jamk.fi

Integration of scientific and engineering applications into CERN's computing resources

Markus Jylhänkangas

Bachelor's thesis
May 2019
Information and communication technology
Degree Programme in Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Jylhänkangas, Markus	Bachelor's thesis	May 2019
		Language of publication: English
	Number of pages: 47	Permission for web publication: x

Title of publication

Integration of scientific and engineering applications into CERN's computing resources

Degree programme

Software Engineering

Supervisor(s)

Kokkonen, Tero

Rantonen, Mika

Assigned by

CERN

Abstract

Following CERN's strategy to consolidate all computing resources under Linux, it was planned to move the Windows HPC service to Linux-based infrastructure. The scientific and engineering applications that were running under Windows HPC were therefore assigned to be migrated to Linux HPC and HTC environments.

The main objective is the migration of engineering applications to Linux-based computing resources. The first objective for the migration was to design a strategy to carry out the migration by identifying potential risks and select target service HPC or HTC, to make efficient use of resources. Second objective was to implement templates, plugins and necessary changes to the HTC and HPC infrastructure. Migration was also going to be tested and validates with users.

The migration was implemented by first installing the applications on the Linux resources. Template script were written to submit applications to HPC or HTC. Modification to HPC were made with Puppet to make it possible to run the applications

As a result, the applications were migrated to either HPC or HTC resources and the Windows HPC was deprecated. Authentication and security inside the Linux HPC cluster was also improved.

Running complex scientific and engineering applications on Linux HPC and HTC resources is possible, but there are some compatibility problems with different parallel computing implementations.

Keywords/tags (subjects)

HPC, HTC, SLURM, HTCondor, Batch

Miscellaneous (Confidential information)

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Jylhänkangas, Markus	Opinnäytetyö, AMK	Toukokuu 2019
	Sivumäärä 47	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Integration of scientific and engineering applications into CERN's computing resources

Tutkinto-ohjelma

Ohjelmistotekniikka

Työn ohjaaja(t)

Tero Kokkonen

Mika Rantonen

Toimeksiantaja(t)

CERN

Tiivistelmä

CERNin strategiaa tietojenkäsittelyn vakauttamisesta seuraten Windows HPC-palvelu suunniteltiin siirrettäväksi Linux-pohjaiseen infrastruktuuriin. Sen vuoksi tehtäväksi annettiin tieteellisten ja teknisten sovellusten siirtäminen Windows HPC-ympäristöstä Linux HPC ja HTC-ympäristöön.

Päätehtävänä sovellusten siirrossa oli teknisten sovellusten siirto Linux-pohjaisiin tietojenkäsittelyresursseihin. Ensimmäisenä tehtävänä siirrossa oli suunnitella strategia siirrosta ja siirtoon liittyvistä mahdollisista riskeistä sekä valita kohdeympäristö HPC tai HTC. Toisenä tehtävänä oli toteuttaa ja kehittää sapluunat, lisäosat sekä tarvittavat muutokset HTC- ja HPC-infrastruktuureihin. Siirron lopputulokset testattiin ja vahvistettiin käyttäjien kanssa.

Siirto suoritettiin ensin asentamalla sovellukset Linux resursseille. Sovellusten suorittamista varten kehitettiin sapluunat, joilla voitiin lähettää töitä Linux-resursseihin. Jotta sovellukset saatiin toimimaan, muutoksia piti tehdä HPC-ympäristöön. Nämä muutokset suoritettiin Puppetilla.

Tuloksena sovellukset siirrettiin joko HPC- tai HTC-resursseille ja Windows HPC-palvelu otettiin pois käytöstä. Käyttäjien identiteetin todennusta sekä turvallisuutta parannettiin Linux HPC-klusterissa.

Monimutkaisten tieteellisten ja teknisten sovellusten suorittaminen Linux HPC- ja HTC-resursseilla on mahdollista, mutta näiden välillä on pieniä yhteensopivuusongelmia.

Avainsanat (asiasanat)

HPC, HTC, SLURM, HTCondor, Batch

Muut tiedot (Salassa pidettävät liitteet)

Contents

Figures

Tables

Acronyms

ARC ANSYS RSM Cluster

API Application programming interface

CERN European Organization for Nuclear Research

GUI Graphical User Interface

HPC High-Performance Computing

HTC High-Throughtput Computing

HEP High Energy Physics

IaaS Infrastructure-as-a-Service

LFS Load Sharing Facility

LHC Large Hadron Collider

MPI Message Passing Interface

RSM Remote Solver Manager

PAM Linux Pluggable Authentication Modules

SGE Sun Gring Engine

SDS Software-Defined Storage

Slurm Simple Linux Utility for Resource Management

SSH Secure Shell

TGT Ticket Generating Ticket

UGE Univa Grid Engine

XVFB X virtual framebuffer

WWW World Wide Web

1 Introduction

1.1 Motivation and background

This project was carried out at the European Organization for Nuclear Research (CERN) for the need to migrate engineering and scientific simulations from Windows HPC to Linux HPC infrastructure. Motivation for this is that significant amount of resources are used for maintaining two separate High-Performance Computing (HPC) services, and developing HPC and High-Throughput Computing (HTC) infrastructures that is already in place and running Linux. By migrating engineering applications from Windows based infrastructure to Linux infrastructure, it is possible to reduce maintenance of the HPC services and simplify support for engineering applications. This migration makes it possible to make better use of resources.

Running different types of simulations is an important part of scientific and engineering work. It is also important that the simulations can be run effectively and fairly on the infrastructure. At CERN, Every group that has access to computing resources has been allocated specific portion of these resources.

1.2 CERN

CERN was founded in 1954, and is known for the Large Hadron Collider, the world's largest and powerful particle accelerator (*The Large Hadron Collider* 2018) and being the place where the World Wide Web (WWW) was founded, by sir Tim Berners-Lee (*The birth of the Web* 2018).

CERN provides infrastructure for high-energy physics research, such as particle accelerators and data center for storing and analyzing data from experiments.

Currently 23 member states are part of CERN: Austria, Belgium, Bulgaria, Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Israel, Italy, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Spain, Sweden, Switzerland and United Kingdom (*Our Member States* 2019).

There is also multiple countries with observer status, as well as co-operation argeements with non-member states.

Multiple experiments are run at CERN. Seven of these are run at LHC: ATLAS, CMS, ALICE, LHCb, TOTEM and LHCf. There are also non-LHC experiments, which include:

- COMPASS which looks at the structure of hadrons
- NA61/SHINE studies properties of hadrons in collision of beam particles with fixed targets
- NA62 uses protons to study rare decays of kaons
- DIRAC is investigating the strong force between quarks.
- CLOUD is investigating a possible link between cosmic rays and cloud formation
- Five experiments that uses antiprotons from the Antiproton Decelerator
  - ACE
  - AEGIS
  - ALPHA
  - ASACUSA
  - ATRAP
- CAST experiment is looking for hypothetical particles coming from the Sun.

(*Our Member States* 2019)

1.3 Scope and objectives

This thesis explores compatibility, deployability and problems that might come up when integrating scientific and engineering applications in to CERN infrastructure. Previous reports have not described how and what problems there might come up integrating these simulation applications on existing infrastructure.

The main objective is the migration of engineering application to Linux-based computing resources. This is going to be accomplished by dividing this objective into sub tasks

1. Strategy is going to be devised, where potential risks are identified
2. To make efficient use of resources, it is decided if application is migrated to either HPC or HTC services

3. Necessary plugins and changes are implemented in order to accommodate new applications
4. Applications are tested and validated with users

HPC and HTC resources are used to run heavy simulations and computations that are difficult or expensive to run on normal desktop computers, due to time and resource constraints.

Most computations at CERN are run at HTC batch environment, but some are better to run on HPC. Applications that benefit from parallel execution, for example using MPI or applications that do not fit HTC model are run on HPC.

This thesis focus specifically on CERN HPC and HTC infrastructure with CERN-specific problems and solutions. Any problems that might come up with the integration to CERN infrastructure are responded to. For testing the applications on HPC there is a test cluster setup, where simulations can be tested without breaking the main cluster.

In section 2.1 the thesis describes research methodology used and chapters 3.1, 4.1 and 5 discusses theories related to migration. After theory, the actual execution of the migration is described in chapter 7, and finally at the end results in chapter 8 and conclusion in chapter 9 is shared with future research.

2 Research methodology

This thesis answers the question, how simulation applications are migrated from Windows HPC cluster to Linux HPC cluster. This is carried out using constructive research methodology, which is explained in chapter 2.1.

2.1 Constructive research

Constructive research methodology was developed in business administration; however, it has been used in many other fields as well.

In constructive research the goal is to solve real world problems by producing a new construction. Construction is an abstract concept that includes every possible artifact created, for example designs, plans, diagrams, organizational structures, products and information systems. One character of these constructions is that they are not found but developed and invented. (Lukka 2001)

2.2 Research process

According to Lukka (2001), the common constructive research process follows few common steps. In this section, these these steps and the ways how the thesis uses them are discussed and described.

Identify relevant problem, where there is possibility for theoretical contribution

A realistic solvable problem needs to be identified and it is possible to construct something for the job.

The problem that this thesis sets out to solve is migrating applications from Windows HPC cluster to Linux HPC cluster with the documentation of the process. There is not much documentation about this process yet.

Acquiring theoretical and practical information

Information required to solve the problem needs to be acquired from different sources. Usually, the problems require theoretical and practical information.

The information acquired for this thesis originates from few different types of sources,

i.e. research papers, websites and tool documentations.

Forming solution to problem, which might also have theoretical contributions

At this stage, a solution to the problem is formed and documentation constructed about the difficulties in solving those problems.

Testing the solution and its functionality

At this stage, the solution needs to be tested in order to make sure that it is working.

