Evaluating scalability and efficiency of the Resource and Job Management System on large HPC Clusters

Yiannis Georgiou and Matthieu Hautreux

Plan

Plan

Introduction

Facts for large HPC supercomputers

- Increased number of resources
- Submission of large number of jobs
- Large network diameters and more complex designs

Introduction

Facts for large HPC supercomputers

- Increased number of resources
- Submission of large number of jobs
- Large network diameters and more complex designs

Resource and Job Management System(RJMS)

The goal of the RJMS is to satisfy users' demands for computation and assign user jobs upon the computational resources in an efficient manner.

Introduction

Facts for large HPC supercomputers

- Increased number of resources
- Submission of large number of jobs
- Large network diameters and more complex designs

Resource and Job Management System(RJMS)

The goal of the RJMS is to satisfy users' demands for computation and assign user jobs upon the computational resources in an efficient manner.

RJMS Issues for large HPC supercomputers

- Resources Scalability
- Large workload management
- Network topology aware placement efficiency

HPC supercomputers

Tera-100 supercomputer

- In production since November 2010
- 17th in June's 2012 Top500 list
- 1.25 Pflops theoretical

Curie supercomputer

- In production since March 2012
- 9th in June's 2012 Top500 list
- 1.6 Petaflops theoretical

Motivations

Evaluate the behaviour of SLURM regarding:

- Large workload management
- Network topology aware placement
- Scalability in terms of number of nodes to manage

Motivations

Evaluate the behaviour of SLURM regarding:

- Large workload management
- Network topology aware placement
- Scalability in terms of number of nodes to manage

Tune Tera100 & Curie configurations for best performances

- Core level allocation, fine topology description, backfill scheduler

Motivations

Evaluate the behaviour of SLURM regarding:

- Large workload management
- Network topology aware placement
- Scalability in terms of number of nodes to manage

Tune Tera100 & Curie configurations for best performances

- Core level allocation, fine topology description, backfill scheduler

Evaluate the behaviour of SLURM for larger machines

- Experimentations beyond the current scale of existing machines

Plan

Experimental Methodology

Method for Throughput Evaluations

- Tera100 nodes during maintenance periods
- Real-Scale experimentations
- Submission bursts generation scripts

Experimental Methodology

Method for Throughput Evaluations

- Tera100 nodes during maintenance periods
- Real-Scale experimentations
- Submission bursts generation scripts

Method for Efficiency and Scalability Evaluations

- Emulated experimentations through `multiple-slurmd` mode
  - Multiple virtual SLURM nodes per physical nodes
  - Enhanced to ease scalable deployement of simulated clusters
  - Patches accepted in SLURM main branch
- Tera100 nodes during maintenance periods
  - Automatic generation of simulated clusters configuration
  - Automatic start of all the SLURM daemons
  - Fully automated to enable reproduction of same experiments
- Automatic run of a single benchmark
  - Based on a known model: ESP
  - Variations of ESP used with different job counts and sizes

Experimental Methodology

Method for Throughput Evaluations

- Tera100 nodes during maintenance periods
- Real-Scale experimentations
- Submission bursts generation scripts

Method for Efficiency and Scalability Evaluations

- Emulated experimentations through `multiple-slurmd` mode
  - Mult
  - Enh ers
  - Patc
- Tera100 r
  - Auto
  - Auto
  - Fully ts
- Automati
  - Base
  - Varia

Emulation methodology validation

- The goal is to evaluate the behavior of SLURM scheduling and topology placement efficiency; these functions take place as internal mechanisms on the `slurmctld` controller deamon.
- Similar results of workloads execution with real and emulated scale experimantations.

