



the **ceph** distributed storage system

sage weil
msst – april 17, 2012

outline

- why you should care
- what is it, what it does
- how it works, how you can use it
 - architecture
 - objects and data placement
 - file system
- big data, cloud
- current status, roadmap
- who we are, why we do this



why should you care about another
storage system?

requirements, time, money



storage requirements

- **scale**
 - terabytes, petabytes, exabytes
 - heterogeneous hardware
 - reliability and fault tolerance
- diverse storage needs
 - object storage
 - block devices
 - shared file system (POSIX, coherent caches)
 - structured data

time

- ease of administration
- no manual data migration, load balancing
- painless scaling
 - expansion **and** contraction
 - seamless migration

money

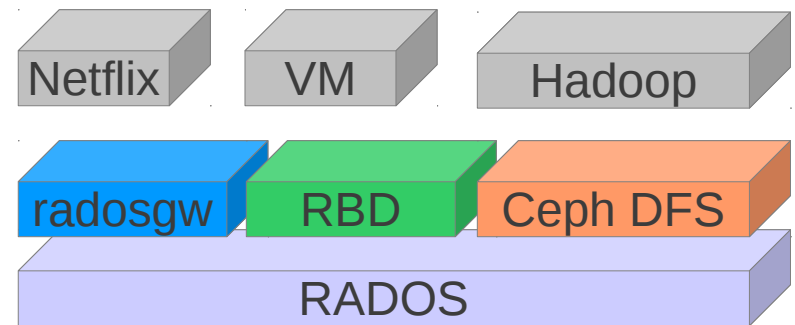
- low cost per gigabyte
- no vendor lock-in
- software solution
- commodity hardware
- open source

what is **ceph**?



unified storage system

- objects
 - small or large
 - multi-protocol
- block devices
 - snapshots, cloning
- files
 - cache coherent
 - snapshots
 - usage accounting



open source

- LGPLv2
 - copyleft
 - free to link to proprietary code
- no copyright assignment
 - no dual licensing
 - no “enterprise-only” feature set
- active community
- commercial support



distributed storage system

- data center (not geo) scale
 - 10s to 10,000s of machines
 - terabytes to exabytes
- fault tolerant
 - no SPoF
 - commodity hardware
 - ethernet, SATA/SAS, HDD/SSD
 - RAID, SAN probably a waste of time, power, and money



architecture

- monitors (ceph-mon)

- 1s-10s, paxos

- lightweight process

- authentication, cluster membership, critical cluster state

- object storage daemons (ceph-osd)

- 1s-10,000s

- smart, coordinate with peers

- clients (librados, librbd)

