Global Filesystem Option for the SKA *UV buffer*

Tests and Development of Lustre

Bojan Nikolic

Blame

Cambridge HPCS

Cambridge IT

SKA SDP Consortium

Industry

Paul Calleja

Wojciech Turek

Mark Calleja

PROT.ISP Team

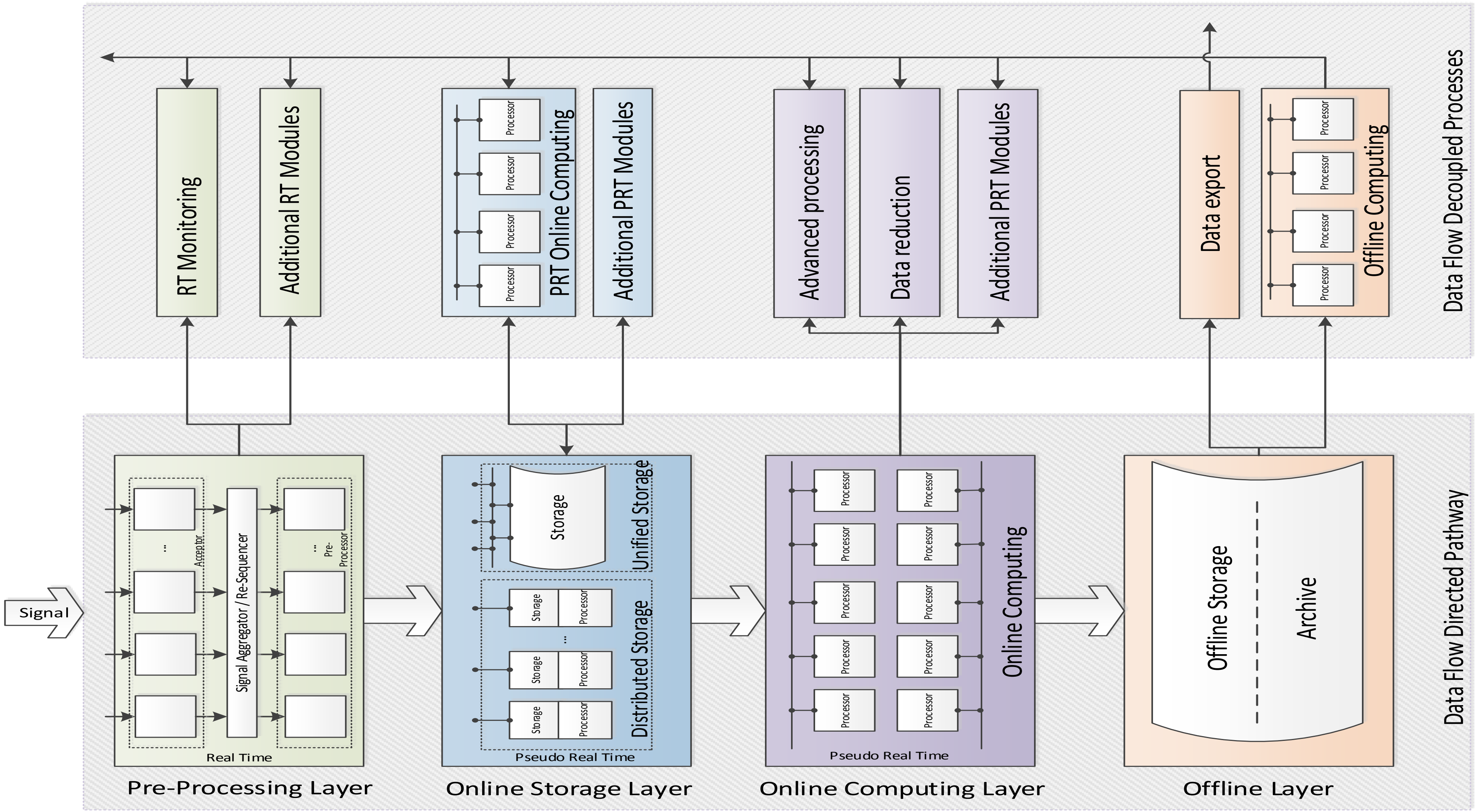
ARCH Team

Peter Braam

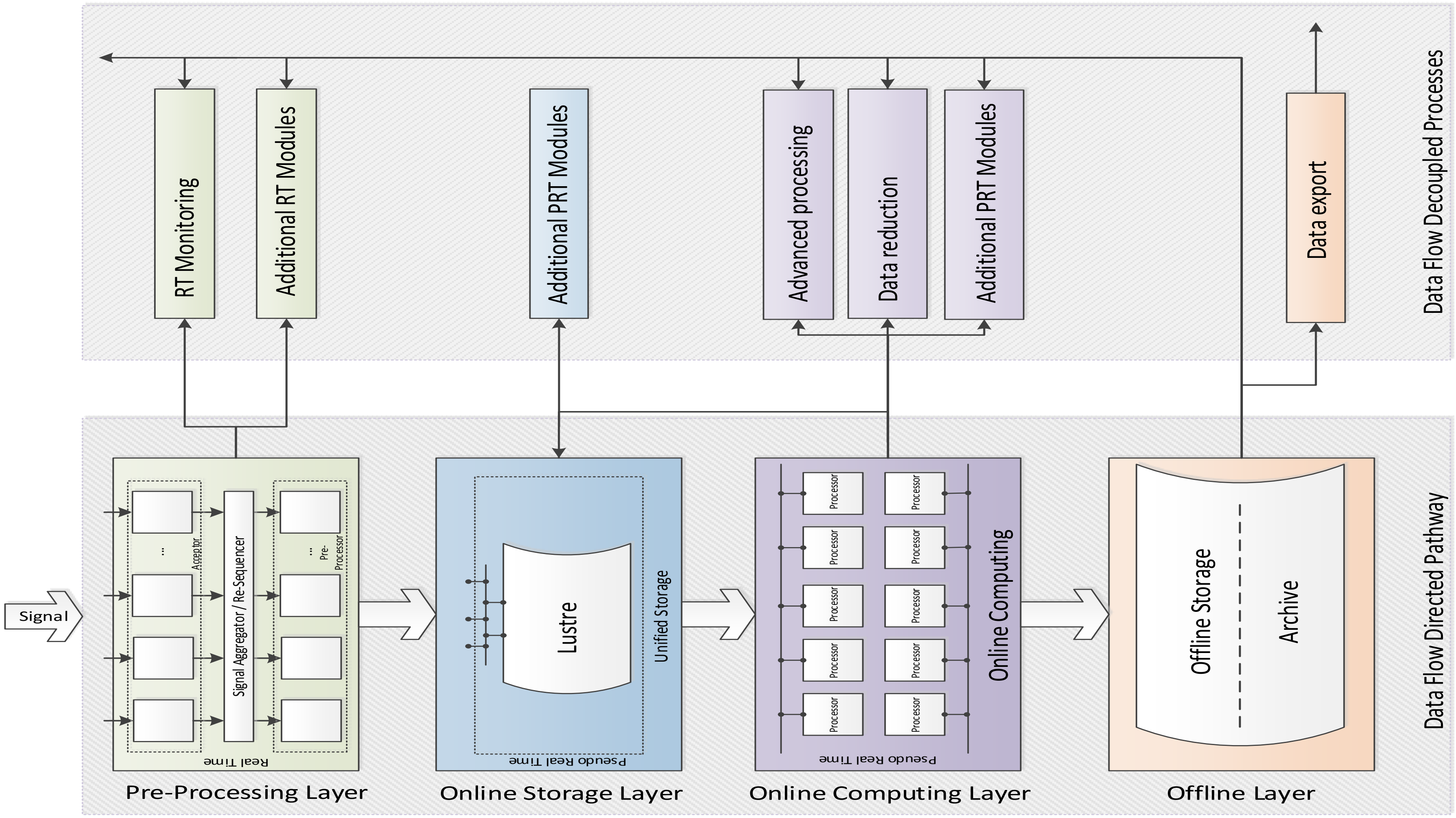
Calsoft Inc

Most of the work

SDP UV Buffer in CRISP Context



Global *UV Buffer* option



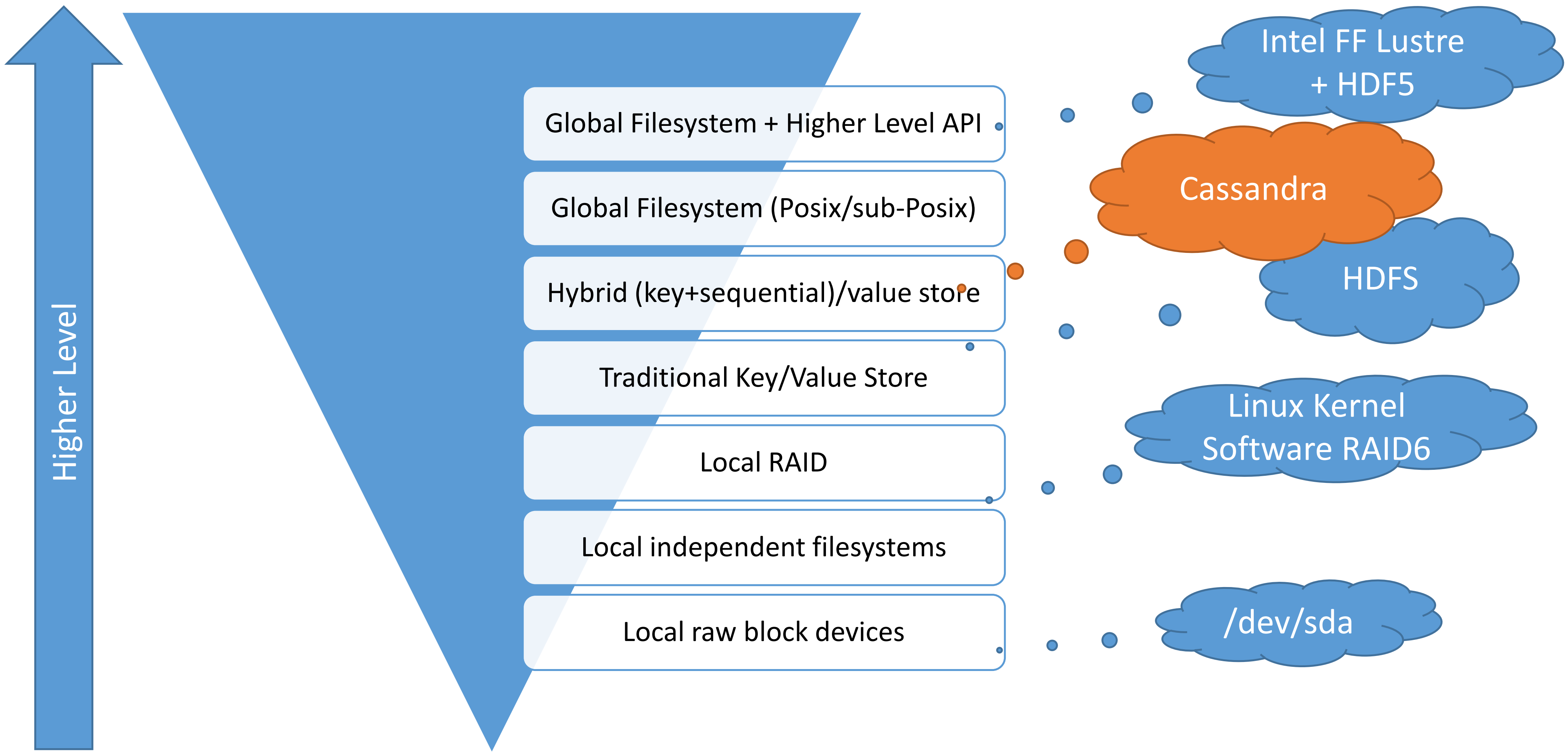
Specifications (maximal case)

Throughput	Write: maximum case 7 Tbyte/s
	Read: maximum ~5 times higher
Capacity	Maximum requirement 300 Pbyte
	Data lifecycle ~ 12 hours
Files	Maximum 10^8 files (one file per ten frequency channels per snapshot)
IO Computational Intensity	>250 Floating Point Operations per Byte of IO to UV Buffer
Deployment Date	Start of commissioning 2019. Science Operations 2021

Global Filesystem Motivation

