

Self-service for software development projects and HPC activities

M. Husejko¹, N. Høimyr¹, A. Gonzalez¹, G. Koloventzos¹, D. Asbury¹, A. Trzcinska¹,
I. Agtzidis¹, G. Botrel², J. Otto³

¹ CERN, CH-1211 Geneva, Switzerland

² Universite Paul Sabatier, F-31062 Toulouse, France

³ University of Warsaw, PL-00-927 Warszawa, Poland

E-mail: Michal.Husejko@cern.ch

Abstract. This contribution describes how CERN has implemented several essential tools for agile software development processes, ranging from version control (Git) to issue tracking (Jira) and documentation (Wikis). Running such services in a large organisation like CERN requires many administrative actions both by users and service providers, such as creating software projects, managing access rights, users and groups, and performing tool-specific customisation. Dealing with these requests manually would be a time-consuming task. Another area of our CERN computing services that has required dedicated manual support has been clusters for specific user communities with special needs. Our aim is to move all our services to a layered approach, with server infrastructure running on the internal cloud computing infrastructure at CERN. This contribution illustrates how we plan to optimise the management of our of services by means of an end-user facing platform acting as a portal into all the related services for software projects, inspired by popular portals for open-source developments such as Sourceforge, GitHub and others. Furthermore, the contribution will discuss recent activities with tests and evaluations of High Performance Computing (HPC) applications on different hardware and software stacks, and plans to offer a dynamically scalable HPC service at CERN, based on affordable hardware.

1. Introduction

CERN IT provides a number of services that are used by software development projects, such as the Git [1] and SVN [2] version control services, the JIRA Issue Tracking Service [3] and the TWiki [4] collaborative web application. Each of these tools have their own interfaces for registration and administration. For users of typical (software) projects, setting up and defining the rules for all the projects can be quite time consuming. Setting up repositories in a version control system, creation of a project for issue tracking, and wiki require requests to the administrators of the respective services. Handling the requests and setting up the project environment in each of the tools are administrative tasks that typically require system privileges, and hence only a limited number of support staff can handle these actions. Automating as many of these tasks as possible can lead to a gain of time both for users of the system in a project start-up phase, as well as IT support personnel.

Thus the team managing these services created a plan for a one-stop portal for software project support named “Cernforge”, along the same lines as offerings like Github [5] and Sourceforge [6]. The difference between an integrated approach like what is offered e.g. on Github and the “Cernforge” is that at CERN one seeks to integrate a set of existing, well established services and tools, instead of offering a complete service from scratch, like Github. This paper will describe the current version of “Cernforge”, as well as the plans for more features and technical implementation details.

While the CERN batch service and computing resources can address the needs of 90% of the regular physics computing, special software for Accelerator Physics and Engineering requires more specialized HPC resources. The engineering and accelerator physics communities at CERN use a range of simulation codes in different fields ranging from Beam Dynamics Studies, Computational



Fluid Dynamics (CFD) and Field Calculations to Electronic FPGA synthesis and Structural Analysis for Mechanical Design. Examples of applications are ANSYS, Comsol, CST, Fluent, HFSS, Orbit and openFOAM. Furthermore there are parallel applications (Lattice QCD) used by the Theoretical Physics group. All these applications can take advantage of a cluster where MPI is available. This contribution will discuss the testing and evaluation of the HPC platform, as well as future plans.

2. Services for software projects

The Version Control Services team in IT/PES provides software code management infrastructure (SVN and Git) to a global user community at CERN and world-wide in the LHC experiment collaborations. In addition, the group provides other tools, such as Issue Tracking based on JIRA and the TWiki wiki for collaborative web documentation. We will here briefly describe these applications.

2.1 Version Control Services

In the HEP community, software code, configuration files, FPGA programs, as well as other files such as LaTeX for certain physics applications, are typically stored in the Version Control Systems SVN or Git (in the past also CVS). At CERN, there are centrally supported services for SVN and Git project hosting for the user community.

