HPC in a HEP lab: lessons learned from setting up cost-effective HPC clusters

2015 J. Phys.: Conf. Ser. 664 092012

(http://iopscience.iop.org/1742-6596/664/9/092012)

Journal of Physics: Conference Series **664** (2015) 092012 doi:10.1088/1742-6596/664/9/092012

HPC in a HEP lab: lessons learned from setting up cost-effective HPC clusters

Michal HUSEJKO, Ioannis AGTZIDIS, Pierre BAEHLER, Tadeusz DUL, John EVANS, Nils HIMYR and Helge MEINHARD

CERN, European Laboratory for Nuclear Research, Switzerland

E-mail: michal.husejko@cern.ch

Abstract. In this paper we present our findings gathered during the evaluation and testing of Windows Server High-Performance Computing (Windows HPC) in view of potentially using it as a production HPC system for engineering applications. The Windows HPC package, an extension of Microsofts Windows Server product, provides all essential interfaces, utilities and management functionality for creating, operating and monitoring a Windows-based HPC cluster infrastructure. The evaluation and test phase was focused on verifying the functionalities of Windows HPC, its performance, support of commercial tools and the integration with the users work environment.

We describe constraints imposed by the way the CERN Data Centre is operated, licensing for engineering tools and scalability and behaviour of the HPC engineering applications used at CERN. We will present an initial set of requirements, which were created based on the above constraints and requests from the CERN engineering user community. We will explain how we have configured Windows HPC clusters to provide job scheduling functionalities required to support the CERN engineering user community, quality of service, user- and project-based priorities, and fair access to limited resources. Finally, we will present several performance tests we carried out to verify Windows HPC performance and scalability.

1. Introduction

Until recently, powerful engineering workstations were sufficient to solve most of the engineering simulation and analysis problems encountered at CERN. However, we saw a growing number of enquiries from users seeking for more computing power, more memory and storage for analyzing more complex models or for more detailed analyses. We looked for solutions based on as much as possible standard CERN Data Center hardware and CERN-IT services with out-of-the-box configured commercial engineering applications used at CERN. In cases where CERN standard hardware/software components were not adequate, we propose cost-effective solutions for enhancing the performance.

In our CHEP 2013 paper [1] we presented our approach to building low-cost Linux-based HPC clusters; in this paper we present our approach to building Windows-based HPC clusters, that have the advantage to integrate smoothly with the Windows desktop infrastructure commonly used by CERN engineers.

2. CERN Data Centre - constraints imposed on design of the HPC cluster

Any new system to be installed in the existing CERN Data Centre (CERN DC) must fit into the established infrastructure. The hardware must be compatible with existing machines; operating the new system should only require services supported by CERN IT. For that reason we decided to build a cluster for HPC engineering applications by utilizing standard equipment procured for CERN DC, only replacing components wherever needed to obtain multi-core and multi-node application scalability. This imposes some constraints on what type of computing hardware, interconnect technology and storage system can be used.

The computing nodes purchased for CERN DC are usually standard dual-socket (rarely quad-socket) systems with 4 GB RAM per physical core. The standard interconnect is Ethernet at a speed of 1Gb/s or 10 Gb/s. Usually the tenders include options for upgrading some of the components (for example different network adapter cards, more RAM, SSD local storage instead of HDD).

To build a low-cost HPC system we had to consider using the standard CERN DC equipment.

Requirement 1: Ethernet interconnect is a standard in CERN DC

3. Engineering applications users and software licensing

The engineering community at CERN uses a variety of applications to perform their daily work. The work includes designing parts of LHC machine and detectors, performing safety-related simulations and others. Most of the design processes require multiple simulations of different types (e.g. finite element analysis, fluid dynamics, electromagnetic simulations). Almost all of these applications support and profit from using some sort of HPC infrastructure for performing the simulations. Some also support accelerator cards like GPUs and Xeon Phi.

