SUSE® Enterprise Storage

Design and Performance for Ceph

Lars Marowsky-Brée
Distinguished Engineer
lmb@suse.com

What is Ceph?

From 10,000 meters

- Open Source Storage Distributed solution
- Most popular choice of distributed storage for openStack[1]
- Lots of goodies
  - Distributed Object Storage
  - Redundancy
  - Efficient Scale-Out
  - Can be built on commodity hardware
  - Lower operational cost

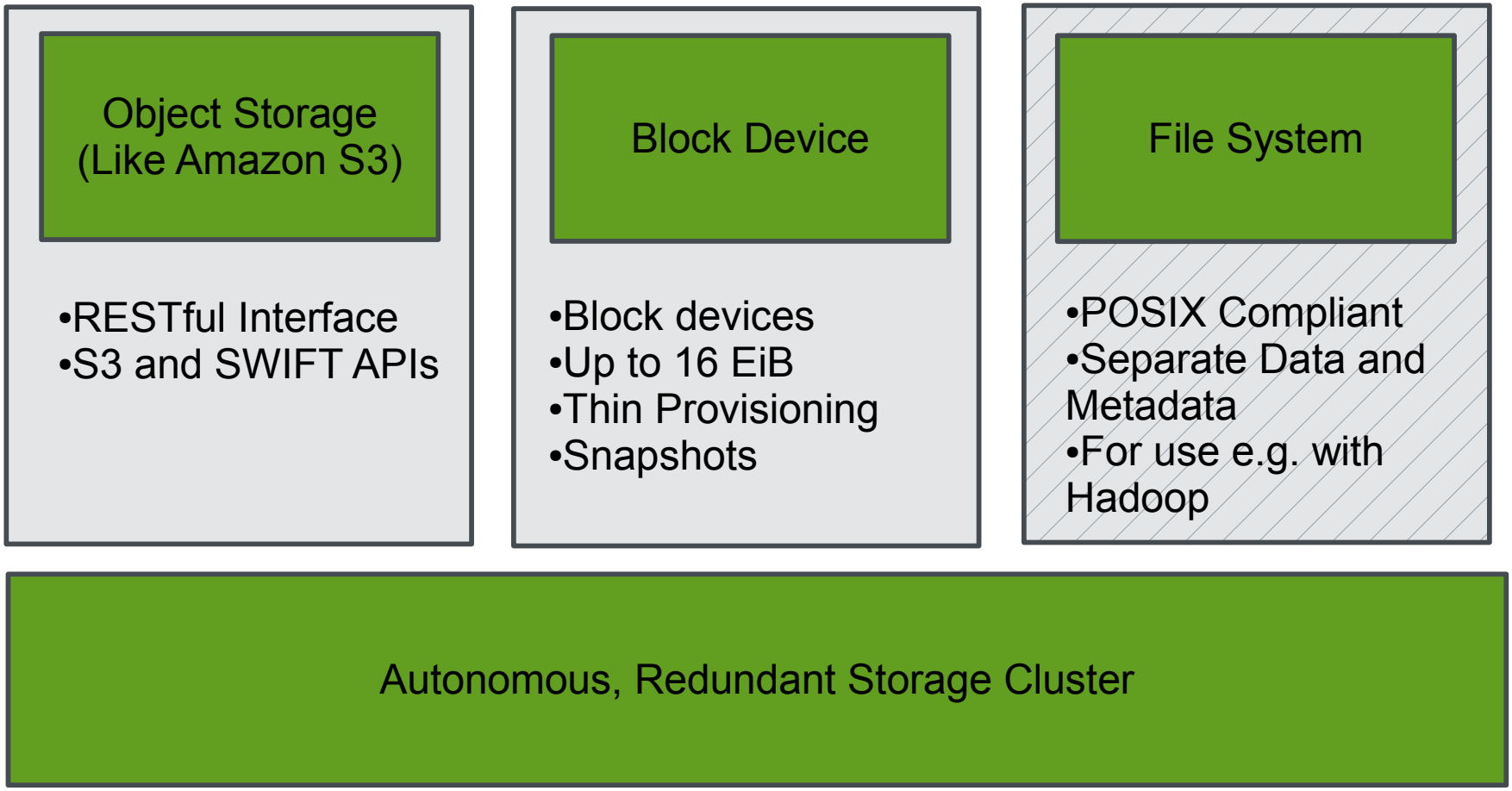
[1] *http://www.openstack.org/blog/2013/11/openstack-user-survey-statistics-november-2013/*

