Multi Tenant HBase

Introducing Use cases and available solutions

Bhupendra Kumar Jain
bhupendra.jain@huawei.com

$ whoami

Senior System Architect @ Huawei India

Tech Lead for Huawei HBase, Zookeeper & Hadoop

~13 years of experience in Big Data and BI Domain

Agenda

Multi Tenancy

- What it is and Why
- Use cases
- Goals & Challenges

Achieving Multi Tenancy

- Single HBase cluster
- Tenant specific cluster
  - HBase on YARN using Apache Slider

Future Work

Agenda

Multi Tenancy

- What it is and Why
- Use cases
- Goals & Challenges

Achieving Multi Tenancy

- Single HBase cluster
- Tenant specific cluster
  - HBase on YARN using Apache Slider

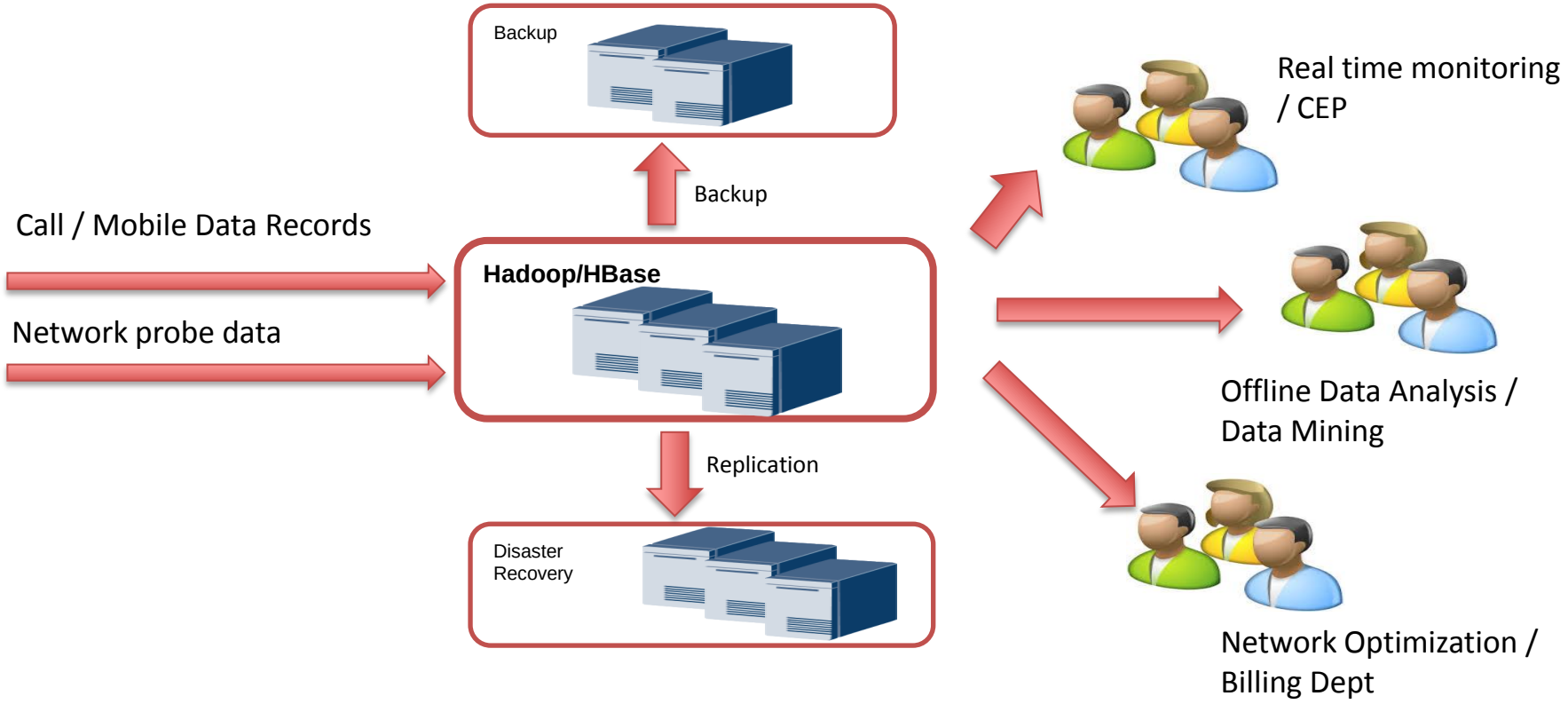
Future Work

Tenants

Tenant can be thought as

- a single role, a single user, administrator, super user, …
- a group of users (a department in an organization …)
  - Share some **common objective**
  - Work on the same set of data
  - **Hierarchy** within tenant is possible
  - developers, analysts, data scientists etc.