Only one of the options being considered for SKA Science Data Processor UV Buffer

Options/Alternatives



Global Filesystem is not *Required*

- But this **always** true:
  - Computation is only ever done on words in the CPU
  - Large storage systems are always made of many smaller drives
  - Global Filesystem is just a software sub-system
- Why global filesystem?
  - Simple, familiar concept
  - Simple well known, very often used API and UI
  - Combine inter-node data communication, inter-node metadata distribution, persistence, inter-node high-availability and failover mechanisms all in one subsystem
  - Off the shelf vendor solutions & open source solutions

Global Filesystem Features revisited

Write to File	• Make data available to any node at any subsequent time • Redundantly distribute data on multiple nodes and disks
Read from File	• Retrieve data from any node from arbitrary previous time • On-the-fly recover from lost nodes and disks
File Open For Write	• Process signals that it is doing a piece of work
File Open For Read	• Process signals that no more work on this can be done until close
Directory list	• What are other nodes doing? • What have other nodes done in the past
File Permissions	• Enforce interfaces between nodes • Enforce interfaces between processes

Trade-off Issues

- Features/ Functions
- Proven Alternatives
- Efficiency
- Maximum Performance
- Scaling Issues
- Capital Expenditure
- Op-Ex/ Power Budget
- Software Development Cost
- Resilience
- Future-proof?
- Implementation Risks

Approaching a Trade-Off Analysis

- What are the inter-node communication features we **need**?
- What do we know we can do efficiently without a global filesystem?
- What is the performance of current global filesystems in providing the features we need?
- What is the capital/op-ex cost of global-filesystem?
- What are the impacts on software development costs?

