SDP Memo: PROT.ISP: File-system Benchmark Report

Document number SKA-TEL-SDP-0000061
Document Type ..... REP
Revision 01C
Author ..... W. Homberg and PROT.ISP partners
Release Date 2016-04-07
Document Classification ..... Unrestricted
Status ..... Draft

Lead author	Designation	Affiliation
W. Homberg	Scientific employee	Forschungszentrum Jülich GmbH
Signature & Date:	 Willi Homberg (Apr 8, 2016) w.homberg@fz-juelich.de	

Released by	Designation	Affiliation
Bojan Nikolic	SDP Project Engineer	University of Cambridge
Signature & Date:	 Bojan Nikolic (Apr 8, 2016) b.nikolic@mrao.cam.ac.uk	

Version	Date of Issue	Prepared by	Comments
1.0	2015-02-09	W. Homberg	
1A	2015-05-29	W.Homberg	

ORGANISATION DETAILS

Name	Science Data Processor Consortium

Table of contents

Introduction

Purpose of the document

This document summarizes the results of prototyping activities performed within the SDP subtask Integrated System Prototype [ISP] and is provided as input for the milestone M11 (Element PDR) document package.

Scope of the document

The preliminary Science Data Processor (SDP) architecture document introduces SDP as a large collection of compute islands, each of which is an independent element within the SDP capable of handling a data-parallel chunk of data [SDPARCH, p.58ff].

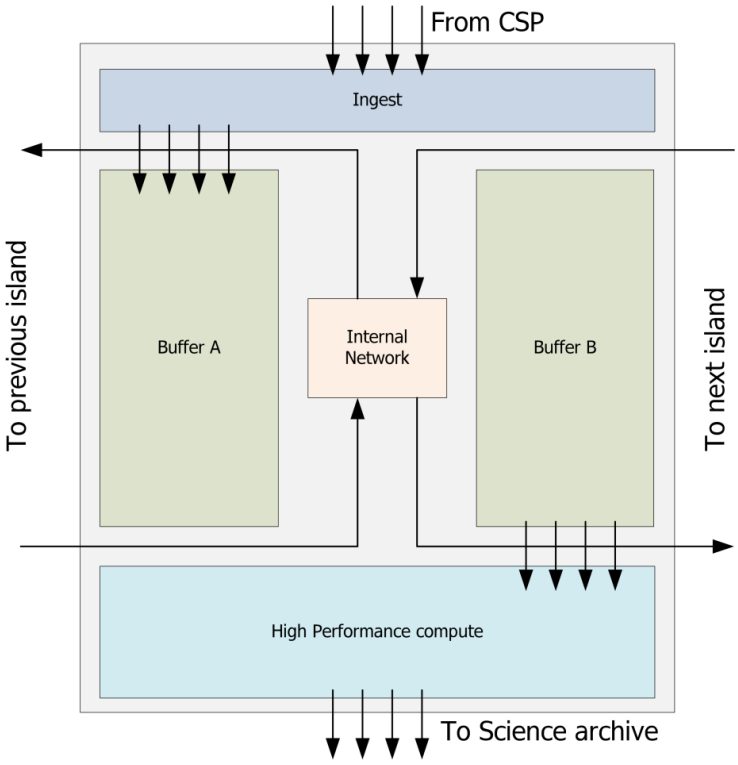
A Compute Island is a number of interconnected compute nodes, with associated infrastructure and facilities, such as shared file systems, management network and master and login nodes, which makes the island largely independent of the rest of the system (see Figure 1). The intention is that the number of islands will determine the SDP size, and that the independent nature allows the number of islands to be scaled virtually indefinitely.

Within a compute island, connectivity is abundant, so communication between nodes within an island, although discouraged, is possible. A low-latency, high-bandwidth interconnect is provided to facilitate communication within the compute island. This same interconnect facilitates communication between islands, but bandwidth is much more limited, and end-to-end transfers may require several hops. All data needed for processing is contained within this island. All processing for data contained within an island is done in that island.

One of the final SDP design challenges will be to provide an I/O concept which does not rely on a global file system where any compute node could reach any storage node with similar bandwidth. Simpler parallel file system concepts might allow to realise one parallel file system per island with acceptable overhead.

In the following we assume that storage for buffering data is to be provided *in a global namespace* through a parallel file system which can either be accessed by all nodes of the system or all nodes of a given island. We note that the parallel file system does not have to be implemented as an interface to physical disk drives, but may well be an interface to *non-volatile memory*.

Receiving data from the central signal processor (CSP) and merging meta-data provided by the telescope manager (TM) into SDP data streams needs to be handled in near real-time. So, the performance of the buffer storage and the implemented file systems is essential.



ISP partners [HPCISP] have run IO benchmarks on their local conventional HPC systems to

Figure 1: Schematic of an SDP compute island.

evaluate the file system performance and find limits of currently available IO technology. These results serve as a starting point for assessing buffer storage solutions that might be in place at the time of the SDP construction phase.

Bandwidth and meta-data tests have been carried out: The IOR benchmark [IOR] has been used to measure file system bandwidth at both POSIX and MPI-IO level. Here, write and read performance is reported under several sets of conditions. Tests have been run both on Lustre and GPFS based systems. The mdtest tool [MDTEST], designed to evaluate meta-data performance, has also been employed to measure file creation/deletion rates under consideration of different depths of the file system directory structure. Results are reported for Lustre, GPFS, BeeGFS, and a local SSD-based ext4 file system. The Jülich Benchmarking Environment [JUBE] has been used for the benchmark execution.