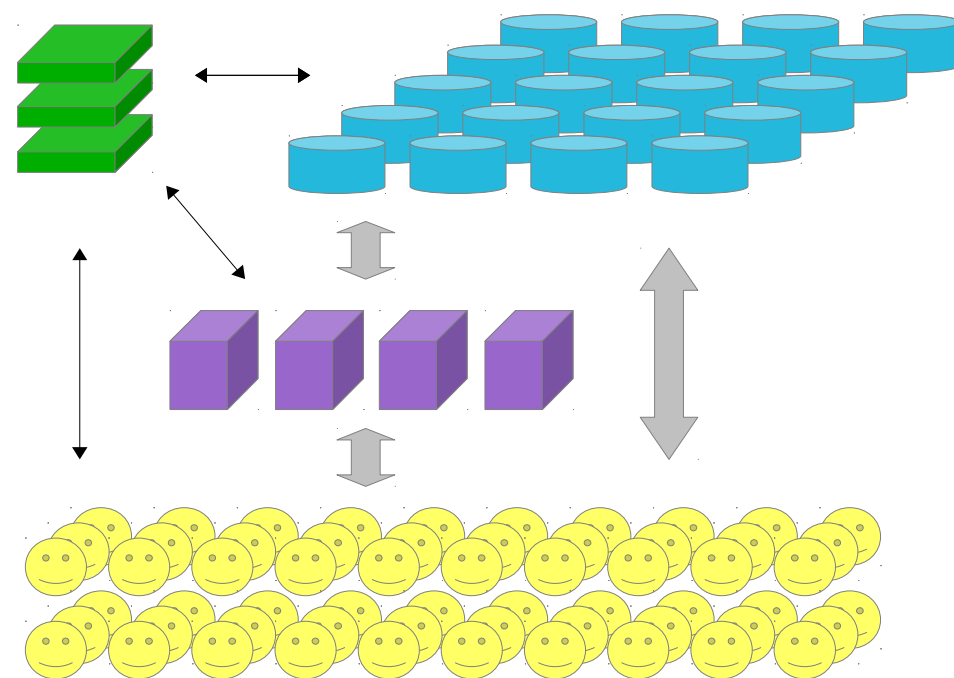
- zillions

- authenticate with monitors, talk directly to ceph-osds

- metadata servers (ceph-mds)

- 1s-10s

- build POSIX file system on top of objects



rados object storage model

- pools
 - 1s to 100s
 - independent namespaces or object collections
 - replication level, placement policy
- objects
 - trillions
 - blob of data (bytes to gigabytes)
 - attributes (e.g., “version=12”; bytes to kilobytes)
 - key/value bundle (bytes to gigabytes)

rados object API

- librados.so
 - C, C++, Python, Java. shell.
- read/write (extent), truncate, remove; get/set/remove xattr or key
 - like a file or .db file
- efficient copy-on-write clone
- atomic compound operations/transactions
 - read + getxattr, write + setxattr
 - compare xattr value, if match write + setxattr
- classes
 - load new code into cluster to implement new methods
 - calc sha1, grep/filter, generate thumbnail
 - encrypt, increment, rotate image



object storage

- client/server, host/device paradigm doesn't scale
 - dumb servers sit idle
 - if storage devices don't coordinate, clients must
- ceph-osds are **intelligent** storage daemons
 - coordinate with peers
 - sensible, cluster-aware protocols
- flexible deployment
 - one per disk, one per host, one per RAID volume
- sit on local file system
 - btrfs, xfs, ext4, etc.



data distribution

- all objects are replicated N times
- objects are automatically placed, balanced, migrated in a **dynamic** cluster
- must consider physical infrastructure
 - ceph-osds on hosts in racks in rows in data centers
- three approaches
 - pick a spot; remember where you put it
 - pick a spot; write down where you put it
 - calculate where to put it, where to find it



CRUSH

- pseudo-random placement algorithm
 - uniform, weighted distribution
 - fast calculation, **no lookup**
- placement rules
 - in terms of physical infrastructure
 - “3 replicas, same row, different racks”
- predictable, bounded migration on changes
 - $N \rightarrow N + 1$ ceph-osds means a bit over $1/N$ th of data moves

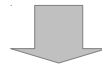
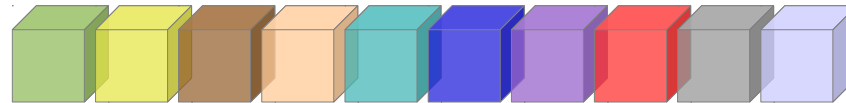
object placement