The engineering applications to be run on the cluster are provided by different suppliers, most of them commercial. The licensing schemes of the commercial suppliers vary greatly; whenever an application starts, one or several licence features are being checked-out from the licence server. Depending on the application, one licence feature is required per core, per node, or per job; more complex schemes exist as well. In the HPC system that we are designing, the number of licences available naturally imposes a limit on the number of jobs that can run on the cluster simultaneously. In addition, users are only able to run a finite number of simulations due to the time needed to close the loop from preparing the simulation, waiting for it to complete, and analysing the results before submitting a new job.

Based on past experience we expect that the HPC cluster will have less than hundred distinct users, who will run less than 50 concurrent jobs on a total of less than 1'500 cores.

Requirement 2: The cluster must support multiple releases of engineering applications from multiple suppliers. Each application must support a range of MPI library implementations.

Requirement 3: The limited number of licences causes a limit on the number of cores that can be used per simulation job.

Requirement 4: The proposed system must be integrated well with the Windows environment as used by the CERN engineering community.

4. Computing scalability of engineering applications

The HPC engineering applications can utilize both multi-core platforms and distributed computing systems and some also support mixed mode (MPI+OpenMP). Distributed computing uses MPI libraries. Some applications are limited in the number of cores per job yet may require large amounts of RAM per core greater than 50 GBytes.

All HPC engineering applications take a user-supplied project description as input. In many cases, changes of a single option can significantly modify the applications behaviour, with a potential impact on the multi-core scalability or MPI profile. As a consequence, it is difficult to define a synthetic benchmark representing engineering applications at CERN, that could then

be used as input for system analysis. We decided to measure a few challenging use cases and select system solution based on these measurements.

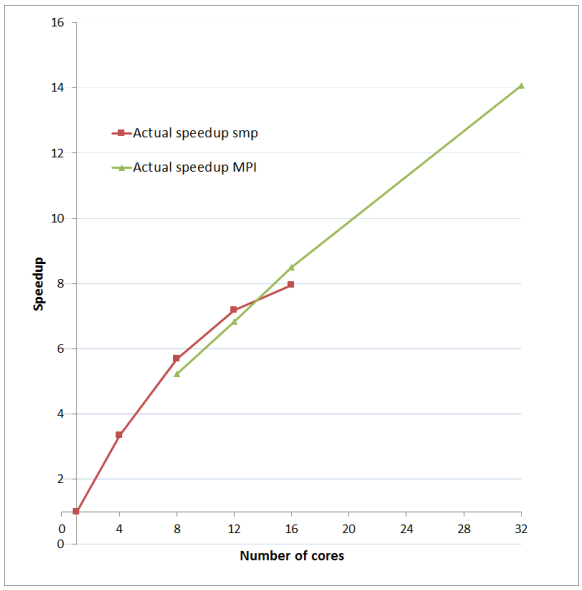

Figure 1. Shared memory (SMP) and distributed memory scalability (MPI).

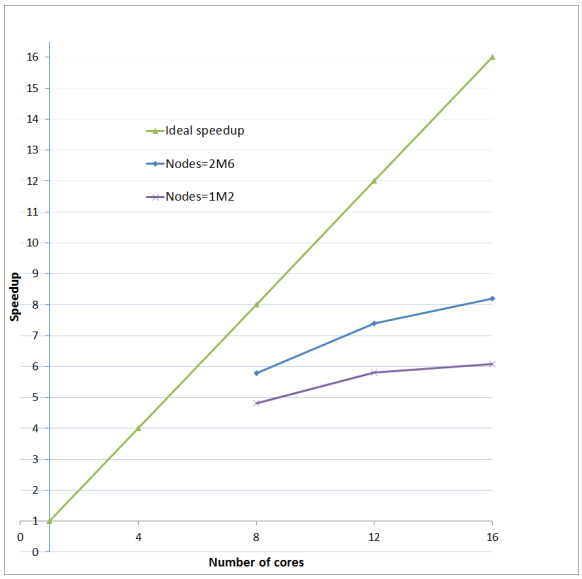

Figure 2. Scalability of simulation engine can be improved by increasing problem size while increasing number of cores.