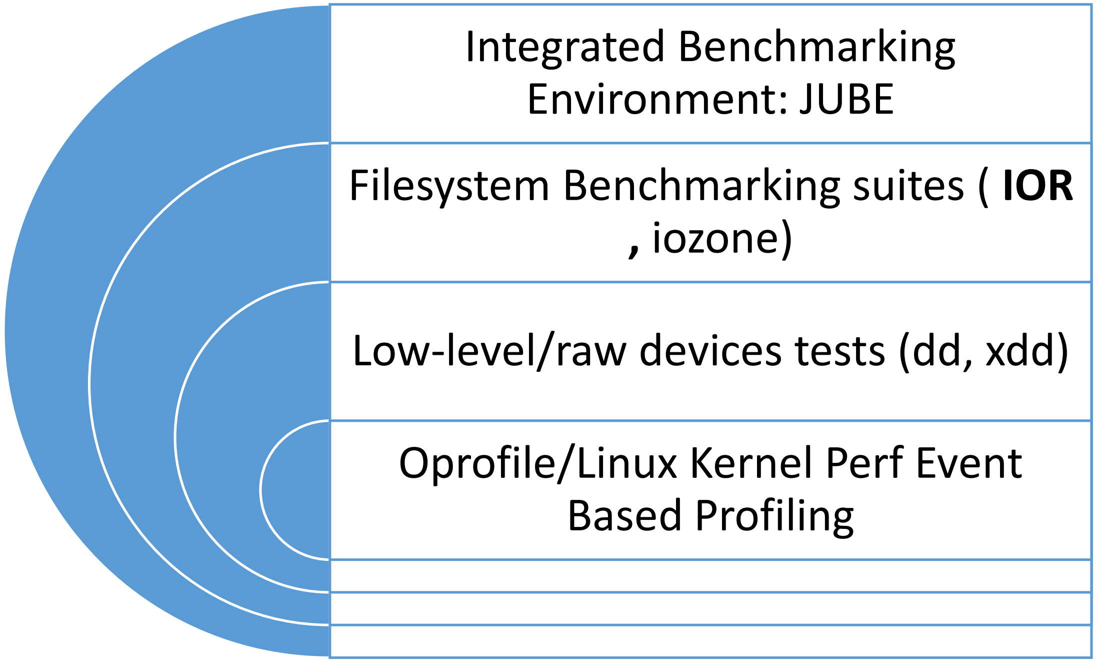
Current Lustre work toward the trade-off

- Performance & efficiency : benchmarks, analysis, hardware tests
- Scalability: Tests on very large integrated systems
- Cap-ex: Commodity Lustre
- High-Availability: Efficient Software RAID
- Implementation Risks: Accumulating and recording know-how

Performance Benchmarking

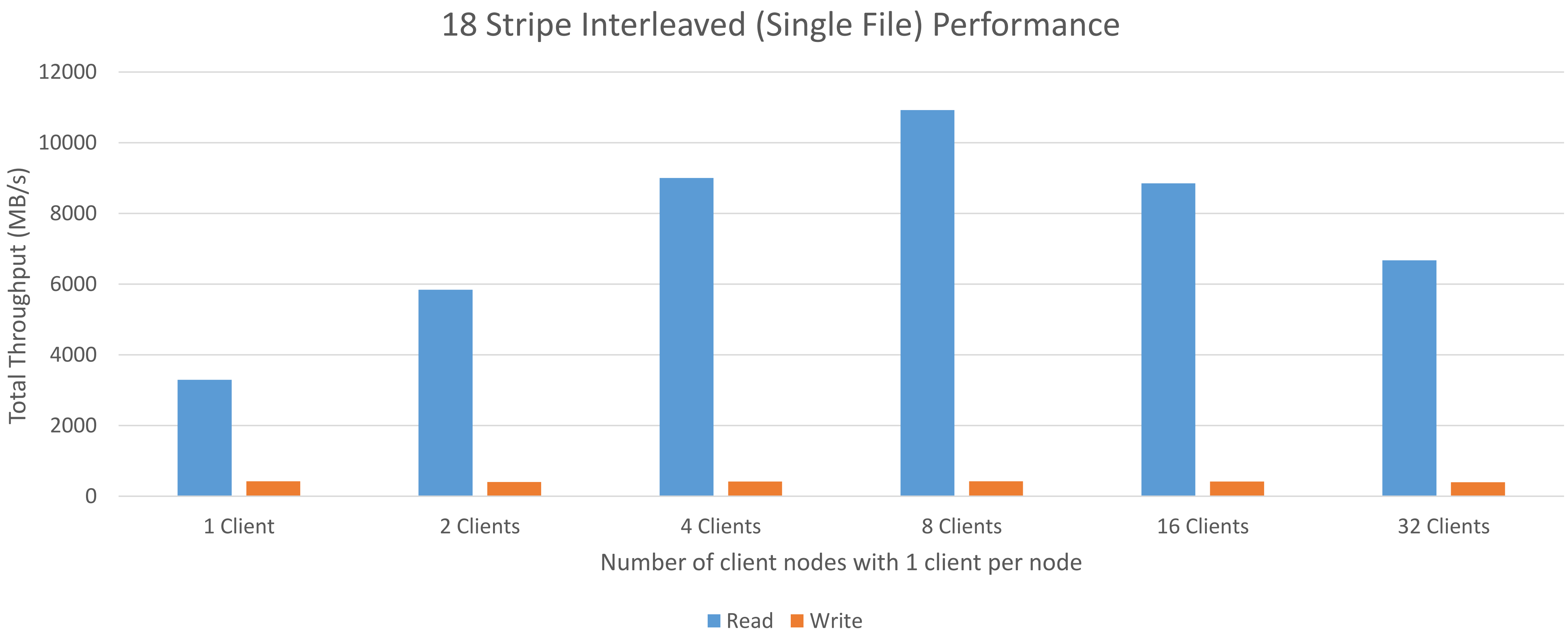
Aims: Measure Real World Performance, Identify Bottlenecks, Quantify Scaling