References

[SDPARCH] https://confluence.ska-sdp.org/download/attachments/145326117/preliminary-architecture.pdf?api=v2

[ISP] https://confluence.ska-sdp.org/display/ISP/PROT.ISP+Home?src=search

[HPCISP] https://confluence.ska-sdp.org/display/ISP/HPC+Facilities+for+prototyping+in+ISP

[IOR] http://sourceforge.net/projects/ior-sio

[MDTEST] http://sourceforge.net/projects/mdtest/

[JUBE] http://www.fz-juelich.de/jsc/jube

[SIONlib] *http://www.fz-juelich.de/jsc/sionlib*

[SHAN] Shan H., Shalf S., Using IOR to Analyze the I/O Performance for HPC Platforms, http://escholarship.org/uc/item/9111c60j#page-1

[BEEGFS] http://www.beegfs.com/docs/BeeGFS_Flyer.pdf

[ADDEND] IO-Benchmark-M5-Addendum, see [ISP]

Context and Background

Benchmark specifications

IOR

The IOR software focuses on measuring the sequential read/write performance under different file sizes, I/O transaction sizes, and scales of concurrency. It also differentiates the modes shared-file and file-per-process. It supports both the traditional POSIX interface and the parallel I/O interfaces, including MPI-IO, HDF5, and parallel NetCDF [SHAN].

In the following we assume version 2.10.3 to be used. From the whole set of parameters to set, we chose to configure the following ones and let the rest unmodified (set to default):

Argument	Value(s)	Comments
api	POSIX / MPI-IO	
filePerProc	1 (one file per proc) / 0 (shared file)	
blockSize	4 GiB, 2 GiB, 1 GiB	vary amount of bytes per task
transferSize	4KiB, 32KiB, 1 MiB, 4 MiB	consider access granularity
(nodes,tasks,blocksize)	(256,1,4096m), (256,2,2048m), (256,4,1024m)	e.g. fixed aggregated file size: 1 TiB of data using different combinations wrt. no of nodes, no of tasks per node, and blocksize

Jobs have been run with reordering of tasks by a constant number.

MDTEST

mdtest [MDTEST] is an MPI-coordinated metadata filesystem benchmark test that performs open/stat/close operations on files and directories and then reports achieved performance.

We assume version 1.9.3 to be used. Relevant mdtest parameters are here:

Option/Value	Comments
-F	perform test on files only (no directories)
-n <files>	number of files created by each task
-C	only create files
-z <depth>	depth of hierarchical directory structure
-np <tasks>	MPI tasks

We perform the test runs using several values for the total number of files to be created in each test:

mpiexec -np <tasks> ./mdtest -n <files> [-z <depth>] -F -C -v -d /work/mdtest

For such a fixed number of total files we vary the number of MPI tasks and the depth of the directory structure, e.g.:

Total Number of Files created	Number of Files per Task	MPI Tasks	Nodes	Tasks per Node	Directory Depth	File creation/ delete
1048576	32768	32	32	1	-|1|2	-C
	16384	64	32	2	-|1|2	-C
	8193	128	32	4	-|1|2	-C
	4096	256	32	8	-|1|2	-C

The input parameters have been chosen after discussions with domain scientists. Given the need for developing significant parts of the software stack from scratch, it was assumed that there is significant freedom in adjusting the access patterns to optimize the use of the I/O architecture. For this reason it was decided to scan the parameter space of the synthetic benchmarks in a reasonable range. No profiling of current radio astronomy software could be performed.

Benchmark environment

JuBE

The Jülich Benchmarking Environment [JUBE] allows for putting together a large set of benchmarks and run them across different systems. Results are produced in a consistent and reproducible way and can easily be compared and presented. JuBE consists of a number of xml files defining various parameters of the system, the environment, and the benchmark application. The xml files need to be adjusted to the system that the benchmark is intended to run on. A few JuBE xml files generated for this report can be found in the report addendum [ADDEND].

Using JuBE, the IOR xml benchmark configuration file accepts different modes:

- File per process: Each process creates its own file writing and reading (typical with POSIX applications). If using Lustre with default striping this means that each file will placed on a single OST (raid group). So the max write/read performance for each file will be limited by a single raid group throughput. In this mode there is only one segment per file with one block per segment and many transfer sizes per block. For optimum performance transfer size should be 1MB and block size multiplied by transfer size will give the total file size.
- MPI-IO shared file: If the api is set to MPIIO and mode is set to “grouped” then –s 1 option will be used in IOR run. Each process will be writing to its own block. There will be one segment for the entire file. If mode=interleaved is used IOR will be run with –s 4. File will be split into four segments and each process will be writing to its own block within each segment [SHAN, p.4ff].

Storage systems

iVEC

The operational and capital costs of a storage system similar to that in iVEC is summarised below, based on (September 2014) list pricing in US dollars, as provided by Cray:

Model	Cray Sonexion 1600 (w/ 2TB disks)
No. scalable storage units (SSUs)	12
Sustained bandwidth (GB/sec)	60
Usable capacity (TB)	1,440
List price (incl. 3-year Cray Gold warranty)	$ 1,614,171 USD

One should note that the working storage for MAGNUS consists of 15 Scalable Storage Units with a sustained bandwidth of 70 GB/sec, and is fitted with 3TB disks.

The infrastructure requirements for the Sonexion 1600 storage system, per rack, are as follows:

- Weight per rack: 1,175 kg (incl. 34.5 kg for water-cooled door);
- Dimensions: standard rack size – 600mm×1200mm×42RU (w×d×h);
- Power-consumption per rack (incl. switch): 20kW.

The installed Lustre system at iVEC has a single metadata server, which would rapidly become a pinch point for such a large file system. Support for multiple metadata servers was introduced in Lustre 2.4 (May 2013) though this version is not yet widely used in HPC environments.