For testing if the solutions are working there is a testing environment where test simulations can be run.

Reflect scope of solution

This chapter reflects on the scope of the solution. Even if the solution did not work, there is an analysis of why and how it did not work.

Recognizing and analyzing theoretical contribution

Here the focus is on what theoretical contributions this research has created. (Lukka 2001)

3 Private cloud

3.1 Agile Infrastructure

CERN developed all of its IT infrastructure based on agile methodologies. As a result, CERN has developed a private cloud and tools that are tightly coupled in a way that enables user-facing IT services to be operated using an agile methodology. This concept has been codenamed Agile Infrastructure. The purpose of Agile Infrastructures is to ease the management of infrastructure resources and configurations. (Llamas, Megino, Kucharczyk, Denis & Cinquilli 2014)

General overview of the Agile infrastructure is described in Figure 1. At the bottom there is the hardware, on which everything is built upon. Agile infrastructure configuration is used to configure everything from hypervisors to IT services. Monitoring is used to monitor the whole stack.

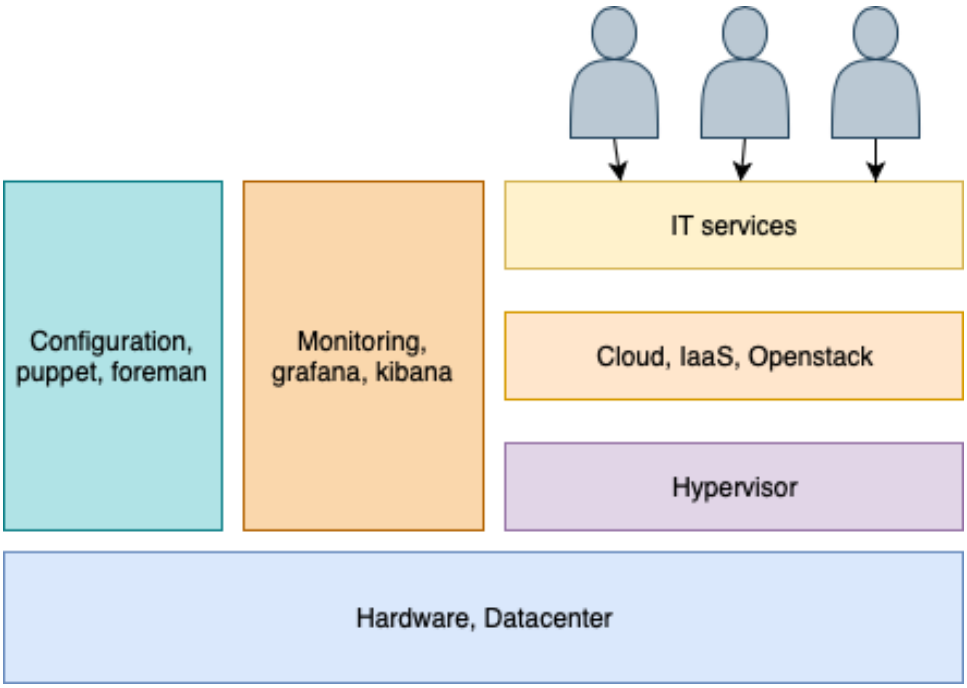

Figure 1. Agile infrastructure

For monitoring the cloud few various tools are used:

- Flume
- Elastic

- Kibana
- and Grafana

All the other components of the agile infrastructure are described here.

3.2 OpenStack

Agile infrastructure is operated by OpenStack and it manages all the machines in the private cloud. Many of the machines are virtual machines running on hypervisors; however, there are also bare metal servers managed by OpenStack Ironic.

OpenStack is a software platform for cloud computing deployed as infrastructure-as-a-service (IaaS). Openstack layer between the hardware and the applications, as seen in Figure 2.

The purpose if OpenStack is to provide for the industry, a massively scalable hosting architecture that is completely open source (Sefraoui, Aissaoui & Eleuldj 2012). OpenStack project began as project of NASA and Rackspace Hosting, but it is now managed by OpenStack foundation.

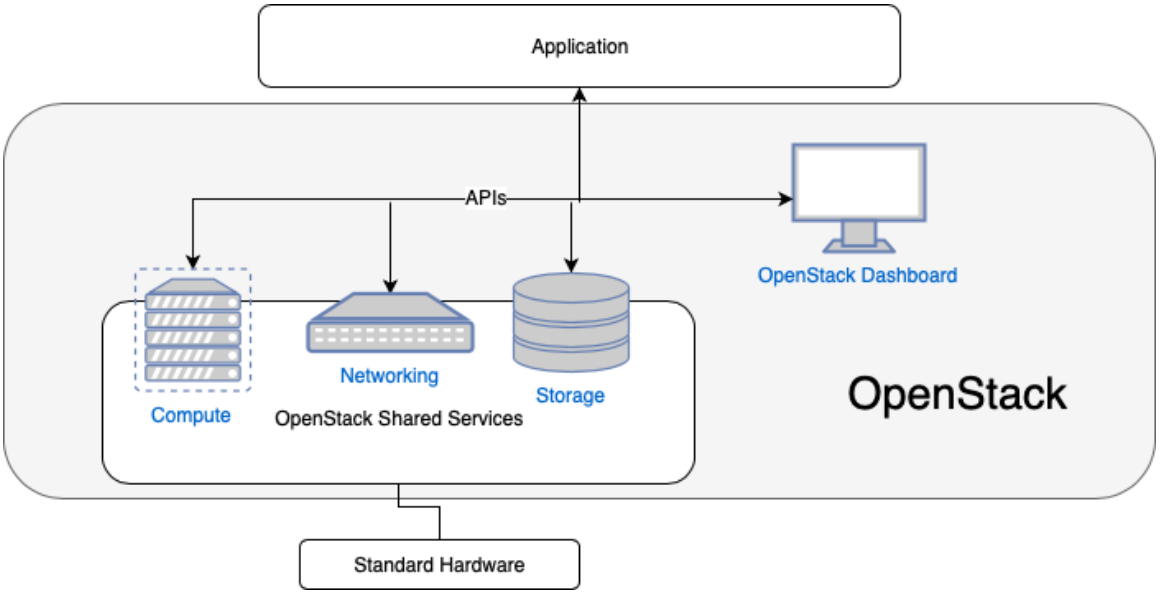

Figure 2. Openstack (Leon 2017)

3.3 Configuration tools

Configuration tools allows automating the configuration of the machines. These automation tools can eliminate issues and save time when configuring a large number of machines. This enables reproducible infrastructure and makes maintaining it easier. Puppet and Foreman are used as configuration tools.

Puppet is a software configuration management tool, which is used for managing server configurations, including HPC cluster. Puppet enables the execution and organization of configuration plans and allows addressing configurable components of the system as objects through a resource abstraction layer. (Loope 2011)

Foreman is used for server orchestration. It is a tool that manage complete life cycle of physical and virtual servers. With Foreman, it is possible to automate repetitive tasks, deploy applications and manage on-premise and cloud servers. (*Foreman* 2019)

3.4 Ceph

Ceph is Software-Defined Storage (SDS) platform. Software-defined storage is virtualization, which integrates different storage devices and storage resources. Ceph has been used for storage at CERN for 5 years. (Yang, Lien, Shen & Leu 2015)

Ceph is distributed file system designed to provide reliability, scalability and excellent performance (Weil, Brandt, Miller, Long & Maltzahn 2006). It provides network-attached block devices for OpenStack, object storage with S3 and NFS like file system with CephFS. CERN made world first entry for CephFS in IO-500 list that was presented at Super Computing 2018 and is currently rated at 21st position (*IO-500* 2019).

3.5 Version control

CERN uses on-premises GitLab installation for hosting source code and Puppet configurations. GitLab continuous integration is used to run automatic test on new commits and merge requests.

4 Computing systems

4.1 Computing at CERN

At CERN, most of the jobs by High Energy Physics (HEP) community are executed on HTC, which is why CERN batch and grid services focus much more on HTC. However, there are numerous applications benefiting hugely from HPC environment, for example engineering simulations. HPC is used for massively parallel applications, scaling across many cores on MPI-enabled infrastructure. (Llopis, Lindqvist, Høimyr, Ster & Ganz 2019)

Engineers at CERN use a variety of different applications in their work. Engineering applications are used for designing parts of the Large Hadron Collider (LHC) machines and detectors. These applications are used to run, for example, safety-related simulations and all different kinds of simulations. (Husejko, Agtzidis, Baehler, Dul, Evans, Høimyr & Meinhard 2015)

Miron Livny gave a good explanation about the differences between HPC and HTC: ”HPC deals with floating-point operations per second, and I like to portray HTC as floating-point operations per year.” - Miron Livny (Beck 1997)

4.2 High-Performance Computing

Supercomputers are used for HPC. HPC is low latency computing distributed between multiple nodes. These machines are physically connected to each other with low latency networking, for example Infiniband networking or low latency Ethernet. Today, big portion of computing is performed with CPU; however, GPUs are getting more common in the industry.

Software stack used to manage and use these supercomputer includes queue systems, resource management and parallellisation libraries. Parallellisation libraries help distribute computing across number of nodes and processes. One of the most common is MPI.

MPI is a standard that provides a message-passing parallel programming model. The official language bindings are for C and FORTRAN. (Forum 2012)

HPC has many use cases, most of which are some sort of simulations in different fields.

Some of the uses cases are or have been:

- Weather forecasting (Russell 1978)
- Aerodynamic research (Russell 1978)
- Probabilistic analysis (Joshi 1997)
- Brute force code breaking (*EFF DES Cracker Source Code* 1998)
- High-energy particle collision simulation

All nodes in the cluster also share a file system; hence, it is possible to share files between nodes. Simulations can use the shared file system if files that are updated or written need to be shared with other nodes. If files that nodes write do not need to be shared between nodes, it is possibly to use local storage and after the calculations move the results to a shared file system. Usually if simulations do not write and read much from the disk, it should be in order to use shared file system; however, if the results from one node do not need to be shared with other nodes, using the local disk is usually much faster, because there is no need to transfer files over the network.

