Sharing server nodes for storage and compute

David Smith[1,*], *Alessandro* Di Girolamo[1], *Ivan* Glushkov[2], *Ben* Jones[1], *Andrey* Kiryanov[3], *Massimo* Lamanna[1], *Luca* Mascetti[1], *Gavin* McCance[1], *Herve* Rousseau[1], *Jaroslava* Schovancová[1], *Markus* Schulz[1], *Havard* Tollefsen[4], and *Andrea* Valassi[1]

[1]CERN, 1211 Geneva CH

[2]University of Texas at Arlington, 701 South Nedderman Drive, Arlington, TX 76019 USA

[3]Petersburg Nuclear Physics Institute, 1, mkr. Orlova roshcha., Gatchina, 188300 Leningradskaya Oblast RU

[4]Norwegian University of Science and Technology (NTNU), Høgskoleringen 1, 7491 Trondheim, NO (seconded to CERN)

Abstract. Based on the observation of low average CPU utilisation of several hundred file storage servers in the EOS storage system at CERN, the Batch on EOS Extra Resources (BEER) project developed an approach to also utilise these resources for batch processing. Initial proof of concept tests showed little interference between batch and storage services on a node. Subsequently a model for production was developed and implemented. This has been deployed on part of the CERN EOS production service. The implementation and test results will be presented. The potential for additional resources at the CERN Tier-0 centre is of the order of ten thousand hardware threads in the near term, as well as being a step towards a hyper-converged infrastructure.

1 Introduction

EOS [1] is the disk-based, low-latency storage service used and developed at CERN. The CERN EOS service is deployed on 1577 nodes. A subset of 1339 of those nodes are called EOS FST (file storage servers). The purpose of the storage servers is to enable access to data contained on attached disks to clients via the network as well as performing other functions such as periodically reading and checksumming data to contribute to data integrity verification. The storage servers are similar in specification to the batch workers at CERN but with the addition of disks and host bus adapter. The storage servers are not virtualised.

2 Motivation

The CPU load of the storage servers of one of the CERN EOS clusters is shown in Figure 1. It can be seen that the idle fraction was at least 80% throughout the period.

[*]e-mail: david.smith@cern.ch

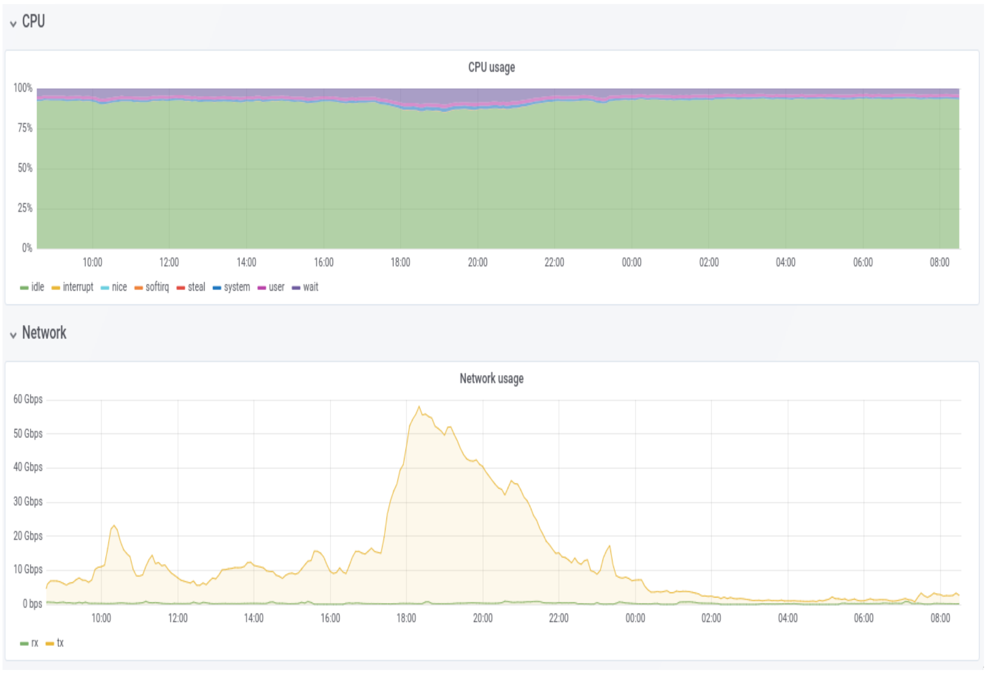

Figure 1: CPU load and network interface usage of storage servers in a CERN EOS cluster over a 24 hour period.