pool

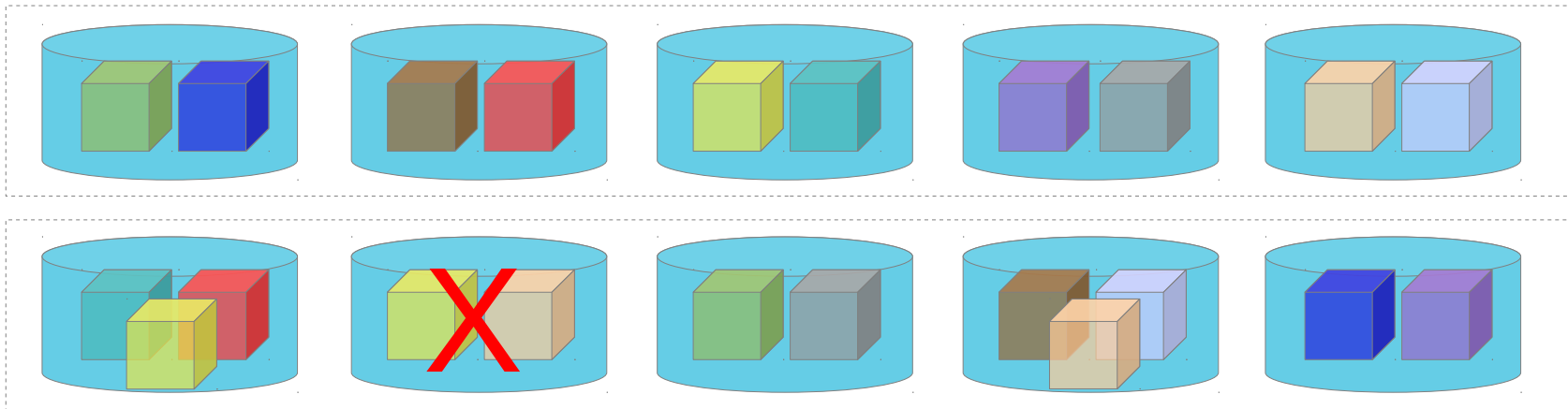


$\text{hash}(\text{object name}) \% \text{num_pg} = \text{pg}$

placement group (PG)

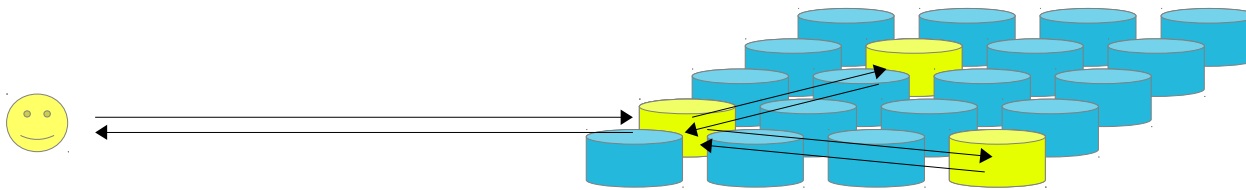


$\text{CRUSH}(\text{pg}, \text{cluster state}, \text{rule}) = [\text{A}, \text{B}]$



replication

- all data replicated N times
- ceph-osd cluster handles replication
 - client writes to first replica



- reduce client bandwidth
- “only once” semantics
- cluster maintains strict consistency

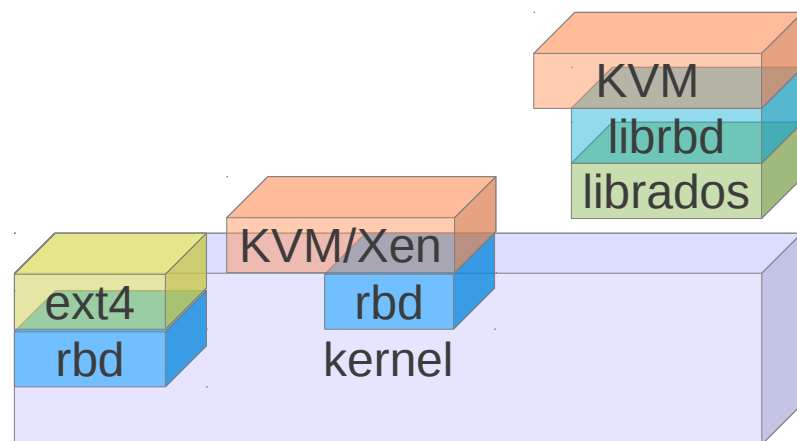
recovery

- dynamic cluster
 - nodes are added, removed
 - nodes reboot, fail, recover
- “recovery” is the norm
 - “map” records cluster state at point in time
 - ceph-osd node status (up/down, weight, IP)
 - CRUSH function specifying desired data distribution
 - ceph-osds cooperatively migrate data to achieve that
- any map update potentially triggers data migration
 - ceph-osds monitor peers for failure
 - new nodes register with monitor
 - administrator adjusts weights, mark out old hardware, etc.



rbd – rados block device

- replicated, reliable, high-performance **virtual disk**
 - striped over objects across entire cluster
 - thinly provisioned, snapshots
 - image cloning (real soon now)
- well integrated
 - Linux kernel driver (/dev/rbd0)
 - qemu/KVM + librbd
 - libvirt, OpenStack
- sever link between virtual machine and host
 - fail-over, live migration