Lustre Benchmarking Approach



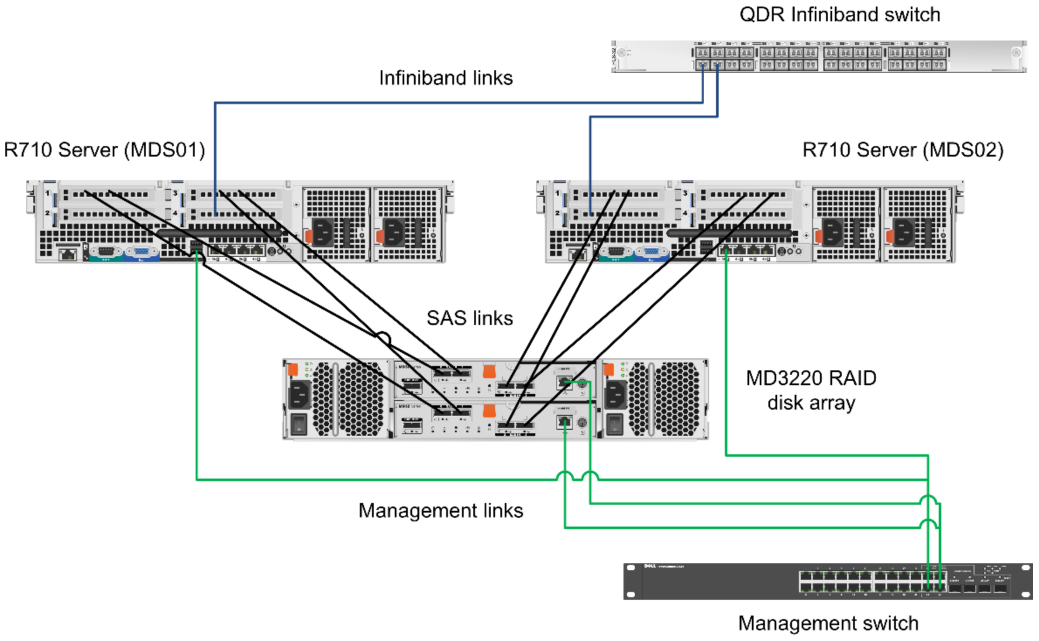
- Application tests: ASKAPSoft (in preparation)

Cambridge HPCS LUSTRE Test Brick (Gen 1)