The Lustre storage is scalable in bandwidth and capacity, this can be achieved simply by adding more object storage targets. However, the (iVEC-deployed) Lustre parallel file system is currently not scalable in metadata since there is a single metadata server. During testing, metadata-server performance was not observed to be a limiting factor, though this would be a limiting factor at some point. In terms of bandwidth, Lustre scaled very well, and should continue to scale as long as the SDP software is designed for this – that is, many large files being read/written from many compute nodes.

Jülich Storage: JUST

In general, the scalability of a system performance depends of the number of storage building blocks used in cooperation with an appropriate configured network. The Jülich Storage System JUST e.g. comprises 31 IBM GSS24 building blocks each including 2 GPFS servers equipped with 6 GE adapters providing roughly 12 GB/s per building block.

Each building block comprises 240 disks resulting in 7,440 disks in total which are the main cause of power consumption of about 3kW per building block in idle mode.

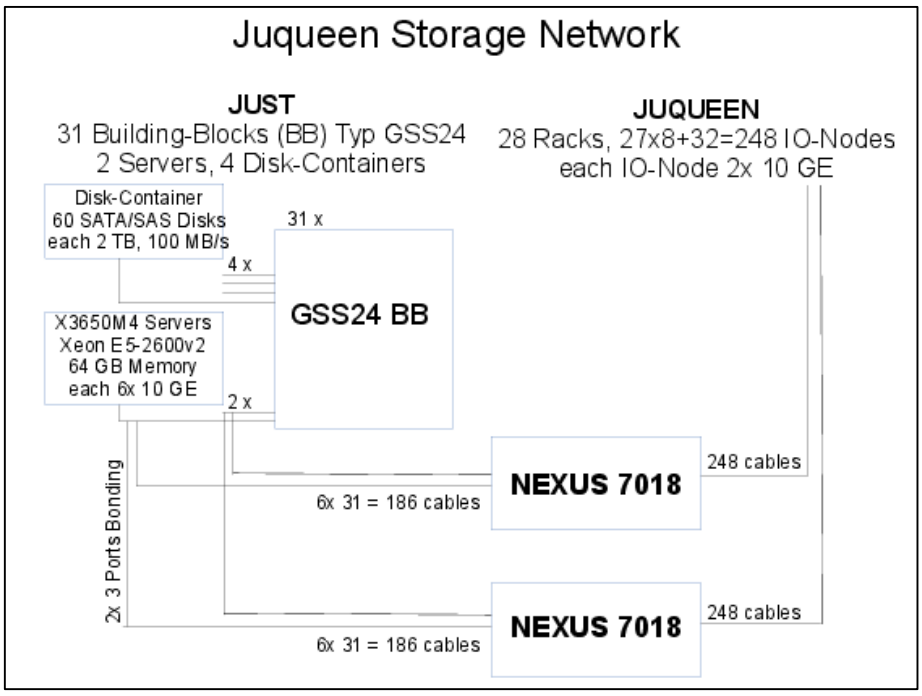


Figure 2: Schematic of the Juqueen storage network.

For SKA1 the limiting factor is the bandwidth provided by each disk. The extrapolation based on today's GSS technology (meanwhile called Elastic Storage) shows that significantly more building blocks are required than affordable for SKA1. SKA1-low expects a data rate of 7.3 TB/s, i.e. about 600 building blocks of the JUST type would be required to achieve this performance. With 2 building blocks per rack this amounts to 300 racks floor space. Based on 15 GB/s measured throughput on one JUQUEEN rack with 8 IO nodes this would require about 500 BG/Q racks (125 BG/Q racks if equipped with 32 IO nodes each).

The project would critically benefit from new non-volatile memory technologies which can sustain much higher data rates than spinning disks. Today available technology, i.e. NAND flash memory, features endurance which is likely too low for SKA. Other memory technologies are products based on these technologies are currently being developed. However, how quickly these will be brought to market and at which costs is difficult to predict.

Benchmark Results

Most of the following results have been obtained in normal operation mode of conventional HPC systems. The JUST storage system is used by several large-scale HPC systems**.** *It is very hard to perform tests in single-user mode. Repeating benchmark tests multiple times could give some indication of the variability of the results*. Thus, the reported data may be affected by the I/O load generated by concurrently active jobs. In this document performance is documented for the single-file-per-process mode as this seems to be most interesting for the SDP. Further results – also covering the shared-file mode - are included in the addendum [ADDEND].