For multi-core scalability within a single compute node, we selected a few cases with Ansys Mechanical, Comsol and Ansys CFX which we measured with up to 32 physical cores in a single node. We concluded that each tested tool can scale up to the maximum number of cores per node, provided that enough of work is provided for all the cores. The scalability was observed for both shared memory computing (OpenMP) and for distributed computing (MPI within a single node, with MPI traffic using shared memory transport). Results are presented in figure 1 and figure 2.

In order to measure the impact of the available memory size on the execution time, we used a fixed- size ANSYS Mechanical simulation problem and ran it on a workstation with both HDD and SSD storage. The simulation was configured to run in so-called out-of-core mode, where the temporary storage of simulation results is split between RAM and disk. The simulation was heavily I/O bandwidth- limited. When executed with an SSD offering twice the bandwidth of the HDD, the simulation ran 100% faster. The same speedup can be achieved by increasing the size of RAM and running the simulation in so-called in-core mode. Results are presented in figure 3.

We present the scalability of multi-node simulations as direct measurements (1Gb Ethernet vs. 10Gb low latency Ethernet) and indirect measurements based on our past experience with comparing Infiniband QDR against 10 Gb low latency Ethernet(presented in [1]).

(i) We compared multiple multi-node simulations running on two nodes of 32 physical cores each, measuring both 1Gb Ethernet and 10 Gb low-latency Ethernet connections. The representative example is show on figure 4. Switching from 10Gb to 1 Gb causes performance reduction.

(ii) We also performed an indirect analysis of multi-node simulations based on MPI profiles and execution times for Lattice QCD simulations published in our previous work [1] and based on performance reports and MPI profiles published by HPC Advisory Council [2]. If we

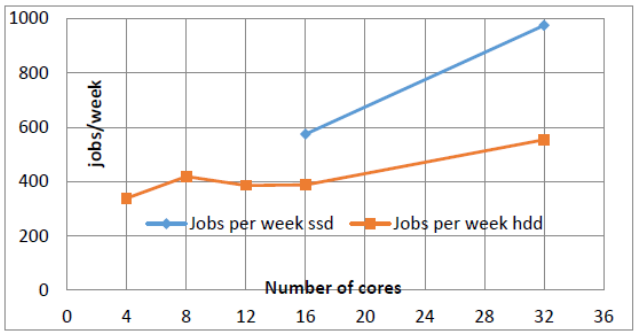

Figure 3. Impact of storage (I/O) performance on multi-core/node scalability.

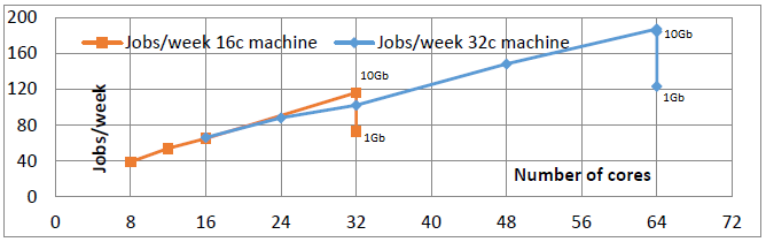

Figure 4. Comparison of distributed computing 1Gb versus 10 Gb Ethernet.

look at the MPI profiles of different engineering applications (ANSYS Mechanical, ANSYS CFX, Fluent and LS-DYNA) published in [2] we can observe that most of them consist of MPI collective operations (Allreduce, Bcast) and MPI point to point communication (Send, Recev). The MPI histograms presented in [2] show less stress that the MPI histograms for Lattice QCD simulations (analysed in[1] and presented in figure 5 for reference). In [1] we concluded that for small scale HPC clusters (up to 12-16 nodes) the simulation wall clock time for Infiniband and 10 Gb low latency clusters differs by less than 15%.