Use Cases


Backup
Backup
Call / Mobile Data Records
Hadoop/HBase
Network probe data
Replication
Disaster Recovery
Real time monitoring / CEP
Offline Data Analysis / Data Mining
Network Optimization / Billing Dept

Use Cases

Real time monitoring / CEP

- **faster** data processing
- **high priority** over other request
- mostly access latest data (**get** / short scan)

Offline Data Analysis /
Data Mining

- hourly / daily / monthly data processing
- multiple **longer scans**
- medium priority

Network Optimization/
Billing Department

- require **service only for shorter duration**
- need High load and Scan performance
- **resource** required is **dynamic** based on data load

Admin

- **high priority** over all other requests
- **security**, data access control
- user management

Business Goals

For each tenant / tenant group

- **Store** the data
- Allow to **access** the data
- Allow to **process** the data
- Do everything in **fully Secure manner**
- **Guarantee** a certain portion of cluster
- Data **Governance**

Challenges

- **Security**
  - Data should not be accessible by other tenant groups
  - Within tenant group, different level of data access
- **Performance**
  - Isolate tenants workload – no impact on other tenant workload
  - Resource guarantee
- **Cost efficiency**
  - Effective resource utilization
- **Maintenance effort**
  - Support tenant specific configurations
  - Different version needs
  - Monitoring workload
  - Dynamic scaling / resizing
- **Priority**
  - Resource fairness
  - Admin / Super user requests

Agenda

Multi Tenancy

- What it is and Why
- Use cases
- Goals & Challenges

Achieving Multi Tenancy

- Single HBase cluster
- Tenant specific cluster
  - HBase on YARN using Apache Slider

Future Work

Single HBase cluster for all

- **Namespaces, Quota (HBASE-8015, HBASE-8410)**

  A namespace is a logical grouping of tables analogous to a database in relation database systems.

  - ✓ Each tenant has their own workspace to deal with.
  - ✓ Permissions at Namespace level.
  - ✓ Quota to restrict the Number of tables and regions.
  - ✓ HDFS Space Quota for Namespace / table

- **RPC Throttling (HBASE-11598)**

  Provides a mechanism to control

  - The Number of request /time for a user / table
  - The bandwidth consumed by a user / per table

  - ✓ Can throttle the analytical user workload
  - ✓ Un-Throttle the real time user workload

Single HBase cluster for all (contd …)

- **Multiple RPC Queues (HBASE-10993, HBASE-10994)**
  - Different RPC Queues for Replication, Read / Write request, Meta request
  - Give priority to Get request over Long running scan request

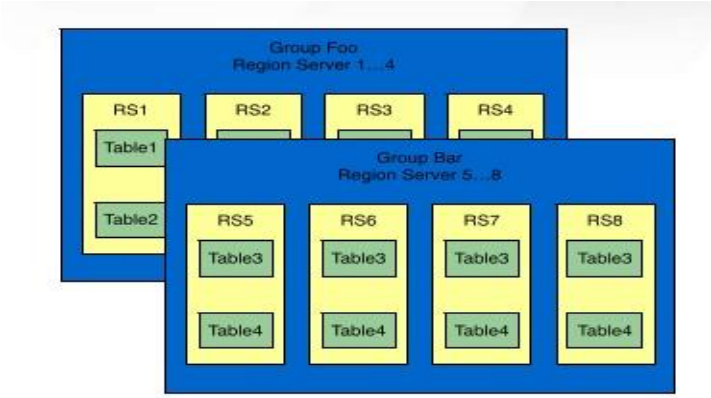
    ✓ longer running scans from analytical tenant gets lower priority
    ✓ Real time tenant's Get request serves better.
    ✓ Admin / Super user request gets higher priority.