Table 1: Number of EOS storage servers with 2 SSDs. Each machine uses a 10Git/s network interface, and have 2x Intel S3510 or S3520 SSDs with 800 or 960GB capacity each. EOS uses 48 HDs on each storage server.

Number	Intel(R) processor	logical cores	RAM/GiB
48	2x Xeon(R) CPU E5-2630 v3 @ 2.40GHz	32	64
337	2x Xeon(R) CPU E5-2630 v4 @ 2.20GHz	40	128

Since the CPU load was seen to be low it was thought that batch tasks could be run on the storage servers with limited impact on the EOS functionality. A project was formed to use some machines for batch jobs in addition to their role in EOS. The project was called Batch on EOS Extra Resources (BEER). The approximate number of storage servers considered to be suitable for BEER are shown in Table 1. The nodes were considered suitable if they had two SSDs, so that one could be dedicated to batch functions. All the storage servers have hyperthreading enabled, and *logical cores* is used to mean the number of hardware threads available.

The combined system which is a result of the BEER project is currently in production on some servers. The initial tests and the current state of production deployment is described in subsequent sections.

3 Tests

Investigation was begun in 2016 on testbeds and continued though a number of iterations and a pilot test system. The tests culminated in a pre-production service and then a production service in 2018. Tests involving the testbeds were reported in the spring 2017 HEPiX [2].

3.1 Testbed systems

A number of test systems are described in the HEPiX report [2]. The composition of these are shown in Table 2. All the test systems used EOS v0.3 (Aquamarine release series). Testbed 1 used older hardware, while testbed 2 hardware is typical of some production servers with 32 logical cores. A pilot test system was built subsequently and is also described below.

Table 2: Composition of testbeds and pilot. The EOS MGM is the service headnode, EOS FST is the file storage server. HD is a count of hard disks used for EOS.

Type	#	RAM/GiB	cores	Network/Gbit/s	HD	OS
Testbed 1						
EOS MGM	1	64	32	10	0	SL6
EOS FST	4	12	8	1	24	Centos 7
Testbed 2						
EOS MGM	1	64	32	10	0	SL6
EOS FST	1	64	32	10	48	Centos 7
(added for pilot)						
EOS FST	2	64	32	10	48	Centos 7

The pilot system used Puppet [3] for configuration management in a similar way as many of the CERN Data Centre systems [4]. It used similar hardware as testbed 2 but consisted of 3 similar storage servers. EOS was configured to make two copies of each file, as is typical in production. The storage space for the memory swap file and HTCondor [5] batch working space is a locally attached SSD. The SSD was dedicated to these purposes. Another SSD was used for the operating system and EOS runtime data and logs.

3.2 Results from testbed 1 and 2

CPU load generation on the testbed 1 and 2 was via vLHC@Home [6]. For testbed 1 disk activity was generated by using `dd` commands on the nodes themselves. Testbed 2 contained a storage server with a 10Gbit/s interface and in that case `xrdstress` was used to generate the disk load via network access. Table 3 shows data rates achieved with and without additional CPU load for testbed 1 and 2.

Table 3: Summary of results of the 2 testbeds.

Configuration	CPU load	Read rate MB/s	Write rate MB/s
Testbed 1	off	357	96
Testbed 1	on	357	91
Testbed 2	off	~500	~770
Testbed 2	on	~500	~770

3.3 Results from the pilot system

The pilot system was integrated into the batch system so that jobs can be sent to the pilot nodes by specifying an attribute in the job submission file. vLHC was no longer used for the generation of CPU load. Instead jobs were sent manually at first and then using HammerCloud [7]. An example of the evolution of the number running and submitted jobs to the 3 pilot nodes via HammerCloud is shown in Figure 2. Often an example of a production job was run which was considered typical of an LHC experiment's Monte Carlo digitisation and reconstruction workload.