RJMS evaluation with workload injection

ESP benchmark [a]

[a]http://www.nersc.gov/assets/RD/ESP/espsc00.pdf

- **quantitative evaluation** of launching and scheduling via a single metric: the smallest elapsed execution time of a representative workload
- Complete **independence** from the hardware performances
- automatic adaptation on cluster sizes
- parametrized Gaussian model of submission
- particular number of jobs derived from different classes with various characteristics

Specific Parameters

- No real computation (sleep jobs)

Proposed models for workload injection

Light-ESP model

- shrinked execution time than ESP
- 230 jobs and 14 classes of jobs
- Largest jobs use 50% of the machine (ignoring Z jobs)
- Smallest jobs use 3.125% of the machine

Parallel Light-ESPx10 model

- shrinked execution time than ESP
- 10x smaller job sizes
- no Z jobs (13 classes)
- 10x more jobs (2280 jobs parallel launches)
- Largest jobs use 5% of the machine
- Smallest jobs use 0.3% of the machine

Proposed models for workload injection

Benchmarks	Normal-ESP	Light-ESP	Parallel Light-ESP
Job Type	Fraction of job size relative to system size (job size for cluster of 80640 cores) Number of Jobs / Run Time (sec)		
A	0.03125 (2520) / 75 / 267s	0.03125 (2520) / 75 / 22s	0.003125 (252) / 750 / 22s
B	0.06250 (5040) / 9 / 322s	0.06250 (5040) / 9 / 27s	0.00625 (504) / 90 / 27s
C	0.50000 (40320) / 3 / 534s	0.50000 (40320) / 3 / 45s	0.05000 (4032) / 30 / 45s
D	0.25000 (20160) / 3 / 616s	0.25000 (20160) / 3 / 51s	0.02500 (2016) / 30 / 51s
E	0.50000 (40320) / 3 / 315s	0.50000 (40320) / 3 / 26s	0.05000 (4032) / 30 / 26s
F	0.06250 (5040) / 9 / 1846s	0.06250 (5040) / 9 / 154s	0.00625 (504) / 90 / 154s
G	0.12500 (10080) / 6 / 1334s	0.12500 (10080) / 6 / 111s	0.01250 (1008) / 60 / 111s
H	0.15820 (12757) / 6 / 1067s	0.15820 (12757) / 6 / 89s	0.01582 (1276) / 60 / 89s
I	0.03125 (2520) / 24 / 1432s	0.03125 (2520) / 24 / 119s	0.003125 (252) / 240 / 119s
J	0.06250 (5040) / 24 / 725s	0.06250 (5040) / 24 / 60s	0.00625 (504) / 240 / 60s
K	0.09570 (7717) / 15 / 487s	0.09570 (7717) / 15 / 41s	0.00957 (772) / 150 / 41s
L	0.12500 (10080) / 36 / 366s	0.12500 (10080) / 36 / 30s	0.01250 (1008)/ 360 / 30s
M	0.25000 (20160) / 15 / 187s	0.25000 (20160) / 15 / 15s	0.02500 (2016) / 150 / 15s
Z	1.00000 (80640) / 2 / 100s	1.00000 (80640) / 2 / 20s	1.00000 (80640) 2 / 20s
Total Jobs / Theoretic Run Time	230 / 10773s	230 / 935s	2282 / 935s

Plan

Jobs High throughput

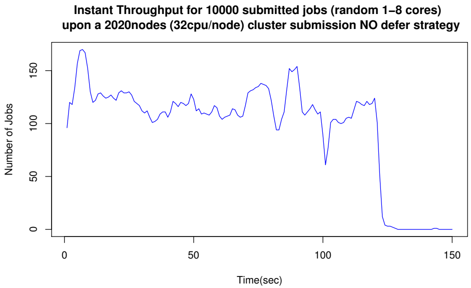
Large workload management

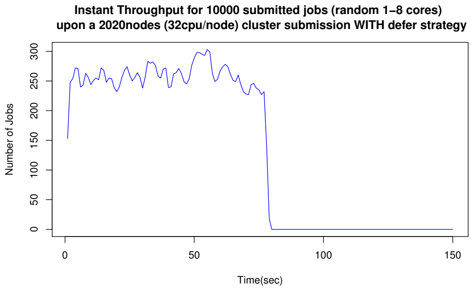
- Large number of directly elligible jobs
  - 10k jobs from 1 to 8 cores to fill an idle machine
- Large number of pending job launches
  - 1 big allocation to ensure that no new jobs can be scheduled
  - 10k jobs from 1 to 8 cores to fill an idle machine
- Large number of job terminations
  - 10k jobs from 1 to 8 cores to fill an idle machine
  - Global cancellation when every jobs are running

