Exploiting Non-volatile memory for HPC I/O

Juan F. R. Herrera, Adrian Jackson, et al.

EPCC, The University of Edinburgh, UK

HPC-IODC workshop. Frankfurt, 28th June 2018

Outline

- What is NEXTGenIO?
- Hardware
  - B-APM
  - Architectures
- Software
  - Systemware SW
  - Profiling tools
  - Applications
- Summary

What is NEXTGenIO?

- *Next Generation I/O for the Exascale*
- Addressing the I/O bottleneck of HPC workloads through exploitation of NVRAM technologies
- Aim: bridging the gap between memory and storage
  - Memory: fast read/writes – small capacity
  - Storage: slow read/writes – large capacity

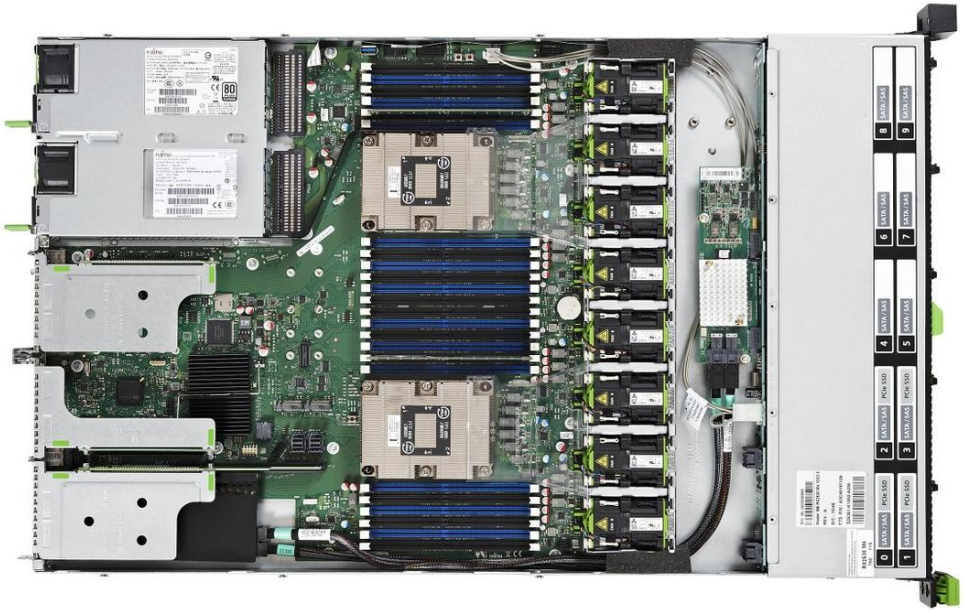
What is NEXTGenIO?

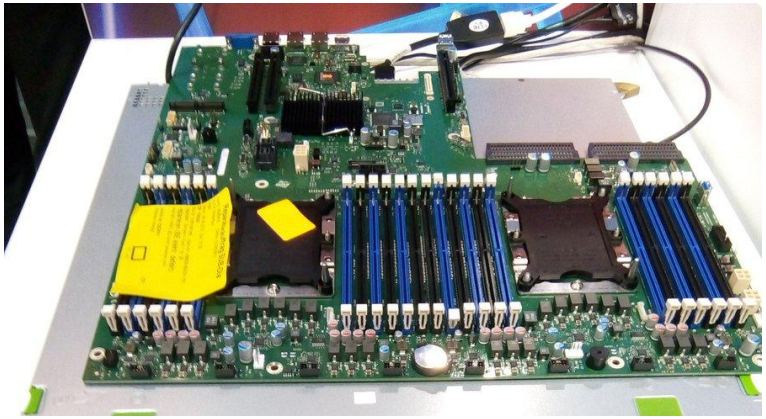
- EC H2020 project
- 48-month duration
- 8.1 million € (50% HW)
- 8 partners, covering:
  - Hardware
  - HPC centres and uses
  - Software
  - Tools development

NEXTGenIO objectives

- Hardware platform prototype to investigate **applicability** for high performance and data-intensive computing
- Understand how best to **exploit NVRAM**
- Develop the necessary **systemware software** to enable (Exascale) application execution on the hardware platform
  - Systemware SW must understand extra level present in the memory hierarchy
- Study application **I/O profiles** and **I/O workloads**
  - How different I/O behaviour and scheduling policies will impact job throughput

Project hardware and software!