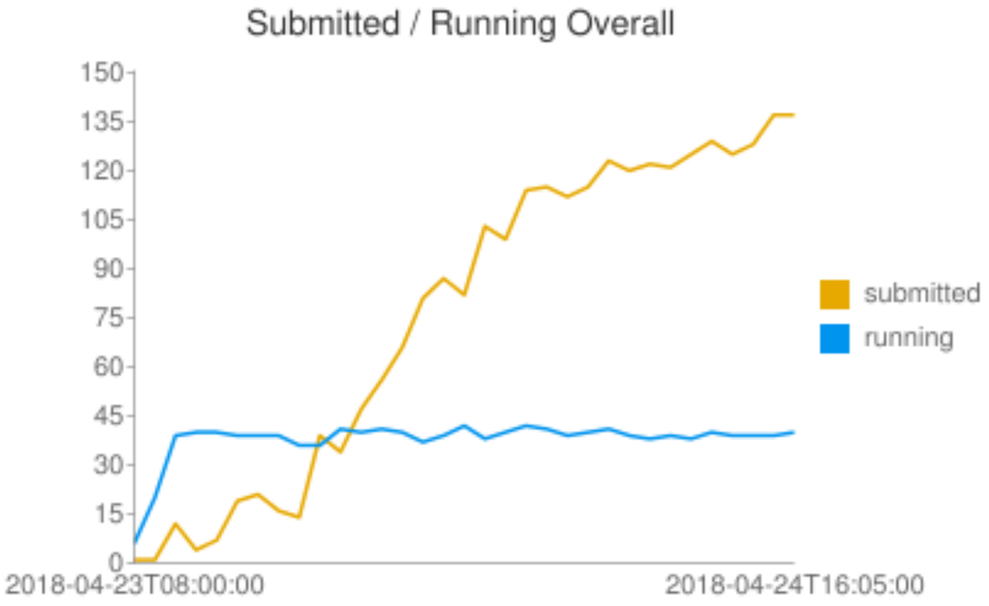

Figure 2: Number of HammerCloud jobs submitted or running on the pilot system against time.

The pilot system was intended to gain experience with the following:

- Integrating a standard EOS storage server with a CERN batch node. To make the combination the Puppet configuration of a storage server had the batch configuration added, with the intent to keep the number of changes to either minimal. A swap file had to be added because the standard EOS storage servers had no swap configured whereas batch workers have a swap file 1.5 times the size of RAM.
- Bring members of the teams involved in batch and the storage server together. The process of combining the Puppet profiles was one way to do this.
- Make the job slots created on the pilot available via the standard CERN HTCondor system.
- Measure the pilot systems in operation.

The pilot system went through a number of changes with respect to how it was initially configured. These changes are:

- HTCondor will start docker [8] containers and will launch each job in a container. The reasons for this are twofold: First it was planned to also follow this route for standard batch, and secondly it reduces the number of packages to be installed on the pilot compared to a standard storage server.

- In order to limit resources that can be used by batch jobs, and therefore be unavailable to EOS, HTCondor was to be run in cgroup [9] sets limiting a number of resources. Initially this included use of *cpusets* to explicitly ensure that certain CPU threads were not available to batch jobs. However this was changed to use *cpushares* instead. See Section 4 for a description of current cgroup use. The use of cpusets to force hardware thread affinity was abandoned due to the need to tailor the configuration across nodes with processors with varying number of cores. It also follows a principal of allowing the Linux process scheduler as much flexibility as possible.

4 Resource limits

A maximum memory limit and process count for the HTCondor processes was set in the systemd unit file `/etc/systemd/system/condor.service`:

```
[Service]
MemoryLimit = 50455563264
TasksMax = 8000
```

Systemd sets the above in the appropriate cgroup controllers. While the above would apply to HTCondor service process or processes directly started by HTCondor, the model chosen to run batch jobs was to have HTCondor start docker and then start the user's job inside the docker container. HTCondor does this by using `docker run`. To ensure all such jobs had overall limits applied the `dockerd` options in `/etc/sysconfig/docker` had a parent cgroup added. i.e. `-cgroup-parent system-htcondor.slice`. The following limits were set in the parent cgroup:

```
memory/memory.memsw.limit_in_bytes = 103079215104
memory/memory.limit_in_bytes = 50455560192
blkio/blkio.weight = 50
pids/pids.max = 8000
```

HTCondor will start the job in sub-cgroups and additionally set memory and cpushares limits derived from the job requirements.

Figure 3 shows the CPU load against time and also the network interface traffic for the same period. The machines have hyperthreading enabled and have 32 logical cores. HTCondor is configured to run up to 24 single core jobs with a maximum of 96GiB of memory. The user can submit jobs with specific requirements for number of cores or memory, in this case the jobs were multi-process and required 8 logical cores. It was concluded from Figure 3 that neither the impact of CPU load on the network activity generated by EOS data access nor the impact on CPU load by serving EOS data were readily evident.