Large workload management

High throughput of directly eligible job launches

- Comparison of 2 SLURM configurations
  - default mode: 1 schedule try per incoming job
  - defer mode: 1 schedule try every 60 seconds (configurable)





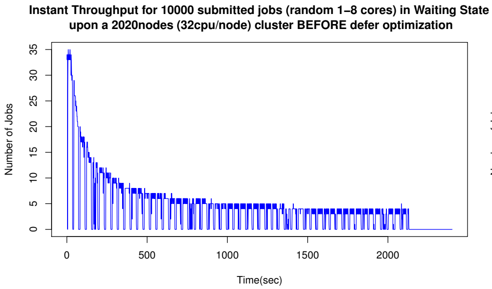
Observations

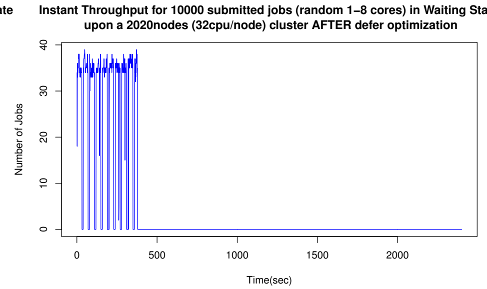
- defer mode greatly increase the submission rate
- but induces latency between submission and execution
- not good choice in case of single interactive jobs

Large workload management

Large number of pending job submissions

- Heavy load induced on the controller
  - Every job submission triggers the scheduler logic even in defer mode
  - Patch done to remove the scheduling logic greatly improves the submission rate (landed in SLURM main branch)
- Lower rate than for elligible jobs
  - No longer the case, corrected in the main branch

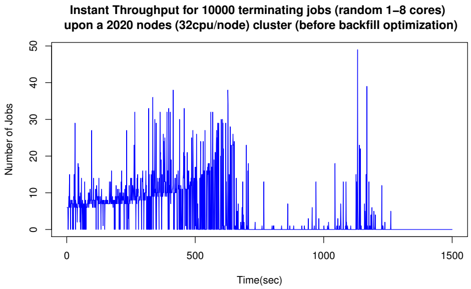


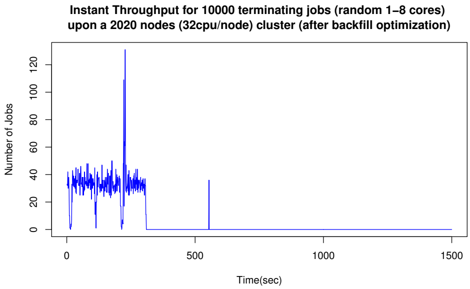


Large workload management

Large number of job terminations

- Heavy load due to a large number of completion messages involving a complexe logic in SLURM internals
  - 20 minutes of unresponsiveness of the system
- Patch from the community to reduce the complexity
  - greatly improves the termination rate
  - No major unresponsiveness after applying the patch





Network Topology aware placement

SLURM topology aware plugin

- to support network tree topology awareness of jobs
- best-fit approach pack the jobs on the minimal number of switches maximizing the amount of remaining free switches
- file for the description of the network design topology

Curie cluster network design

- Current Infiniband technology is based on a 36 ports ASIC that made fat-tree topologies constructed around a pattern of 18 entities.
- Actual design: 280 Infiniband leaves aggregated by 18 324 ports physical switches grouping the 5040 nodes in 16 virtual interme- diate switches.