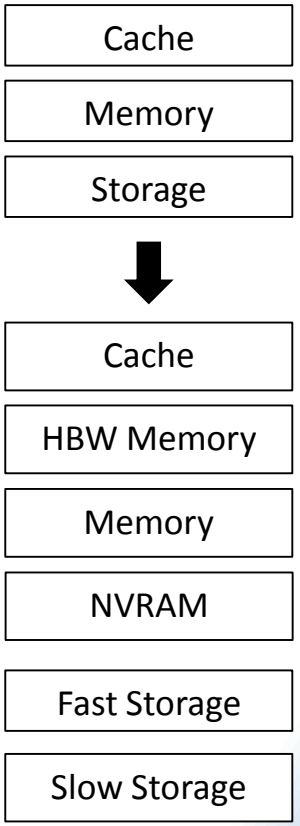




OpenFOAM running on Intel NVDIMMs in Fujitsu motherboard running on BSC echoFS filesystem

New Memory Hierarchies

- High bandwidth, on processor memory
  - Large, high bandwidth cache
  - Latency cost for individual access may be an issue
- Main memory
  - DRAM
  - Costly in terms of energy, potential for lower latencies than high bandwidth memory
- Byte-Addressable Persistent Memory (B-APM)
  - High capacity, ultra fast storage
  - Low energy (when at rest) but still slower than DRAM
  - Available through same memory controller as main memory, programs have access to memory address space



Non-volatile memory

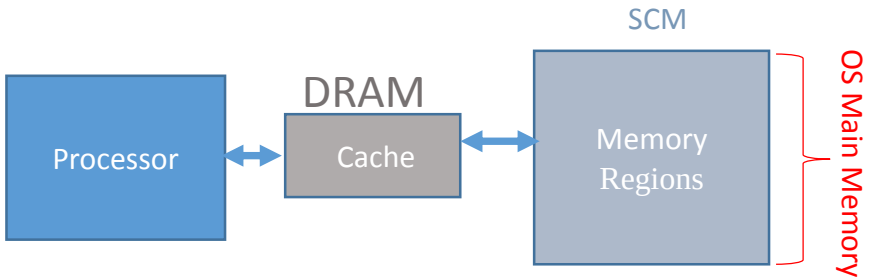
- Non-volatile RAM
  - 3D Xpoint (Intel/Micron) (Intel Optane DC Persistent Memory)
  - Other technology is being developed: STT-RAM
- Much larger capacity than DRAM
  - Hosted in the DRAM slots, controlled by a standard memory controller
- Slower than DRAM by a small factor, but significantly faster than SSDs
- NVDIMM offers
  - Persistency
  - Direct access (DAX) (cache line data access)

NVDIMMs

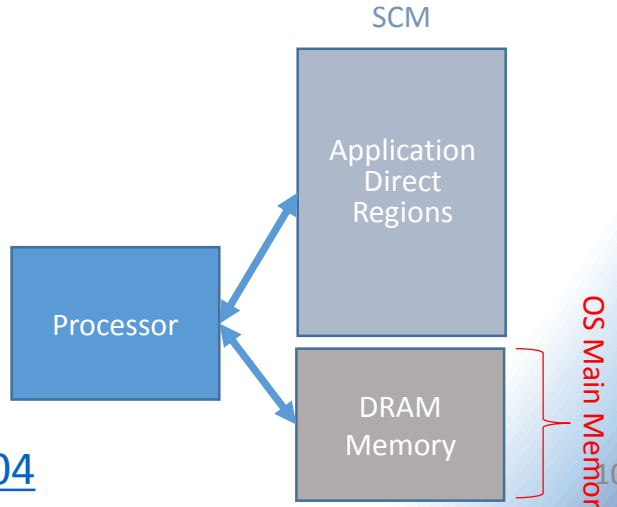
- 3D Xpoint -> Intel Optane DC Persistent Memory
  - NVDIMM-P like (i.e. direct memory access and block device access can be added on top)
  - No DRAM on board
  - Likely to be paired with DRAM in the memory channel
  - Real differentiator (from NVDIMM-N) likely to be capacity and cost