2.1.1 Subversion (SVN). The central SVN service at CERN hosts about 2200 projects for CERN and the HEP community. The projects range from mission-critical on-line and off-line configurations of experiments like ATLAS and LHCb, to physics code of users in the experiments. Individual project settings are managed by project librarians, who can manage access rights and other settings for each repository. The SVN service is layered, taking advantage of central IT infrastructure services such as AFS for storage and Active Directory for authentication and authorization.. The service is hosted on two clusters: one, (svn) for SVN client access (https and ssh) and (svnweb) for web browser access. Furthermore, there are dedicated project management pages for SVN librarians, using a dedicated web interface.

The number of active users has averaged about 2500 per. month over the last couple of years, with peaks up to 5000 when many new repositories were setup. The number of commits per month is of the order of 50,000.

Setting up a new SVN project is automated, and users can currently request projects from a dedicated web page. The request triggers actions for the project creation, such as allocations of a dedicated AFS storage volume and the setting of access rights. Authorization to access data in repositories hosted in the CERN SVN service is granted via the traditional facilities with SVN, as well as e-groups, using established CERN LDAP groups.

2.1.2 Git. The Git service at CERN offers an alternative to SVN for project teams wishing to use a distributed software version control system. Git has become increasingly popular among open source projects over the years, and following user demand, the CERN Git service was launched in 2013 to complement SVN. Currently the service hosts about 700 projects. The usage is growing rapidly, with the number of commits per month now exceeding 15000 (figure 1)

The Git service architecture is similar to SVN, with a load-balanced cluster and storage currently on AFS. Git projects are requested directly from the “Cernforge” page, as described later in this paper. Authorization and access control is controlled via e-groups, via the underlying Gitolite Git repository management framework.

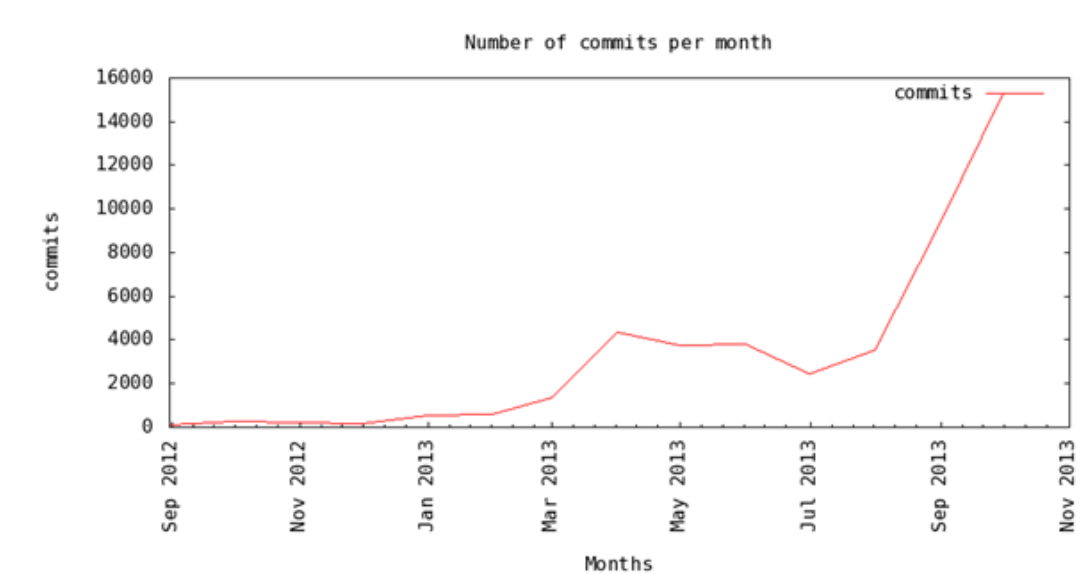


Figure 1. Git service – number of commits per month since the start of a pilot service.

2.2 Issue Tracking - JIRA

The central Issue Tracking service at CERN is based on Atlassian JIRA[7], and was set up in 2012. The service allows project teams to track bugs, tasks or other issues that may be relevant for their processes via the JIRA web application. Access authentication to JIRA is provided via the regular CERN Single Sign-on (SSO), while authorization is done via a combination of e-groups, mapped to JIRA-groups, and the roles that are defined within the JIRA application. The usage of the service is growing, and currently about 200 projects with about 25k issues are hosted in the central JIRA instance, with additional projects in custom hosted instances.