Network Topology aware placement

Emulation of Curie using 200 Tera-100 physical nodes

- 5040 nodes with an Infiniband fat tree (18-ary fat tree)
- The real topology of the emulated cluster is different than the physical nodes topology. Not an issue since the goal is to evaluate the behavior of SLURM scheduling and not the behavior of the workload execution.

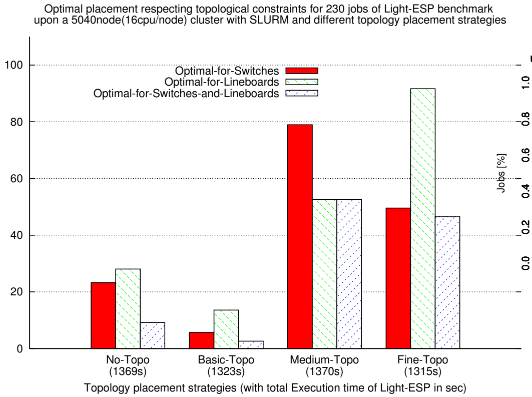
10 runs of Light-ESP benchmark for 4 scenarios

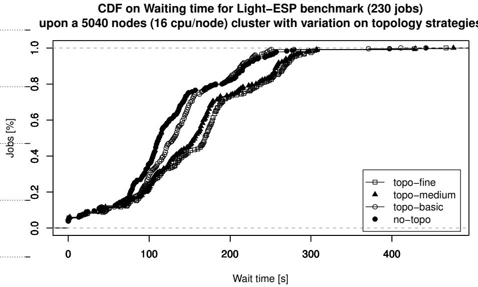
- no topology
- Basic topology:single virtual switch
- Medium topology: intermediate level only (324 nodes switches only)
- Fine topology: all levels (324 nodes switches, 18 nodes lineboards)

Comparison of placement versus optimal placement

- Optimal numbers of intermediate and leaf levels switches
- Evaluate random success of not sufficiently defined descriptions to provide good solution

Network Topology placement





Topology effectiveness

- Fine topology not efficient in providing the optimal number of intermediate levels switches
  - Because the best fit selection of switches is only made at leaf levels when the system is fragmented
  - Pruned topology, like Tera-100, should not use such a fine description
  - Full fat tree can use it as it still gives very good results for leaf levels

Nodes Scalability

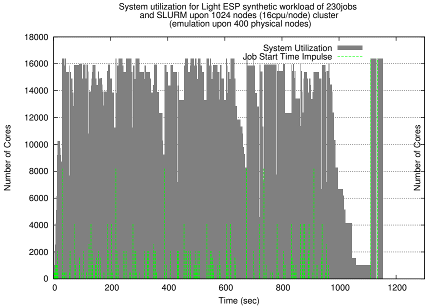
Scalability evaluation (standard jobs)

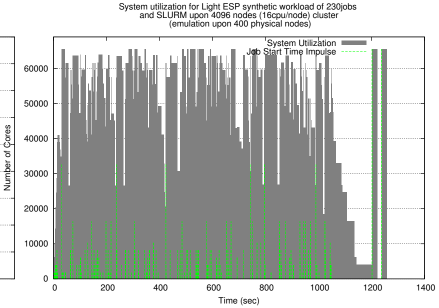
- 400 physical nodes of Tera-100
- 10 runs of Light-ESP benchmark for 6 scenarios
    - 1024, 2048, 4096, 8192, 12288 and 16384 nodes
- Global execution time measurement
    - Reflect the internal scheduler logic scalability when number of resources increased at the same pace as job sizes

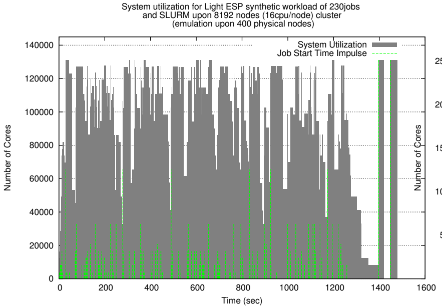
Scalability evaluation (small jobs)