From 1,000 meters

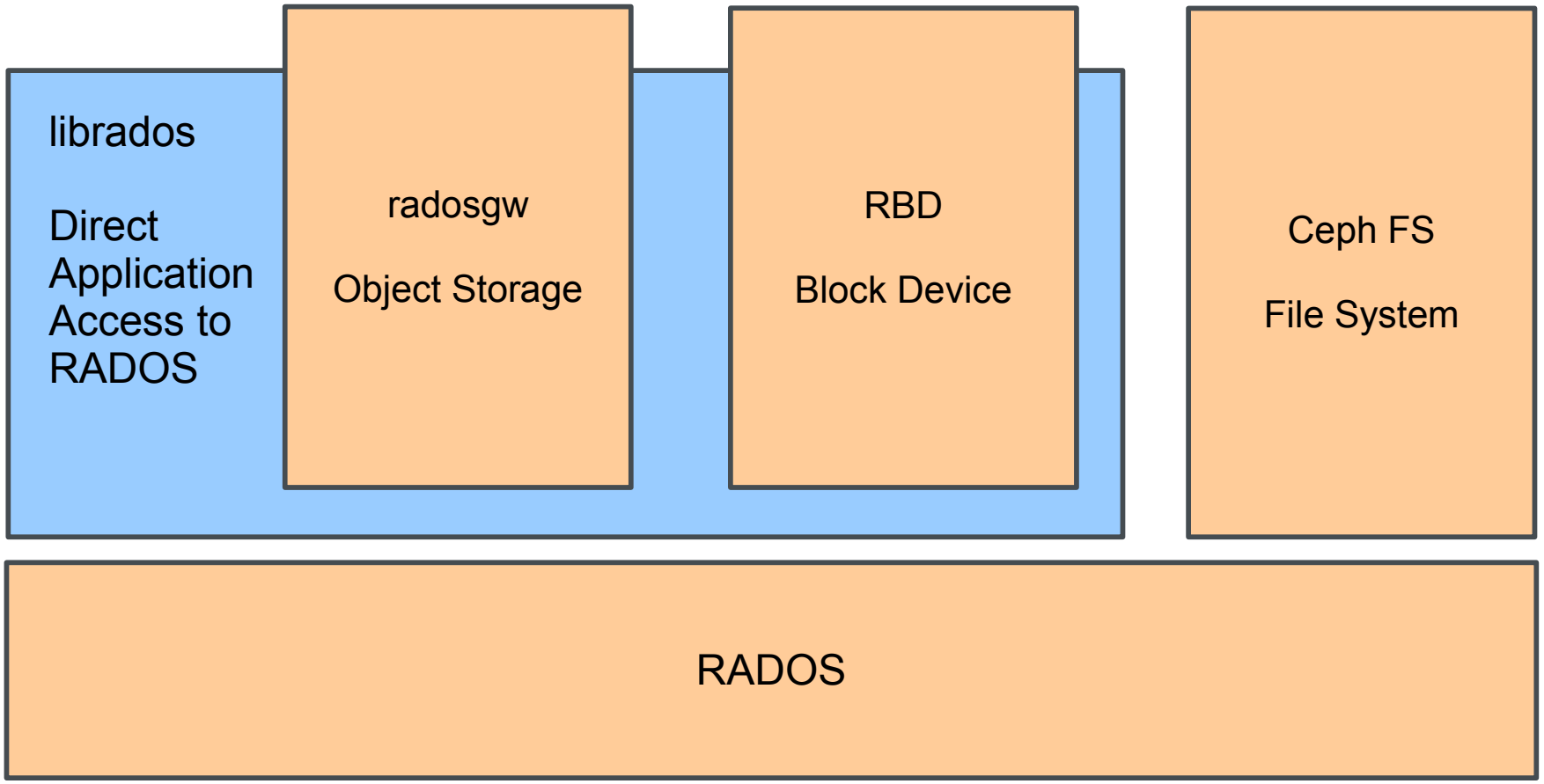
- Three interfaces rolled into one
  - Object Access (like Amazon S3)
  - Block Access
  - (Distributed File System)
- Sitting on top of a Storage Cluster
  - Self Healing
  - Self Managed
  - No Bottlenecks