JIRA is a rather flexible tool and allows for the use of a number of add-ons, such as a module “JIRA-Agile”, for Agile project workflows and a Gantt chart plugin for project planning purposes. Each project may use different settings for issue types and permission schemes, allowing certain flexibility. However, the issue types, permission schemes and work-flow schemes can only be defined or customised by the system administrators. Also certain operations like linking a JIRA project to an SVN or Git repository requires JIRA system privileges. Thus having a way to grant project administrators a way to change some of their project settings is potentially interesting. A REST API to JIRA can be used to automate certain system tasks.

2.3 Collaborative web documentation - TWiki

The wiki application “TWiki” has been in use at CERN since 2003 to host web documentation for software projects as well as other purposes, ranging from physics paper discussions to IT Service documentation. TWiki is widely used by the LHC experiment collaborations, and has more than 10000 registered users.

TWiki is a project-oriented wiki application, and information in TWiki is structured into Topics and Webs, where webs are used for major experiments or other groups, and topics for individual web pages. Thus a typical software project would either be documented in a TWiki Web or TWiki topic. The TWiki application is flexible and allows numerous plug-ins. At CERN, plugins in use range from Math-mode text rendering to dynamic use of variables generated by an external application in one of the LHC experiments.

3. Software as a self-service - portal

In order to facilitate the administration of projects using our services, we wish to enable project administrators to do operations that currently require a support intervention, such as linking a JIRA project to a Git repository. For this, we aim to provide a web portal, based on the experience from the <http://cernforge.cern.ch> prototype and the SVN management panel for librarians. In the first iteration of the “Cernforge” portal, users are simply presented with links to create projects in SVN or to the CERN Service Portal to request a new JIRA project.

Our goal is to extend and develop the portal to fully automate creation and management of Git and SVN projects with links to issue tracking in JIRA as well as documentation pages in TWiki. The aim is to have a one-stop management portal for our services and to streamline the administration procedures for project administrators and relieve the support team of repetitive administrative tasks.

3.1 Design and architecture of portal

The user interface of the portal is deliberately simplistic, as you can see in figure 2.

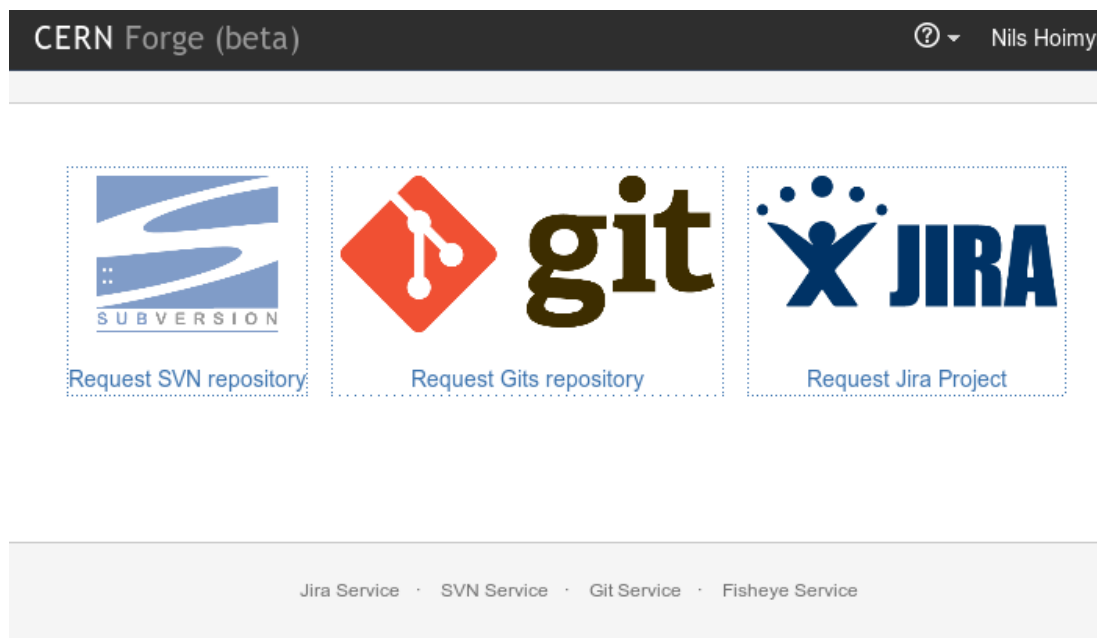


Figure 2- Portal user interface.

