Building and Running a Solr-as-a-Service

SHAI ERERA

IBM

Who Am I?

- Working at IBM – Social Analytics & Technologies
- Lucene/Solr committer and PMC member
- http://shaierera.blogspot.com
- shaie@apache.org

Background

- More and more teams develop solutions with Solr
  - Different use cases: search, analytics, key-value store…
- Many solutions become cloud-based
- Similar challenges deploying Solr in the cloud
  - Security, cloud infrastructure
  - Solr version upgrades
  - Data center awareness / multi-DC support
  - …

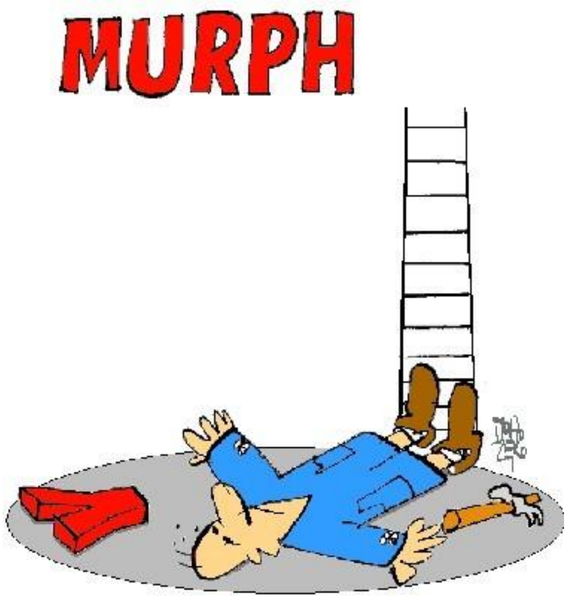
Mission

Provide a cloud-based service for managing hosted Solr instances

- Let users focus on indexing, search, collections management
  - **NOT** worry about **cluster health, deployment, high-availability …**
- Support the **full Solr API**
- **Adapt** Solr to the challenging cloud environment

Developing Cloud-Based Software is Fun!

- A world of micro-services: Auth, Logging, Service Discovery, Uptime, PagerDuty …
- Infrastructure decisions
  - Virtual Machines or Containers?
  - Local or Remote storage?
  - Single or Multi Data Center support?
- Software development and maintenance challenges
  - How to test the code?
  - How to perform software upgrades?
  - How to migrate the infrastructure?
- Stability/Recovery – “edge” cases are not so rare

