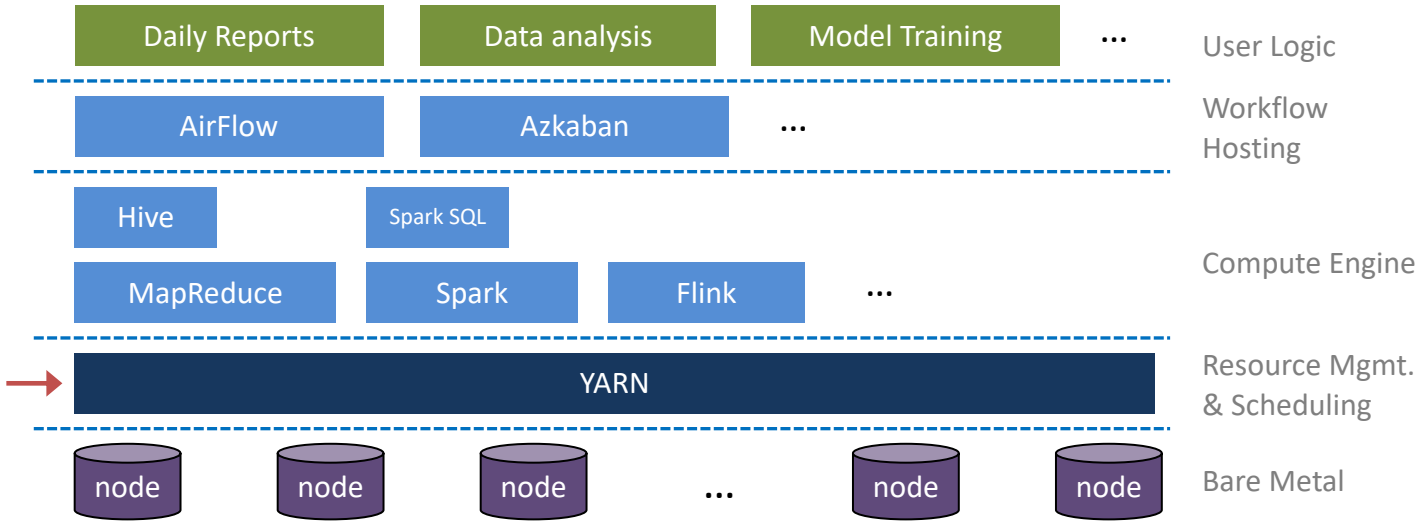
Key Customizations of YARN@ByteDance

李亚坤
liyakun.hit@bytedance.com
字节跳动基础架构工程师

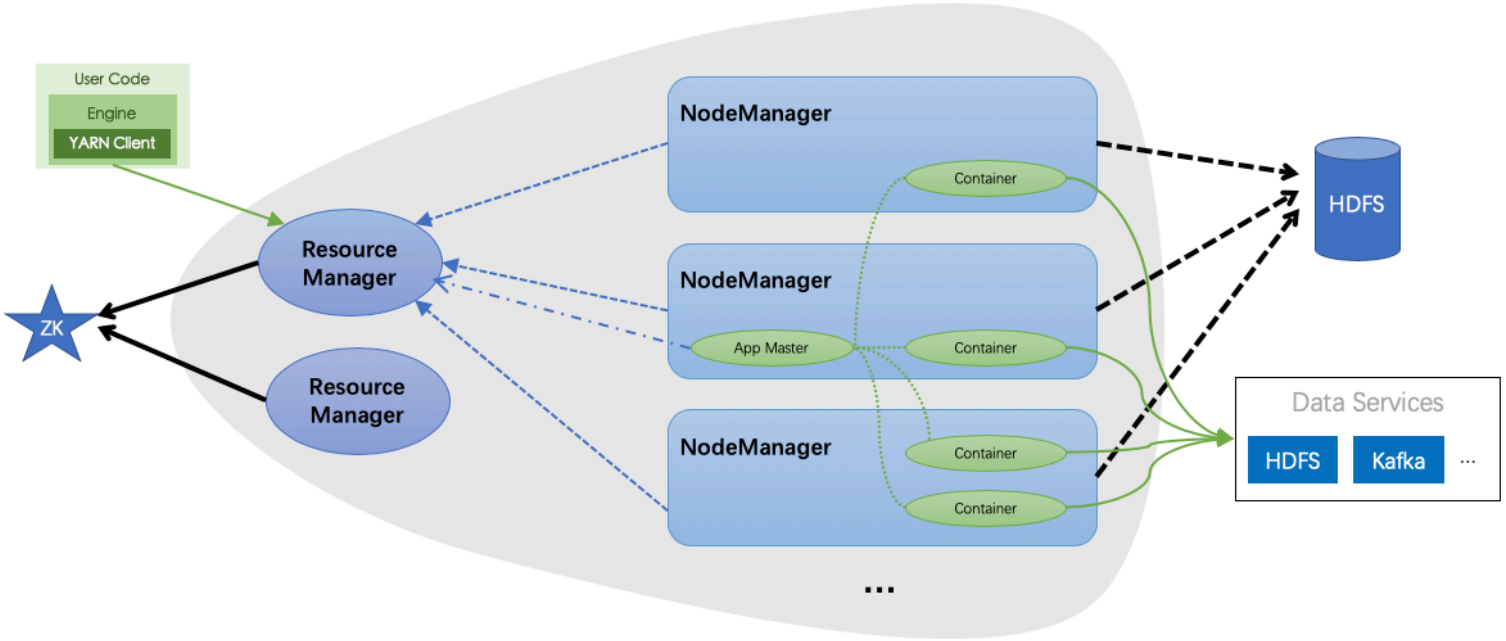
OUTLINE