The portal is designed to be a flexible application that can communicate with the different services via a REST API. The portal itself is implemented with the Django Python framework and uses a database back-end to store project meta-data. An example set of menus for a JIRA project request is shown in figure 3.

The screenshot shows the 'Request a Jira Project' interface on the CERN Forge (beta) website. At the top, there's a dark header with 'CERN Forge (beta)' and a help icon. Below the header, the title 'Request a Jira Project' is centered. A message states: 'Your project will be created on the [Central Instance](#)'. The form includes a 'Project leader' field with the name 'Nils Holmyr' and a note 'Can be changed later in Jira'. There are two required text input fields: 'Name' and 'Key'. Below the 'Key' field, a note says 'This is the unique project key.' followed by a question mark icon. Below these are four dropdown menus: 'Notification Scheme' (set to 'Default'), 'Workflow Scheme' (set to 'CERN Default'), 'Issue Type Screen Scheme' (set to 'Project Management'), and 'Screen Scheme' (set to 'Restricted Creation'). At the bottom, there are two buttons: a 'Basic' button and a blue 'Send Request' button.

Figure 3. JIRA project request, with options to pass to API.

Only project creation is currently supported. However, administrative functions that change project settings will soon be possible, such as git project visibility and links between JIRA and SVN/GIT.

4. HPC tests & evaluation

Some 95% of CERN applications are served well with stock machines: Dual CPU, with 3 to 4 GB RAM/core, modest disk space, 1 Gb Ethernet or at best standard 10 Gb Ethernet. In order to operate these machines more efficiently, we have heavily invested in the Agile Infrastructure (AI) including layered approach to responsibilities, virtualization and private cloud. A small fraction of applications (engineering, theory, and accelerator) have requirements that are not well fulfilled by these stock machines, due to core count, memory size/latency/bandwidth, and inter-process communication. Even though all these applications are HPC applications, the requirements (and hence the optimal technical solutions) can be very different. **Every effort must be made to fulfil these requirements as much as possible by the same, layered approach.**

In order to understand the requirements of CERN HPC applications (from this point on called simply HPC applications), we have consulted our user community to gather user test cases. We have prepared system analysis environments, based on available hardware and standard performance monitoring tools, with the main goal to understand better the requirements of user applications. In this article we would like to introduce some of the HPC applications, present at CERN and present preliminary performance analysis.

4.1. HPC applications

Based on our interaction with the user community, we have concentrated on two different groups of HPC applications: engineering applications and physics simulation applications.

For HPC engineering applications we have major commercial engineering tools like ANSYS Mechanical [9], Comsol [10], FLUENT [11], CST [12], and HFSS [13]. These tools are being extensively used in different CERN departments to model and design parts of the LHC machine. The most common usage scenarios involve structural analysis (stress and thermal), fluid dynamics, electromagnetic simulations, and also recently multiphysics [10][14], i.e. solving different physics aspects in a same single simulation run. It is very important to note, that most of CERN HPC engineering applications need commercial licenses to run, creating natural scalability barriers for these applications. Understanding that there are many users of these tools, and that the number of licenses is limited, we expect the number of cores utilized per single simulation job will be in range of 128-256. HPC physics applications include HEP community developed simulation packages for theoretical and accelerator physics studies, with the main representative being Lattice QCD simulations [15][21], but also LINAC4 plasma simulations.

4.2. Analysis of HPC applications – methodology

To assess if the existing infrastructure can be used to run HPC applications, we started detailed analysis of these applications, including scalability of numbers of cores, impact of temporary storage, impact of interconnect latency, and most importantly, the impact of low latency interconnect type on total wall clock time of HPC applications.

This work involves the analysis of applications where we have access to source code (HPC physics applications and a few engineering applications) and applications where we do not (most of HPC engineering applications). According to our knowledge there are only few publications available [22][23][24] which try to compare 10 Gb Ethernet against Infiniband for applications which we are concerned. We agree with the findings that Infiniband gives better performance, but we would like to show that 10Gb Ethernet interconnect is a “good enough” solution for CERN’s specific HPC applications. With this in mind we decided that the common denominator is to perform the analysis at the highest level possible, concentrating on the impact of different system parameters on total wall clock time (number of simulation jobs per week).