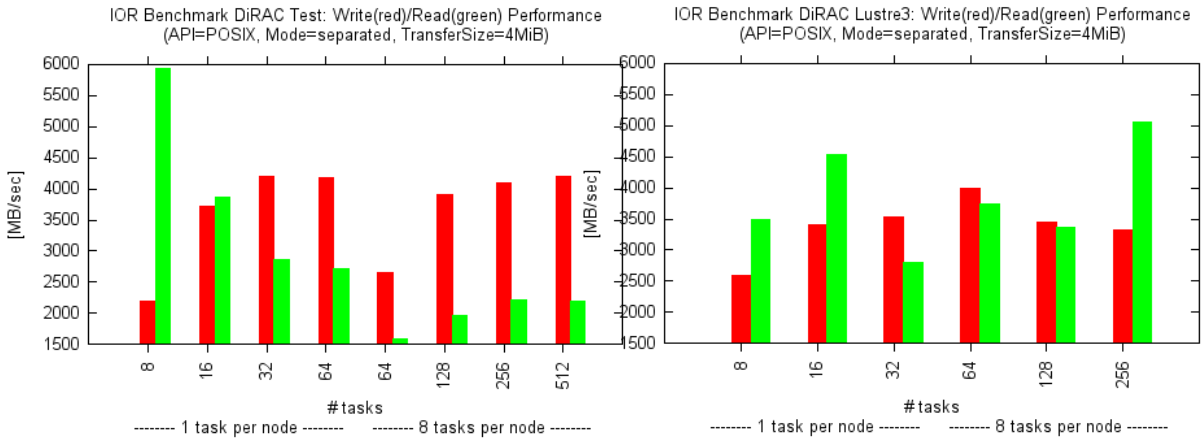
Lustre file system implementations

DiRAC

Site web address: www.dirac.ac.uk

System	Lustre test system / Lustre3 production system
Compute node	Intel E5-2670 2.6 GHz /Sandy-Bridge) eight-core
Processors	1200 (9600 cores)
Main memory	53.8 TB
Network	InfiniBand
Storage	80 HDDs / 180 HDDs
File system	Lustre

Bandwidth measured with IOR:



Further IOR benchmark results of the DiRAC systems also covering the shared file case and the MPI-IO interface are included in the addendum [ADDEND].

Jülich

Site web address: www.fz-juelich.de/jsc

System	**JUROPA (**3288 nodes)
Compute node	Intel Xeon X5570 (Nehalem-EP) quad-core, 24 GB RAM
Processors	6576 (26304 cores)
Main memory	79 TB
Network	InfiniBand QDR

OS	SuSE SLES 11
IO nodes	1 MDS serving 1 MDT, 8 OSS serving 120 OSTs in total
Storage	• DDN SFA10000 storage systems • OSTs are raid6 devices with 8+2 disks (SW RAID) • each disk a 128KB chunk is written (x8 = 1MB, theoretical optimal chunk size) • each disk is 931GB x8=7TB of data for each OST
File system	• Lustre 1.8.4 (underlying ext3 file system) • stripe size 1 MB, default stripe count 4 (striping over 4 OSTs) • 833TB total capacity of the file-system • Maximum IO bandwidth ~20GB/s.

Bandwidth evaluation based on IOR:

JobId	Tasks	Mode	API	Job Description
n256p1t1	256	separated	MPIIO	256 nodes, 1 task per node, file-per-process, MPIIO
n256p1t1	256	separated	POSIX	256 nodes, 1 task per node, file-per-process, POSIX
n256p2t1	512	separated	MPIIO	256 nodes, 2 task per node, file-per-process, MPIIO
n256p2t1	512	separated	POSIX	256 nodes, 2 task per node, file-per-process, POSIX
n256p4t1	1024	separated	MPIIO	256 nodes, 4 task per node, file-per-process, MPIIO
n256p5t1	1024	separated	POSIX	256 nodes, 4 task per node, file-per-process, POSIX

The aggregated filesize is 1 TiB, the block size is varied dependent of the number of tasks.

Tasks	BlockSize	TransferSize
256	4 GiB	32KiB and 4 KiB
512	2 GiB	32KiB and 4 KiB
1024	1 GiB	32KiB and 4 KiB

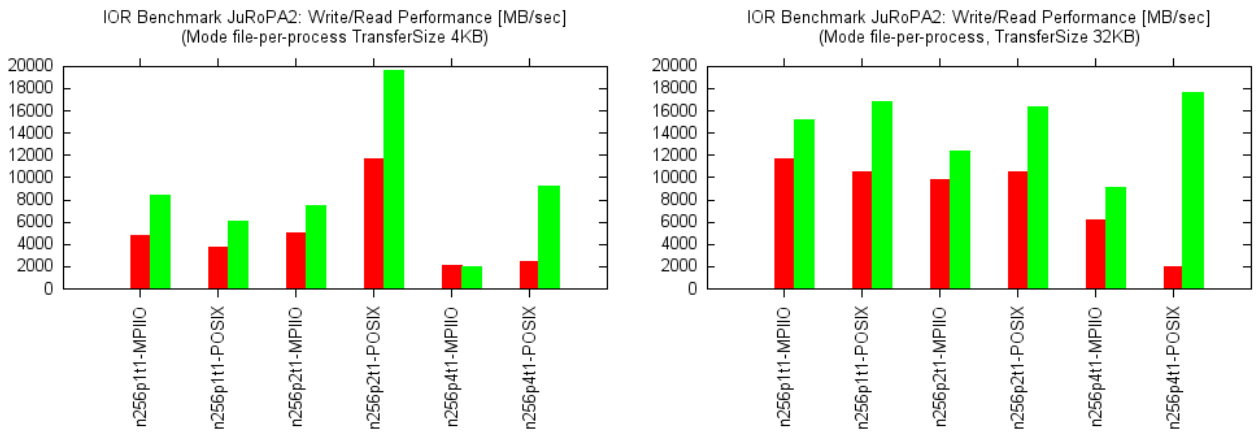


Figure 3: Results from the IOR benchmark. The colours read and green in the above charts represent write and read performance, respectively.

The corresponding JuBE xml files and further results are embodied in the addendum.