5 Transition to production

Four machines in the pre-production storage server pool had the pilot configuration applied. The machines were required to run Centos 7 and have an SSD which could be dedicated to swap file and batch scratch space. The second SSD had to be drained if it had previously been used as EOS storage space.

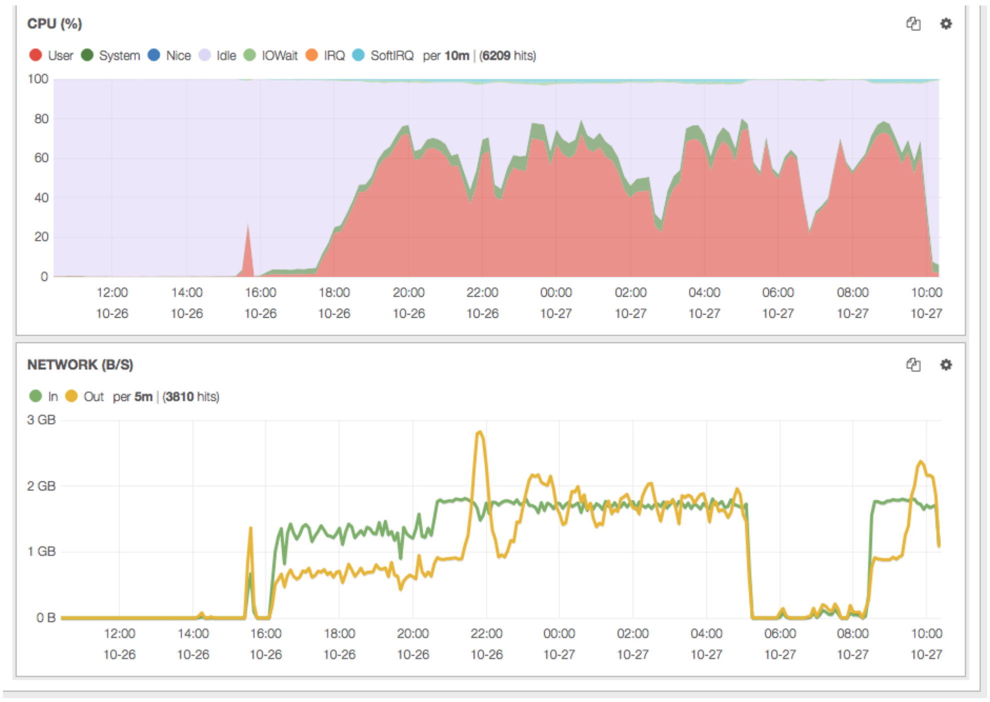

Figure 3: CPU load and network interface traffic against time of the pilot test system. The pilot system used 3 storage servers.

6 Production

The BEER configuration entered production in July 2018, initially with 70 storage server nodes associated with the ATLAS experiment. A month later a number of LHCb experiment storage servers were added. Currently there are 85 ATLAS and 41 LHCb storage servers presenting up to 3780 single core job slots.

Figure 4 shows the batch system occupancy for the BEER nodes for a week during November. At the moment use of the BEER compute resources is split by LHC experiment, such that LHCb jobs run on storage servers which are part of the LHCb EOS instance etc.

7 Next Steps

BEER resources will be made available to the CMS experiment on a number of the storage servers in the CMS EOS instance. A part of the CMS workflow requires use of singularity [10] by the job. As explained, HTCondor was set to start docker and run each job inside docker. Thus there is the situation where singularity needs to work inside docker. As of November 2018 this is not supported as the docker container does not have the correct privileges to allow singularity to work. Additional *linux capabilities* need to be granted to the docker container. A configuration change is being prepared to allow this.