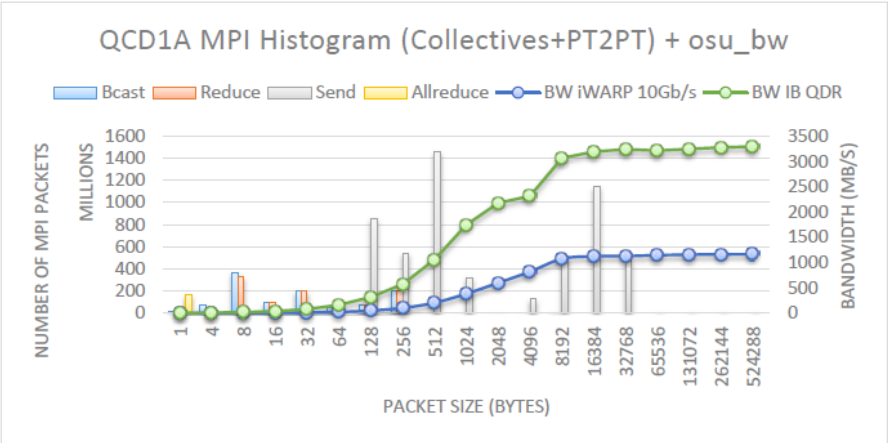

Figure 5. An example MPI profile for Lattice QCD simulation, analysed in [1].

Observation 1: When simulation job is spanning limited number of nodes (measured up to 12/16 nodes in [1]) 10 Gb low latency Ethernet can keep the pace with Infiniband QDR (with 15% performance difference).

Observation 2: Impact of 1 Gb Ethernet interconnect on multi-node simulations is strong. A lower and higher bandwidth interconnect is required.

Observation 3: If simulation is able to run in the in-core mode, then requirements on storage system can be relaxed.

5. Initial set of requirements

Based on the analysis and measurements presented above we can synthetize a set of constraints which will be imposed on the design of CERN Engineering HPC system:

(i) Cluster shall be well integrated into Windows based engineering environment and easy to use - GUI job submission preferred over command line.

(ii) System shall be based on standard machines purchased into CERN DC with optionally modified network interface card (NIC) and increased size of RAM.

(iii) Increase RAM size to maximum allowable inside single node to support small number of cores computation kernels (RAM requirement cannot be distributed among many nodes) and for running simulations.

(iv) Interconnect shall be based on Ethernet and could utilize low latency NIC.

(v) If increased RAM size is used, then the system shall utilize one of standard CERN DC storage systems. In case of Windows OS this is either DFS or SMB shares.

The system shall initially support these engineering applications: ANSYS, COMSOL, CST, HFSS, and LS-DYNA for both multi-core and multi-node (distributed) simulations.

6. Proposed architecture

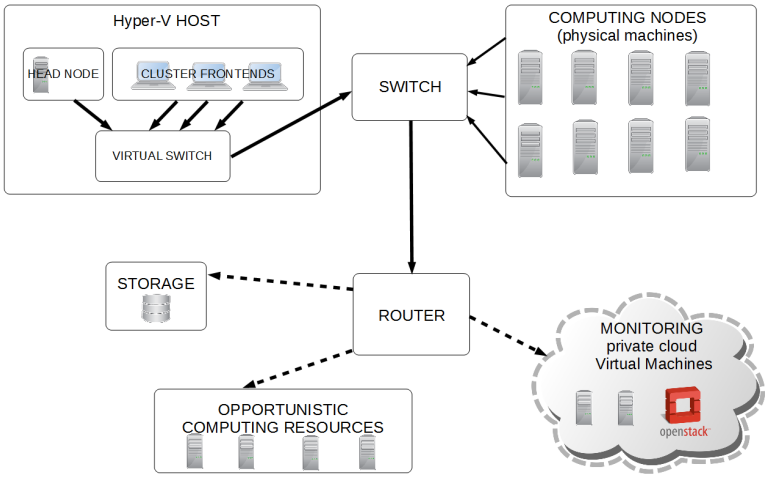

Figure 6. Proposed system architecture of Windows HPC cluster.

Based on the constraints and requirements presented above, we propose to build the HPC cluster based on Windows HPC. The Windows HPC is the upgrade package which is being installed on top of Windows Server, in our case Windows Server 2012 R2 64-bit. This package provides cluster management software, job submission and management software and Microsoft MPI (MSMPI). Each user who wants to utilize the cluster to run his simulation jobs has to install Windows HPC pack on his machine to be able to submit and monitor his jobs.

The proposed architecture is presented in figure 6. The cluster is built with both virtual and bare metal machines. The basic building block is a 32-core machine with 512 GB of RAM.