In a simulation application, some simulation solvers are better optimized for scaling for multiple nodes and these are run in HPC cluster.

4.2.1 Windows HPC cluster

Windows HPC cluster was designed to integrate well with Windows and most engineering applications have good Windows HPC integration.

The Windows HPC is an upgrade package for Windows Server that can be installed on top of the Windows server. All users who want to use the Windows cluster have to install Windows HPC package on their machines. This package gives users the ability to submit and monitor the submitted jobs. (Husejko et al. 2015)

This cluster had few different queue types and each queue had different kind of hardware specifications; therefore, users could specify what kind of hardware they wanted to run by selecting appropriate queue. Windows HPC includes few different types of machines, as seen in Figure 3.

The goal of the migration is to deprecate Windows HPC service. Windows cluster was

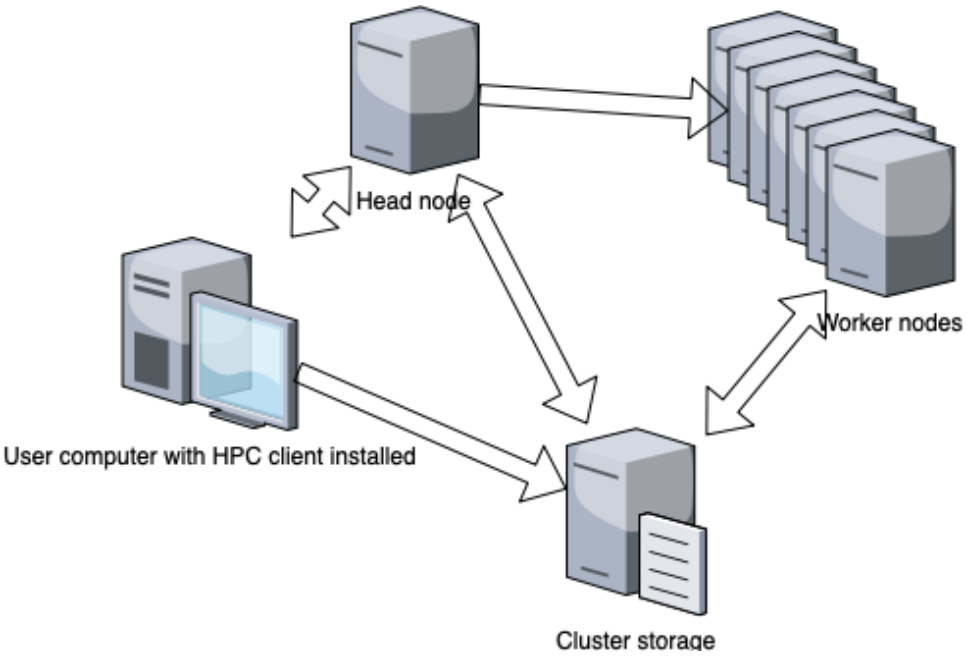

Figure 3. Windows cluster

used at CERN, because it provided excellent integration for Windows and engineering application and because significant portion of engineers at CERN uses Windows desktop. The decision to deprecate Windows services was influenced by CERN deciding to move focus more to Linux-based services.

4.2.2 Linux HPC cluster

Linux HPC cluster takes advantage of the agile infrastructure for configuration, updates and deployment. The HPC cluster uses the same tools to orchestrate the cluster as the CERN internal OpenStack cloud. Currently, HPC cluster contains a total of about 5,600 physical cores and 11,200 with hyperthreading enabled. Linux cluster at CERN uses Slurm for HPC job scheduling. (Llopis et al. 2019)

In the cluster, there are multiple different node types: submit node, head node, worker node and database nodes. The nodes that are relevant for the migration are submit node, head nodes and worker nodes.

- Submit nodes are used by users to submit and monitor jobs with Slurm.
- Worker nodes run the jobs independently.
- Head node manages the jobs, but does not run anything.

- Database node archives job accounting records

(*Quick Start Administrator Guide* 2019)

There are two different node resources types, which all are running CentOS 7. These described in Table 1.

Table 1. HPC node types

Cores	**RAM**	**Connection**
16	128	Low-latency 10Gbit Ethernet
20	128	Infiniband

The cluster installation has been divided into to two installations: production cluster and testing cluster. The testing cluster is much smaller than production cluster, and it can be used for testing new configurations without breaking the production cluster.

The main difference between the production cluster and the testing cluster is their size. Testing cluster is significantly smaller, it only contains the minimum amount of nodes required to test and validate new configurations. This test cluster contains:

- 2 Head nodes
- 1 Submit node
- 2 Database nodes
- 5 Worker nodes (in two queues)

Figure 4 shows different type of nodes and how they communicate bewteen each other.

In Puppet hierarchy the test nodes are situated in hostgroup below the production nodes hostgroup. Having test node below production hostgroup means that the test nodes inherit the same configurations that the production nodes have, however, this makes it possible to add testing specific configurations.

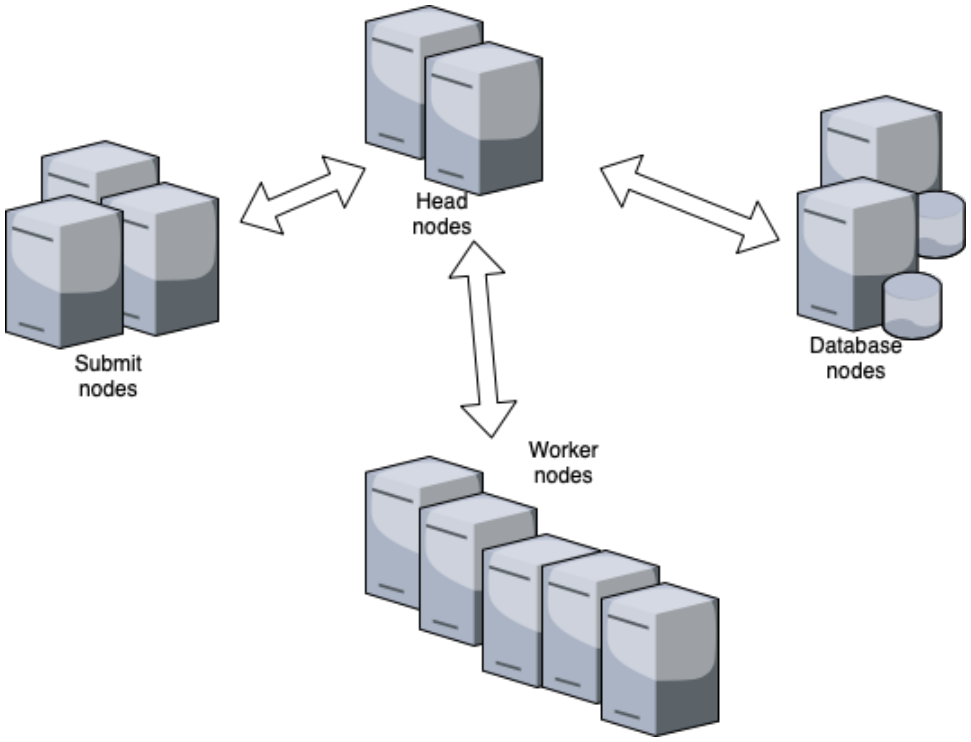

Figure 4. Linux cluster (Llopis, Lindqvist, Høimyr, Ster & Ganz 2019)

4.3 Slurm

Slurm is cluster resource manager with three main tasks. It allocates access to compute nodes for a specified time and user. Slurm provides a framework for job lifetime management (start, execute, monitor). Lastly, it provides a queue for jobs. (Yoo, Jette & Grondona 2003)

Slurm is used as workload manager on about 60% of the TOP 500 supercomputers (*Running a Job on HPC using Slurm* 2019).

Users interact with the cluster mainly through a command line. Jobs are submitted from the submit node using one of two commands, *sbatch* or *srun*. Command srun runs the specified command and shows the run in the shell. With sbatch, one can define all the parameters required to run the job in the shell script file by using special comments.

```
1 #!/usr/bin/bash
2 #SBATCH -p queue
3 #SBATCH -t 1:00:00
4 #SBATCH -n 16
5
6 ./program parameters
```

After submitting the job script, Slurm will put the job to a queue and return the job ID. With ID one can query status of one's job.

4.4 Linux Control Groups

The cluster uses Linux Control Groups to limit the amount resources the applications can access on the nodes. Linux Control Groups offers the grouping tasks and binding groups to a subset of resources, such as processor, memory and I/O bandwidth (Bellasi, Massari & Fornaciari 2015). Linux control groups ensure that user applications do not use too much resources and that there is enough resources for the system to function properly.

4.5 High-Throughput Computing

Where HPC is ranked as floating point operations per second, HTC delivers large amounts of processing over long periods of time.

In HTC, most important aspects are effective management and using all available computing resources possible. (Livny, Basney, Raman & Tannenbaum 1997)

Simulations that do not benefit from parallelism or do not scale to multiple nodes are better run in HTC.

Currently, CERN HTC batch farm contains about 175,000 hyper-threaded cores and these resources are divided in few different node types which are described in Table 2. CERN uses HTCondor for HTC batch infrastructure. (Llopis et al. 2019)

Table 2. HTC node types

Cores	**RAM**
8	2GB/core
16	2GB/core
24	5 GB/core
24	1 TB
48	500 GB

Submission with Condor takes place with ClassAd. ClassAd is a file where all the parameters needed to run the job are specified, for example executable, environment vari-

ables, output files and what to run and how many times.

```
executable = script.sh
arguments  = $(ClusterID) $(ProcId)
output     = output/$(ClusterId).$(ProcId).out
error      = error/$(ClusterId).$(ProcId).err
log        = log/$(ClusterId).$(ProcId).log

queue
```

After submitting ClassAd to Condor grid, this ClassAd is advertised to all the machines and if there is a machine that fills all the requirements, the job will be run on that machine.