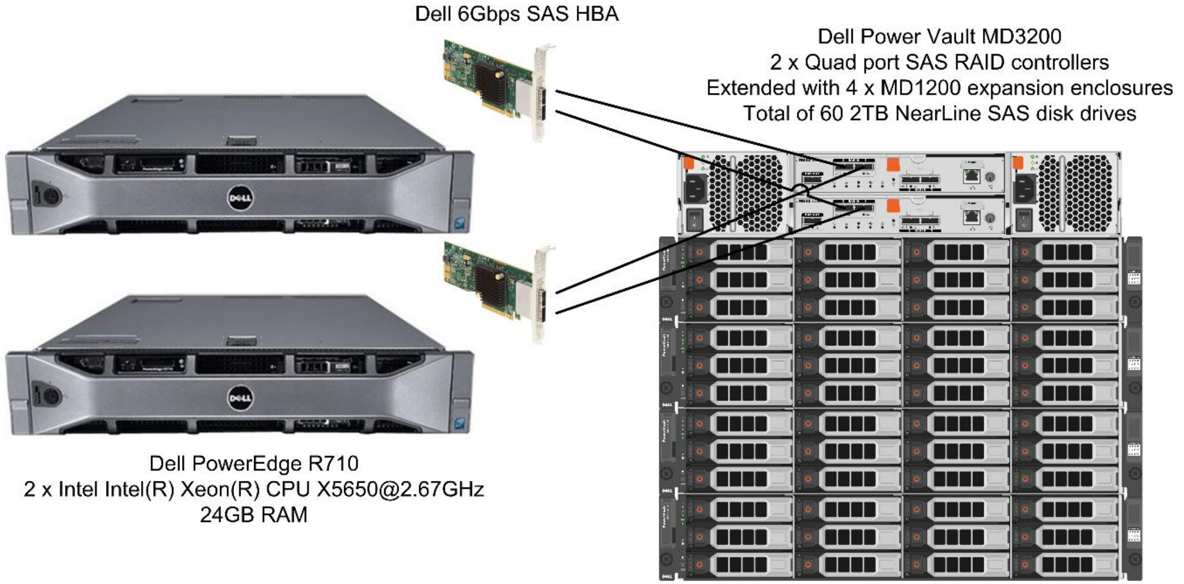


Cambridge HPCS Lustre Test Brick Gen 1

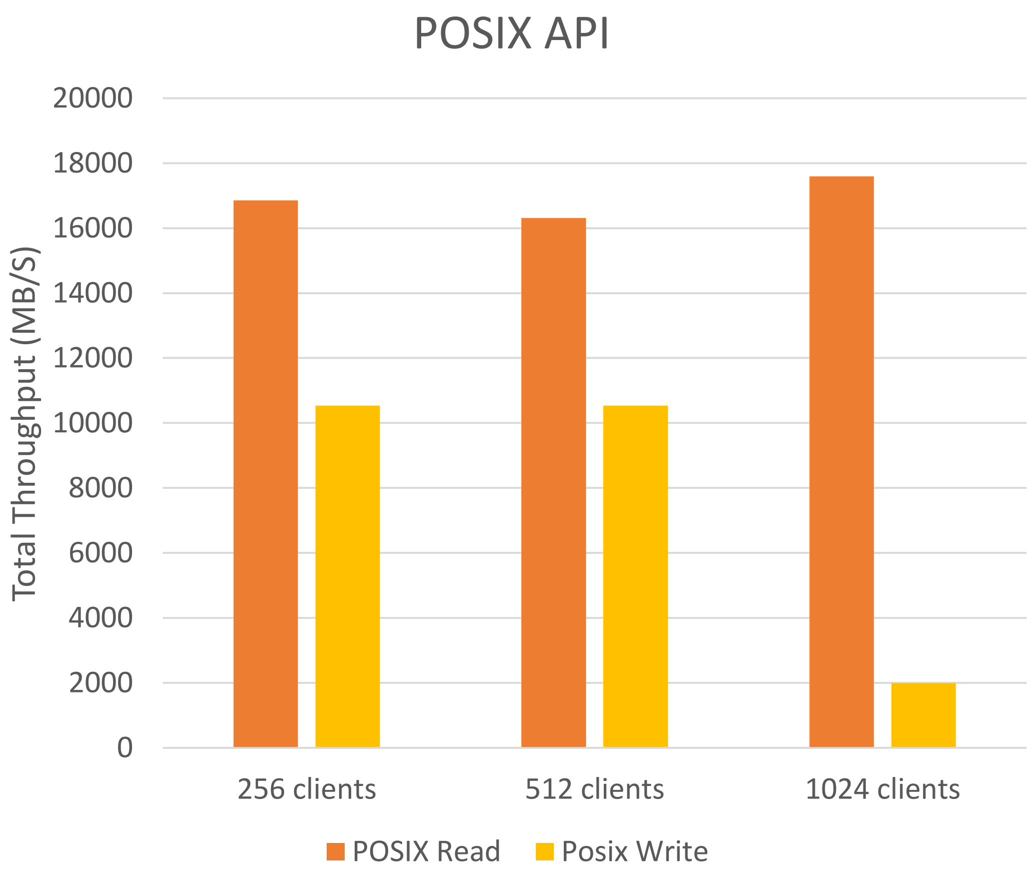
Metadata

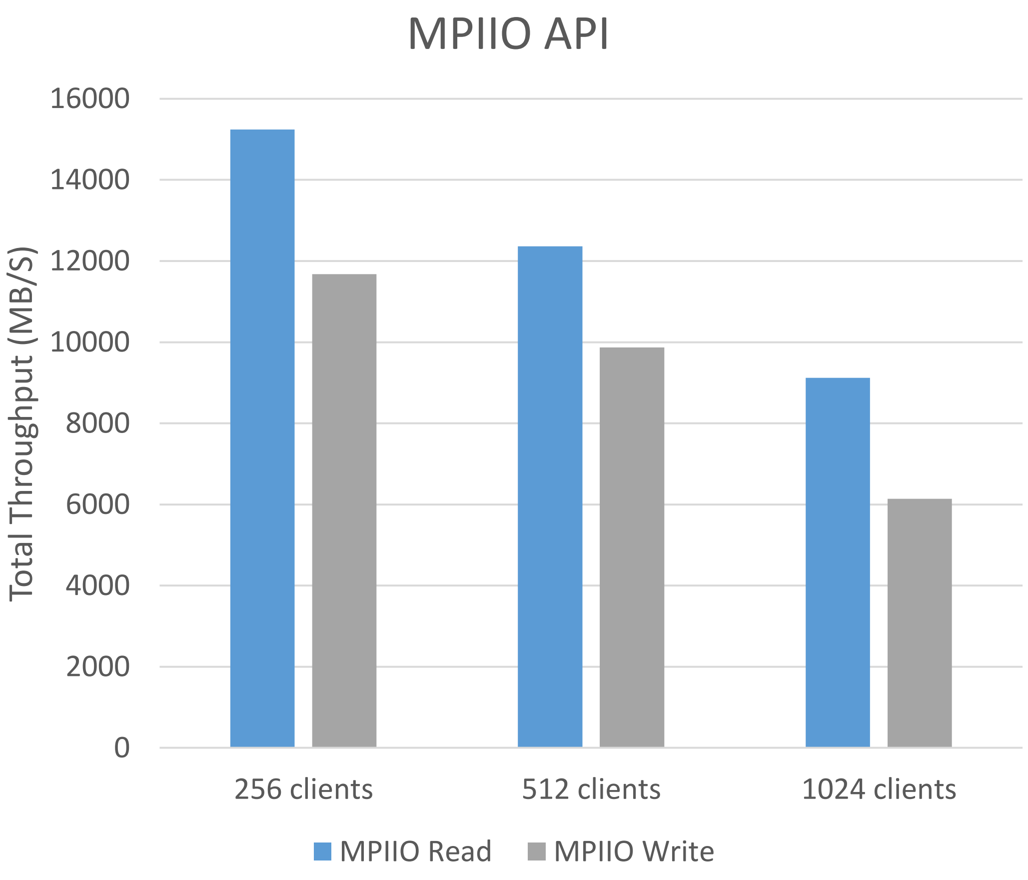


Object storage



Lustre tests on JuRoPA2 (file-per-process)





Lustre tests on JuRoPA2 (separate files)

JobId	Tasks	Mode	API	MaxWrite	MaxRead	BS	TS
n256p1t1	256	separated	MPIIO	11678.41	15237.87	4GiB	32KiB
n256p1t1	256	separated	POSIX	10535.22	16856.16	4GiB	32KiB
n256p2t1	512	separated	MPIIO	9871.79	12365.02	2GiB	32KiB
n256p2t1	512	separated	POSIX	10538.96	16314.26	2GiB	32KiB
n256p4t1	1024	separated	MPIIO	6141.25	9124.83	1GiB	32KiB
n256p5t1	1024	separated	POSIX	1982.93	17590.65	1GiB	32KiB

Lustre Kernel Profiling

		Unpatched Kernel				Patched Kernel		