Each machine uses low latency 10 Gb iWARP/RDMA NIC. The machines are interconnected by low latency 10Gb Ethernet Switch. Total number of servers in the cluster is 16. One server is utilized as a (VM) virtual machine (Hyper-V) host. This machine hosts head node VM of the cluster and additional VMs which are used as a front end remote desktop machines. The remaining 15 servers are used as computing nodes providing to user community a computing system of 480 cores and 7.5 TB of RAM.

The cluster uses standard Windows storage system available as a service from one of CERN IT groups. The storage is provided under DFS namespace and available to user community as a scratch and project space. The low latency NICs utilized in the cluster are supporting SMB shares with RDMA off- load.

7. Windows HPC monitoring system

7.1. Concept

The overall idea for the monitoring came from a requirement to monitor the real hardware real usage by all running jobs on the cluster. It is needed because sometimes users request more computing resources than is actually used by the job (either by omission of some command line switch by the user or limitations of given solver) which results in suboptimal hardware usage.

Before developing a monitoring solution in order to collect such data cluster administrators have to manually run performance monitoring tools on specific node before starting job, and then look for the data collected. In the future there are plans to give access to the monitoring tools to the cluster users, which will allow them to investigate basic scalability and performance bottlenecks by themselves.

7.2. Architecture

The solution is developed by standardized ELK (Elasticsearch + Logstash + Kibana) stack with customizations specific for Windows HPC scheduler and monitoring machines by WMI protocol.

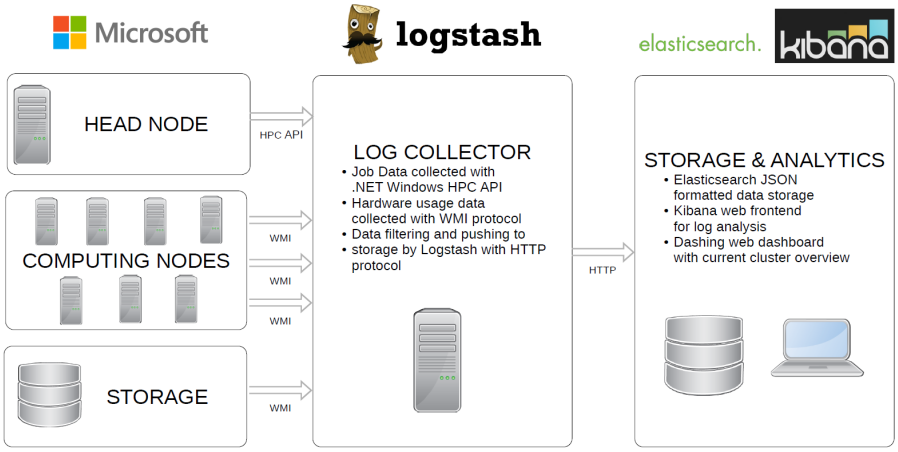

Figure 7. Proposed monitoring architecture of Windows HPC cluster .

7.2.1. Job Log Collector Job Data is collected by custom application which uses .NET Win HPC API provided by Microsoft. The application collects information about currently running and queued jobs. The collection is repeated every 30 seconds (one can think that each job sends a heartbeat to the monitoring when its either running or queued). All the information is stored in a file which is observed and read by Logstash and then pushed by standard Logstash to Elasticsearch output.

7.2.2. Hardware usage data Hardware usage data is collected by WMI protocol. The standard Logstash WMI plugin is designed to collect WMI data only from a local machine. It was modified to collect data remotely, in parallel from the hosts specified in Logstash configuration file. This data is then pushed by standard Logstash to Elasticsearch output.

7.2.3. Kibana frontend Monitoring data is customized in a set of predefined Kibana dashboards. We created separate dashboards for all the data regarding given machine and dashboards for each monitored component (cpu, memory, network, storage, jobs etc.). We find this set useful for basic monitoring needs and in case of more sophisticated analysis it allows us easy and fast extensions.

7.2.4. Dashing dashboard Using Dashing technology we created static dashboard which sums up what is currently happening on the cluster with some basic statistics. Such a dashboard is meant to be projected on a big screen in the administrators room.

