Creating massively scalable distributed storage systems

History, experience, mistakes and successes

Evgeniy Polyakov

Little bit about myself

- MIPT DPQE
- linux kernel hacker (dst, pohmelfs, w1, connector, kevent, osf/pof, carp…), hacking (DNS poisoning, earned NYT article)
- Yandex (head of linux filesystem development group)
- Reverbrain.com (CEO, continue developing massively scalable distributed storage systems)
- ioremap.net - storage and other tech ramblings

Little bit about this presentation

- dreams and requirements for the ideal distributed storage system
- naive empirical distributed storage types dichotomy (hardware level, block storage, filesystems, clouds, global clouds)
- dissecting each storage type, pros and cons
- Elliptics - distributed storage made by Reverbrain.com, pros and cons
- discussion and buzzwords

Little bit about Elliptics distributed storage

- medium to large objects
- massively scalable (53 billions of objects in 2013, ~20% of Facebook)
- geographical replication, survives whole continent going offline
- p2p server-client streaming
- very simple, fast, repairable

Dreams and requirements for the ideal distributed storage system

- truly scalable - the more servers we have, the higher the performance and storage size
- parallel, performance is proportional to the number of servers
- no single points (or clusters) of failures
- geographical replication, should survive datacenter going offline
- as simple as possible, but not less

Naive empirical distributed storage types dichotomy

- hardware level (AoE, iSCSI, RDMA, iWARP)
- block storage (dst, drbd, rados)
- filesystems (pohmelfs, ceph, lustre, gpfs, glusterfs, nfs)
- local clouds (mongodb, cassandra, riak, hbase, hdfs, Amazon S3)
- global clouds (elliptics)

Hardware level (AoE, iSCSI, RDMA, iWARP)

Pros:

- Legacy just works
- Performance

Cons:

- vendor lock-in
- network limitations
- debug is kind of impossible
- what is high availability, redundancy and failover?
- brain split problem hasn’t been raised yet, other issues prevail

Block storage level (dst, drbd, rados)

Pros:

- more flexible, some features (encryption, networking)
- baby-style scalability

Cons:

- brain split
- “client-side” failover and replication
- way too easy to corrupt data
- no cluster awareness (need for keepalived, carp)

Filesystems (pohmelfs, ceph, lustre, gpfs, glusterfs, nfs)

Pros:

- POSIX

Cons:

- network filesystems client side scalability (how to work with multiple mount points?)
- POSIX
- linux VFS (`ls -l` in the billion objects directory, nfs/lustre hacks)
- hard to work in lousy (even cross-datacenter) network environment

Cloud storages (mongodb, cassandra, riak, hbase, hdfs, Amazon S3)

Pros:

- very convenient flexible stateless interfaces (including http)
- high availability
- high performance (replica selection)

Cons:

- CAP - always have to drop something (which you will then regret)
- hard to work in lousy (even cross-datacenter) network environment
- when datacenter (AZ) is in trouble, the whole world screams

Global cloud storage (Elliptics)

Pros:

- medium-to-large objects, kilobytes-to-gigabytes
- geo-replication, storage just works when datacenter goes offline
- massive scalability (53+ billions of keys in 2013, ~20% of Facebook)
- performance (parallel access, low-level Eblob storage)
- byte/flv streaming, easy to extend to other objects like mp3
- distributed Nginx cache

Cons:

- CAS - we sacrificed consistency for scalability

Dreams and requirements for the ideal distributed storage system

- truly scalable - the more servers we have, the higher the performance and storage size
- parallel, performance is proportional to the number of servers
- no single (cluster) points of failures
- geographical replication, should survive datacenter going offline
- as simple as possible, but not less

Questions and generic storage discussion

http://reverbrain.com/elliptics
http://github.com/reverbrain/elliptics
Google groups -> reverbrain