4.6 Scientific and engineering applications

Applications to be migrated in this project, originates from three commercial companies: Ansys, DAssault Systems and Comsol.

Ansys family of products includes many types of simulation applications. This thesis concentrates mainly on Ansys mechanical, Ansys Fluent and Ansys CFX. (*Ansys homepage* 2019)

- Ansys Mechanical is software solution that uses finite element analysis for structural analysis (*Ansys Mechanical Enterprise* 2019).
- Ansys Fluent is used to simulate flow, turbulence, heat transfer and reactions for industrial applications (*Ansys Fluent* 2019).
- Ansys CFX is high-performance fluid dynamics software. It is used to simulate turbomachinery; for example: pumps, fans, compressors also gas and hydraulic turbines (*Ansys CFX* 2019).
- CST or Computer Simulation Technology by DAssault Systems is used for electromagnetic design and analysis (*Computer Simulation Technology Homepage* 2019).
- Comsol is multiphysics simulator software for scientific and engineering applications (*Comsol homepage* 2019).

Most application from these companies support and can make use of HPC infrastruc-

ture for simulations. HPC engineering applications can make use of both distributed computing and multi-core platforms. (Husejko et al. 2015)

5 Migration Plan

The starting point of the migration is as follows: all of the simulations run on Windows HPC cluster and there is no documentation for users on how to run these simulations on Linux HPC cluster or HTC grid. All of the applications have Linux support, however, the support for Slurm and HTCondor is not built-in in all applications. In addition, some of the applications use built-in MPI libraries, which might not be optimized for CERN HPC cluster, and there might be compatibility problems. There is a risk that some of the advanced features might not work with CERN's Linux resources.

For initial testing of the applications on Linux cluster, a small testing cluster is used. Changes to cluster is made if necessary. Template script to submit job to batch system are made and documented.

After the tests in the testing cluster are passed without problems, the simulations are tested with the production cluster. After successful tests in the production cluster, documentation on how to run the simulation in production is given to early adopters.

When testing applications with HTCondor, the production grid is used. After successful tests, the documentation about running the simulations is given to early adopters.

This process is laid out in Figure 5.

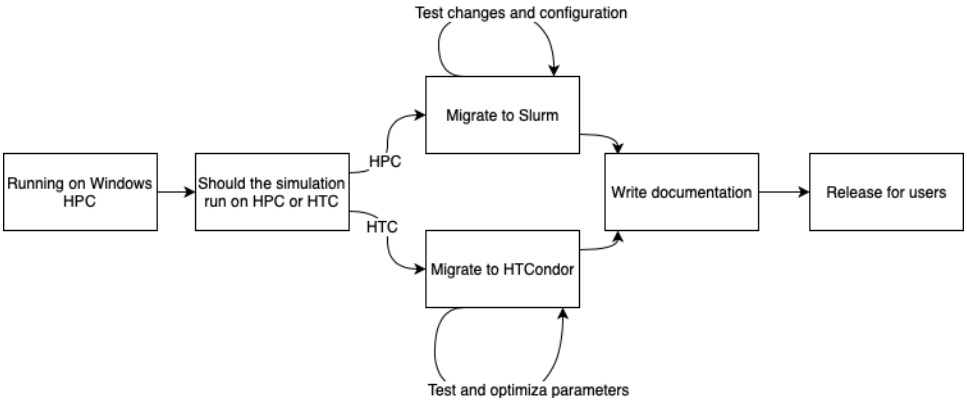

Figure 5. HPC or HTC migration process

The goal of the migration is to move all engineering applications from Windows HPC cluster to either Slurm HPC cluster or HTCondor HTC grid, depending on the simulation

type. If the simulation is optimized for parallel computing and scales across multiple nodes, it is migrated to HPC cluster. If the simulation does not scale for multiple nodes, does not make use of parallel computing, or it needs much memory, it is migrated to HTC grid. Some applications need to be migrated to both HPC and HTC because it might depend on the problem how these applications scale. Table 3 shows what application are run on which batch system.

Table 3. What batch system application should run

Ansys classic/mechanical	HTCondor
Ansys CFX	HTCondor or slurm. generally HTCondor
Ansys Fluent	Slurm
Comsol	HTCondor
CST	HTCondor

All the technologies used in migration were defined by the existing infrastructure. They all needed to be fitted on this existing infrastructure.

Most issues are likely to be related to missing libraries and communication issues between nodes. CERN used Kerberos for authentication but Slurm does not support Kerberos out-of-the-box. This could be fixed using a Slurm plugin called Auks. One limitation with Auks and Kerberos is that with current Kerberos configurations it is not possible to renew Kerberos ticket for more than one week. for jobs that need to be run more that week there is a workaround with SSH keys; however, the problem with SSH keys is that they allow users connect to machines where there are no jobs running. A possible fix for this is a package called slrum-pam_slurm. This package makes sure that users can only access machines there are jobs running.

6 Development workflow

The workflow of working with the HPC cluster is quite simple. All configurations are stored and versioned inside Git repositories and hosted on CERN's own GitLab installation. Puppet configurations are separated into modules containing isolated configurations, for example, different services, packages or applications. This workflow is outlined in Figure 6.

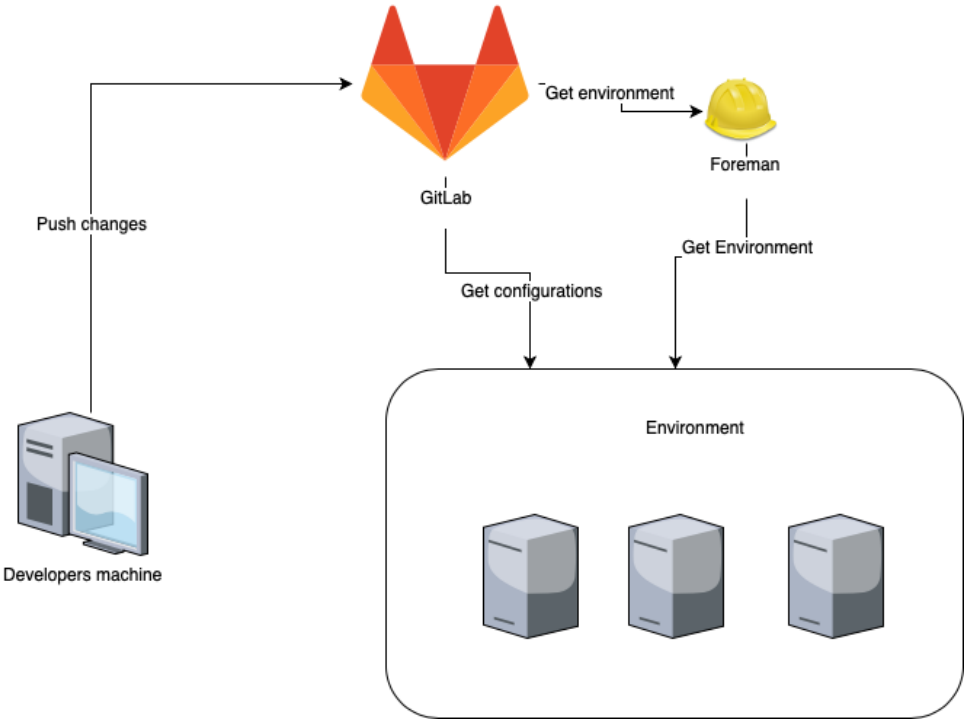

Figure 6. Development environment

To start experimenting with new modifications, one first has to create a new environment for the changes. With the new environment, one can overwrite Puppet modules to use, for example, testing or feature branches, instead of a production branch. After the new environment is created, the machines with the configurations to be tested are moved to that environment with Foreman.

When Puppet is run on the machine, it will update the configurations with its environment's configurations. For the migration, a new Puppet module for Auks was created and this is described in section 7.5. Changes for the modules are created in feature branches and that branch is defined in the environment, so the machines use the fea-

ture branches for configuration. After the changes are in Git, the running of Puppet on the machine will update the configuration with the latest configurations from the Git.

Foreman is used to manage machines environments and what hostgroup they belong. State of that machine can be queried and updated with web interface, as well as with command line tool. Creating new environment, work by creating new file inside Git repository of environments. For example to create new environment called test_env which would override Slurm module to use testing branch called feature1, but use master branches for everything else. One would first create test_end.yaml with following content:

```
---
default: master
notifications: developer@example.org
overrides:
  modules:
    slurm: feature1
```

This file is pushed to repository and after that the new environment is available in Foreman. To add machine to the new environment one can use Foreman's web interface. Puppet is run periodically on machines so after a while the machine will get changes from the new environment, however, it is possible to run Puppet manually on nodes.

All of the testing and first development is happening inside the testing cluster, so errors in configurations do not break the production cluster. When testing changes for first time, is usually easier to first test the changes by applying them manually in the testing cluster. It is good idea to disable Puppet service on the machines while testing the changes, otherwise Puppet might override one's changes. After configurations have been confirmed to be working, these changes should be Puppetized either in new Puppet module or existing one. These changes will first be applied to the testing cluster, and later to production.

7 Migration

7.1 HTC and HTCondor

Migrations to HTCondor were really straightforward, because most applications on HTCondor only run on one node. To run applications on HTCondor there are only a few things to check:

1. Application is installed in a location where it can be accessed from HTCondor node.
2. All dependencies are installed.
3. Required environment variables are defined.
4. Correct application parameters are used for the simulation, application and environment.