1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Introduction to YARN
  - YARN @ Hadoop Ecosystem



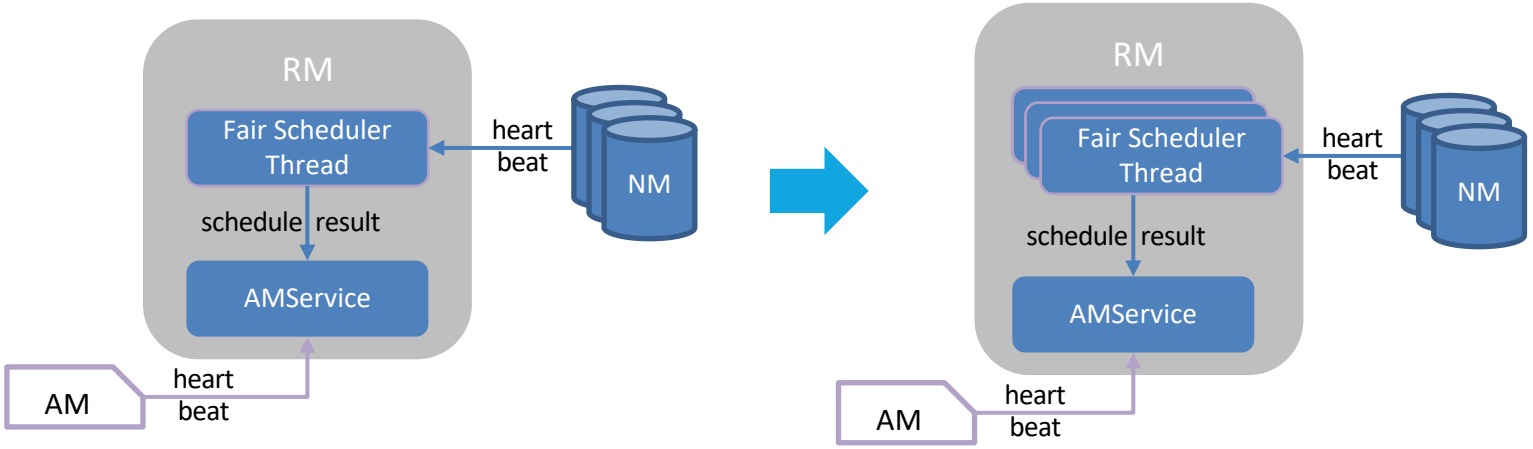
- Introduction to YARN
  - YARN Architecture