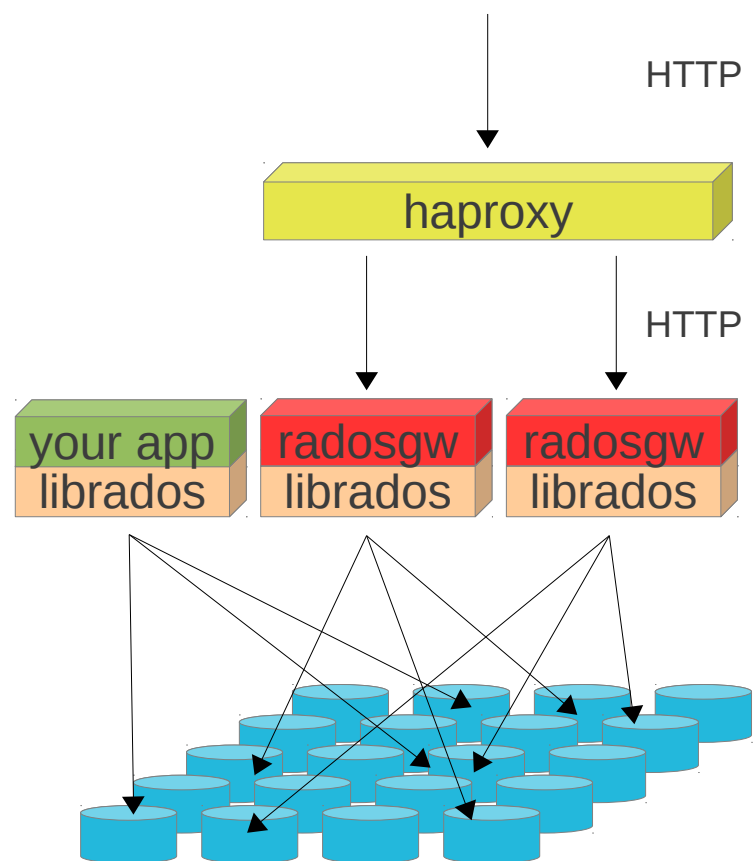
librados, radosgw

- librados

- direct parallel access to cluster
- rich API
- SaaS infrastructure

- radosgw

- RESTful object storage
 - S3, Swift APIs
- proxy HTTP to rados
- ACL-based security for the big bad internet