From 1,000 meters

Unified Data Handling for 3 Purposes

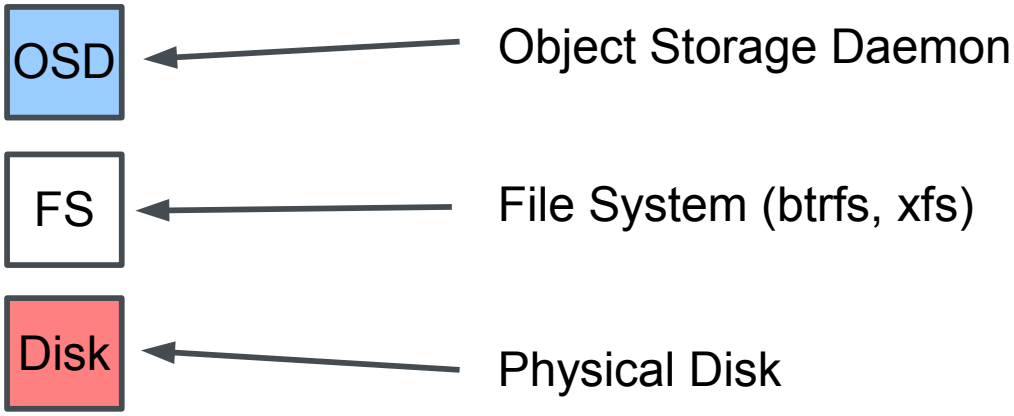


Component Names



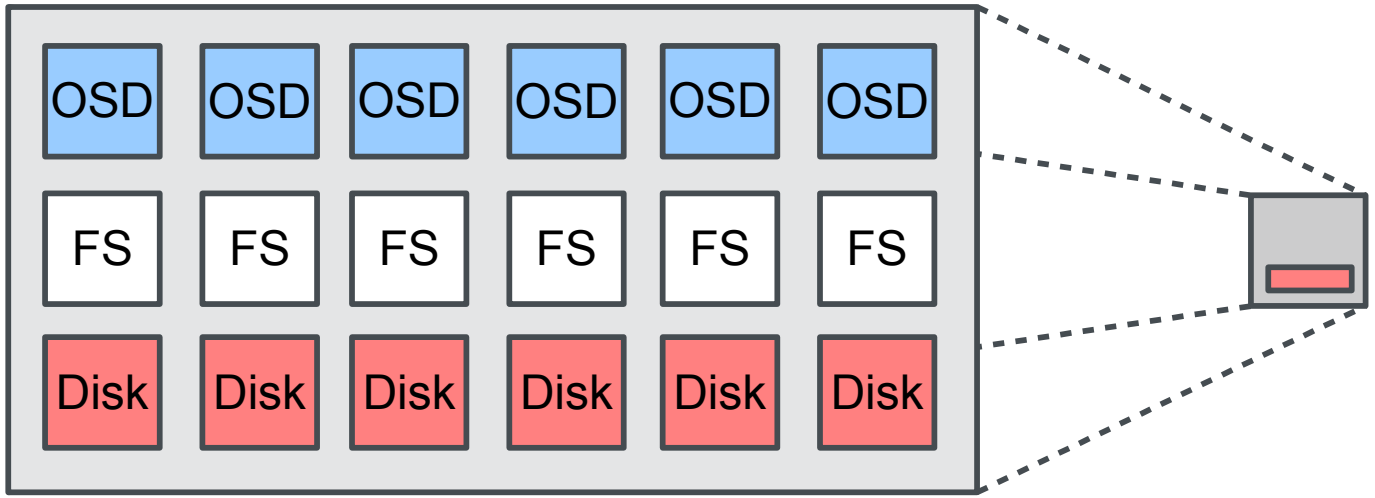
How does Ceph work?