Memory levels

- B-APM in general is likely to have different memory modes* (like MCDRAM on KNL):
  - Two-level memory (2LM)



  - One-level memory (1LM)



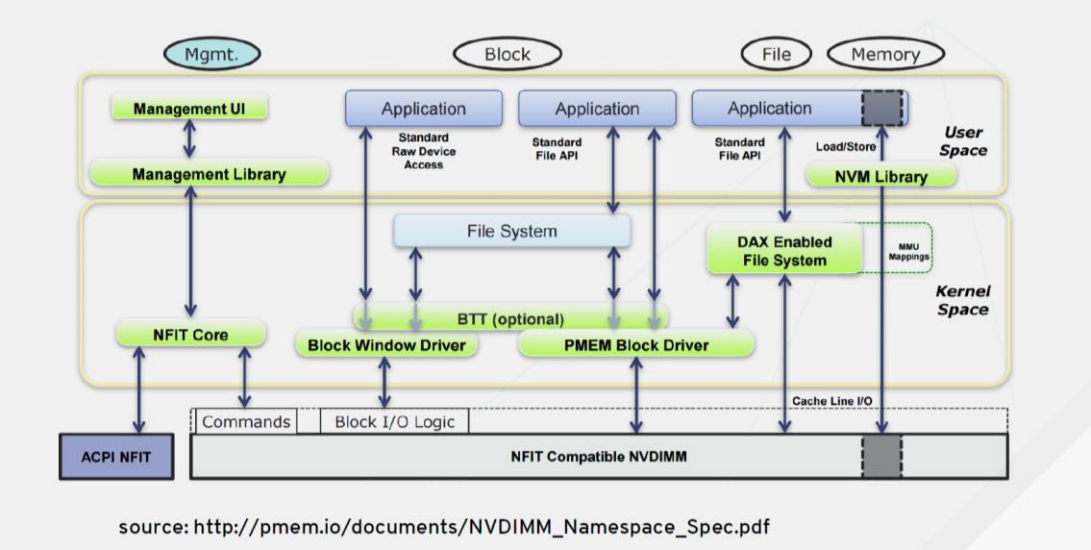
*https://www.google.com/patents/US20150178204

Byte-Addressable Persistent Memory

- The “memory” usage model allows for the extension of the main memory
  - The data is volatile like normal DRAM based main memory
  - Potential for very large memory spaces at reduced cost (capital and recurrent) compared to DRAM
- The “storage” usage model which supports the use of NVRAM like a classic block device
  - E.g. like a very fast SSD
- The “application direct” (DAX) usage model maps persistent storage from the NVRAM directly into the main memory address space
  - Direct CPU load/store instructions for persistent main memory regions