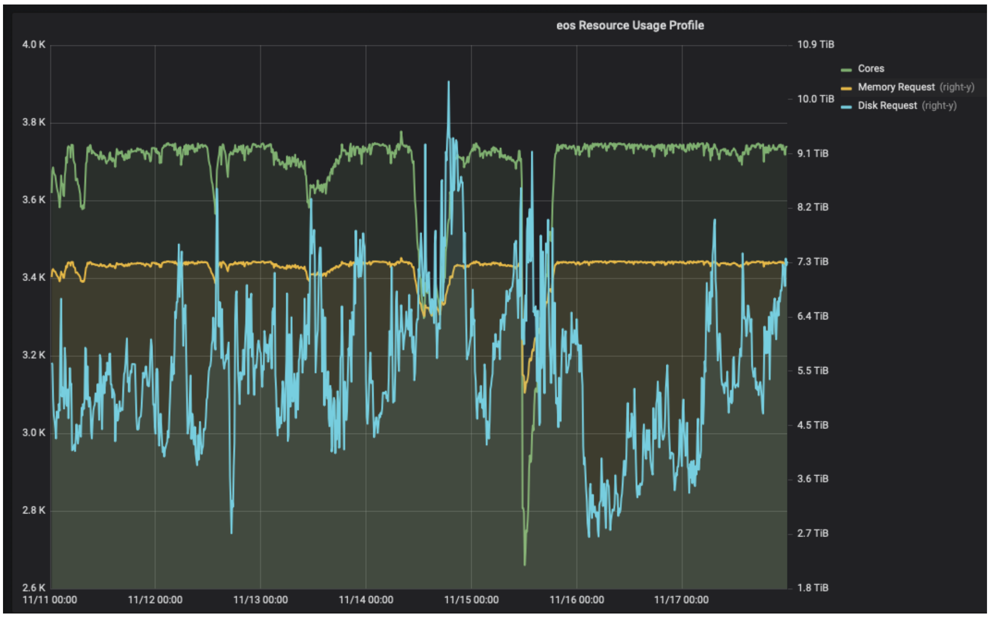

Figure 4: Batch system occupancy of BEER in production, over a week during November 2018.

7.1 Estimated resources that can be made available

Table 4 shows estimated HEP-SPEC06 (HS06) that could be made available with the introduction and expansion of BEER in the near term. The near term estimate is based on the addition of nodes listed in Table 1. The long term outlook is based on hypothetically extending BEER to the majority of currently availble storage servers. However further expansion beyond the near term is expected to be via new hardware not yet available in the Data Centre.

Table 4: Estimated HS06 that may be available: Assuming machines will offer 75% of logical cores and an estimated 10 HS06 per logical core. The HS06 count is compared to the 2018 Tier-0 pledge size for reference.

Storage servers	note	BEER use	cores	HS06	T0 pledge
385	near term	75%	11262	112620	8.9%
1200	long term	75%	30800	308000	~24%

8 Conclusion

The BEER project has allowed for the use of compute resource that exists within the EOS storage servers at CERN. ATLAS and LHCb have been using these resources. It is planned that use will be extended to CMS shortly. To date no impact on EOS operations due to BEER has been found.

References

[1] X. Espinal et al., Disk storage at CERN: Handling LHC data and beyond *J. Phys.: Conf. Ser.* **513** 042017 (2014)

[2] A. Kiryanov et al., Harvesting Cycles on Service Nodes *HEPiX Spring 2017 conference* (2017)
https://indico.cern.ch/event/595396/contributions/2532584/ [accessed 2018-11-28]

[3] Overview of Puppet's architecture
https://puppet.com/docs/puppet/4.6/architecture.html [accessed 2018-11-28]

[4] P. Andrade et al., Review of CERN Data Centre Infrastructure *J. Phys.: Conf. Ser.* **396** 042002 (2012)

[5] D. Thain, T. Tannenbaum, and M. Livny, Distributed Computing in Practice: The Condor Experience *Concurrency and Computation: Practice and Experience* **Vol. 17, No. 2-4**, pages 323-356 (2005)

[6] N. Høimyr et al., BOINC service for volunteer cloud computing *J. Phys.: Conf. Ser.* **396** 032057 (2012)

[7] J. Schovancová et al., Evolution of HammerCloud to commission CERN Compute resources *The 23rd International Conference on Computing in High Energy and Nuclear Physics, Sofia* (2018) Draft https://cds.cern.ch/record/2646247 [accessed 2018-11-29]

[8] Docker overview. Docker Documentation
https://docs.docker.com/engine/docker-overview/ [accessed 2018-11-28]

[9] Linux kernel documentation on cgroups
https://www.kernel.org/doc/Documentation/cgroup-v2.txt [accessed 2018-11-28]

[10] Singularity Documentation https://www.sylabs.io/docs [accessed 2019-02-15]