For a Moment, zooming to Atom Level



- OSDs serve storage objects to clients
- Peer to perform replication and recovery

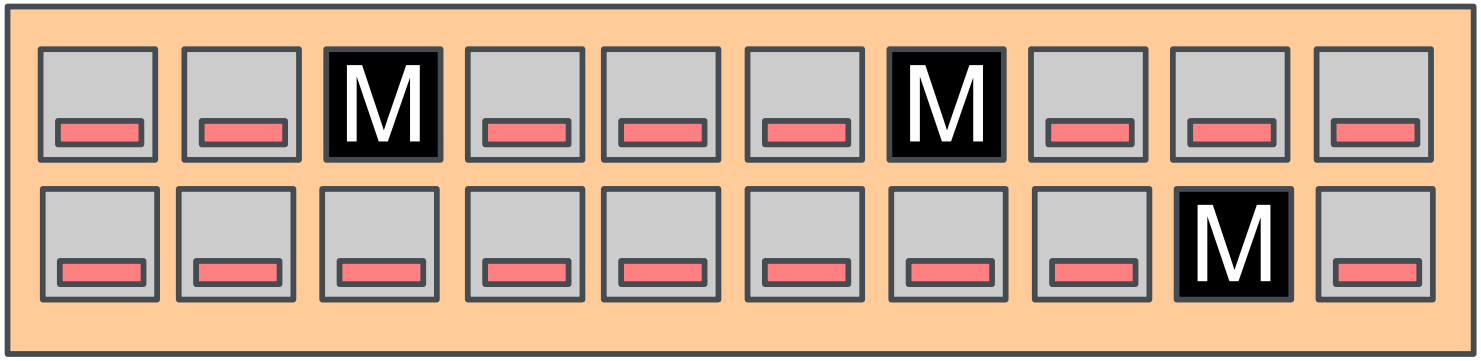
Put Several of These in One Node



Mix in a Few Monitor Nodes

- Monitors are the brain cells of the cluster
  - Cluster Membership
  - Consensus for Distributed Decision Making
- Do not serve stored objects to clients

Voilà, a Small RADOS Cluster



Several Ingredients

- Basic Idea
  - Coarse grained partitioning of storage supports policy based mapping (don't put all copies of my data in one rack)
  - Topology map and Rules allow clients to “compute” the exact location of any storage object
- Three conceptual components
  - Pools
  - Placement groups
  - CRUSH: deterministic decentralized placement algorithm

Pools

- A pool is a logical container for storage objects
- A pool has a set of parameters
  - a name
  - a numerical ID (internal to RADOS)
  - number of replicas OR erasure encoding settings
  - number of placement groups
  - placement rule set
  - owner
- Pools support certain operations
  - create/remove/read/write entire objects
  - snapshot of the entire pool