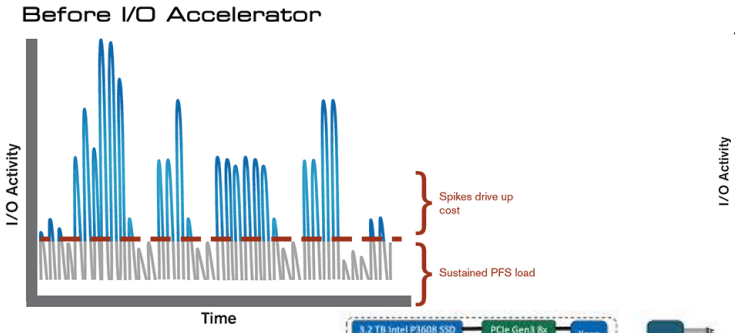
Programming B-APM

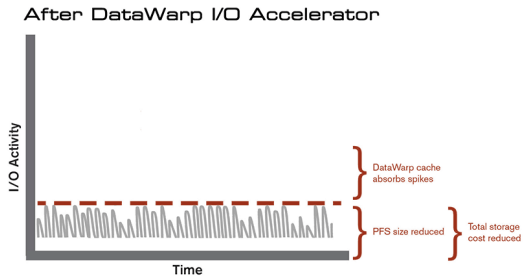
- Block memory mode
  - Standard filesystem API's
  - Will incur block mode overheads (not byte granularity, kernel interrupts, etc…)
- App Direct/DAX mode
  - Volatile memory access can use standard load/store
  - NVM library
    - pmem.io/PMDK
    - Persistent load/store
    - memory mapped file like functionality

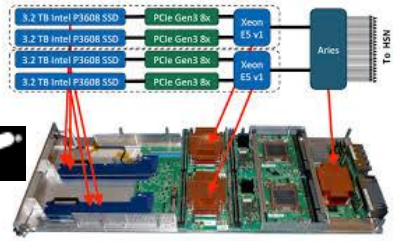


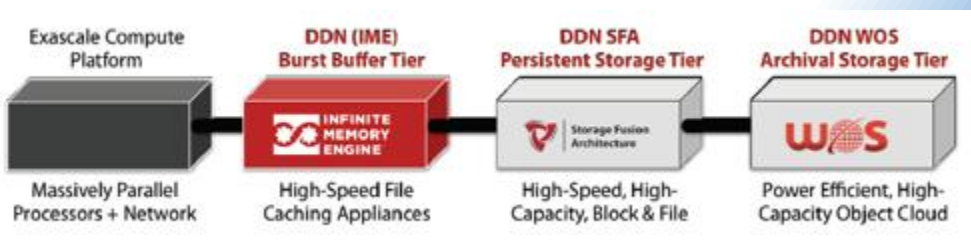
Burst Buffer

- Non-volatile already becoming part of HPC hardware stack
- SSDs offer high I/O performance but at a cost
  - How to utilise in large scale systems?
- Burst-buffer hardware accelerating parallel filesystem
  - Cray DataWarp
  - DDN IME (Infinite Memory Engine)