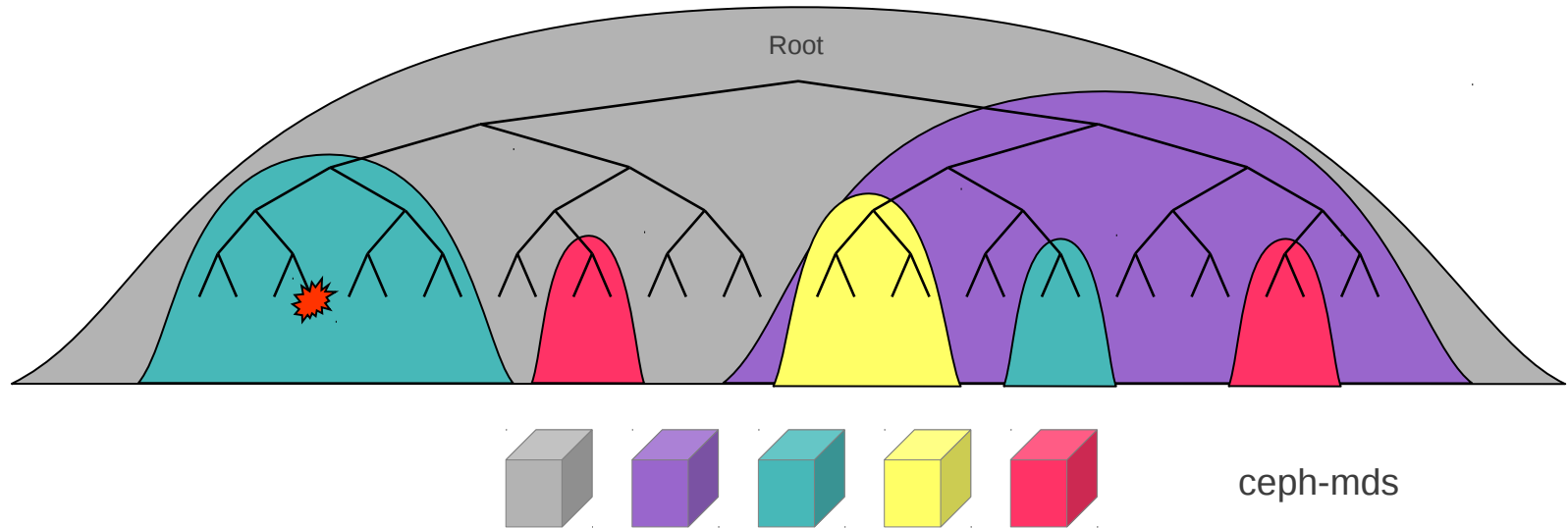


ceph distributed file system

- shared **cluster-coherent** file system
- separate metadata and data paths
 - avoid “server” bottleneck inherent in NFS etc
- ceph-mds cluster
 - manages file system hierarchy
 - redistributes load based on workload
 - ultimately stores everything in objects
- highly stateful client sessions
 - lots of caching, prefetching, locks and leases



dynamic subtree partitioning



- efficient
 - hierarchical partition preserve locality
- dynamic
 - daemons can join/leave
 - take over for failed nodes
- scalable
 - arbitrarily partition metadata
- **adaptive**
 - move work from busy to idle servers
 - replicate hot metadata

slick mds features

- ceph-mds tracks recursive directory stats
 - file sizes, counts, ctime
 - efficient

```
$ ls -alSh | head
total 0
drwxr-xr-x 1 root          root          9.7T 2011-02-04 15:51 .
drwxr-xr-x 1 root          root          9.7T 2010-12-16 15:06 ..
drwxr-xr-x 1 pomceph      pg4194980    9.6T 2011-02-24 08:25 pomceph
drwxr-xr-x 1 mcg_test1    pg2419992    23G  2011-02-02 08:57 mcg_test1
drwx--x--- 1 luko        adm          19G  2011-01-21 12:17 luko
drwx--x--- 1 eest        adm          14G  2011-02-04 16:29 eest
drwxr-xr-x 1 mcg_test2    pg2419992    3.0G 2011-02-02 09:34 mcg_test2
drwx--x--- 1 fuzyceph     adm          1.5G 2011-01-18 10:46 fuzyceph
drwxr-xr-x 1 dallasceph   pg275        596M 2011-01-14 10:06 dallasceph
```



snapshots

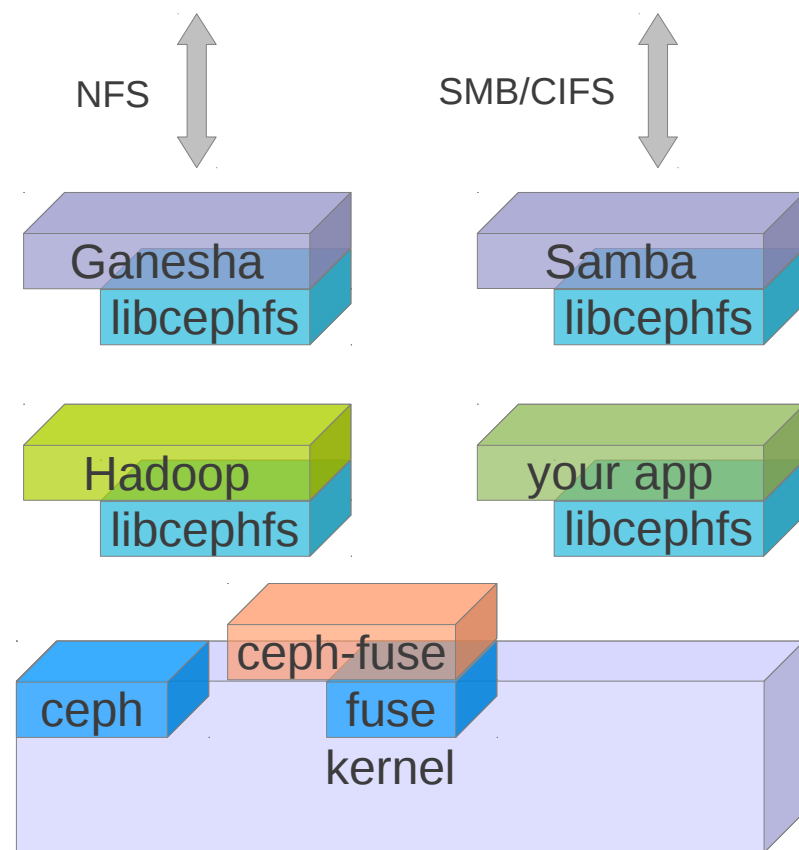
- volume or subvolume snapshots unusable at petabyte scale
 - snapshot arbitrary subdirectories
- simple interface
 - hidden '.snap' directory
 - no special tools

```
$ mkdir foo/.snap/one      # create snapshot
$ ls foo/.snap
one
$ ls foo/bar/.snap
_one_1099511627776      # parent's snap name is mangled
$ rm foo/myfile
$ ls -F foo
bar/
$ ls -F foo/.snap/one
myfile bar/
$ rmdir foo/.snap/one     # remove snapshot
```