Meta-data performance on JUROPA has been measured using 32 nodes creating 32768 files:

Total Number of Files created	**Nodes**	**Tasks per Node**	**Directory Depth** **-z DEPTH**	**File creation only/file delete**	**File creations per second**
32768	32	1	0	-C	1266.90
32768	32	2	0	-C	824.52
32768	32	4	0	-C	747.5
32768	32	8	0	-C	562.63
32768	32	1	1	-C	780.51
32768	32	2	1	-C	854.01
32768	32	4	1	-C	733.11
32768	32	8	1	-C	552.54
32768	32	1	2	-C	935.42
32768	32	2	2	-C	1032.08
32768	32	4	2	-C	816.83

32768	32	8	2	-C	620.79

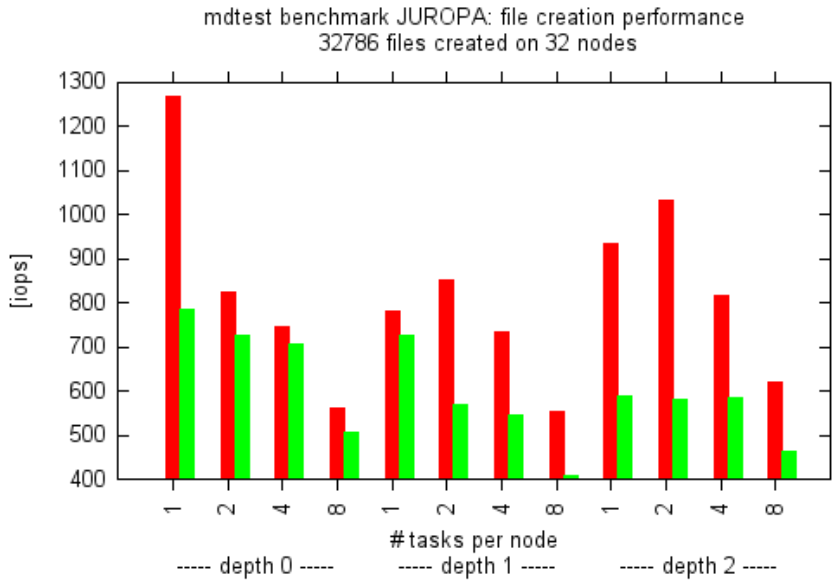


Figure 4: Here, red and green colour in the graphics above indicate maximum and minimum file creation rates.

iVEC

Site web address: http://www.ivec.org/services/supercomputing

System	**Magnus** (208 nodes)
Compute node	Intel Xeon X5570 (Sandy-Bridge) eight-core
Processors	416 (3328 cores)
Main memory	13.3 TB
Network	Cray Aries, InfiniBand FDR
OS	
IO nodes	1 MDS serving 1 MDT, 28 OSS serving 112 OSTs in total
Storage	• Cray Sonexion 1600 appliance • OSTs are raid6 devices with 8+2 disks

File system	• Lustre 2.1 on Sonexion (server), Lustre 2.4 on compute nodes (client) • 3 PB total capacity of the file-system • Maximum IO bandwidth ~70GB/sec peak / 60 GB/sec sustained

All Magnus runs were done on a production system, never in maintenance mode. They were done using Cray-MPICH, CLE5.2. Throughput results for Magnus were consistent. Each run did 3 benchmarks which were internally consistent. However, an order of magnitude difference results were observed when running on different days. Observing a write bandwidth larger than read bandwidth might indicate buffering effects. The following IOR tests have been run:

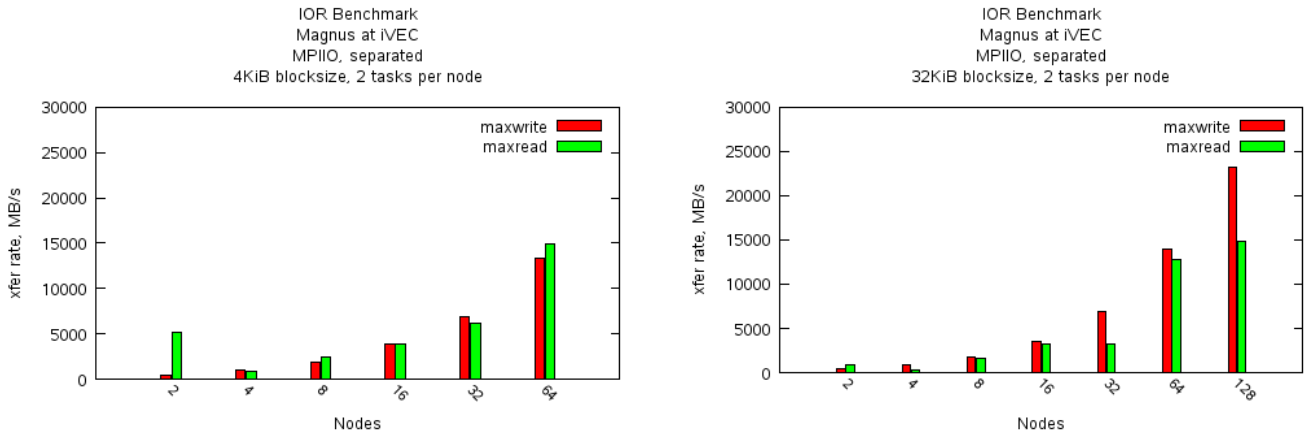


Figure 5: IOR benchmark results.

For further results pls. refer to the addendum.

In 2014, the Magnus system was upgraded (cf. http://www.ivec.org/systems/magnus):

System	**Magnus** (1488 nodes)
Compute node	2x Intel Xeon E5-2690V3 (Haswell) twelve-core
Processors	2976 (35,712 cores)
Main memory	93 TB

Meta-data performance on Magnus has been measured using 32 nodes creating 32786 files. Tests were consistent within a run, but varied significantly between days.