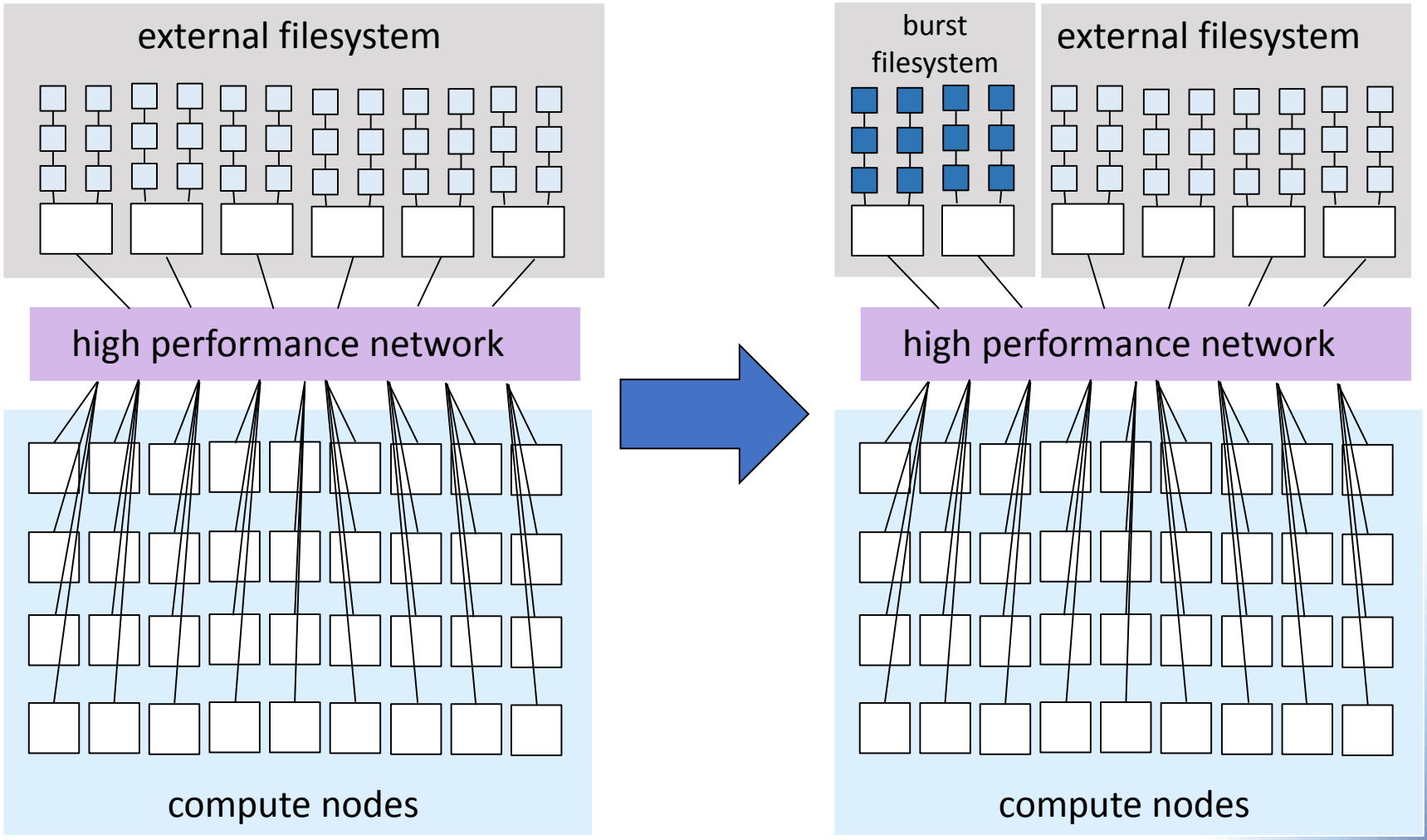




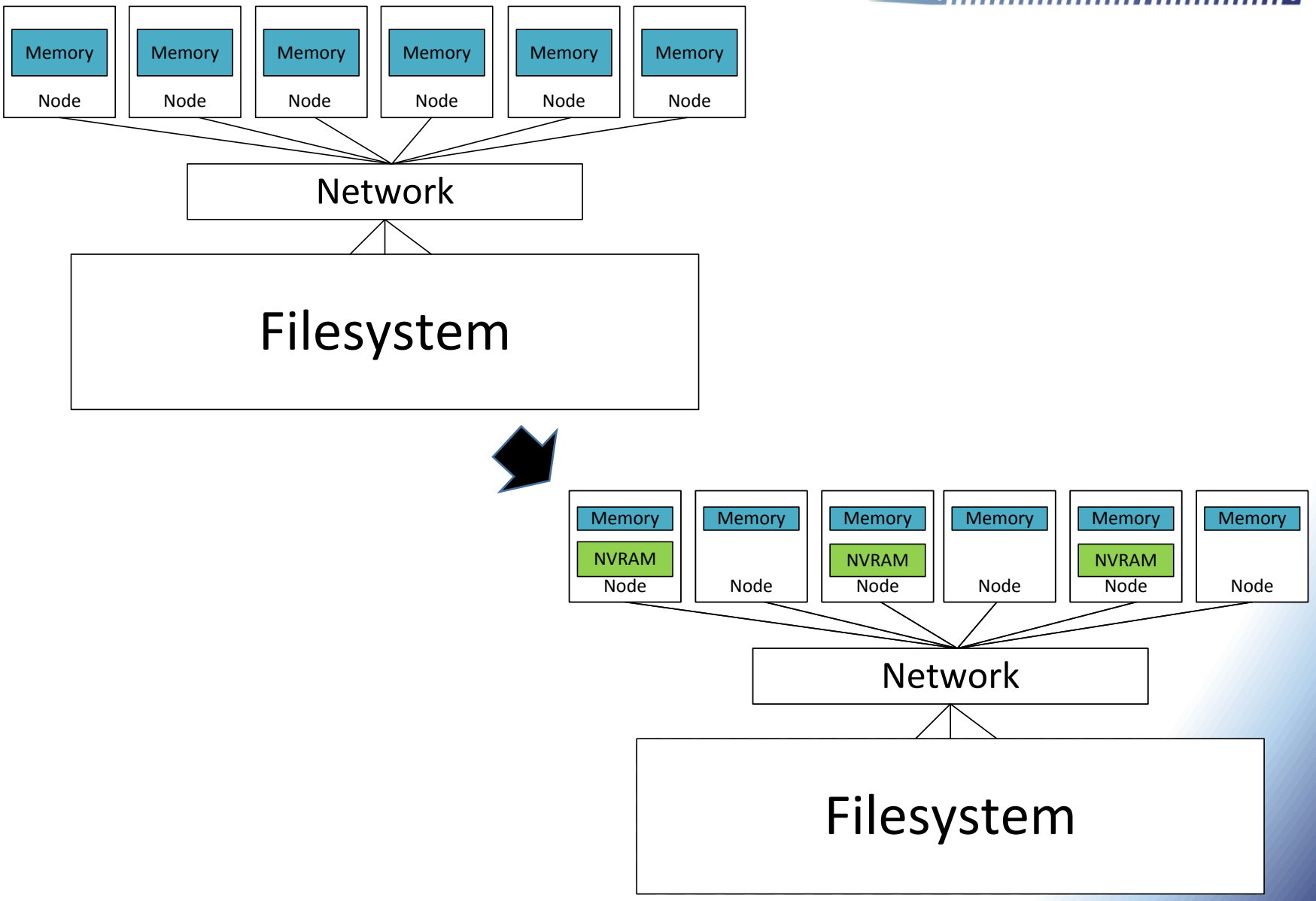




Burst buffer

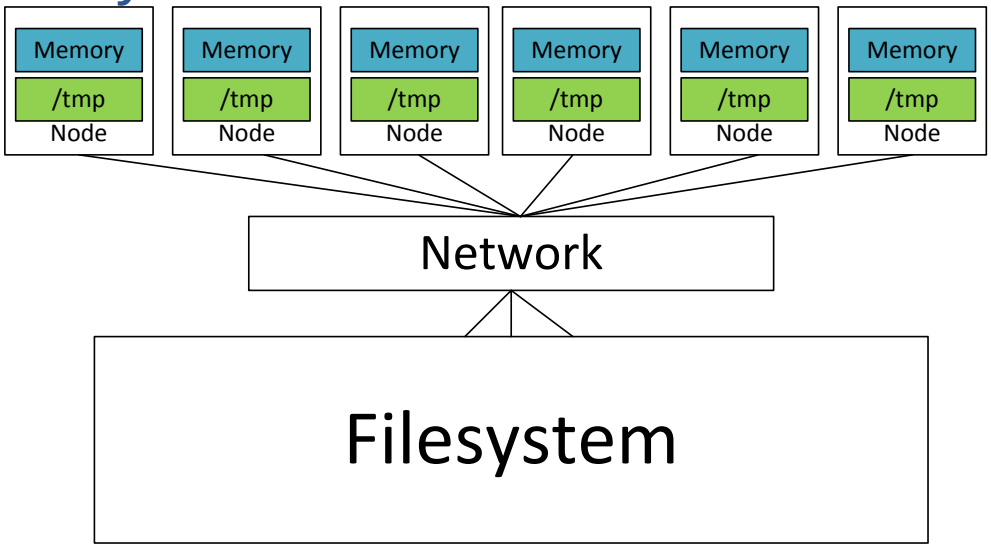


Exploiting distributed storage



Using distributed storage