To make it easier for users to launch jobs, template scripts for launching different applications were developed.

Some applications required X virtual framebuffer (XVFB) to function properly.

7.2 HPC and Slurm

Migration started by checking if the applications had any support for submitting jobs to queue systems. It was decided that best way would be to create template scripts where all the required parameters would be defined for every application.

Usually these applications take simulation specific parameters, e.g. what version to use, 2D or 3D simulation, what license to use or what type of solver to use. When trying to scale the application across many nodes, usually one has to provide more parameters that describe how to scale the application, e.g. number of tasks, hosts where to run or should MPI be used.

Here example from part of the script which is used to launch Fluent. This script is launched with either srun or sbatch:

```
1 HOSTSFILE=.hostlist-job$SLURM_JOB_ID
2 srun hostname -f > $HOSTSFILE
```

```
3 if [ "$SLURM_PROCID" == "0" ]; then
4   /path/to/fluent/bin/fluent 2d -i $PATH_TO_JOURNAL -t$SLURM_NTASKS -
      cnf=$HOSTSFILE -mpi=intel -g
5
6   rm -f $HOSTSFILE
7 fi
```

Here is explanation of the parameters

- 2d, version to use for simulation. Could be 2d, 2ddp, 3d, 3ddp
- -i $PATH_TO_JOURNAL, Journal file is used for automating interaction with Fluent
- -t$SLURM_NTASKS, Defines number of processors to use
- -cnf=$HOSTSFILE, Path to file that defines all the hosts
- -mpi=intel, Fluent come with couple different MPI versions.
- -g, application should be run without GUI

In the example, the hosts file is defined dynamically, so that users do not have to manually reserve nodes, list them in a file and the give that file to the application. Name of the HOSTSFILE is appended with slurm job id, which is retrieved from SLURM_JOB_ID environment variable set by Slurm. Command hostname is run on every reserved node by using srun command, and these outputs are appended to the hosts file.

Some applications need to be launched only from one node and other applications needs to be launched on all the nodes. If the application launches other processes on other nodes, the main process should be launched only once and not on every node. This can be done by executing the application inside "$SLURM_PROCID" == "0" conditional. This ensures that the application is only ran once. Some application should be run on every node, and in those cases this conditional is not needed.

The most common problem when migrating applications to Slurm was the way in which the job communicates between nodes. The Linux cluster uses Kerberos for authentication. Many of the engineering applications used SSH for communicating between the nodes, however, forwarding Kerberos ticket to Slurm job is not supported, which results in jobs missing a valid Kerberos ticket. Running the job without a Kerberos token will cause that the job is unable to access the shared file system and other nodes without interaction. The applications using SSH for communication usually required passwordless

authentication between nodes.

Many of the applications lacked the option to run jobs custom MPI. Some applications provided multiple MPI flavours. Choosing the MPI version made a difference for some applications. Some MPI versions were obsolete and not compatible with the cluster; hence, they did not run at all.

These communication issues were resolved by introducing Auks plugin to Slurm. This is described more in detail in section 7.5. Because Kerberos configuration imposes a one-week limitation for the ticket renewal, Auks does not work for longer jobs. One way to fix this is to use SSH keys, however, this gives users access to all nodes, even the ones where there are no jobs running. Also with setup of SSH keys, there is some room for errors.

The problem with simulation using SSH to communicate is that the Linux Control Groups are not applied to new processes. Also when using SSH, Slurm does not know about the other processes and this poses a problem when applications crash, or job is cancelled. If application crashes on the process that was launched through Slurm, the nodes are freed; however, the processes that were launched through SSH are left as zombie processes. The fix for this is to use pam_slurm_adopt Linux Pluggable Authentication Module (PAM).

7.3 Slurm PAM

Slurm adopt PAM module is provided with Slurm in slurm_pam-slurm package. This module restrict user's access to nodes where they do not have jobs running. The module also ensures that Linux control group restrictions are applied to processes launched through SSH, and also makes Slurm aware of these processes. The benefit of Slurm being aware of these processes is that if the application crashes, Slurm is able to terminate all the user's other processes on all the other nodes, thus not leaving any zombie processes.

Linux Pluggable Authentication Module (PAM) provides pluggable authentication support for services and applications. With PAM one can change the way application or services authenticate without changing the applications themselves. (Morgan & Kukuk 2006)

When executing some simulation applications, it was noticed that nodes became unresponsive. The applications were able to use more resources than it was assigned to. When checking Linux control group status for the jobs it was noticed that the control groups were not applied to these processes. When launching job without Slurm PAM the control groups would look like this:

```
11:devices:/user.slice
10:freezer:/
9:pids:/user.slice
8:blkio:/user.slice
7:cpuacct,cpu:/user.slice
6:hugetlb:/
5:memory:/user.slice
4:perf_event:/
3:net_prio,net_cls:/
2:cpuset:/
1:name=systemd:/user.slice/user-110578.slice/session-49608.scope
```

but after enabling the Slurm PAM it looked like this:

```
11:perf_event:/
10:memory:/slurm/uid_110578/job_2194/step_extern
9:freezer:/slurm/uid_110578/job_2194/step_extern
8:cpuset:/slurm/uid_110578/job_2194/step_extern
7:hugetlb:/
6:devices:/system.slice/sshd.service
5:pids:/system.slice/sshd.service
4:net_prio,net_cls:/
3:blkio:/system.slice/sshd.service
2:cpuacct,cpu:/system.slice/sshd.service
1:name=systemd:/system.slice/sshd.service
```

Here the difference between the control group settings can be seen. After enabling Slurm PAM, the application has control group settings applied by Slurm. As mentioned before, the problem was with SSH communication.

PAM module was tested with few different tests. First after enabling the module, logging in to a node without and with job reservation was tested, and module worked correctly. It blocked the login to nodes where there were no jobs running by the user.

Other test was to reserve two nodes with short time limit, launch script that would

connect to second node through SSH in the background and would run command that took longer to execute than the time limit. With this it could be seen if Slurm was able to kill the application so that it would not leave any zombie processes hanging. When this test was tried without PAM module it leaved a zombie process hanging on the other node. With PAM module enabled, Slurm was able to kill processes from the other node.

7.3.1 Puppet

When starting to think about puppetizing this module, first procedure that was tried was to include configuration for the Slurm PAM in the Slurm Puppet module. The idea was to provide flag for users of the plugin, who could set the flag as true, and the puppet module would take care of the configuration.

Installation of this PAM module requires editing a few lines in /etc/pam.d folder, and the order of the lines is important for the system.

There were three options to edit these files. First was to use Puppet's file_line function, but this was discovered not to be flexible enough to allow it to used in general way. Second option was to use puppet modules which were meant for managing PAM files. This probably would be the cleanest option. Last option was to import whole files in puppet as template files.

It was decided to not to include this inside the Slurm puppet module, but to put the configuration inside CERN specific configuration. From the three options, last one was selected, because it offered safest way to enable the plugin, without using great amount of resources into enabling PAM management Puppet module.

Changes were made into CERN's Puppet configuration files to ensure that slurm-pam_slurm package would be installed. Also the imported PAM files were copied to system from puppet. Lastly PrologFlags settings was set to 'contain' in slurm configuration.

7.4 Applications

7.4.1 CST

The developer of CST, DAssault System, provides a submission script with the application. This script can be used to submit simulations to different batch systems. It generates a command for the selected batch system that will launch the simulation with the correct parameters.

The problem with CST was also with the node communication. CST assumes SSH access to any node and mpiexec launcher is hardcoded to use SSH. One way to fix this is to use SSH keys, and this is discussed earlier in section 7.2.

After running issues with HPC migration, focus was moved to HTCondor. There are few solvers in CST that benefit from HPC, but most of the simulations are likely to be run on HTC.

Because HTC jobs only run on one node, there were no problems with communication. One dependency was missing from HTC nodes, and that was XVFB package. This package was installed. Also HTCondor submission file and script that launches the CST design environment was required. In the HTCondor submission file, one needs to define all normal parameters, executable, output, error, log, queue and environment.

```
environment             = CST_INSTALLPATH="/path/to/cst/installation/
    cst2018"; CST_WAIT_FOR_LICENSE=on; CST_LICENSE_SERVER="123
    @licenseserver1","123@licenseserver2","123@licenseserver3"; HOME="
    Path/to/home"
```

In the launch script one needs to launch the CST application itself with right parameters for the jobs, for example:

```
/path/to/cst/installation/cst2018/cst_design_environment -m -e ./
    simulation_file.cst
```

This uses CST MICROWAVE STUDIO which is defined by '-m ' parameter, and the simulation uses Eigenmode solver which is defined with -e parameter.

Problem with the HPC migration was found to be mainly because of the SSH usage. After the Auks plugin was enabled the communication part started working correctly. The Auks process is described in section 7.5. After trying to run few test simulations

it was discovered that the nodes became unresponsive. This was discovered to be, because when using SSH, the Linux Control Group limitations were not applied. So without limitations, the simulations were able to use more resources than what was reserved for the job. Fix for this was to install Slurm PAM module. This is describe in section 7.3

7.4.2 Ansys

Ansys version 19.1 was used for testing. There are two possible ways to launch Ansys simulations: either from command-line or with the Remote Solver Manager (RSM). There are advantages and disadvantages to using both. With RSM, users can launch jobs directly from their Windows desktop using Graphical User Interface (GUI), however, when launching from command-line one have to take few more extra steps to be able to submit jobs.

With command-line users, one have more control over how to run the simulations. The command-line has more parameters and options, which makes it more powerful but because of that, it has a steeper learning curve.