multiple protocols, implementations

- Linux kernel client
 - `mount -t ceph 1.2.3.4:/ /mnt`
 - `export` (NFS), Samba (CIFS)
- ceph-fuse
- `libcephfs.so`
 - your app
 - Samba (CIFS)
 - Ganesha (NFS)
 - Hadoop (map/reduce)



hadoop

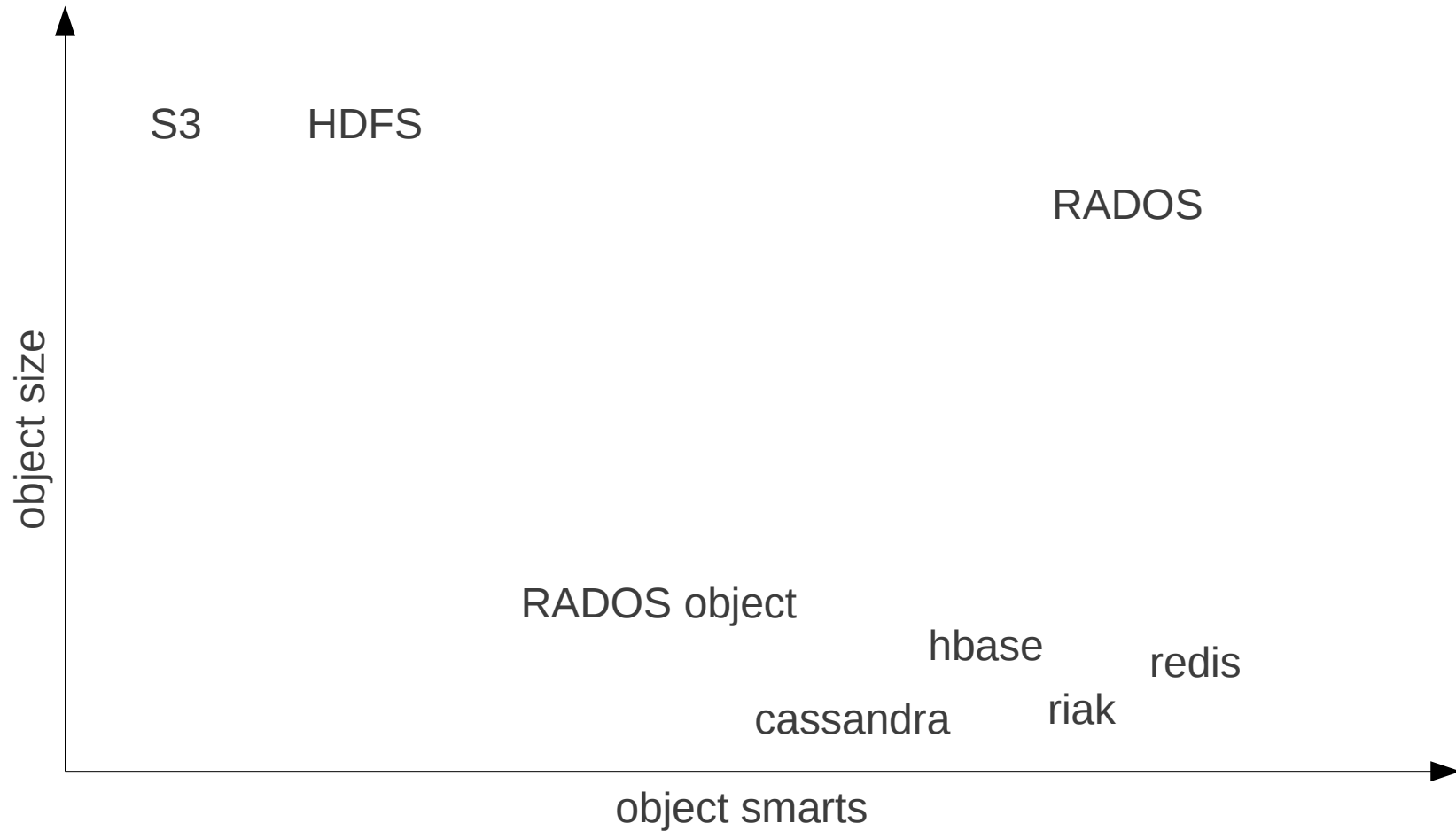
- seamless integration
 - Java libcephfs wrapper
 - Hadoop CephFileSystem
 - drop-in replacement for HDFS
- locality
 - exposes data layout
 - reads from local replica
 - first write does not go to local node
- can interact “normally” with Hadoop data
 - kernel mount
 - ceph-fuse
 - NFS/CIFS
- can colocate Hadoop with “normal” storage
 - avoid staging/destaging