4.3. Analysis of HPC applications – performance analysis tools and hardware test systems

For system-level performance analysis we decided to use open source tools and system level analysis tools present in the Scientific Linux (SLC) distribution (iostat, dstat, sar) to monitor CPU performance, disk i/o access patterns and network traffic. We extended this with the Intel Performance Counter Monitor (PCM) [16] to analyse memory bandwidth consumed by each application at the memory controller's level. In addition we used mpiP [17][18] and IPM [19] to analyze the MPI communication.

To compare the performance impact of low latency interconnects two clusters were set up with the same type of compute nodes but the following two different interconnects. One used 10 Gb/s low latency Ethernet interconnects (iWARP), whereas the other Infiniband QDR (40Gb/s). For both clusters the MVAPICH2 MPI library was used [20].

4.4. HPC test results

All tested applications show good scalability with the number of cores. If strong scalability is not achievable, we can still profit from weak scalability by increasing the problem size, typically by increasing number of mesh elements used to numerically represent simulated structure. Another observation is that if simulation is being executed on distributed nodes, a low latency interconnect is necessary to achieve strong or weak scalability beyond 1-2 compute nodes.

In almost all tested cases, CPU time is spent either in computation or communication, with almost no time wasted in system or iowait as long as the simulation job can fit in system memory, which

implies that we have to provide enough fast memory for each compute node. Based on analysis of our cases, we estimate that 256 GB of RAM per dual socket compute node is a practical solution. Last, but not least, the results obtained with Lattice QCD simulations show that low latency Ethernet interconnect can keep pace with theoretically better performing Infiniband. The results obtained with our tests show that Ethernet can keep pace with Infiniband QDR up to 12 nodes (144 cores), this perfectly fits the requirements of our physicists (simulations are usually being run with no more than 256 cores) and engineers (with natural commercial license limits of 128-192 cores).

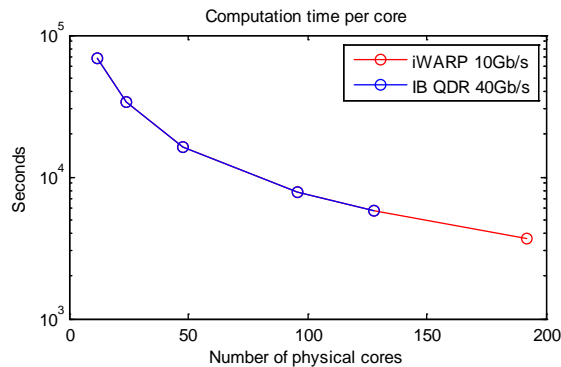


Figure 4. LatticeQCD – Part of wall clock time spent performing computations. Curves for IB and iWARP clusters are aligned – same computing nodes are used in both clusters.

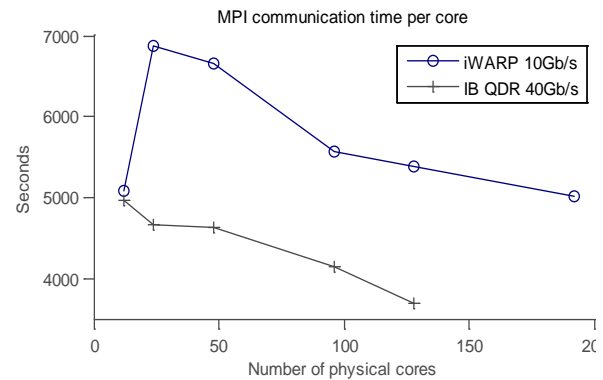


Figure 5. LatticeQCD – Part of wall clock time spent on MPI communication. iWARP cluster consumes more wall clock time compared to IB cluster, but the trend is similar.