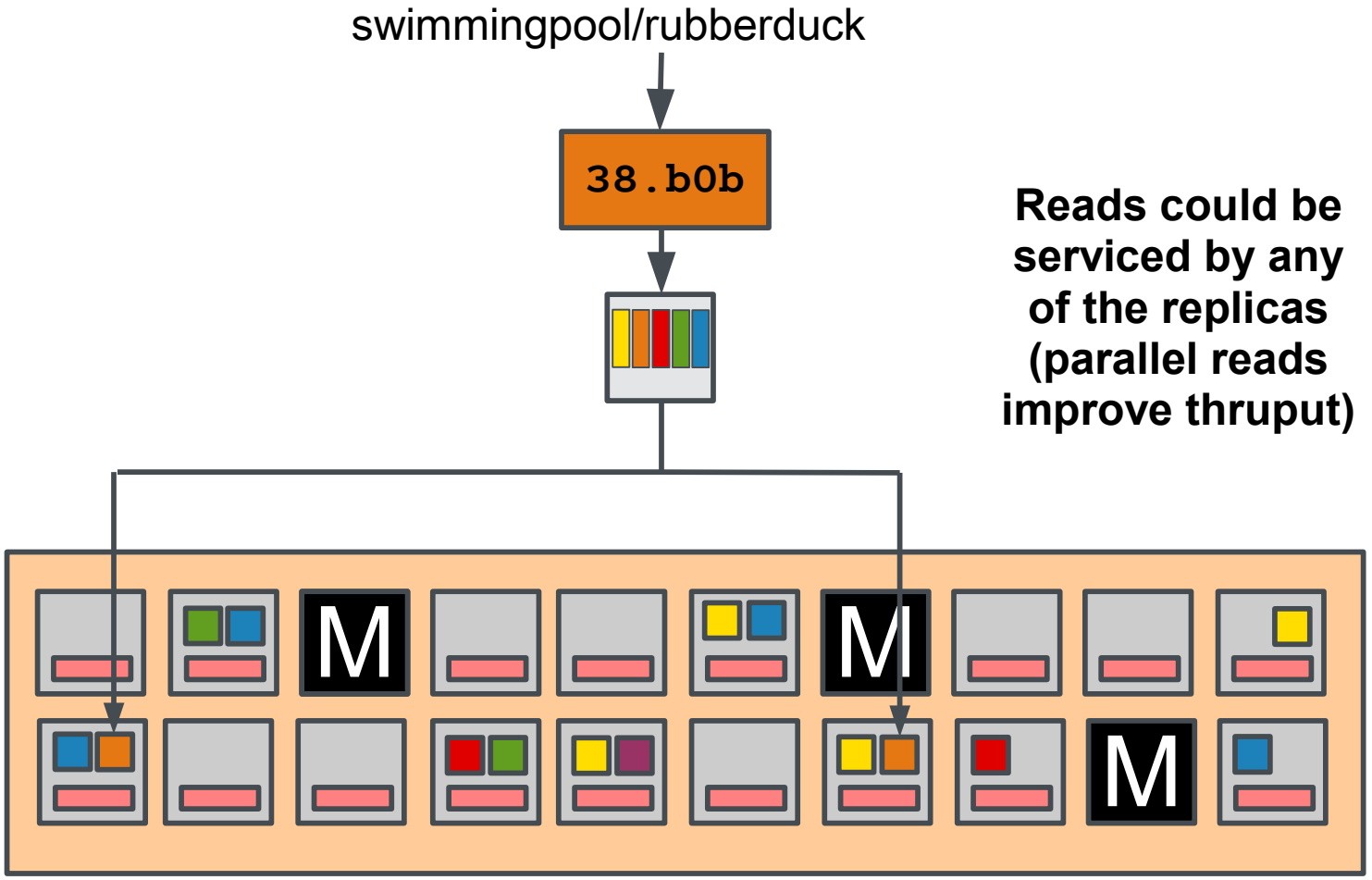
Placement Groups

- Placement groups help balance data across OSDs
- Consider a pool named “swimmingpool”
  - with a pool ID of 38 and 8192 placement groups (PGs)
- Consider object “rubberduck” in “swimmingpool”
  - hash(“rubberduck”) % 8192 = 0xb0b
  - The resulting PG is 38.b0b
- One PG typically spans several OSDs
  - for balancing
  - for replication
- One OSD typically serves many PGs