- **Security**
  - Authentication
  - Authorization: ACLs at all levels (Namespace, table, cf, cq, cell)

    ✓ tenant have access to only his required data.
    ✓ restrictive permissions for shared data. (admin-write, others-read)

Single HBase cluster for all (contd …)

- **Region Sever Grouping (HBASE-6721)**
  - Logical grouping of RS.
  - One RS group handles regions of one or more similar tenant.
  - Namespace /Table level group binding
    - ✓ Complete tenant workload Isolation
    - ✓ Better Qos for each tenant
    - ✓ Configure group as per tenant work load



- **FileSystem Quotas (HBASE-16961)**
  - track how HBase namespaces and tables use filesystem space
  - impose limitations via centralized policies (disable table if violation …)
  - includes space used by archive, snapshot etc. (future …)
    - ✓ Better File system space management at HBase level
    - ✓ Not limited to any file system

Single HBase cluster for all (contd …)

Is this sufficient ??

- Security - **Yes**
- Resource Isolation
  - Disk space: Space Quota
  - Disk I/O: Not available
  - CPU: Yes with Region grouping
  - Memory: Yes with Region grouping
  - Network I/O : Yes with RPC throttling
- Workload isolation ? – **Yes with RS Group**
- Too many tenants and regions ? – **Single Master may not handle**
- Tenant specific configurations ? – **Yes with RS Group**
- Need for different versions ? – **Not possible**
- Short running services ? – **Not possible**
- Priority – **User based priority is not available**
- Effective resource utilization ? – **Not completely**

Agenda

Multi Tenancy

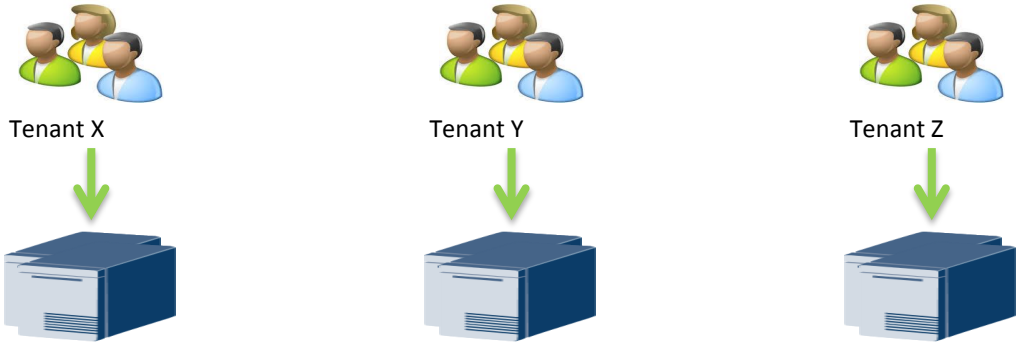
- What it is and Why
- Use cases
- Goals & Challenges

Achieving Multi Tenancy

- Single HBase cluster
- Tenant specific cluster
  - HBase on YARN using Apache Slider

Future Work

Tenant have their own cluster



Let each tenant have their own cluster for HBase and share HDFS, YARN, ZK.

- ✓ Complete Isolation of workload
- ✓ Tenant specific configurations, Versions
- ✓ Short running services
- ✓ Resource Isolation

Tenant have their own cluster

Is this sufficient ??