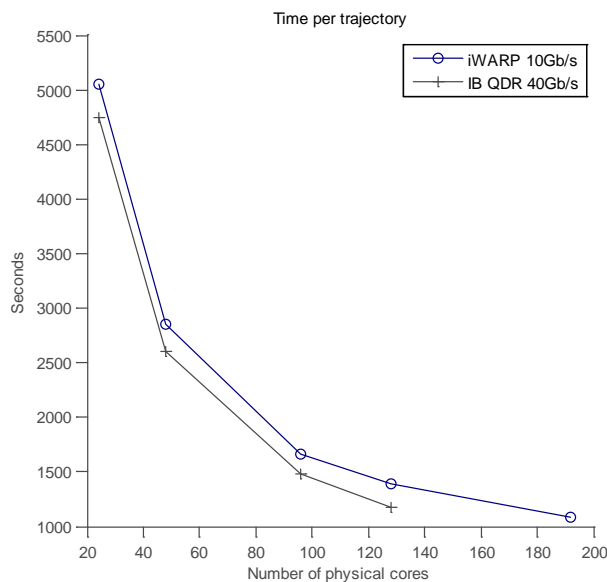


Figure 6. LatticeQCD – Simulation Wall clock time for iWARP and IB clusters. Even though the IB cluster has a superior interconnect, we observe at a system - level that iWARP cluster is slower by 15% for this particular 128 core simulation.

5. Conclusion

The two projects on a software portal for self service and cost-effective HPC computing at CERN are part of an overall change of IT service strategy to offer on-demand services that can scale according to need, based on a layered set of building blocks of IT services. For the software portal, the approach is to automate and link existing computing services to grant users and project administrators the power to

scale and manage their service according to need. The HPC project is currently in the evaluation phase, but also here the aim is to build a scalable self-service HPC cloud facility on top of CERN's internal cloud and batch infrastructure. Our evaluation has shown that with 10Gb Ethernet interconnects, one can achieve HPC cluster performance that is good enough for most use cases. By optimizing the setup on commodity hardware, an affordable generic HPC platform can be provided.

6. Acknowledgements

We would like to thank the authors of the software packages and IT infrastructure teams who provide the foundation for the work on these new projects for self-service software support and HPC.

References:

- [1] Git service at CERN URL <http://cern.ch/git>
- [2] SVN service at CERN URL <http://cern.ch/svn>
- [3] Issue Tracking Service at CERN URL <http://cern.ch/its/>
- [4] TWiki Service at CERN URL <http://twiki.cern.ch>
- [5] Github URL <https://github.com>
- [6] Sourceforge URL <http://sourceforge.net/>
- [7] Atlassian JIRA URL <https://www.atlassian.com/software/jira>
- [8] Django URL <https://www.djangoproject.com/>
- [9] ANSYS Mechanical URL <http://www.ansys.com/Products/Simulation+Technology/Structural+Mechanics/ANSYS+Mechanical>
- [10] Comsol Multiphysics URL <http://www.comsol.com/comsol-multiphysics>
- [11] FLUENT URL <http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+Fluent>
- [12] CST STUDIO URL <https://www.cst.com/Products/CSTS2>
- [13] ANSYS HFSS URL <http://www.ansys.com/Products/Simulation+Technology/Electromagnetics/Signal+Integrity/ANSYS+HFSS>
- [14] ANSYS Multiphysics URL <http://www.ansys.com/Products/Simulation+Technology/Systems+&+Multiphysics>
- [15] Simulation Program for Lattice QCD with open boundary conditions URL <http://luscher.web.cern.ch/luscher/openQCD/>
- [16] Intel Performance Counter Monitor URL <http://software.intel.com/en-us/articles/intel-performance-counter-monitor-a-better-way-to-measure-cpu-utilization>
- [17] mpiP Lightweight Scalable MPI Profiling URL <http://mpip.sourceforge.net/>
- [18] Vetter J S and McCracken M O 2001 [Statistical scalability analysis of communication operations in distributed applications](#) *Proc. ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPOPP)*
- [19] IPM - Integrated Performance Monitoring URL <http://ipm-hpc.org/>
- [20] MVAPICH – MPI Library URL <http://mvapich.cse.ohio-state.edu/>
- [21] Lüscher M 2010 CERN-PH-TH-2010-047
- [22] HPC Advisory Council 2013 MILC Performance Benchmark and Profiling *HPC Advisory Council Best Practices*
- [23] HPC Advisory Council 2010 ANSYS Performance Benchmark and Profiling *HPC Advisory Council Best Practices*
- [24] The MIMD Lattice Computation (MILC) Collaboration URL <http://www.physics.utah.edu/~detar/milc/>