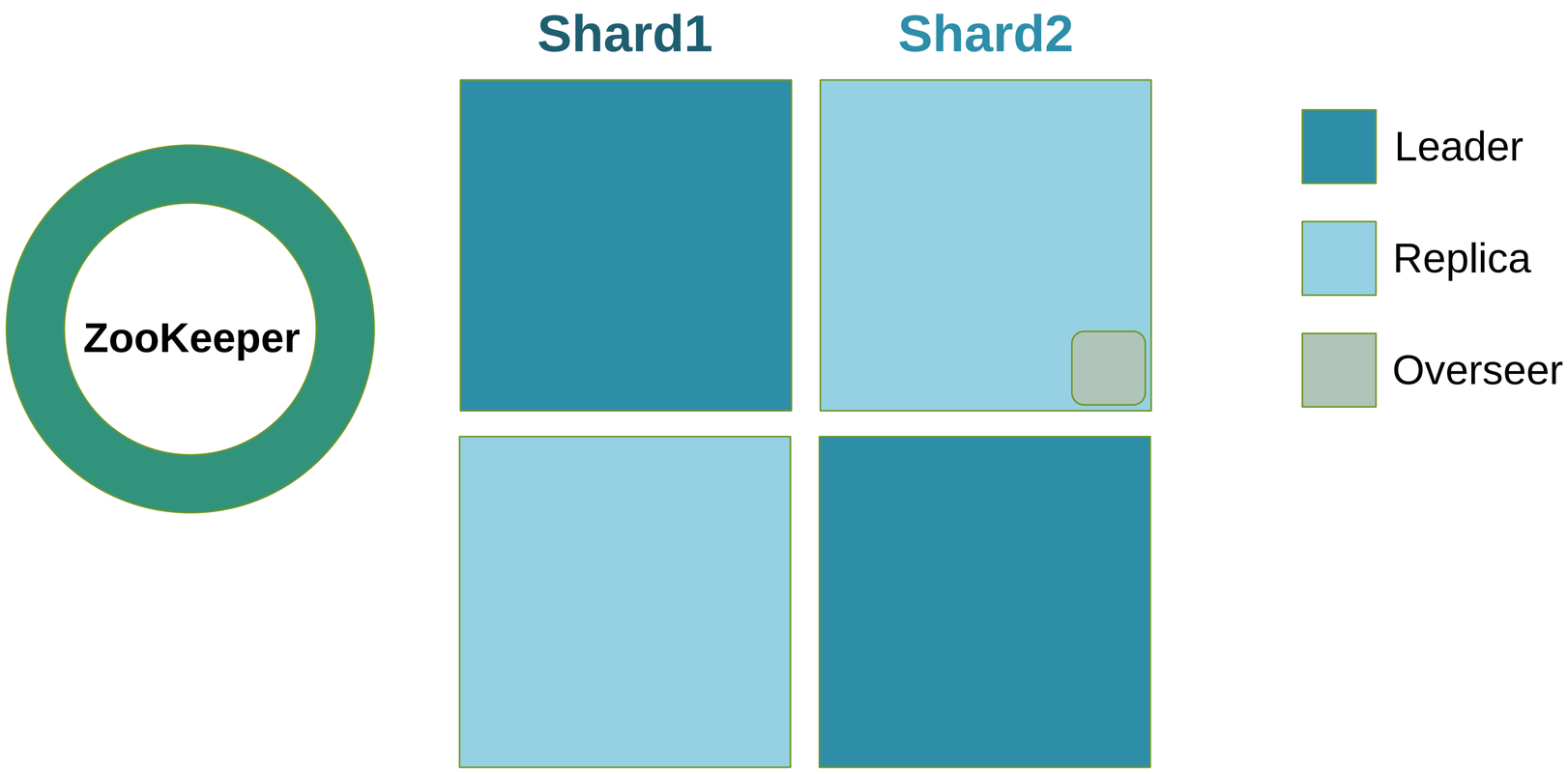


** Whatever can go wrong, will go wrong!*

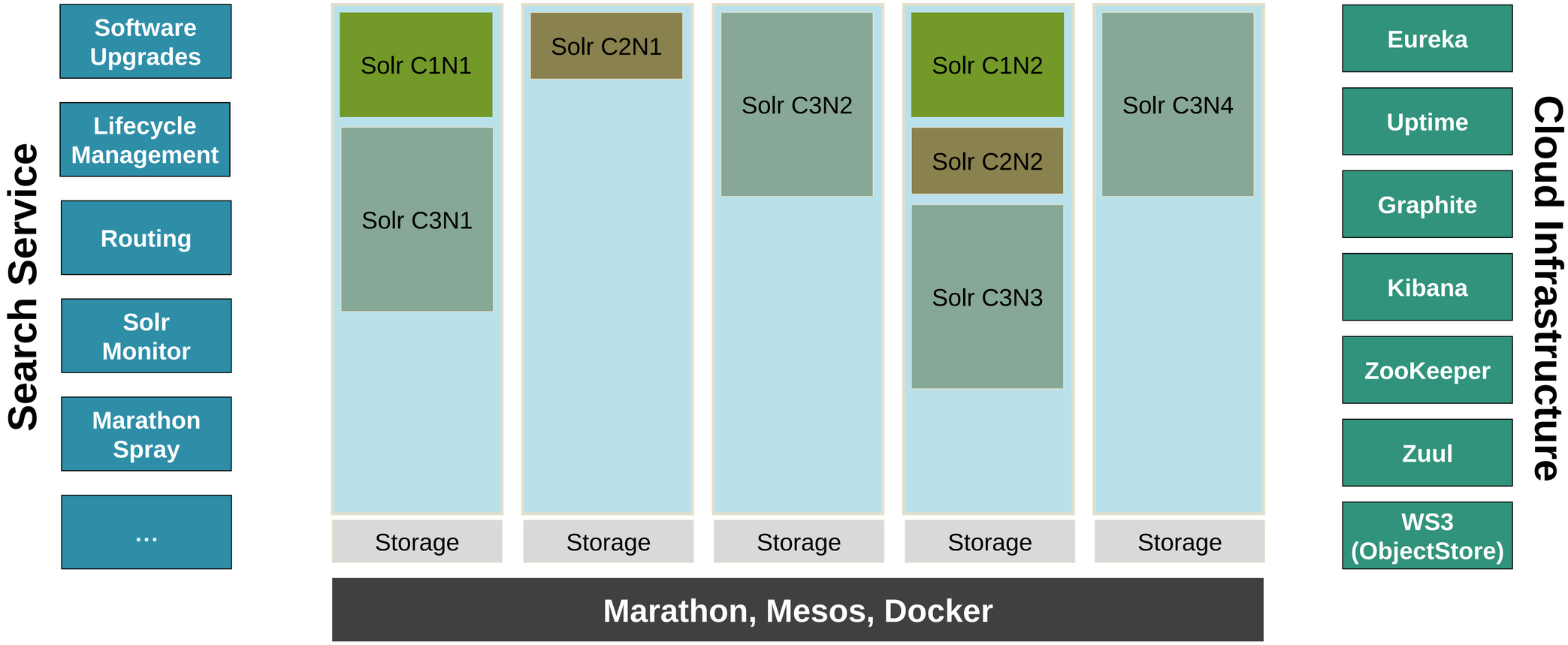
Multi-Tenancy

- A cluster per tenant
- Each cluster is isolated from other clusters
  - Resources
  - Collections
  - Configurations
  - ZK chroot
  - Different Solr versions…
- Every tenant can create multiple Solr cluster instances
  - Department indexes, dev/staging/production …

SolrCloud 101

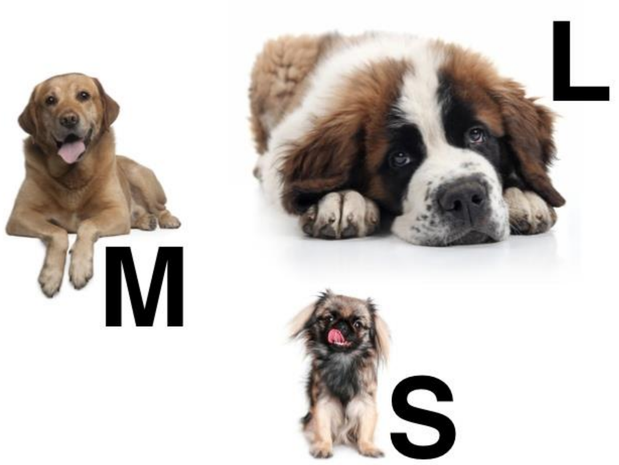


Architecture



Sizing Your Cluster

- A Solr cluster's size is measured in units
  - Each unit translates to **memory**, **storage** and **CPU** resources
  - A size-7 cluster has 7X more resources than a size-1
  - All collections have the same number of shards and a replicationFactor of 2
- Bigger clusters also mean sharding and more Solr nodes
  - Cluster sizes are divided into (conceptual) tiers
  - $Tier_1$ = 1 shard, 2 nodes
  - $Tier_2$ = 2 shards, 4 nodes
  - $Tier_n$ = 2^{n-1} shards, 2^n nodes
- Example, a size-16 ($Tier_3$) cluster has
  - 4 shards, 2 replicas each, 8 nodes
  - Total 32 cores
  - Total 64 GB (effective) memory
  - Total 512 GB (effective) storage



Software Upgrades

- Need to upgrade Solr version, but also own code
- Software upgrade means a full Docker image upgrade (even if only replacing a single .jar)
- SSH and upgrade software **forbidden** (security)
- Important: no down-time
- **Data-replication Upgrade**
  - Replicate data to new nodes
  - Expensive: a lot of data is copied around
  - Useful when resizing a cluster, migrating data center etc.
- **In-place Upgrade**
  - Relies on Marathon's *pinning* of applications to host
  - Very fast: re-deploy a Marathon application + Solr restart; **No data replication**
  - The default upgrade mechanism, unless a data-replication is needed