distributed computation models

- object classes
 - tightly couple computation with data
 - carefully sandboxed
 - part of I/O pipeline
 - atomic transactions
 - rich data abstraction
 - blob of bytes (file)
 - xattrs
 - key/value bundle
- map/reduce
 - colocation of computation and data is optimization only
 - more loosely sandboxed
 - orchestrated data flow between files, nodes
 - job scheduling
 - limited storage abstraction

size vs (intra-object) smarts



cloud infrastructure

- compute
 - KVM, Xen
- scratch storage
 - can't migrate
- persistent block storage
 - EBS, RBD
- object storage
 - librados, or RESTful
- POSIX file system
 - NFS, Gluster, Ceph, etc.
- key/value (NoSQL)
 - Cassandra, riak, etc.
 - RADOS
- RDBMS
 - MySQL, Postgres, etc.
- distributed computation
 - Hadoop
 - RADOS classes
 - new hybrids



can I deploy it already?

- rados object store is stable
 - librados
 - radosgw (RESTful APIs)
 - rbd rados block device
 - commercial support
- file system is almost ready
 - feature complete
 - suitable for testing, PoC, benchmarking
 - needs testing, deliberate qa effort for production



roadmap

- RADOS performance and stability
- RBD caching and cloning
- radosgw
 - additional protocols (Google, CDMI)
 - versioning
 - snapshots
- Ceph DFS
 - QA
 - fine-grained security (better multi-tenancy)
 - fsck and online repair
 - robust NFS reexport
- async replication



community efforts

- openstack and RBD
 - volume manager improvements
- CloudStack and RBD
 - based on improved libvirt support
- Samba
 - plug libcephfs into Samba VFS
- Ganesha
 - libcephfs-base FSAL
 - pNFS
- Hadoop
 - Java wrappers
- Chef, Juju
- RDMA



why we do this

- limited options for scalable open source storage
 - oragefs, lustre
 - glusterfs
 - HDFS
- proprietary solutions
 - marry hardware and software
 - expensive
 - don't scale (well or out)
- industry needs to change



who we are

- created at UC Santa Cruz (2007)
- supported by DreamHost (2008-2011)
- spun off as NewCo (2012)
 - downtown Los Angeles, downtown San Francisco
- growing user and developer community
 - Linux distros
 - cloud computing stacks
 - OEMs
- we are **hiring**
 - C/C++/Python developers
 - devops, testing, proserv engineers

<http://ceph.com/>





why we like btrfs

- pervasive checksumming
- snapshots, **copy-on-write**
- efficient metadata (xattrs)
- inline data for small files
- transparent compression
- integrated volume management
 - software RAID, mirroring, error recovery
 - SSD-aware
- online fsck
- active development community

