios and New Relic, AppDynamics.





How do you test your system?

Selenium, Junit, Nose, nightwatch and manual testing.

How you analyze performance?

New Relic, AppDynamics is used to monitor production tomcat nodes performance. We use graphite/nagios /internal tools and other tools to monitor performance for other parts of the system. More details on this is in this blog post Debugging Performance Issues in Distributed Systems

How do you handle security?

Dedicated Security team runs automated security benchmark tests before every release. Continuous automation pen tests are ran in production. We also use bug bounty programs and engage whitehat testing companies. Some customers do their own security testing using third parties.

How Do You Handle Customer Support?

We have a dedicated 24X7 distributed Customer success team, we use Zendesk and JIRA

Do you implement web analytics?

We use Google Analytics, Mixpanel, Flurry to measure feature usage

Do you do A/B testing?

Yes we use feature flags to do A/B testing. More on this is Using feature flags at Egnyte

How many data centers do you run in?

3 primary data centers including one in Europe (due to safe harbor rules) and network pops all around the

world.

How is your system deployed in data centers?

Puppet is used for deploying most of the new code. We are still rewriting few last pieces in architecture to be puppetized and those pieces currently are deployed using shell scripts.

Which firewall product do you use?

We use a mix of Juniper SRX and Fortigate firewalls.

Which DNS service do you use?

PowerDNS

Which routers do you use?

Cisco

Which switches do you use?

Arista

Which email system do you use?

We use our own SMTP server for outbound emails, for some internal tools we use SendGrid and for inbound

emails we use GMail.

How do you backup and restore your system?

For MySQL we use Percona XTraBackup , for Elasticsearch the data is replicated 3 times and we also take a backup to GCS using ElasticSearch GCS backup plugin. For customer files we replicate them 3 times. If a storage Filer fails to recover, we discard it, add a new Filer and replicate the copies again. For some customers we additionally replicate their data to the provider they choose. For customers using S3,Azure or GCS as pluggable storage layer it will ensure replication to prevent data loss.

How are software and hardware upgrades rolled out?

Most of the nodes are stateless and stateful component have an active-active failover. Upgrades are handled by taking the node out of the pool and upgrading and putting it back in the pool.

How do you handle major changes in database schemas on upgrades?

Different services use different types of databases and

they are upgraded in a different manner. At a 10000 ft they look like below screenshot :

1. EOS db stores object metadata and grows very fast, it's sharded and we keep adding more of these.
2. MDB grows even faster, it's sharded and we keep adding more of these.

3. DC_central is a dns database and remains fairly static. We run many replicas of this for scalability.
4. Pod_central has fast mutating data but does not grow beyond 20M rows per table. We run many replicas of this for scalability.

- Every database schema is always forward and backward compatible i.e. we never drop columns and code in same release, we first deploy the code in release-1 that stops using the column and after 2 weeks in release-2 we drop the column.
- non sharded dbs get upgraded as often as every 2 weeks. They are the ones storing all kind of feature driven tables. We currently upgrade them using a script but this is getting changed to use Sqitch
- Sharded db new column alter happens using an automated script
- Sharded db migration is a pain as we have 7000+ shards and growing, you can't do it in the 1 hour upgrade window. The way to do is:
- Live Code migrates the row as they need it. This means migration can happen over years.
- Migrate using feature flags, you have both old/new code live at same time and you migrate customer in

background and then flip a flag to switch them to go to new code path without downtime, more on this is here and here

- When we migrated from lucene to ElasticSearch we had no option than to reindex all the content and we did it using feature flags and it took some 3-4 months to finish.
- Schema consistency checker reports ensure that all schemas are same in all data centers after the upgrade.

Do you have a separate operations team managing your website?

Yes we have a dedicated devops team and an IT/Ops team responsible for monitoring and managing the system.

Miscellaneous

Who do you admire?

AWS: Their pace of innovation is admiring.

Google: Their tools like BigQuery, Analytics are awesome.

Elasticsearch : The rest api simplicity and architecture is awesome.

MySQL : It just works.

Eclipse/Jenkins: The plugin architecture is nice.

Have you patterned your company/approach on someone else?

We are a regular reader of http://highscalability.com/ , many designs are inspired by it.

- The POD architecture was inspired on Cell architecture at Tumblr, it's not an exact match but concept of isolating failures is same.
- The architecture to have a jitter in memcached and flush keys after 12 hours was inspired by facebook.
- Adding fingerprint to each database query by inspired by some article at http://highscalability.com/

We are hiring, check us out at Jobs Page and contact us at jobs@egnyte.com if you are interested in being a part of our amazing team at Egnyte.

On HackerNews