OUTLINE

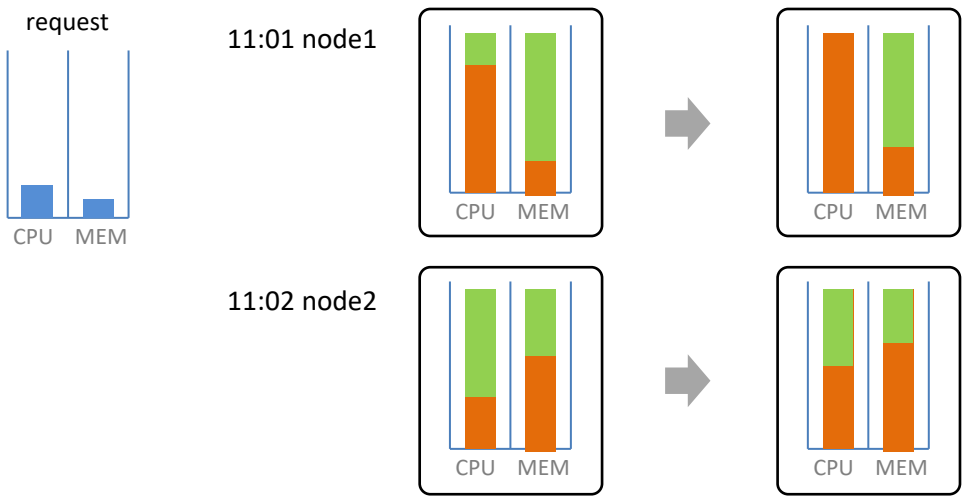
1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Utilization Opt.
  - Quota Utilization Opt.
    - **Multithreading** Version of Fair Scheduler



Performance : container throughput rate 3K / sec

- Utilization Opt.
  - Quota Utilization Opt.
    - Node **DRF waiting** to Reduce Fragmentation



Performance : 24h avg quota utilization ~90% / ~95%

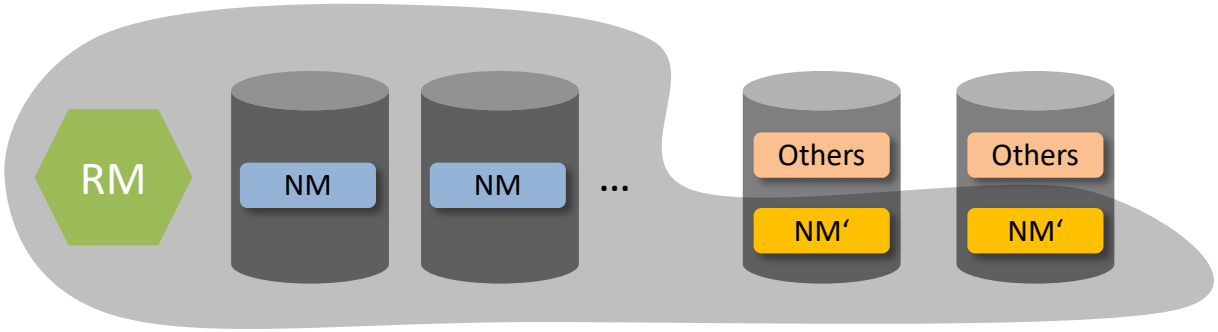
- Utilization Opt.
  - Quota Utilization Opt.
    - **Scale Out** a Cluster

