Testing SLURM open source batch system for a Tier1/Tier2 HEP computing facility

DONVITO GIACINTO (INFN-BARI)
ALESSANDRO ITALIANO (INFN-BARI)
DAVIDE SALOMONI (INFN-CNAF)

Outline

- Why we need a “new” batch system
  - INFN-Bari use case
- What do we want from a batch system?
- SLURM short overview
- SLURM functionalities test
  - … fail-tolerance considerations
  - … pros & cons
- SLURM performance test
- CREAM support to SLURM
- Future Works
- Conclusions

Why we need a “new” batch system

- Multi-Core CPU are putting pressure on batch system as it is becoming quite common to have computing farms with O(1000) CPU/cores
- Torque/MAUI is a common and easy-to-use solution for small farms
  - It is open source and free
  - Good documentation
  - and wide user base
- …but it could start suffering as soon as the farm becomes larger
  - in terms of Cores
  - and of WN
  - … but especially in terms of users

Why we need a "new" batch system:
INFN-Bari use case

- We started with few WN in 2004 and constantly growing
  - we now have about:
    - 5000 CORES
    - 250 WNs
- We have Torque 2.5.x + MAUI:
  - We see a few problem with this setup:
    - "Standard" MAUI supports up-to ~4000 queued jobs
      - All the "others" jobs are not considered in the scheduling
    - We modified the MAUI code to support up to 18000 queued jobs and now it works
      - … but it often saturates the CPU where it is running and soon it becomes un-responsive to client interaction

Why we need a “new” batch system: INFN-Bari use case (2)

- Torque is suffering from memory leak:
  - It usually use ~2GB of memory under stress condition
  - We need to restart it from time to time
- Network connectivity problems to a few nodes could affect the whole Torque cluster

- We need a more reliable and scalable batch system and (possibly) … open source and with a low TCO

What we need from a batch system

- Scalability:
  - How it deals with the increasing number of Cores, jobs submitted and users …
- Reliability and Fault-tolerance
  - HighAvailability features, client behavior in case of service failures
- Scheduling functionalities:
  - The INFN-Bari site is a mixed site, both grid and local users share the same resources
    - We need complex scheduling rules and full set of scheduling capabilities
- Low TCO
- Grid enabled

SLURM short overview

- OpenSource (https://computing.llnl.gov/linux/slurm/)
- Used by many of the TOP500 super-computing centers
- Documentation states that:
  - It supports up to 65'000 WNs
  - 120'000 jobs/hour sustained
  - High Availability features
  - Accounting on Relational DataBase
  - Powerful scheduling functionalities
  - Lightweight
  - It is possible to use MAUI/MOAB or LSF as scheduler on top of SLURM

SLURM functionalities test

- Functionalities tested:
  - QoS
  - Hierarchical Fair-share
  - Priorities on users/queue/group etc.
  - Different pre-emption policies
  - Client resilience on temporary failures
    - The client catchs the error and retries automatically after a while
  - The server could be configured with HighAvailability configuration
    - This is not so easy to configure
    - It is based on “events”
  - The accounting information stored on MySQL/PostgreSQL DB
    - This is also the only way to configure the Fair-Share

SLURM functionalities test (2)

- Functionalities tested:
  - Age based priority
  - Support for Cgroup for limiting the usage of resources on the WN
  - Support for *pluggable* "consumable resources" scheduling
  - "Network topology" aware scheduling
  - Job suspend and resume
  - Different kind of jobs tested:
    - MPI jobs
    - "Whole node" jobs
    - Multi-threaded jobs
  - Limits on amount of resources usable at a given time for:
    - Users, groups, etc.
    - It is possible to limit also the number of submitted jobs (Queued)

SLURM functionalities test (3)

- Functionalities tested:
  - Computing resources could be associated to:
    - Users, group, queue, etc
  - ACL on queues, or on each of the associated nodes
  - Job Size scheduling (Large MPI Jobs first or small jobs first)
  - It is possible to submit executable directly from CLI instead of writing a script and submitting it
  - The jobs lands on the WN exactly in the same directory where the user was when it is submitting the jobs
  - Triggers on events
  - Any batch job running on a failed node will be re-queued for execution on different nodes
  - Security can be managed using well-known “munge” server