With RSM, one can configure and monitor job submissions from central framework. RSM is configured with wizard-line interface. Configurations enable integrations with third-party job schedulers. RSM includes support for multiple HPC resource types, for example, Windows HPC, Load Sharing Facility (LSF), PBS Pro, TORQUE with Moab, Univa Grid Engine (UGE)/Sun Gring Engine (SGE) and ANSYS RSM Cluster (ARC). RSM also provides a to extend its functionality with plugins. (*Remote Solve Manager user's guide. v19.1* 2019)

RSM works by taking a user's job and wrapping it in its own submit script. After this, RSM copies the files to be configured staging a directory; in this case a shared file system in which RSM used SSH to copy all the relevant files to the headnode, where the files were placed in this shared file system. When all the files have been copied to staging directory, RSM submits the job to the cluster using shell commands.

It also makes difference, where one launches the simulation from Ansys. If user uses Ansys Workbench, there are two different ways to submit simulation. One way is to define whole solution to use RSM. This way, whole Workbench project is sent to batch system and whole project is updated remotely using Workbench. Other way is to define

only the solution to use RSM. This way only the simulation is sent to the batch system to be updated with that specific application. Both of the update methods can be seen in Figure 7.

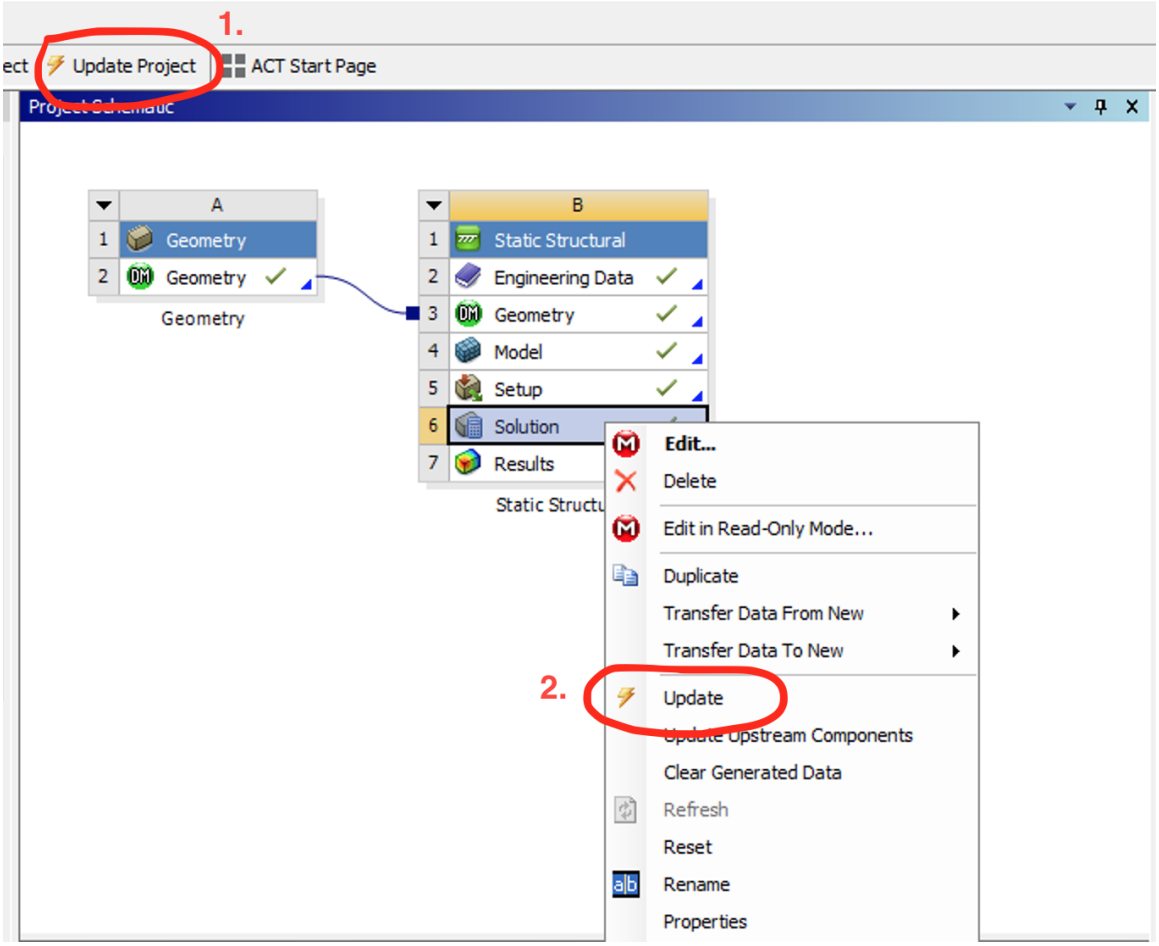

Figure 7. Ansys Workbench update

The easiest way to launch jobs from RSM to Slurm is to use Portable Batch System (PBS) plugin. Slurm has PBS compatibility layer, which enables launching jobs using PBS commands. Ansys RSM supports PBS and using this to launch jobs to Slurm seems to be working without problems.

```
#!/bin/bash
#SBATCH -p queue-name
#SBATCH -t 1-0
#SBATCH -N 2
#SBATCH -n 2

PATH_TO_JOURNAL="./journalfile"
VERSION="2d" # 2d, 2ddp, 3d, 3ddp
```

```
9
10 HOSTSFILE=.hostlist_job$Slurm_JOB_ID
11 srun hostname -f > $HOSTSFILE
12 if [ "$Slurm_PROCID" == "0" ]; then
13   /path/to/ansys191/v191/fluent/bin/fluent $VERSION -i
       $PATH_TO_JOURNAL -t$Slurm_NTASKS -cnf=$HOSTSFILE -mpi=intel -p -g
14
15   rm -f $HOSTSFILE
16 fi
17
18 exit 0
```

Launching jobs to HTCondor using RSM is not supported by default; however, it is possible to extend RSM with new plugins. The plugins use XML and Python for their functionality. This is described in more detail in section 7.4.3.

It is possible to export simulations from these programs for running in the cluster. First, one needs to export the simulation in the right format, transfer it to a storage where the cluster nodes can access it, and launch it from command line with the right parameters.

Ansys products use automation scripts called journal files for automating the launching simulation and settings to launch parameters for the simulation.

Ansys Fluent had the same common communication issues. Fluent used SSH for communication between nodes. The access problem was fixed with Auks plugin, but Fluent started leaving zombie processes on the nodes. Fix for the zombie job is described in section 7.2, and it uses PAM module to fix this.

7.4.3 Ansys RSM HTCondor plugin

Plugin is divided into two main parts: commands XML and parse script. In XML, one defines the shell commands run on the submit node and python script is used to parse the outputs of those commands.

In the XML file there are multiple functions defined. These functions are:

- Submit
- Cancel

- Query status
- Query queues
- Get all queues
- Get all status

Each one of these functions is defined with XML block. Here is example of a submit declaration for HTCondor. First submit tag defines that this whole block is for the submit command. Inside the block, there is primaryCommand and postCommands blocks. The primaryCommand block is executed first and the output is passed to postCommands.

How the job is submitted to batch system is defined in the app block that is inside the primaryCommand. This command is executed in the submit node of the HTCondor.

The postCommands block handles the parsing of the output and handing the parsed information back to the Ansys, so that users can monitor their jobs inside the Ansys GUI. Python script is called with *submit* parameter. This parameter informs the parsing script that what kind of output it should parse.

```
1  <submit>
2      <primaryCommand name="submit">
3        <application>
4          <app>echo -e 'executable=%RSM_HPC_STAGING%/%RSM_HPC_COMMAND%
       \n %RSM_HPC_NATIVEOPTIONS% \n +JobFlavour = "%RSM_HPC_QUEUE%" \n
       environment = AWP_ROOT191=/path/to/ansys191/v191/;HOME=%
       RSM_HPC_STAGING%; \n queue' | condor_submit log=condor.log output=
       condor.out error=condor.err</app>
5        </application>
6      </primaryCommand>
7      <postcommands>
8        <command name="parseSubmit">
9          <properties>
10           <property name="MustRemainLocal">true</property>
11         </properties>
12         <application>
13           <pythonapp>%RSM_HPC_SCRIPTS_DIRECTORY_LOCAL%/
       htcondorParsing.py</pythonapp>
14         </application>
```

```
15          <arguments>
16            <arg>--submit</arg>
17            <arg>
18              <value>%RSM_HPC_PARSE_MARKER%</value>
19              <condition>
20                <env name="RSM_HPC_PARSE_MARKER">ANY_VALUE</env>
21              </condition>
22            </arg>
23          </arguments>
24          <outputs>
25            <variableName>RSM_HPC_OUTPUT_JOBID</variableName>
26          </outputs>
27        </command>
28      </postcommands>
29    </submit>
```

Ansys RSM comes with IronPython. The example scripts include few utilities that give access to job submission information, e.g. job ID input, job queues input. In the main method of the parse script, the script checks the parameter which defines what type of output should be parsed. For most of the commands declared in the XML, there is matching function is the parse script. These functions might include

- parseSubmitCommand
- parseStatusCommand
- parseQueuesCommand
- parseAllQueuesCommand
- parseAllStatusCommand
- parseLoadCommand

Each of these functions first take the input and transform it to more readable form. Then relevant information is extracted from the text and *defineRsmVariableFunc* function is called. This function returns the values back to Ansys.

For RSM to be able to find these plugins, they should be defined in *jobConfiguration* XML file. Her is example of declaring HTCondor plugin, which uses SSH to communicate with the submit node:

```
1 <keyword name="HTCondor">
```

```
2       <hpcCommands name="hpc_commands_HTCondor.xml">
3       </hpcCommands>
4       <protocol name="protocol_definition_SSH.xml"/>
5     </keyword>
```

With Windows HPC, users were able to get graph of the solving process in their Ansys Mechanical GUI. The custom plugin was unable to produce these graphs for unknown reason. No documentation was found about how this would be achieved.