- **Remove unnecessary events** to avoid failover avalanche

- heartbeat **back pressure**

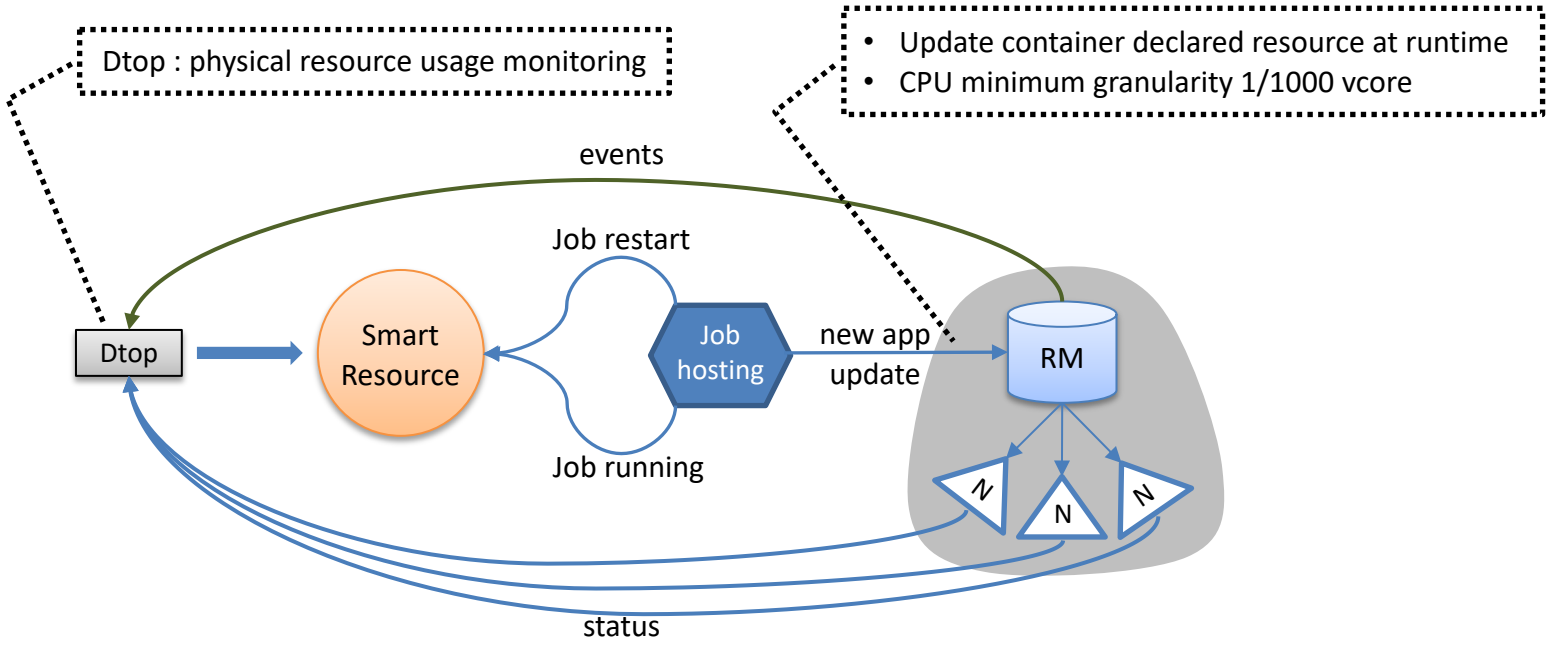
Performance : 10k nodes per cluster

- Utilization Opt.
  - Physical Utilization Opt.
    - Gemini: **Co-location** with Streaming Job & Online Service



Performance : 20%+ absolute CPU avg utilization increase

- Utilization Opt.
  - Physical Utilization Opt.
    - **Smart Resource** : Adjust Resource at Runtime / Restarting



OUTLINE

1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Multi Workloads Opt.
  - For Streaming/Training Workload