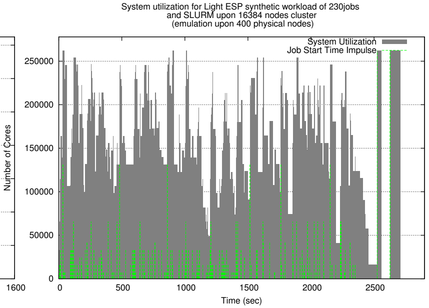
- 200 physical nodes of Tera-100
    - same results as with 400
- Parallel Light-ESP x10 benchmark for 6 scenarios
    - 1024, 2048, 4096, 8192, 12288 and 16384 nodes
    - Jobs are ten times smaller (in size) than for the previous evaluation
- Global execution time measurement
    - Reflect the internal scheduler logic scalability when number of resources increased but with a workload composed of a larger number of small jobs

Resources scalability for standard jobs

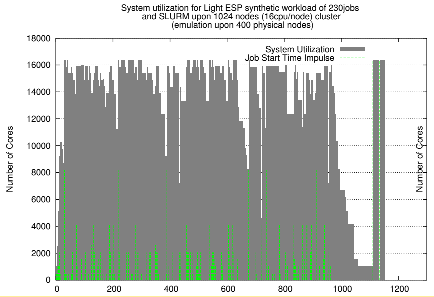


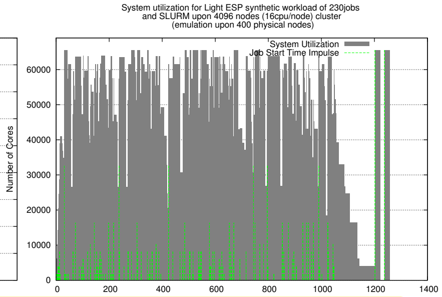






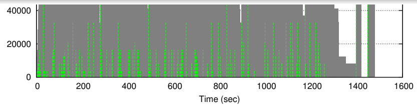
Resources scalability for standard jobs

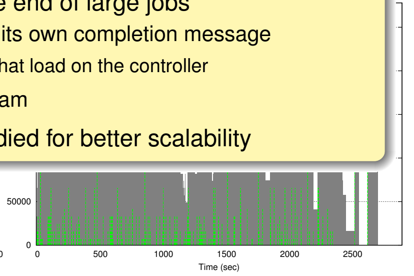




Scalability is great up to a threshold between 4096 and 8192 nodes

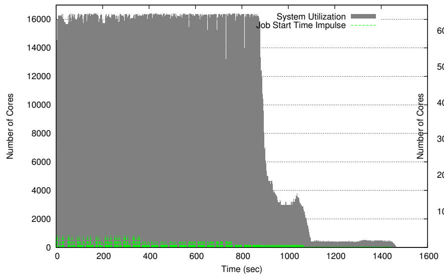
- The controller is heavily loaded at the end of large jobs
  - Every node involved in a job sends its own completion message
    - too much time spent in handling that load on the controller
  - Visible in the stretching of the diagram
- An aggregation strategy must be studied for better scalability



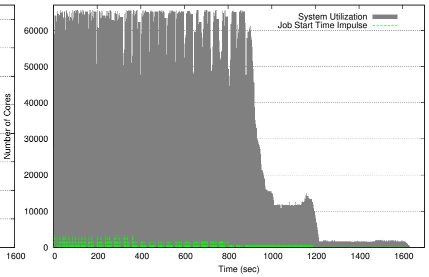


Resources scalability for small jobs

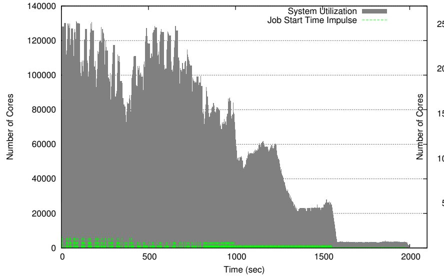
System utilization for Light ESPx10 synthetic workload of 2280jobs
and SLURM upon 1024 nodes (16cpus/node) cluster
(emulation upon 200 physical nodes)