7.5 AUKS

AUKS is a Slurm plugin enabling Kerberos credential support. CERN's shared file systems use Kerberos for authentication, however, Slurm does not have built-in Kerberos support . To be able to access these shared file systems, users need a valid Kerberos token to locate in the worker nodes. AUKS distributes user's Kerberos credentials to all worker nodes and renews them automatically. With this plugin, users are able to access their files in these shared file systems.

In Figure 8 one can see the general communication between nodes with auks. AUKS works by first copying one's Kerberos Ticket Generating Ticket (TGT) from login node to AUKS management node when one submits a new job. After the job has been submitted, the worker nodes fetch Kerberos tokens from the management node and store them in the worker node. The worker nodes authenticate with AUKS management server by using their Kerberos principal.

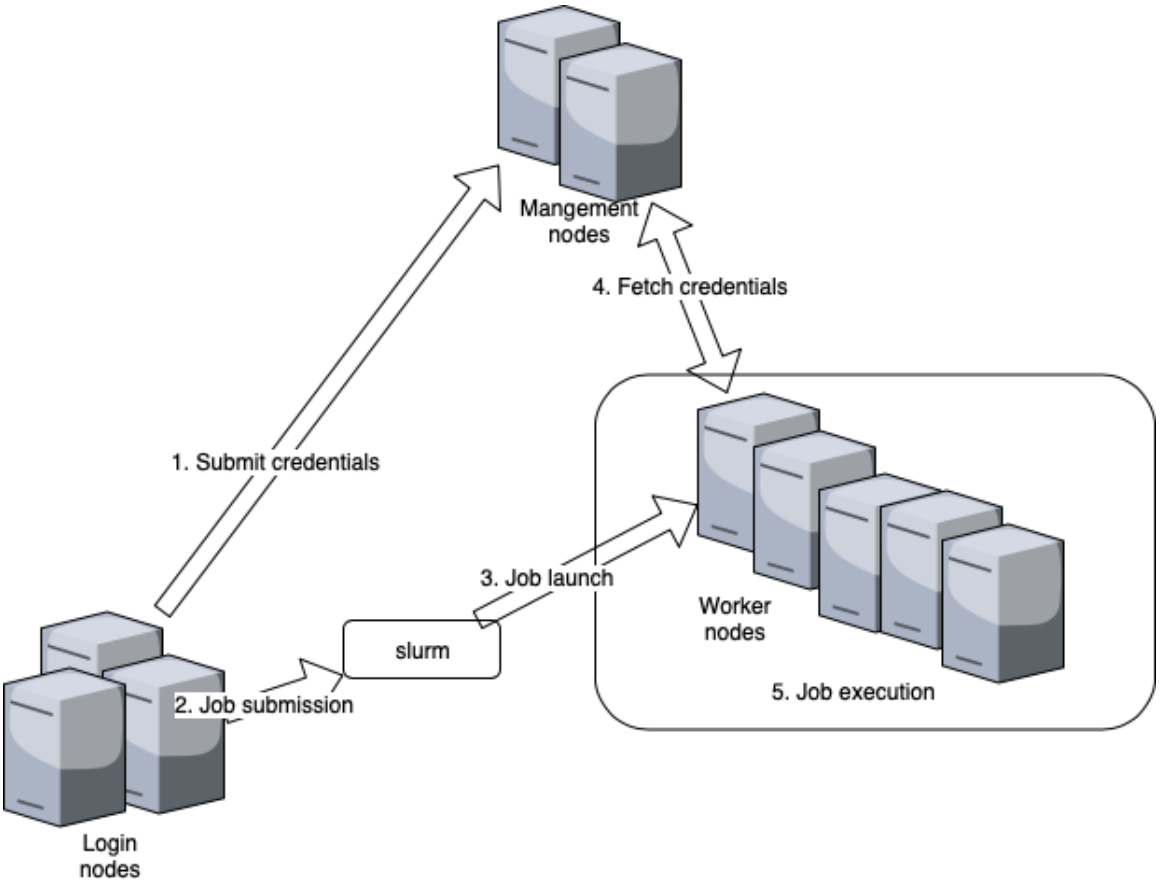

Figure 8. Auks

AUKS consist of four components:

- Auks server, named auksd service, is the central repository for credentials
- auksdrenewer service, which is responsible of renewing user credentials on the server
- aukspriv service, which ensures that credential cache is accessible for auksdrenewer and slurmd
- Slurm SPANK plugin, this plugin communicates with auks server from Slurm

Auks server provides application programming interface (API) which is used by the slurm plugin to request user credentials.

There are also three configuration files. Slum plugin configuration file that resides inside slurm's *plugstack.conf.d* folder. This configuration file is responsible for the plugin which slurm uses to communicate with Auks. In */etc/auks* folder two configuration files reside for the auks services, *auks.conf* and *auksd.acl*. In file *auksd.acl*, one can configure Kerberos principals in to different permission groups. There are three groups, user,

admin and guest. The *auks.conf* file is more general configuration for all auks services. Here one can configure API, auksd and renewer specific configuration, as well as common configuration for all the services.

There were many issues with the installation of AUKS. The lack of documentation was a significant contributor to these issues.

There were issues with how Auks chose what Kerberos principal to use. Auks listed all the available Kerberos principals and chose the last one from the list. The problem with this was that for some reason in CERN Kerberos setup, this principal was not working and a new working principal had to be defined manually for Auks services system variables. This was fixed by defining AUKS_PRIV_PRINC environment variable for aukspriv service.

There was also an issue with Auks credential buffer size, because CERN uses e-groups system to give user's permission for different systems, clubs and groups. Users are able to join multiple e-groups, which increases one's Kerberos credential size. AUKS has static credential buffer size, which had to be patched to be able to hold larger credential sizes.

7.5.1 Puppet module

Puppet module for AUKS was developed to automate the deployment and configuration of AUKS. This module takes care of installing the required packages for the right machines and configures all configurations properly.

The Puppet configurations are divided into multiple files. Each node type has its own files for login, management and compute.

The module configurations variables are defined in config.pp file. The variables included are:

- auks_port
- auks_primary_hostname
- auks_secondary_hostname
- auks_secondary_principal
- aukspriv_principal_domain

- auks_debug_level
- auks_log_level

The login node configurations just make sure that auks and auks-slurm is installed, and that Auks slurm plugin is installed from template file.

Compute node configurations install auks and auks-slurm packages. It is also ensured that auks slum plugin is installed from template file, and that aukspriv service is running with *AUKS_PRIV_PRINC* environment variable set in */etc/sysconfig/aukspriv*. With *AUKS_PRIV_PRINC* variable one can define what Kerberos principal the node uses when trying to get user credentials.

In management node, one can set three variables:

- auksd_acl_user_regexp
- auksd_acl_management_regexp
- auksd_acl_compute_regexp

The configurations makes sure that auks package is installed and that port is opened for auks server. In addition, it makes sure that auksd, auksdrenewer and aukspriv services are running. It is important that /var/cache/auks directory exists, otherwise auks will fail. Auks configuration file is installed from template file and *AUKS_PRIV_PRINC* variable is defined in */etc/sysconfig/aukspriv*

8 Results

The main objective of the project was to migrate scientific and engineering applications from Windows HPC resources to Linux HPC resources. The results of this migration are described in this section.

To migrate simulation applications from Windows HPC to Linux HPC or HTC, one first needs to check Linux compatibility. Developers of the applications have usually documented if their applications support Linux based operating systems. The next thing is to check what kind of resources does the application uses. Does it benefit from parallelization, does it use MPI, or does it require significant amount of memory?

Based on the applications' resource usage, it is decided if the application should be migrated to HPC or HTC. Migration process to HTC resources is described in section 7.1 and to HPC resources in section 7.2.

After this, it should be checked if the application supports the queue system in use, e.g. Slurm, HTCondor, PBS. Also it is good idea to check if the queue system in use offers any compatibility layers for other queue systems e.g. Slurm's PBS compatibility layer.

Some applications provide own scripts or gateway application which could be used for submitting jobs to queue systems, but if these does not support the used queue system, there is usually way to manually launch these applications.

To manually launch these applications, one needs to find right executable to launch and figure out the right parameters for that executable. Good way to figure those out is to look from documentation if those are provided there, if not the one can look inside the submission script. Most of the scripts have help pages built-in that can be seen by launching these with -h parameter. If this submission's script is Shell or Bash script, it is helpful to launch it with bash debug mode: bash -x. Bash debug mode will show what the script would run, and what environment variables would be set.

Often these applications require specific environment variables to be set in order for them to work properly.

The migration of the application started with checking if the manufacturer of the applications supports Linux, Slurm and HTCondor. All of the applications supported Linux, but official Slurm and HTCondor support was lacking. For running HPC applications it is

recommendable to check if there is Message Passing Interface (MPI) support and if the application allows custom MPI versions.

Submission script templates were created for Slurm and HTCondor. The templates included all the parameters required to run the applications either on Slurm or HTCondor. These template files define necessary parameters for the applications to run properly.

The most significant problem was with the HPC cluster and authentication between the nodes. This was fixed with Auks and for longer jobs with SSH keys and Slurm PAM plugin. Auks provides Kerberos authentication support in Slurm cluster, which makes it convenient for the users, because Kerberos is used for authentication across CERN services.

Auks's renewing of tickets is limited by Kerberos server configuration. In the configurations the system administrator can define the maximum renewal time. Auks would need to store the user's keytab in order to renew over this limit, however, this would create a security problem.

Other bigger problem that came up during the migrations was zombie processes that were left on the nodes, and applications escaping the Linux control group restrictions and using more resources than was reserved for the jobs. Both of these were part of the same problem which was also related to how these applications communicate between nodes.