IOTool Type	**#**	**Time taken seconds**	**Speed MBps**	**Oprofile CPU utilization details**		**Time taken seconds**	**Speed MBps**	**Oprofile CPU utilization details**
xdd QueueDepth=32	Avg.	481	8.9	1.97% memcpy 0.07% async_copy_data 5.75% raid456 module		419	10.2	1.5% memcpy 0% async_copy_data 6.8% raid456 module
xdd QueueDepth=1	Avg.	167	25.6	3.8% memcpy 0.31% async_copy_data 12.15% raid456 module		169	25.2	2.65% memcpy 0% async_copy_data 11.7% raid456 module
dd	Avg.	173	24.7	4.3% memcpy 0.37% async_copy_data 13.7% raid456 module		173	24.7	2.85% memcpy 0% async_copy_data 13.4% raid456 module

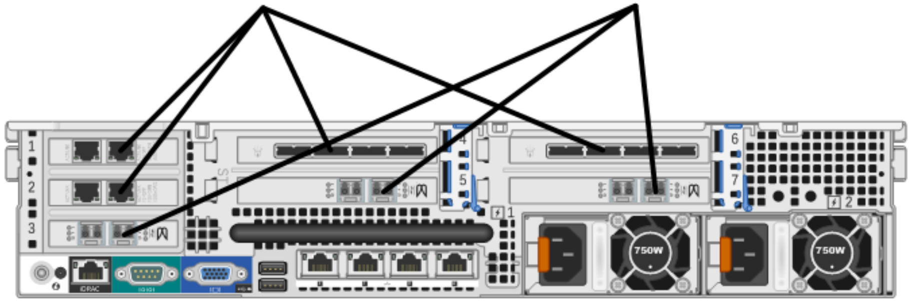
Lustre Development Hardware Platform at HPCS

Test Brick Gen 2

Object Storage Servers: Dell R720

2 x quad 6Gb/s SAS HBA
2 x dual 6Gb/s SAS HBA
Each port has 4 lanes 6Gb/s lanes
In total all for HBAs can deliver 25GBytes/s