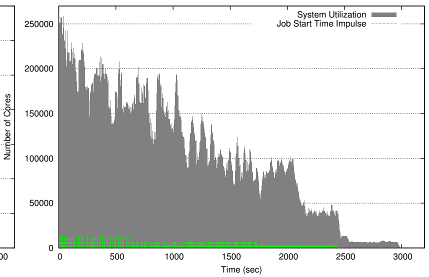
System utilization for Light ESPx10 synthetic workload of 2280jobs
and SLURM upon 4096 nodes (16cpus/node) cluster
(emulation upon 200 physical nodes)



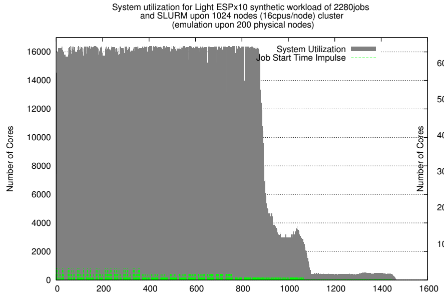
System utilization for Light ESPx10 synthetic workload of 2280jobs
and SLURM upon 8192 nodes cluster
(emulation upon 200 physical nodes)

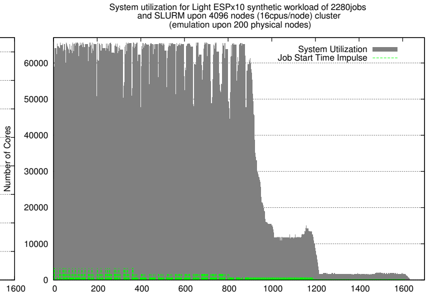


System utilization for Light ESPx10 synthetic workload of 2280jobs
and SLURM upon 16384 nodes cluster
(emulation upon 200 physical nodes)



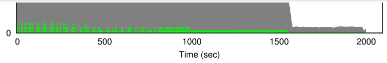
Resources scalability for small jobs

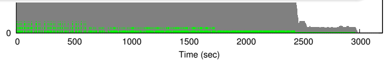




Scalability is great up to a threshold between 4096 and 8192 nodes

- Very high efficiency for clusters up to 4096 nodes
  - Visible on the high packing of the 2 first diagrams
- Good responsiveness of the system but system utilizations for large clusters collapse
- Further investigations required, probably a smoothed version of the previous issue (small jobs no longer so small at that scale)





Plan

Conclusions-Future Works

Experimental Methodology

- Emulation of large clusters on current existing systems is really interesting
  - Good to evaluate the behavior of the RJMS internals at scale
- Automation of emulation in production jobs
  - Great to avoid a particular set of physical nodes as the back-end
    - Avoid to reconfigure everything when a single node has a failure
  - Ease reproduction of identical benchmarks to guarantee the results
- More workloads, synthetic and real, should be tested
  - Evaluate more scenarios and detect issues to correct them

Conclusions-Future Works

Evaluation Results and Observations

- Large workload management
  - Good results up to 10k jobs with minor modifications (all included now)
- Topology effectiveness
  - Good results but gains are possible with an enhanced management of intermediate levels
    - Needs to add a multi-levels best-fit logic of selection
- Nodes scalability
  - Scalability threshold between 4096 and 8192 nodes (depends on the workload)
  - Improvements in completion messages handling seems mandatory
    - Large impact on performances on large clusters
    - One of the few n-to-1 communication patterns of SLURM
  - More tests on large workload and large clusters are required