CRUSH

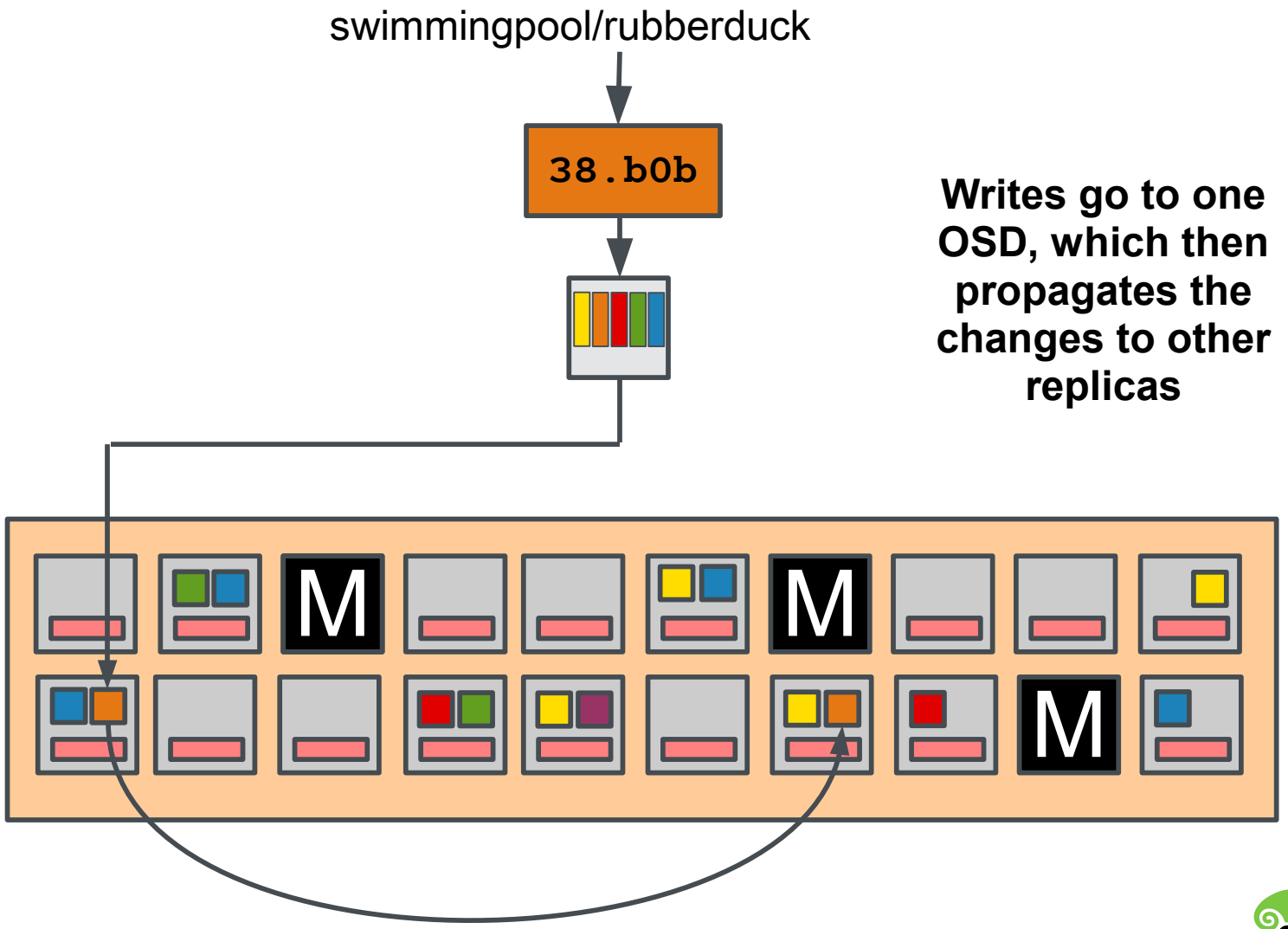
- CRUSH uses a map of all OSDs in your cluster
  - includes physical topology, like row, rack, host
  - includes rules describing which OSDs to consider for what type of pool/PG
- This map is maintained by the monitor nodes
  - Monitor nodes use standard cluster algorithms for consensus building, etc

CRUSH in Action: reading



Reads could be serviced by any of the replicas (parallel reads improve thruput)

CRUSH in Action: writing



Writes go to one OSD, which then propagates the changes to other replicas

Software Defined Storage

Legacy Storage arrays

- Limits:
  - Tightly controlled environment
  - Limited scalability
  - Few options
    - Only certain approved drives
    - Constrained number of disk slots
    - Few memory variations
    - Only very few networking choices
    - Typically fixed controller and CPU
- Benefits:
  - Reasonably easy to understand
  - Long-term experience and “gut instincts”
  - Somewhat deterministic behavior and pricing

Software Defined Storage (SDS)

- Limits:
  - ?
- Benefits:
  - Infinite scalability
  - Infinite adaptability
  - Infinite choices
  - Infinite flexibility
  - ... right.

Properties of a SDS system

- Throughput
- Latency
- IOPS

- Availability
- Reliability

- Capacity
- Density

- **Cost**

Architecting a SDS system

- These goals often conflict:
  - Availability versus Density
  - IOPS versus Density
  - Everything versus Cost
- Many hardware options
- Software topology offers many configuration choices
- There is no one size fits all

Setup choices

Networking (public and private)

- Ethernet (1, 10, 40 GbE)
  - Reasonably inexpensive (except for 40 GbE)
  - Can easily be bonded for availability
  - Use jumbo frames
- Infiniband
  - High bandwidth
  - Low latency
  - Typically more expensive
  - No support for RDMA yet in Ceph, need to use IPoIB