Software Upgrades

Data-Replication

- Start with 2 containers on version X
- Create 2 **additional** containers on version Y
- Add replicas on new Solr nodes
- Re-assign shard leadership to new replicas
- Route traffic to the new nodes
- Delete old containers

In-Place

- Start with 2 containers on version X
- Update one container's Marathon application configuration to version Y
  - Marathon re-deploys the applications **on the same host**
- Wait for Solr to come up and report "healthy"
- Repeat with second container

Resize Your Cluster

- As your index grows, you will need to increase the available resources to your cluster
- Resizing a cluster means allocating bigger containers (RAM, CPU, Storage)
- A cluster resize behaves very similar to a **data-replication** upgrade
  - New containers with appropriate size are allocated and the data is replicated to them
- Resize across tiers is a bit different
  - More containers are allocated
  - Each new container is potentially smaller than the previous ones, but overall you have more resources
  - Simply replicating data isn’t possible – index may not fit in the new containers
  - Before the resize is carried on, shards are split
  - Each shard eventually lands on its own container

Collection Configuration Has Too Many Options

- Lock factory must stay “native”
- No messing with uLog
- Do not override dataDir!
- No XSLT
- Only Classic/Managed schema factory allowed
- No update listeners
- No custom replication handler
- No JMX

Replicas Housekeeping

- In some cases containers are re-spawned on a different host than where their data is located
- Missing replicas
  - Solr does not automatically add replicas to shards that do not meet their `replicationFactor`
  - **Add missing replicas to those shards**
- Dead replicas
  - Replicas are not automatically removed from CLUSTERSTATUS
  - **When a shard has enough ACTIVE replicas, delete those “dead” replicas**
- Extra replicas
  - Many replicas added to shards (“Stuck Overseer”)
  - Cluster re-balancing
  - **Delete “extra” replicas from most occupied nodes**

Cluster Balancing

- In some cases, Solr nodes may host more replicas than others
  - Cluster resize: shard splitting does not distribute all sub-shards' replicas across all nodes
  - Fill missing replicas: always aim to achieve HA
- Cluster balancing involves multiple operations
  - Find collections with replicas of more than one shard on same host
  - Find candidate nodes to host those replicas (least occupied nodes #replicas-wise)
  - Add additional replicas of those shards on those nodes
  - Invoke the "delete extra replicas" procedure to delete the replicas on the overbooked node

More Solr Challenges

- `CLOSE_WAIT` (SOLR-9290)
  - DOWN replicas
  - `<int name="maxUpdateConnections">10000</int>`
  - `<int name="maxUpdateConnectionsPerHost">100</int>`

  ✓Fixed in 5.5.3
- “Stuck” Overseer
  - Various tasks accumulated in Overseer queue
  - Cluster is unable to get to a healthy state (missing replicas)

  ✓Many Overseer changes in recent releases + CLOSE_WAIT fix

More Solr Challenges

- Admin APIs are too powerful (and irrelevant)
  - Users need not worry about Solr cluster deployment aspects

  ✓Block most admin APIs (shard split, leaders handling, replicas management, roles…)

  ✓Create collection with minimum set of parameters: configuration and collection names

- Collection Configuration API
  - Users do not have access to ZK

  ✓API to manage a collection’s configuration in ZK

Running a Marathon (successfully!)

- Each Solr instance is deployed as a Marathon **application**
  - Needed for pinning an instance to an agent/host
- Marathon's performance drops **substantially** when managing thousands of applications
  - Communication errors, timeouts
  - Simple tasks take **minutes** to complete
- Marathon Sprayer
  - Manage multiple Marathon clusters (but same Mesos cluster)
  - Track which Marathon hosts a Solr cluster's applications
- Think positive: errors and timeouts don't necessarily mean failure!

Current Status

- Two years in production, currently running Solr 5.5.3
- Usage / Capacity
  - 450 Baremetal servers
  - 3000+ Solr clusters
  - 6000+ Solr nodes
- 300,000+ API calls per day
- 99.5% uptime

Questions?