3 x dual port FDR IB HCAs
In total 3 x dual port HCAs
can deliver 24GBytes/s

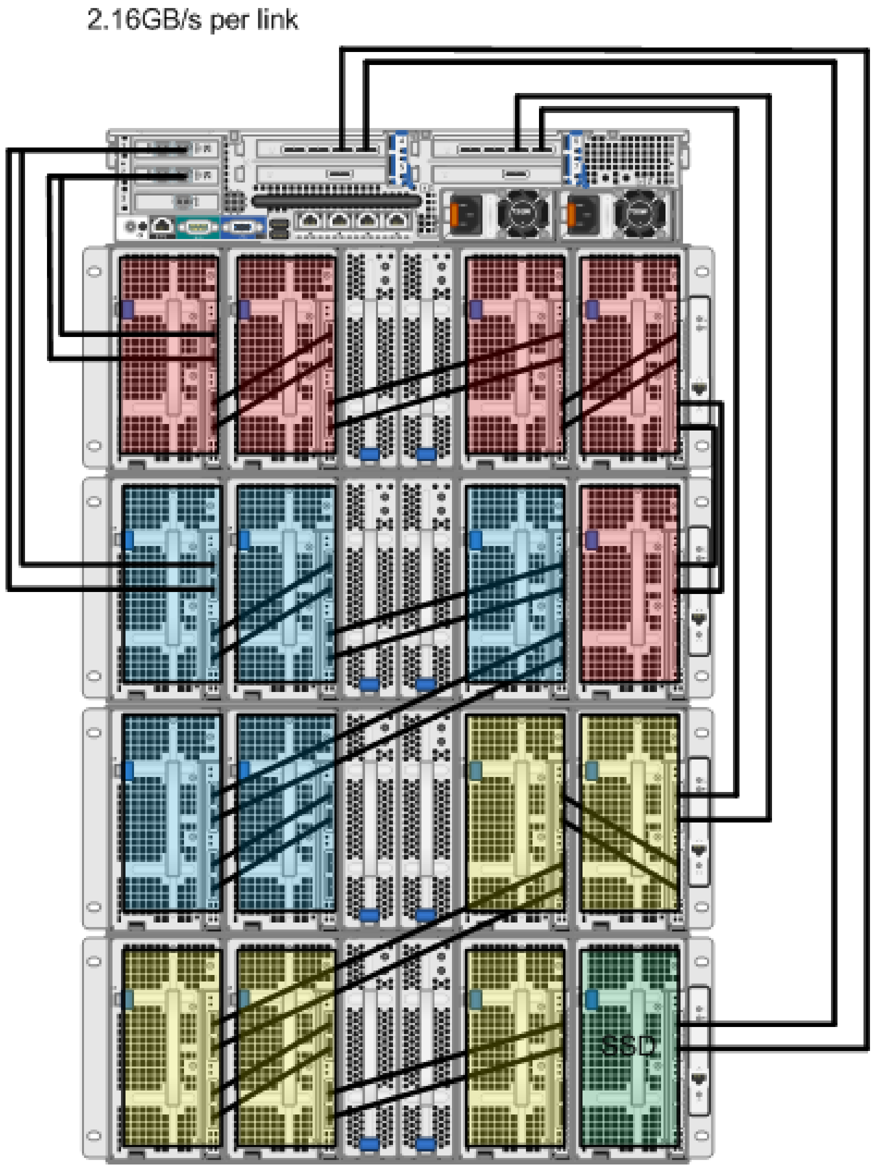


40 units theoretically give 1 Tbyte/s

Object Storage Block

- 16 x C8000XD sleds
- Each Sled: 12 4TB NL-SAS 7.2k HDDs
- Total capacity: 768 TB
- Total throughput: 20 GB/s

- Theoretically 50 units to reach 1 Tbyte/s
- Theoretically ~130 units to reach 100 Pbyte of capacity



Lustre Development at HCPS

High Performance Lustre On Commodity Hardware

- Know-how in setting up and tuning Lustre systems for maximum performance
- Linux Kernel development (in collaboration with Calsoft Inc and Peter Braam)
    - High-Performance (*zero copy)* Software RAID6 for Lustre
- Know-how on maintenance and high-uptime for large Lustre implementations

Future SDP Plans

- Demonstration of very high performance Lustre (> 40 GB/s) on commodity hardware
- Creation IOR scenarios that precisely mimic SDP UV buffer access patterns. Tests on large computer clusters in UK, Germany, Australia and South Africa
- Global Filesystem -> Accelerator IO tests (Cambridge has a 256 K20x cluster)
- Report on scalability limitations of global filesystems
- Investigation of object/key and hybrid global storage systems, compare to filesystems
- Impact of data formatting (e.g., HDF5) on global filesystem performance