Network

- Choose the fastest network you can afford
- Switches should be low latency with fully meshed backplane
- Separate public and cluster network
- Cluster network should typically be twice the public bandwidth
  - Incoming writes are replicated over the cluster network
  - Re-balancing and re-mirroring take utilize the cluster network

Different access modes

- radosgw:
  - An additional gateway in front of your RADOS cluster
  - Little impact on throughput, but it does affect latency
- User-space RADOS access:
  - More feature rich than in-kernel rbd.ko module
  - Typically provides higher performance

Storage node

- CPU
  - Number and speed of cores
- Memory
- Storage controller
  - Bandwidth, performance, cache size
- SSDs for OSD journal
  - SSD to HDD ratio
- HDDs
  - Count, capacity, performance

Adding more nodes

- Capacity increases
- Total throughput increases
- IOPS increase
- Redundancy increases
- Latency unchanged
- Eventually: network topology limitations
- Temporary impact during re-balancing

Adding more disks to a node

- Capacity increases
- Redundancy increases
- Throughput might increase
- IOPS might increase

- Internal node bandwidth is consumed
- Higher CPU and memory load
- Cache contention
- Latency unchanged

OSD file system

- btrfs
  - Typically better write throughput performance
  - Higher CPU utilization
  - Feature rich
    - Compression, checksums, copy on write
  - The choice for the future!
- XFS
  - Good all around choice
  - Very mature for data partitions
  - Typically lower CPU utilization
  - The choice for today!

Impact of caches

- Cache on the client side
  - Typically, biggest impact on performance
  - Does not help with write performance
- Server OS cache
  - Low impact: reads have already been cached on the client
  - Still, helps with readahead
- Caching controller, battery backed:
  - Significant impact for writes

Impact of SSD journals

- SSD journals accelerate bursts and random write IO
- For sustained writes that overflow the journal, performance degrades to HDD levels
- SSDs help very little with read performance
- SSDs are very costly
  - ... and consume storage slots -> lower density
- A large battery-backed cache on the storage controller is highly recommended if not using SSD journals

Hard disk parameters

- Capacity matters
  - Often, highest density is not most cost effective
- On-disk cache matters less
- Reliability advantage of Enterprise drives typically marginal compared to cost
  - Buy more drives instead
- RPM:
  - Increase IOPS & throughput
  - Increases power consumption
  - 15k drives quite expensive still

Impact of redundancy choices

- Replication:
  - n number of exact, full-size copies
  - Potentially increased read performance due to striping
  - Increased cluster network utilization for writes
  - Rebuilds can leverage multiple sources
  - Significant capacity impact
- Erasure coding:
  - Data split into k parts plus m redundancy codes
  - Better space efficiency
  - Higher CPU overhead
  - Significant CPU and cluster network impact, especially during rebuild
  - Cannot directly be used to with block devices (see next slide)

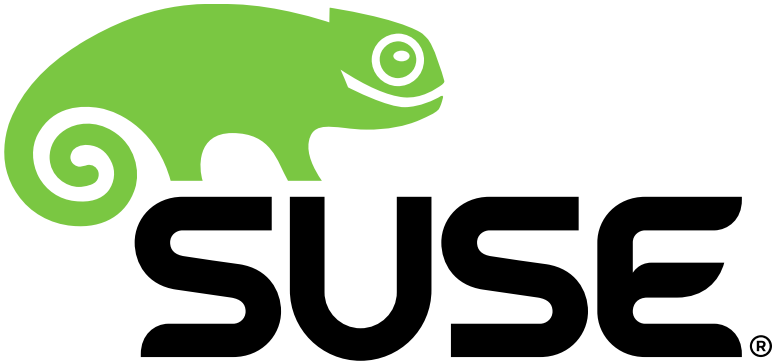
Cache tiering

- Multi-tier storage architecture:
  - Pool acts as a transparent write-back overlay for another
    - e.g., SSD 3-way replication over HDDs with erasure coding
  - Can flush either on relative or absolute dirty levels, or age
  - Additional configuration complexity and requires workload-specific tuning
  - Also available: read-only mode (no write acceleration)
  - Some downsides (no snapshots)
- A good way to combine the advantages of replication and erasure coding

Conclusion

Questions and answers?

Thank you.



Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org