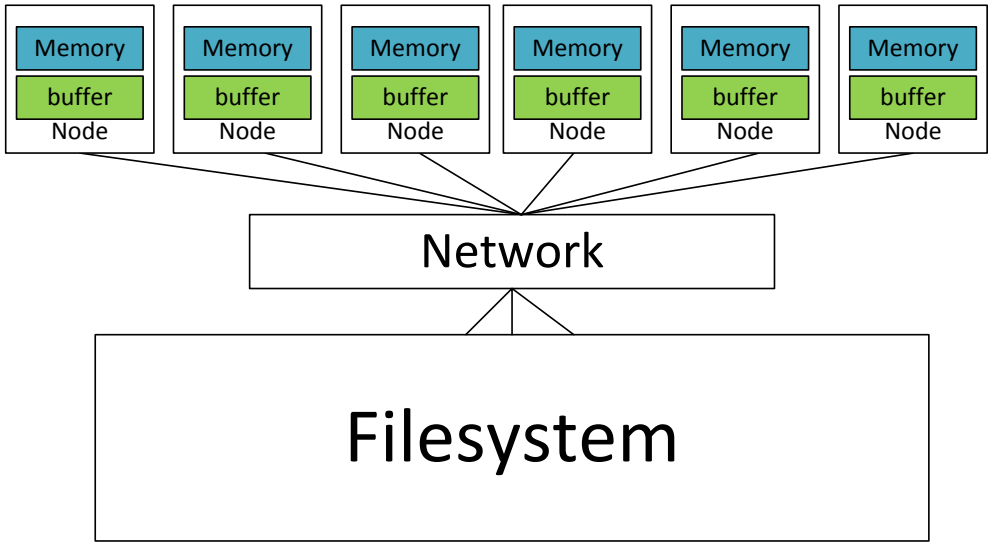
- Without changing applications
  - Large memory space/in-memory database etc…
  - Local filesystem



  - Users manage data themselves
  - No global data access/namespace, large number of files