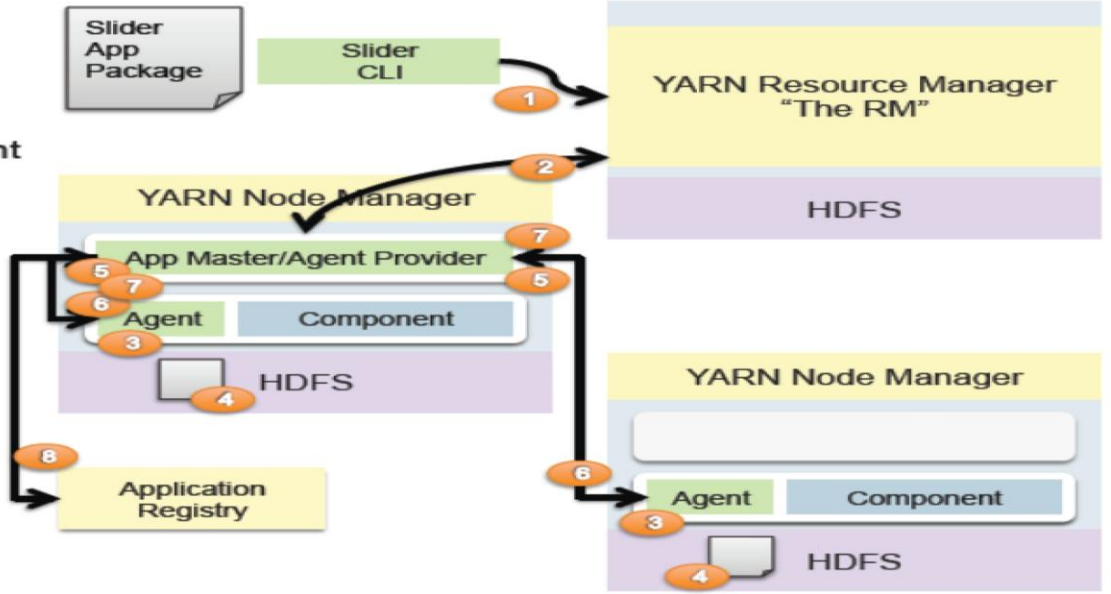
- Maintenance ? – **Very High** (**Need to manage many cluster now)**
- Effective resource utilization ? – **No, Free resources not used by others**
- Tenant workload isolation – **Possible**
- Isolate the HBase service from other running services ? **-- Custom solution** (using cgroup or others)
- Cost ? – **Very high**
- Tenant specific configurations, Versions – **Easy**
- Shared data ? – **Need to replicate**

HBase on YARN using Apache Slider

Slider is a YARN application to deploy non-YARN-enabled applications in a YARN cluster

Similar to any YARN application

1. CLI starts an instance of the AM
2. AM requests containers
3. Containers activate with an Agent
4. Agent gets application definition
5. Agent registers with AM
6. AM issues commands
7. Agent reports back, status, configuration, etc.
8. AM publishes endpoints, configurations



Slider client: Communicates with YARN and the Slider AM via remote procedure calls and/or REST requests
Slider AM: Application master to manage containers(actual application)
Slider Agent: Communicate with AM and mange one container instance.
Component : HBase application component. [HM,RS etc.]

https://slider.incubator.apache.org/
http://events.linuxfoundation.org/sites/events/files/slides/apachecon-slider-2015.pdf

HBase on YARN using Apache Slider

Feature	Remarks
Ease of Deployment	install, start, stop, configure, rolling upgrade
Container allocation and placement	anti-affinity feature (do not place containers together in same node) - Work in progress.
Dynamic Resizing	Increase / Decrease containers at run time
Data Locality	Best effort to restarts failed RS in same node
Resource Allocation and Control	Same capability as YARN.
Fault Tolerance	Slider AM Failure recovery Container (HM/RS) failure recovery
Maintenance	Supports Log aggregation Provides the API to monitor the running containers status
Integration	RPC / REST API (Work in progress)

Agenda

Multi Tenancy

- What it is and Why
- Use cases
- Goals & Challenges

Achieving Multi Tenancy

- Single HBase cluster
- Tenant specific cluster
  - HBase on YARN using Apache Slider

Future Work

Summary

Need	Single Cluster	Tenant specific cluster using slider
Security	Yes	Yes
Resource Isolation	Yes	Yes
Effective resource utilization	Partial	Better comparatively
Cost	Optimal	High comparatively
Tenant specific conf / version	Partial	Yes
Ease of Maintenance	Easy	Requires High
Priority	Partial	Partial
Dynamic resizing	Not Easy	Easy
Short running services	No	Yes

- Mix of both the solution may be needed based on the use cases

Future Work

- **Quota Sharing :** Share the quota with other tenant if possible
- **Effective Resource utilization:** Share the Region server group with other tenant
- **HMaster Federation** : For large number of regions, Distribute heavy workload from Master to other nodes
- **Region Labels for Heterogenous cluster** : Tenant can choose to assign regions to particular Region Server with label (i.e. SSD, HighMemory ..)
- **System table partitioning:** Region group level System Table partitions
- **Disk I/O control :** File system level throttling for disk I/O control (for large compaction …)
- **Separate Block cache for each tenant :** Each tenant have their own share for block cache
- **RPC priority based on tenant:** Priority user's request always can get high priority then others.

Thank You !

mail to: bhupendra.jain@huawei.com