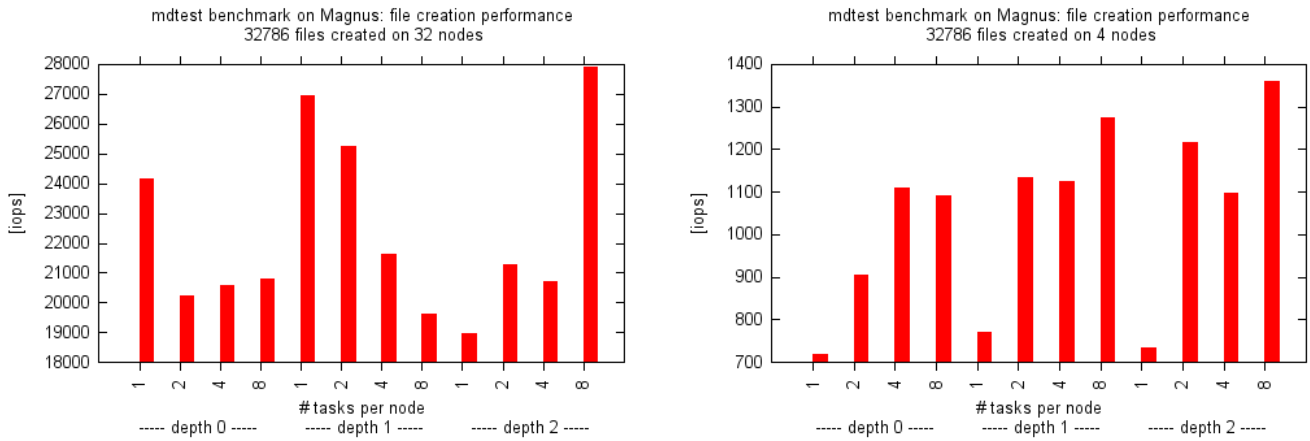


Figure 6: mdtest benchmark results.

GPFS file system implementations at partner sites

STFC Hartree Centre

Site web address: http://www.stfc.ac.uk/Hartree/default.aspx

System	**BlueWonder (512 nodes)**
Compute node	Intel Xeon E5-2670 2.6 GHz (SandyBridge)
Processors	1024 (8192 cores)
Main memory	40 TB
Network	InfiniBand
OS	CentOS Linux 6.2
IO nodes	512 nodes (all nodes have GPFS access)
Storage	GPFS, 5.7 PB
File system	Lustre

Bandwidth measured with IOR:

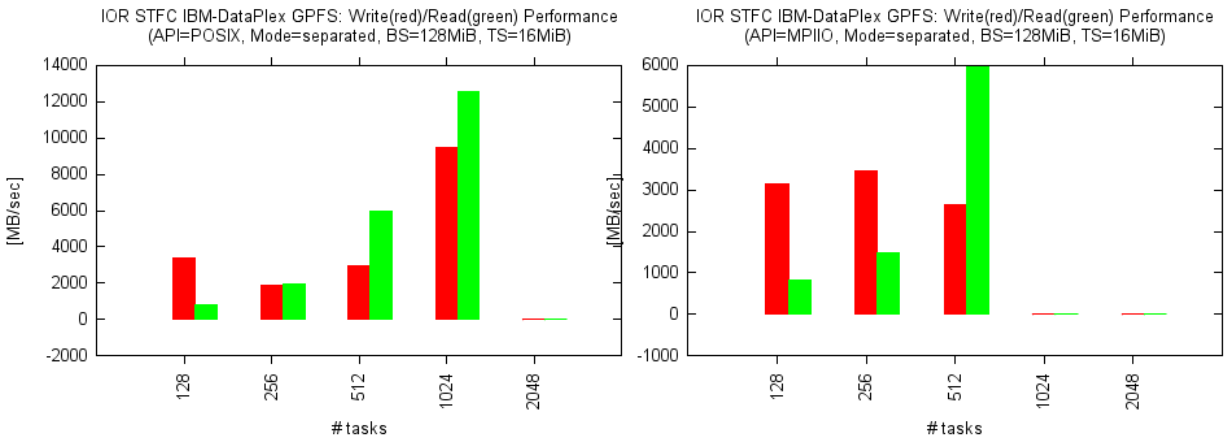


Figure 7: IOR benchmark results.

A table of IOR results measured on BlueWonder can be found in the addendum.

Jülich

Site web address: www.fz-juelich.de/jsc

System	**JUQUEEN (28 racks, 56 midplanes, 28672 nodes)**
Compute node	IBM PowerPC A2
Processors	28672 (458752 cores)
Main memory	448 TB
Network	5D torus
OS	
IO nodes	• I/O nodes are equipped with a 16-way SMP processor and a PCIe Gen2 port; configurable in 8,16 or 32 I/O nodes per rack; 248 I/O nodes in total (27x8 + 1x32) connected to 2 CISCO Switches. 2x 10GbE ports link each I/O node to the GPFS servers. • Every 128 BG/Q compute nodes are linked via the torus network and two so-called bridge nodes to a dedicated I/O node (2 GB/s raw, 1.8 GB/s), i.e. a midplane (a rack) of 512 (1024) compute nodes is limited by a bandwidth of 8 (16) GB/s.
Storage	• 31 building blocks type GSS24 (2 servers, 4 disk containers) • 60 SATA/SAS disks (2TB, 100 MB/s each) • 6x 10 GE adapters per server

File system	• GPFS • 14 PB online, WORK file system: 7 PB • blocksize: 4 MB (cannot be changed); • maximum IO bandwidth ~ 160 GB/s

Bandwidth measured on JUQUEEN using an aggregated file size of 512 GB:

Tasks	BlockSize	TransferSize
8192	64 MiB	4 MiB
16384	32 MiB	4 MiB
32768	16 MiB	4 MiB

