BESIII physical offline data analysis on virtualization platform

Qiulan Huang
huangql@ihep.ac.cn
Computing Center, IHEP, CAS
CHEP 2015
Outline

• Overview of HEP computing in IHEP
• What is virtualized computing cluster?
• Why virtualized computing cluster?
• What we have done?
  • Schedule BESIII jobs to virtual computing cluster
• Current status
• Conclusion
HEP computing in IHEP

- Support several experiments
  - BEPCII & BESIII
  - Cosmic Ray/Astrophysics in Tibet
  - DayaBay
  - CMS, ATLAS experiments on LHC
  - Accelerator-driven Sub-critical System (ADS)
  - China Spallation Neutron Source (CSNS)
- Future experiments
  - Jiangmen Underground Neutrino Observatory: $\sim500\text{TB} \times 10\text{years}$
  - Lhaaoso: 2PB per year after 2017, accumulate 20PB+ in 10 years
Computing status in IHEP

• CPU cores
  • ~ 12000
  • ~ 60 queues, managed by Torque

• Problems
  • Low resource utilization
  • Poor resource sharing
What is virtualized computing cluster?

- **KVM+OpenStack+Torque /maui**: Integrated OpenStack and Torque /maui to provide computing service based on IHEPCloud
- Virtual cluster and physical cluster work together
- When a job queue is busy, the jobs can be allocated to a virtual queue
- VMs are created according to application requirement

![Diagram](image)

- **Distributed computing User interface**
  - Send results
  - Submit jobs

- **Resource Management**
  - CloudAPI (create/stop VMs)
  - Query load
  - Schedule to virtual Queue

- **Unified deployment (puppet)**
- **IHEPCloud**
- **Monitoring**
Why?

• Advantages
  • Improve resource utilization
  • Improve the efficiency of resource scheduling
  • Simplify management
  • Elasticity
  • Resources heterogeneous is transparent to applications and users
  • Energy saving

• To solve problems
  • The overall computing resources utilization rate is relative low
    ◦ IHEP Computing cluster supports various experiments such as BES, Daya Bay, YBJ…
    ◦ Computing resource is separated by each experiment which cannot be shared.
    ◦ At a certain period of time, some subsets are busy, some subsets are idle. It leads to take much time to queue.
What we have done?
BESIII Offline software optimization

- BESIII analysis is I/O heavy
- Creating event metadata and doing pre-selection by metadata according to event property to reduce IO throughput significantly

Detailed: see xiaofeng's talk, track2, 12:00 16/4 (BESIII Physics Data Storing and Processing on HBase and MapReduce)
Optimized KVM performance

- Benchmark testing by default
  - CPU performance penalty of KVM is about 10% and IO is about 12%
  - Network performance penalty is about 3%
- CPU affinity
  - The process is bound to a specific CPU but not allowed to dispatch to the other ones
  - No need to migrate process between processors frequently to improve cache hit rate
- Extended Page Table
  - Disabled EPT: modprobe kvm-intel enable_ept=0
Optimized CPU performance

- CPU benchmark testing
  - Specifications
    - Intel(R) Xeon(R) CPU X5650 (2.67GHz), 8 CPU cores, 24GB memory
    - OS: SLC release 5.5 (Boron) 2.6.18-194.11.3.el5.cve20103081, 64bit KVM-83
  - Tools
    - HEP-SPEC06
  - Optimized CPU performance increased about 3%

- Tools
  - HEP-SPEC06

- Optimized CPU performance increased about 3%
BES simulation job (BOSS 6.6.0)
- event number = 1000

Test environment
- VM: 2 cores, 2GB memory
- Physics machine: 8 cores, 16GB memory

Test results
- Jobs running time in VM is 1:45:05 while in physical machine is 1:42:04 which indicates the performance penalty is about 2.9%
- The penalty is optimistic
BESIII jobs running in VMs(2)

- **BES analysis job ( BOSS 6.6.2 )**
  - `ApplicationMgrEvtMax = 1E9`

```cpp
"/besfs2/offline/data/663-1/jpsi/tmp2/120520/run_0028145_All_file006_SFO-1.dst",
"/besfs2/offline/data/663-1/jpsi/tmp2/120520/run_0028145_All_file006_SFO-2.dst"
```

// Set output level threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL)
MessageSvc.OutputLevel = 6;

// Number of events to be processed (default is 10)
ApplicationMgrEvtMax = 1E9;

ApplicationMgr.HistogramPersistency = "ROOT";

- **Testing specification**
  - **VM:** 1 CPU cores, 2GB memory
  - **Physical machine:** 8 CPU cores, 16GB memory

- **Testing results**
  - Job running time in VM is 8:04:47 while in physical machine is 7:48:32
  - It takes more 975s in VM than in physical machine which illustrated the performance penalty is about **3%**
IHEPCloud

- Launched in May 2014.
- A private IaaS platform aiming to provide a self-service cloud platform for users and IHEP scientific computing
- Open for any user who has IHEP email account (>1000 users, >70 active users)
CloudScheduler

- Integrated virtual computing cluster into the traditional physical cluster to optimize the resource utilization.
- Take fine-grained resource allocation to schedule tasks instead of taking nodes.
- Design flexible allocating policy to provisioning VMs dynamically, considering job types, system load and cluster real-time status.
- Schedule jobs to IHEPCloud
Architecture of Cloudscheduler

- **PBS (VM Queue)**
  - Expends the original Torque PBS to support vm queue.

- **VM central controller**
  - A matcher between the various modules.
  - Polling → Calculate → Publish

- **VM job controller**
  - Provide job query service
  - schedule the jobs running in virtual queue (record the jobs in database)

- **VM resource controller**
  - Policymaker: make vm allocation strategy.
  - VM controller: start or stop vm
  - CloudAPI: a packaged module based on the openstack api and with some extension.

- **Job agent (deployed in VMs)**
  - Pull jobs to run, return job exit status and transfer the output files
Workflow

PBS

VM central controller

Request: ask running number for each queue
Reply: job running number of each queue
Reply: The maximum job number for queue

VM Job Controller

Request: queue name
Reply: job run and queue number of queue, VM ip, addr scheduled jobs
Request: VM type, total VM num and VM ip list scheduled jobs
Reply: the num of active vm

VM Resource Controller

Job Agent

check the vm queue and resource status

start/stop vm

active vm number
Push+Pull mode

- Pull and push mode
  - Pull: allocate the cpu/core for BESIII jobs with suitable resource
    - When new job is coming, the PBS will request VM central controller to get vm resource.
    - VM central controller prepares corresponding resource for the job
    - VMs request matched jobs
  - Push: Cluster internal keeps the original way of "push", schedule job into the virtualized queue.
    - Be transparent to users
VM allocating policy

• Allocating VMs dynamically
  • Provisioning VM number is determined by current virtual cluster status
  • How to allocate VMs according to load of cluster especially considering the information from monitoring system

• Configurable VM allocating strategy interface.
  • Linear addition and subtraction.
  • And so on…
**Current status**

- Completed the testbed
- Submitted hundreds of test jobs
- Still has some problems
  - Sometimes message communicated between modules lost
  - Jobagent service go to offline when not connect to the server side
- **Next steps:**
  - Fix bugs
  - Implement more vm allocation strategies
  - Applied to other experiments like JUNO, DAYABAY, YBJ and so on.
  - Provide online service in this year.
Summary

• Creating event metadata can reduce the IO throughput significantly
• CPU and network performance of KVM is optimistic, which can meet the BESIII experiment’s requirement
• Virtual computing cluster is a good supplement for the existing physical cluster
Any Question?