SLURM functionalities test (4)

- Functionalities tested:
  - Job Memory Limit tested -> OK
    - If the job uses more memory than it was configured it is killed.
  - It is possible to use interactive jobs
    - Also forwarding the X display
      - srun.X11
  - Adding or deleting a node, is quite easy:
    - Change the configuration file and run: "*scontrol reconfigure*"
  - The behaviour in case of failure of the pre-exec, is different from what available in Torque or LSF
    - The job after few attempt is cancelled from the queue
    - We proposed a patch to the code and the community accepted it…
    - …since SLURM 2.5 a failure in the pre-exec leads to re-enqueue the job

SLURM results: cons

- Configuring complex scheduling policy is quite complex and requires a good knowledge of the system
  - Documentation could be improved with more advanced and complete examples
  - There are only few source of information apart from the official site
- There is no possibility to transport output/error files after job execution back to the submitter users/host
  - SLURM assumes you have a shared file-system among WNs and “frontends”

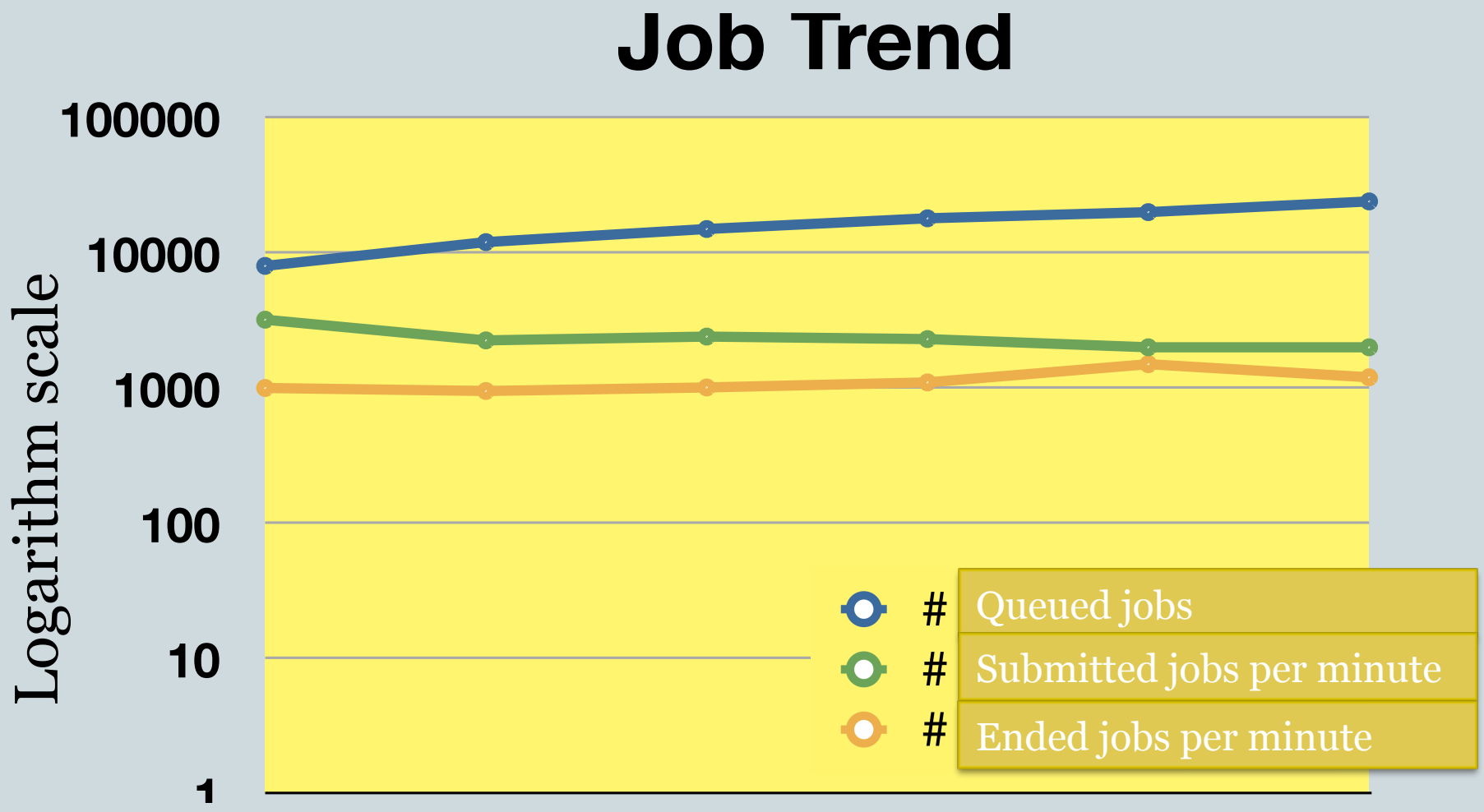
Performance test: description

- We have tested the SLURM batch system in different stressing conditions:
  - High amount of jobs in queue
  - Fairly high number of WNs
  - High number of concurrent submitting users
  - Huge amount of jobs submitted in a small time interval
  - Long run & Stess Test
- The accounting on the MySQL databases is always enabled

Performance test: description (2)

- High number of jobs in the queue:
  - One single client is constantly submitting jobs to the server for more than 24 hours
  - The jobs are fairly long…
  - … so the number of jobs in the queue are increasing constantly
  - We measured:
    - the number of queued jobs
    - the number of submitted job per minutes
    - the number of ended jobs per minutes
- The goal is to prove:
  - the reliability of the system under high load
  - the ability to cope with the huge amount of jobs in the queue keeping the number of executed and submitted job as constant as possible

Performance test: results (1)



Performance test: results (2)

- The test was measured up to 25kjobs in queue
- No problems registered
  - The server was always responsive and the
  - usage is as low as ~200MB
  - The submission rate is decreasing slowly and gracefully
  - … the number of executed jobs is not decreasing
    - This means that the jobs scheduling on the nodes is not suffering
  - We were able to keep a scheduling period of 20 seconds without any problem
  - The loadaverage on the machine is stable at ~1
- TEST PASSED ☺

Performance test: description (3)

- High amount of WNs
- High number of concurrent clients submitting jobs:
- Huge number of jobs to processed a short period of time:
  - 250 WNs
    - ~6000 Cores
  - 10 concurrent client …
  - … each submitting 10'000 jobs
  - Up to 100'000 job to be processed

- The goal is to prove:
  - the reliability of the system under high load from the clients
  - The ability to deal with a huge pick of job submission
  - Managing a quite large farm

Performance test: results (3)