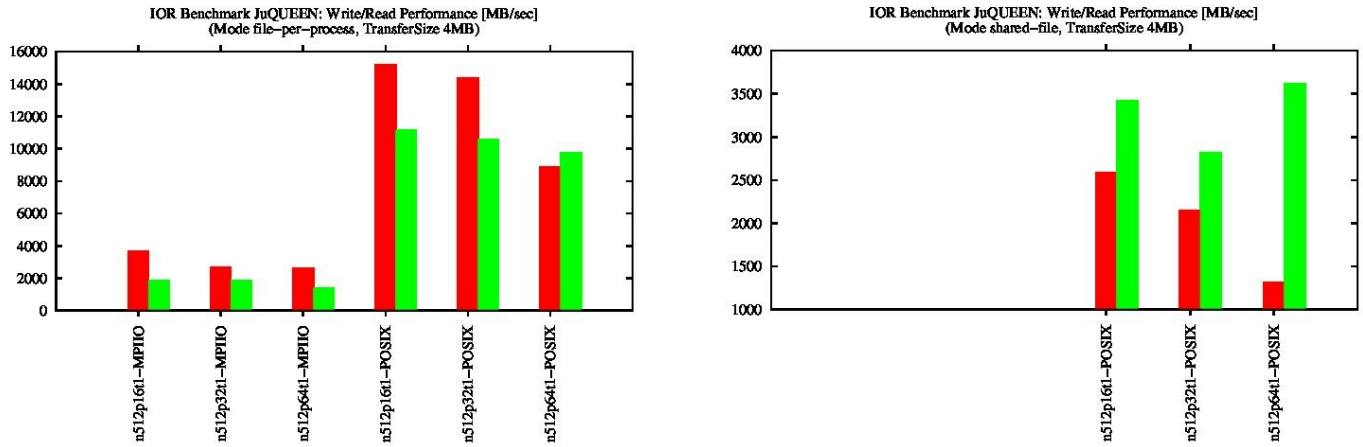


Figure 8: IOR benchmark results.

As we are using only one midplane we are restricted to 8 I/O nodes and can only expect a maximum bandwidth of 8 * 2 GB/s, i.e. 16 GB/s. Pls. refer to the addendum for further results.

Meta-data performance on JUQUEEN has been measured using 32 nodes creating 32768 files:

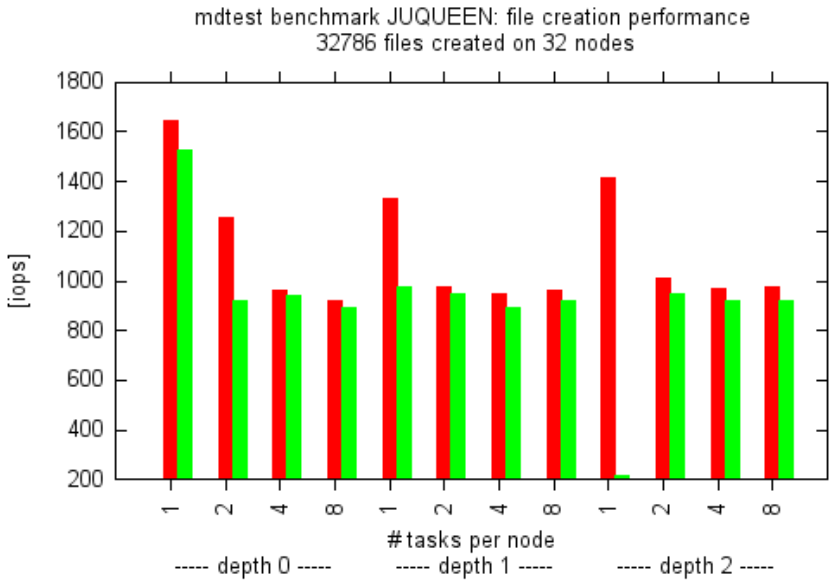


Figure 9: mdtest benchmark results.

Total Number of Files created	Nodes	Tasks per Node	Directory Depth -z DEPTH	File creation only/file delete	File creations per second
32768	32	1	0	-C	1643.94
32768	32	2	0	-C	1251.51
32768	32	4	0	-C	962.80
32768	32	8	0	-C	918.52
32768	32	1	1	-C	1332.29
32768	32	2	1	-C	976.45
32768	32	4	1	-C	944.48
32768	32	8	1	-C	963.32
32768	32	1	2	-C	1413.44
32768	32	2	2	-C	1013.83
32768	32	4	2	-C	967.43

32768	32	8	2	-C	974.10

Tests have been run with three iterations and the maximum value (red colour) and the minimum value (green) are displayed.

BeeGFS file system

Jülich

Site web address: www.fz-juelich.de/jsc

System	**DEEP (1 rack with 8 backplanes x 16 nodes)**
Compute node	2x Intel E5-2680 (8 cores each), 32 GB RAM
Processors	256 (2048 cores)
Main memory	4 TB (aggregate)
Network	InfiniBand QDR
OS	CentOS 6.3
IO nodes	4x storage server, 2x metadata server (same specification as compute nodes)
Storage	• SGI JBOD 2245 (45 SAS disks in total) • 4 x 8+2 RAID6 disk sets • 2 x 1+1 RAID1 mirrored disks • 1 x swap disk attached via 6 Gbit/s SAS 4x wide ports
File system	• BeeGFS/FhGFS version Oct-2013 • WORK: 60 TB size, Blocksize 512 KB

Bandwidth measured on DEEP using IOR:

Tasks	Nodes	Tasks per Node	API	Transfer size	Block size	Aggregate size	Write MiB/s	Read MiB/s
128	8	16	MPIIO	8 MiB	4 GiB	512 GiB	1891.74	1781.39
128	8	16	POSIX	8 MiB	4 GiB	512 GiB	2021.18	1885.67

Jobs have been run in single-file-per-process mode and constant reordering of tasks.

