Report on HPC resources integration at LHCb

Introduction

High Performance Computing (HPC) systems are highly not standard facilities, and are custom built, having in mind use cases largely different from High Energy Physics (HEP) ones. The utilization of these system by HEP experiments is not trivial: each HPC center is different and, of course, this increases the level of complexity from the integration and operations perspectives. One clear HPC design driver is the capability to show the best performance on standard benchmarks, in order to be listed as high as possible in the official HPC ranking[1] . Science related use cases range from Lattice QCD, astrophysical simulations, material sciences, simulations of nuclear systems. Such systems show performant node-to-node interconnection, needed for large scale MPI tasks, scarce local scratch disk, and are not meant for accessing data residing outside the facility. Recently, HPC systems have included node per node accelerator cards, in order to boost total performance and hence global ranking. Storage systems are optimized for latency and speed, and not for total size. On the other hand, current HEP High Throughput Computing (HTC) systems are built using different technical needs in mind: note-to-node connectivity is scarcely relevant, large on-node scratch areas are important, and global connectivity is needed in order to access remote datasets. The use of accelerator cards is marginal if not absent. The HEP workflows are typically data intensive, and systems deploy large storage systems close to the computing farms. Almost 100% of the experiment software stacks are designed and optimized on Intel x86_64 architecture, definitely the best choice for affordable computing in the last decade. This highlights two distinct categories of obstacles to the HPC based resources integration in HEP, namely:

- Utilization of system accelerators:
    - this comprises capability to benefit from additional accelerator hardware such as GPUs, FPGAs, and in turn translates in a experiment software stacks designed and optimized for NOT Intel x86_64 architecture.

[1] https://www.top500.org/

- Computing Facility integration:
    - this comprises peculiarities like the amount of available storage including scratch area on nodes, network setup and related policies, security constraints etc. Often HPC systems are not suitable for I/O intensive workflows.
    - Outbound connectivity may not be available on the compute nodes due to local security policies.

These are largely different types of problematics, in term of people's effort required, type of skills, as well as impact on experiment operations (e.g. software validation etc). The current document is intended to focus only on the second category.

1. Motivations for the document

This document aims at identifying a minimal set of requirements on HPC based resources in order to run LHCb Workflows. In this respect, it does not intend to address LHCb motivations for looking into HPC, but rather to sketch the strategy for a effective exploitation of them. This needs an identification of the requirements in order to try to use HPC as other owned site, and thus dealing with data output, stressing storage and network etc… LHCb's computing needs in Run 3 show that simulation will use the vast majority of available CPU, so the goal should be to make HPC based resources looking like any other LHCb site that run simulation payloads.

Starting with a collection of all the experiences, we try to define the minimal set of requirements any HPC system should satisfy in order to allow LHCb simulation workflow running (efficiently). This will allow to identify:

- Integration limitations and obstacles if any
    - technical / operational / human effort limitation
- Common areas of development and integration
    - Possibly identifying a list of tasks and actions

The next section will summarise the motivations of LHCb requirements to run jobs. Then a collection of current initiatives and R&D HPC related will be given. Based on the current experience all the possible integration options for each requirements will be summarised. Finally we will collect a minimal set of requirements and to conclude a list of open questions and items to improve will be presented.

2. Requirements to run LHCb jobs

The LHCb distributed computing is managed by DIRAC. A late binding approach is obtained via the submission of pilot jobs, which can also be started in the vacuum. The pilot checks for the runtime environment of the node. If all the checks are successful, the pilot job pulls payload jobs from DIRAC central WMS jobs queue, and executes them. LHCb pilots, like all DIRAC pilots, are completely generic. Pilots require python 2.6+ on the worker nodes in order to start.

The pilots checks for:

- The possibility to access the LHCb Software repository
  - which is currently distributed via CVMFS.
- The local architecture, for match-making.
- The possibility to temporarily stage the produced data to the local disk.
- The length of the local queue (which is maybe only logical).

In order to run LHCb jobs, a resource provider need to guarantee that:

- submission is enabled, meaning that DIRAC pilot factories can start pilots on the worker nodes
  - pilot factories may be remote (hosted centrally on the DIRAC servers), or local to the HPC
- Pilots can call back DIRAC Central Job Manager servers to fetch work.
- The local architecture is supported by LHCb software stack.
- Access to LHCb software and conditions (normally both distributed through CVMFS) is granted

3. HPC Challenges and identified scenarios

In this chapter the goal is to define all possible scenarios foreseen for each one of the above identified category of requirements (pilot submission, runtime and data management) We distinguish two macro scenario: HPCs where Worker Nodes have outbound connectivity and where doesn't. In the following we will use a color code:

- Green means the preferred (often standard) solution
- Orange means a change from the standard configuration which is viable given some effort from central LHCb operations and/or local site support

- Red means a blocking problem.

3.1 Availability of Internet on WNs

If worker nodes have internet access, they allow pilots to communicate back with DIRAC fetching payload so the issue to solve is how to get Pilots into the WNs.

Submission

Depending on the presence of a Computing Element three possibilities are available:

- If there is a CE: nothing needed apart from central configuration changes
- If there is no a CE, there are 2 possibilities:
  - the first is to use a “DIRAC SSH CE”, which is a very simple virtual Computing Element that only requires a SSH key pair to be established.
  - a DIRAC pilot factory can be setup, local to the HPC.

Runtime environment

Referring to the classification done above there are two aspects related to the runtime: Operating system and required dependencies. All this can also run on container so the question here is:

- if the local architecture is one supported by LHCb, then there’s nothing to do
- If singularity is available, then the LHCb software can make use of it
- If none of the above is true, then the effort for running LHCb workflows can be simply too high.

Access to LHCb software and resource specific settings is normally provided via CVMFS and thus:

- If CVMFS is available, LHCb jobs can make use of it out of the box
- If CVMFS is not provided LHCb will have, as of today, a hard time running jobs. There are some solutions being discussed, that could lead to positive outcomes, but that have not yet being tried out.

3.2 WNs do not have Internet access

Under this condition LHCb is currently not able to provide a verified technical solution to use the resources at all. DIRAC has in theory the technical capabilities to cope with this scenario, but it has never been tested with a real use case.