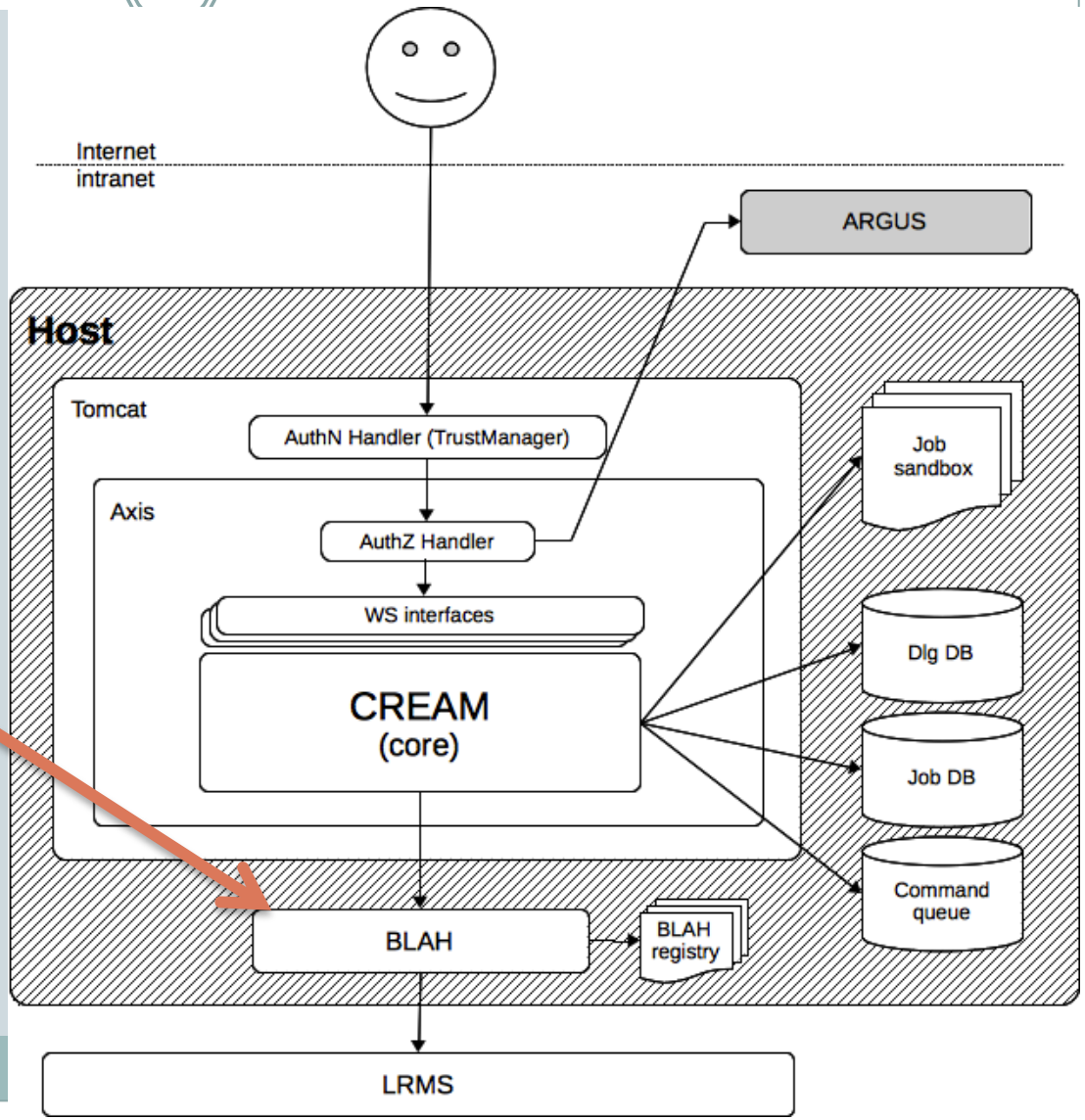
- The test was executed in about 3.5 hours
- No problems registered
  - The submission do not experienced problems
  - the memory used on the server always less than 500MB
  - The loadaverage on the machine is stable at ~1.20
  - At the beginning of the test the submission/execution rate is 5,5kjob per minute
  - During the pick of the load:
    - the rate of submission/execution is about 350 job/minute
  - It was evident that the bottleneck is on the single CPU/Core computing power
- TEST PASSED ☺

Stress test: results (3)

- 6000 Cores available
- 4 days of continuous job submission and execution with ~20kjob always in the queue:
  - No crush, no memory leak
  - Load under control (~1 Load average)
- TEST PASSED ☺

CREAM CE & SLURM

- Interaction with the underlying resource management system implemented via BLAH
- Already supported batch systems: LSF, Torque/PBS, Condor, SGE, BQS…
- … and SLURM…
  - since EMI 3



Status test Cream-CE

- Blah/job submission works -> ☺
- Infoproviders -> ☺
- Accounting (Apel) -> ☺

- Functionalities test are working fine

Conclusion

- SLURM is a fast and reliable batch system solution
- It is completely OpenSource and community driven
  - We already interacted successfully with the developers team proposing patch
- We have been able to implement all the needed configuration
  - Both coming from torque/maui and LSF experience

Work-in-progress

- Stress test on CREM-CE with SLURM
- Test the compatibility layer (torque-slurm)
  - In order to make the migration as easy as possible to the local users
- Test the implementation of SLURM on WNoDeS cloud solution
  - It exploit the same logic of LSF

Not only SLURM

- INFN-CNAF is testing also GridEngine:
  - Poster presentations / 369
  - Changing the batch system in a Tier 1 computing center: why and how