7.2.5. Hardware used for setting up monitoring All the log collecting and log storage machines are set up using CERN private cloud virtual machines (Openstack). We found such an infrastructure perfect for our needs it is completely decoupled from the hardware we are monitoring and private cloud infrastructure allows for easy snapshotting and rebuilding of the machines (in case of need of migration or backup).

7.2.6. Data storage needed for the logs The monitoring system currently generates around 150 MB per day. This is not a lot, but if the amount of machines/parameters we need to monitor will increase in the future the whole monitoring should scale well just by adding more machines dedicated to the monitoring. Elasticsearch is a distributed storage and analytics system, and the log collection can be easily scaled horizontally by adding more machines to the log collection (each machine will collect hardware data from a subset of the whole set of monitored machines).

8. Performance testing

The initial deployment of our cluster was used by a community of around 70 engineers. During a few months of operation we were able to observe real-life usage patterns of our users. Most are running two type of jobs: daily "developer" jobs and "batch" jobs. Developer jobs typically take from a few minutes to a few hours and usually time to result is critical. Batch jobs can run for dozens of hours or days. To assess performance of the Windows HPC, we decided to test job insertion rate under the full load. We have allocated maximum number of cores available on the cluster (480 cores) to the test, and we were able to continuously injected jobs at the rate of 4 jobs per second (4 Hz) with close to zero impact on jobs startup and execution time. This performance is way above our requirement which is 1 job per second, which by itself is much more than we observed on our cluster with user jobs. Figure 8 shows that most of the jobs on our cluster are submitted not often than 5 seconds. Majority of the jobs are submitted not often than 1 minute and above. This proves that the system is fulfilling requirements.

9. Future work

We have already started prototyping Windows Cluster Failover which we plan to deploy on our HPC cluster to increase system resiliency to unplanned events and hardware failures.

All current engineering tools on the cluster use commercial licences which are a scarce resources during the workday. To minimize number of jobs failing due to licence limitations, we decided to implement licence aware job scheduling. If there are no licence features available for that job to run efficiently the job will be returned back to the scheduling queue.

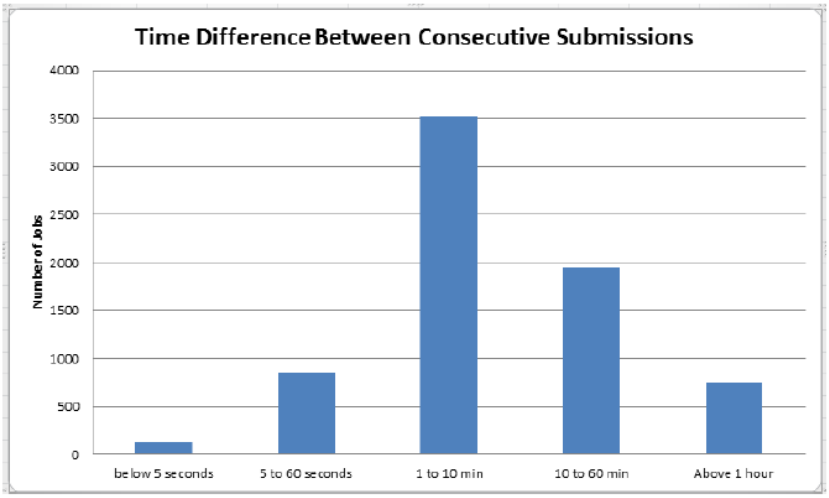

Figure 8. Distribution of spacing between job submissions performed by users.

With the extensive monitoring system that we have built around our cluster, we plan to implement automatic job benchmarking and performance reporting. This will allow us to understand better the behaviour of the cluster and the jobs which are being run on the cluster.

10. Conclusion

In this paper we have shown that it is possible to build a cost efficient system for engineering HPC application based on the standard technologies used in CERN Data Centre.

References

[1] Self-service for software development projects and HPC activities, Husejko et al. CHEP 2013.
[2] ANSYS Performance Benchmark and Profiling, HPC Advisory Council, http://www.hpcadvisorycouncil.com