  - Still require global filesystem for persistence

Using distributed storage

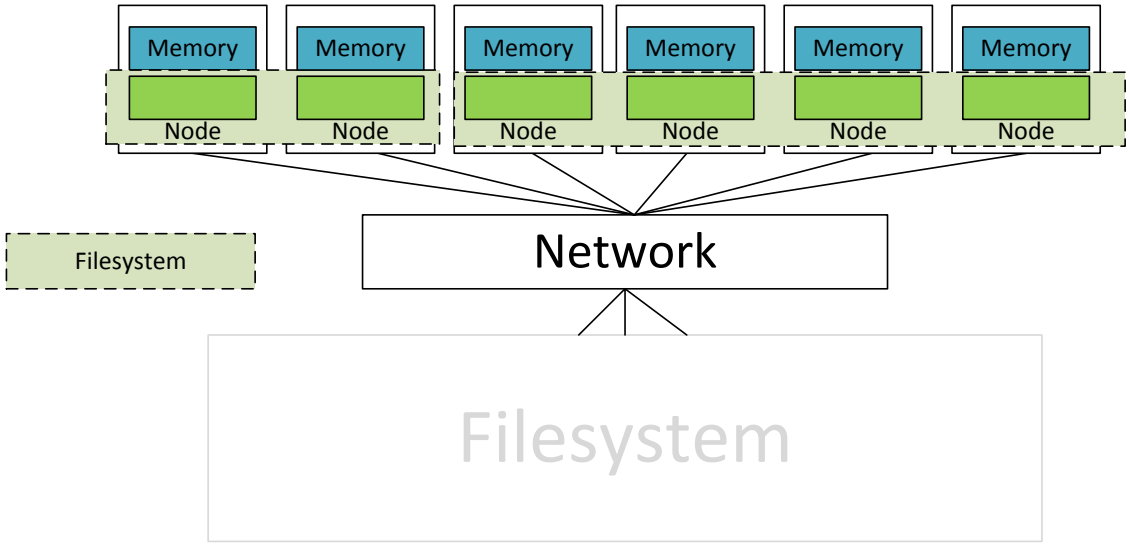
- Without changing applications
  - Filesystem buffer