YARN Gang Scheduler

- All or nothing *s*emantic for a request
- Schedule for application instead of node
    - Low latency (RT in milliseconds per request)
    - Global view
        - Hard constraints
            - eg. *Attributes*, *Load Avg* ..
        - Soft constraints (with priority & weight)
            - eg. *Attributes*, *Load Avg*, *Container Decentralize*, *Quota Avg*, *GPU Affinity*, …

- Multi Workloads Opt.
  - For Streaming/Training Workload

More CPU Usage Strategies

- Expose min-max ratio at share mode

- CPUSET mode support

- NUMA mode support

- Multi Workloads Opt.
  - For Training Workload

Training Customizations

- Docker supporting *GPU* and *Ceph*

- Resource Value Range

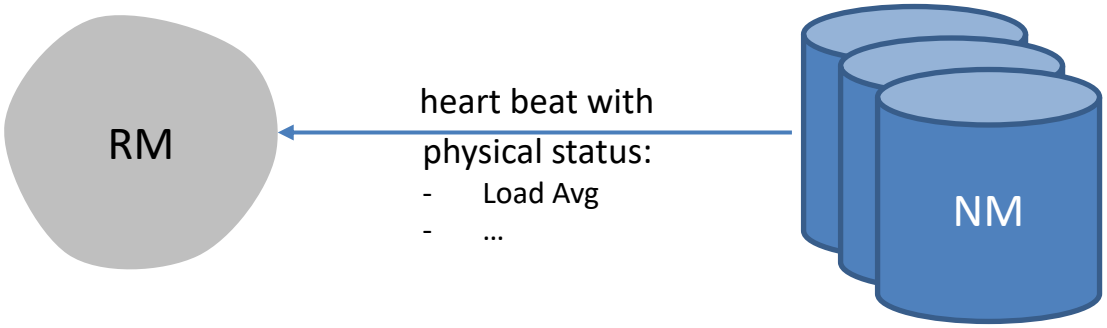
  - *GPU* as a Value Range Resource

  - *Port* as a Value Range Resource

- Node Attributes for *GPU*/*CPU* with different roles

- Multi Workloads Opt.
  - For Batch Workload

Temporarily skip nodes with high physical utilization



Performance : ~40% fetch failed rate decrease

OUTLINE

1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Stability Opt.
  - Make HDFS as a weak dependence

    - *Node Label* stored into ZKRMStateStore

    - *Container logs* upload asynchronously

    - Initialize the *container log directory* asynchronously

  - Container rating and eviction

    - *Disk capacity* usage

    - *Load avg* contribution

  - Unmanaged container cleanup mechanism

OUTLINE

1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Multi Datacenter
  - **Unified** YARN client for all clusters
    - Specify a cluster by *environment variable*

```
$ export YARN_CLUSTER_NAME=c1
$ ./spark-shell --master yarn --queue root.q1
```

    - Specify a cluster by *configuration*

```
$ ./spark-shell --master yarn --queue root.q1 \
  --conf spark.hadoop.yarn.cluster.name=c1
```

  - **Unified** YARN UI for all clusters
  - **Relax** locality for non-owned resource at beginning
  - YARN safemode

OUTLINE

1. Introduction to YARN
2. Key Customizations@ByteDance
   A. Utilization Opt. (Quota / Physical)
   B. Multi Workloads Opt. (Batch / Streaming / Training)
   C. Stability Opt.
   D. Multi Datacenter
3. Future Works

- Future Works
  - Co-location with Streaming & Service

    - Physical utilization increase

    - Better isolation

    - More controllable container kill rate

    - GPU resource co-location

  - YARN Gang Scheduler

    - Richer scheduling predicates

    - Lower latency

Acknowledgement to Team

李亚坤_好吧

北京 海淀

扫一扫上面的二维码图案，加我微信

欢迎加入我们，
与优秀的人一起做有挑战的事情.

邮箱: liyakun.hit@bytedance.com