Communication with applications which were compiled to user CERN provided MPI libraries, such as mvapich or OpenMPI, worked fine. When applications do their own implementations and did not provide way to use custom versions, this usually caused some issues, e.g. applications escaping from Slurm and resource restrictions.

This was fixed with PAM module provided by Slurm called pam_slurm_adopt. This module restrict access to nodes where the user does not have any jobs. Also when user uses SSH inside the cluster for communication, Slurm applies Linux control group restrictions and is able to track which jobs does the new processes belong. This makes fixes the problem of application escaping resources restrictions and also Slurm is able to kill all the zombie processes.

There have been previous publications somewhat related to this topic: migrating from

Unix to Windows (Group 2008) and moving HPC to Cloud (Eadline 2019).

In the end applications were migrated from Windows HPC service to either Linux HPC or HTC services. Using these application with the Linux services was documented and shared with the users. Also few improvements were made to the cluster configurations, which allows easier and more secure communication between nodes.

On the user side after the migration users of the Windows HPC now have access to more computing resources, but because these computing resources are shared between more users, there might be longer wait times. The wait time depends on many factors, how many jobs are in the queue, how much computing resources is requested and overall job priority.

From service's side, there is less maintenance work. Not having to maintain different Windows and Linux services means that it is possible to have more focus to improve and maintain the Linux services.

9 Conclusions

The objective of this thesis was the migration of engineering applications from Windows resources to the Linux-based computing resources. This thesis covered background for this objective, motivations and scope. Research methodology was chosen and described. CERN's private cloud called Agile Infrastructure was discussed and some motivations why it was created. This thesis also covered theory about HPC and HTC, as well as different scientific and engineering applications and some of their use cases.

This project was assigned by CERN. CERN's strategy to consolidate all computing resources to Linux meant that applications had to be migrated from Windows HPC to Linux resources. Running all applications in Linux-based services meant that there would be no need to maintain multiple services with different operating systems.

Biggest issues were related to finding right parameters for launching the applications and how these application communicate between nodes in HPC. All applications had Linux support; however, most of the applications make assumptions on the infrastructure, which makes it harder to fit if the assumptions are not compatible with the existing infrastructure.

The research question was: how are simulation applications migrated from Windows HPC cluster to Linux HPC cluster? The answers to the question is found in more detail in section 8.

The constructive research methodology was chosen because its goal of solving real world problems by producing new constructions suited this project well. The problem for this project was identified as how to migrate from Windows HPC to Linux resources because of the need to unify these computing services. Theoretical information was gathered from application and tool documentation, as well as research papers. By running test applications and making small changes for the existing infrastructure, more practical information was acquired. The solution was formed with the help of gathered theoretical and practical information. After forming the solution, it was tested, refined and fixes with multiple iterations.

At beginning the solutions did not work mainly because of the author's lacking knowledge about the problem area. There were issues that were hard to foresee because of

the author's lack of experience in the field; nevertheless, after iterations and with the knowledge gathered from these iterations, the final solution was formed.

All of the applications were successfully migrated to Linux resources, but some of the features were missing. Two main missing features that were noticed were licence queue and Ansys simulation status graph. Slurm had license queue support; however, it was not easily integrateable into existing license service; hence, this feature was not implemented in this project. another missing feature was with Ansys and how to get some specific information about the simulation during run time. For the missing Ansys feature, no documentation were found about how to implement this with custom RSM plugin, or if it is possible.

For the project, job submission templates were created, custom plugin for Ansys RSM and also changes to HPC and HTC services were implemented.

HPC, HTC and supercomputers in general is quite niche field with very specific use cases. These systems are not that accessible compared to some other technologies. Even though the field is quite niche, there is still significant amount of people working on these systems and a lot of investments into development.

Future research could look into containerization of these applications. These applications could be run from containers using e.g. Singularity (Kurtzer, Sochat & Bauer n.d.). The containerization has the potential to make these applications more portable and easier to migrate from one system to another.

References

Ansys CFX. 2019. Accessed 12.4.2019. Retrieved from https://www.ansys.com/products/fluids/ansys-cfx.

Ansys Fluent. 2019. Accessed 12.4.2019. Retrieved from https://www.ansys.com/products/fluids/ansys-fluent.

Ansys homepage. 2019. Accessed 18.1.2019. Retrieved from https://www.ansys.com/.

Ansys Mechanical Enterprise. 2019. Accessed 12.4.2019. Retrieved from https://www.ansys.com/products/structures/ansys-mechanical-enterprise.

Beck, A. 1997. *HIGH THROUGHPUT COMPUTING: AN INTERVIEW WITH MIRON LIVNY*. Accessed 27.9.2018. Retrieved from http://research.cs.wisc.edu/htcondor/HPCwire.1.

Bellasi, P., Massari, G. & Fornaciari, W. 2015. *Effective runtime resource management using Linux control groups with the BarbequeRTRM framework*. *ACM Transactions on Embedded Computing Systems (TECS)* 14.2, 39.

Computer Simulation Technology Homepage. 2019. Accessed 18.1.2019. Retrieved from https://www.cst.com.

Comsol homepage. 2019. Accessed 18.1.2019. Retrieved from https://www.comsol.com/products.

Eadline, D. 2019. Moving HPC to the Cloud. Accessed 7.4.2019. Retrieved from http://www.admin-magazine.com/HPC/Articles/Moving-HPC-to-the-Cloud.

EFF DES Cracker Source Code. 1998. Accessed 24.4.2019. Retrieved from https://www.cosic.esat.kuleuven.be/des/.

Foreman. 2019. Accessed 19.3.2019. Retrieved from https://www.theforeman.org.

Forum, M. P. I. 2012. *A Message-Passing Interface Standard*.

Group, T. P. 2008. *Migrating HPC Applications from UNIX to Windows*.

Husejko, M., Agtzidis, I., Baehler, P., Dul, T., Evans, J., Høimyr, N. & Meinhard, H. 2015. *HPC in a HEP lab: lessons learned from setting up cost-effective HPC clusters*. *Journal of Physics: Conference Series* 664.9. DOI: 10.1088/1742-6596/664/9/092012. Retrieved from https://doi.org/10.1088%2F1742-6596%2F664%2F9%2F092012.

IO-500. 2019. Accessed 7.4.2019. Retrieved from https://www.vi4io.org/io500/start.

Joshi, R. R. 1997. *A new heuristic algorithm for probabilistic optimization*. *Computers & operations research* 24.7, 687–697.

Kurtzer, G. M., Sochat, V. & Bauer, M. W. N.d. *Singularity: Scientific containers for mobility of compute*. *PloS one* 12.5.

Leon, J. C. 2017. *CERN OpenStack Cloud*. Accessed 19.3.2019. Retrieved from https://indico.cern.ch/event/609294/attachments/1425868/2187452/BE-BASICS-CLOUD-20170310.pdf.

Livny, M., Basney, J., Raman, R. & Tannenbaum, T. 1997. *Mechanisms for high throughput computing*. *SPEEDUP journal* 11.1, 36–40.

Llamas, R. M., Megino, F. H. B., Kucharczyk, K., Denis, M. K. & Cinquilli, M. 2014. *Commissioning the CERN IT Agile Infrastructure with experiment workloads*. *Journal of Physics: Conference Series*. Vol. 513. 3. IOP Publishing.

Llopis, P., Lindqvist, C., Høimyr, N., Ster, D. van der & Ganz, P. 2019. *Integrating HPC into an agile and cloud-focused environment at CERN*.

Loope, J. 2011. *Managing infrastructure with puppet: configuration management at scale*. " O'Reilly Media, Inc."

Lukka, K. 2001. Konstruktiivinen tutkimusote. Accessed 14.2.2019. Retrieved from https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/.

Morgan, A. G. & Kukuk, T. 2006. *The Linux-PAM System Administrators' Guide*. *Url: http://www. kernel. org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG. html*.

Our Member States. 2019. Accessed 29.4.2019. Retrieved from https://home.cern/about/who-we-are/our-governance/member-states.

Quick Start Administrator Guide. 2019. Accessed 10.4.2019. Retrieved from https://slurm.schedmd.com/quickstart_admin.html.

Remote Solve Manager user's guide. v19.1. 2019. Accessed 26.3.2019. Retrieved from https://ansyshelp.ansys.com/account/secured?returnurl=/Views/Secured/corp/v191/wb_rsm/wb_solution_job_status.html.

Running a Job on HPC using Slurm. 2019. Accessed 24.4.2019. Retrieved from https://hpcc.usc.edu/support/documentation/slurm/.

Russell, R. M. 1978. *The CRAY-1 computer system*. *Communications of the ACM* 21.1, 63–72.

Sefraoui, O., Aissaoui, M. & Eleuldj, M. 2012. *OpenStack: toward an open-source solution for cloud computing*. *International Journal of Computer Applications* 55.3, 38–42.

The birth of the Web. 2018. Accessed 19.11.2018. Retrieved from https://home.cern/science/computing/birth-web.

The Large Hadron Collider. 2018. Accessed 19.11.2018. Retrieved from https://home.cern/science/accelerators/large-hadron-collider.

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E. & Maltzahn, C. 2006. *Ceph: A Scalable, High-performance Distributed File System*. *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. Seattle, Washington: USENIX Association, 307–320. ISBN: 1-931971-47-1. Retrieved from http://dl.acm.org/citation.cfm?id=1298455.1298485.

Yang, C., Lien, W., Shen, Y. & Leu, F. 2015. *Implementation of a Software-Defined Storage Service with Heterogeneous Storage Technologies*. *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 102–107. DOI: 10.1109/WAINA.2015.50.

Yoo, A. B., Jette, M. A. & Grondona, M. 2003. *Slurm: Simple linux utility for resource management*. *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 44–60.