Meta-data tests using mdtest (maximum (red) and minimum (green) values are depicted):

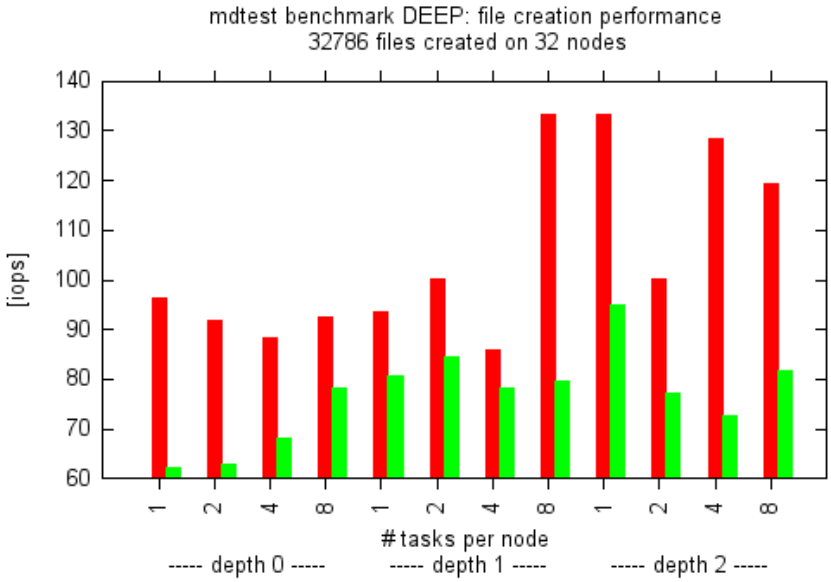


Figure 10: mdtest benchmark results.

Metadata measurement results published in [BEEGFS] show the performance of file creation on 1-20

storage/metadata servers. Rates of 34693 creates/s using one IO server, 68094 creates/s with two

servers, and 539724 creates/s for 20 servers are reported which is far beyond what our tests provide.

These high performance numbers have been obtained using IO servers which are equipped with 4x

Intel 510 series SSD (RAID0 with mdraid) whereas in the DEEP storage environment two RAID1

mirrored disks are employed for storing the metadata (for failsafe operation).

Local SSD-based ext4 file system

Moreover, a test using an **ext4 file system directly ceated on the Ramsan 70 device** displaying a significantly higher file creation rate:

mdtest-1.9.3 was launched with 8 total task(s) on 1 node(s), Command line used:
mdtest-1.9.3/mdtest -n 100000 -i 4 -d /ramsan/.../AFPhome/Temp/mdtest/

FS: 826.8 GiB Used FS: 38.4% Inodes: 52.5 Mi Used Inodes: 0.2%
8 tasks, 800000 files/directories

SUMMARY: (of 4 iterations)

Operation	Max Performance
Directory creation	14952.31
Directory stat	2345661.66
Directory removal	26841.38
File creation	**42944.31**
File stat	2300251.25
File read	1526691.37
File removal	60822.45
Tree creation	47662.55
Tree removal	3.05

Summary and conclusions

- File-system benchmarking tests have been performed at several ISP partner sites (DiRAC, iVEC, Jülich, STFC Hartree) covering throughput and meta-data evaluations for Lustre, GPFS, and BeeGFS implementations in conventional HPC systems. *The currently foreseen island concept makes the SDP architecture significantly different from today's typical HPC architectures. However, each island is expected to reach the scale of HPC systems as of today. The benchmarks have been performed assuming that today's HPC systems could in terms of size and complexity be considered prototypes of future SDP islands.*
- Regarding IO throughput both GPFS and Lustre file systems have been evaluated and it turned out that a high fraction of peak bandwidth could be achieved. It is expected that similar values also hold for larger tests which could not be executed within the small budget available at the time of the experiment.
    - On the JUROPA system (5 years old and currently being replaced by a new general purpose cluster), the Lustre (version 1.8.4) work file system provided an IO bandwidth 18 GB/s (of 20 GB/s maximum available due to network restrictions).
    - On JUQUEEN, using 1 rack and 8 IO nodes for testing, 15 GB/s has been measured (maximum performance is here 16 GB/s on 1 rack equipped with 8 IO nodes). In order to achieve higher performance values a larger part of the clusters would have to be reserved for the tests, e.g. when running installation approval tests on JUQUEEN, 12 racks have been required to reach the maximum file system performance of 160 GB/s.
- For state-of-the-art HPC systems we observed a throughput of metadata operations of at least 1000/s for JUQUEEN, where client operations might be limited by the relatively week cores, and more than 18,000/s on a more recent CRAY system when using 32 nodes.
- *The goal of this benchmark analysis was not a fair comparison of the performance of the different technologies and architectures (e.g. GPFS vs. Lustre) but more an assessment of the performance which can be achieved for existing production file systems with a size of multiple PByte. Of particular interest was a comparison between very sophisticated and simpler solutions, like BeeGFS, which could be interesting candidates for SDP.*
- *Parallel file systems had been designed such that data access can be highly parallelized and data read and write bandwidth can be maximized. The performed tests indicate that sustaining the bandwidth on the storage network or to the disk arrays (whatever the bottleneck is) is feasible for realistic I/O patterns (sequential write or read of large blocks). The more likely bottleneck arises from meta-data operations, where for popular solutions like Lustre parallelization is not yet an option. Significant performance differences have been observed, but they are above the upper performance target of 100 meta-data operations per second and island. At the moment meta-data performance is not expected to be a limiting factor, in particular if no global but only island-wide parallel file systems are assumed. Here furthermore it is assumed that a reasonable design of the software stack avoids excessive number of meta-data operations. There are concepts, like those realised in SIONlib, which allow for reducing the number of meta-data operations in a way which is transparent for the user.*