  - Pre-load data into NVRAM from filesystem
  - Use NVRAM for I/O and write data back to filesystem at the end
  - Requires systemware to preload and postmove data
  - Uses filesystem as namespace manager

Using distributed storage

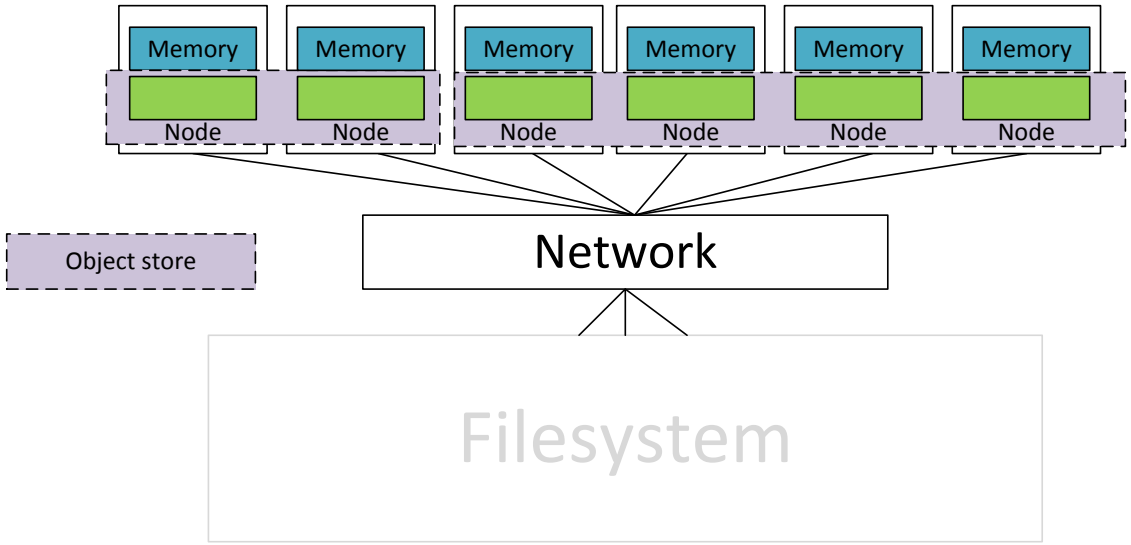
- Without changing applications
  - Global filesystem



  - Requires functionality to create and tear down global filesystems for individual jobs
  - Requires filesystem that works across nodes
  - Requires functionality to preload and postmove filesystems
  - Need to be able to support multiple filesystems across system

Using distributed storage

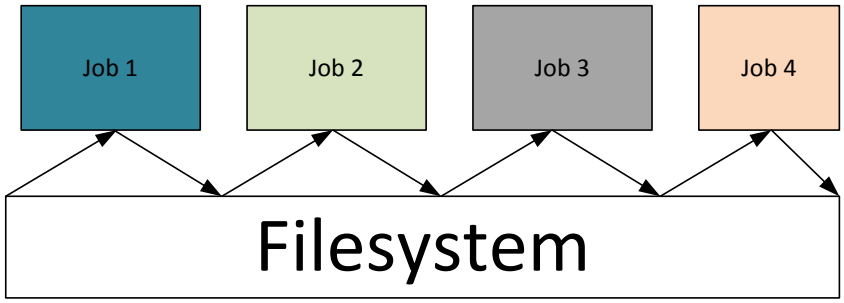
- With changes to applications
  - Object store



- Needs same functionality as global filesystem
- Removes need for POSIX, or POSIX-like functionality

Using distributed storage

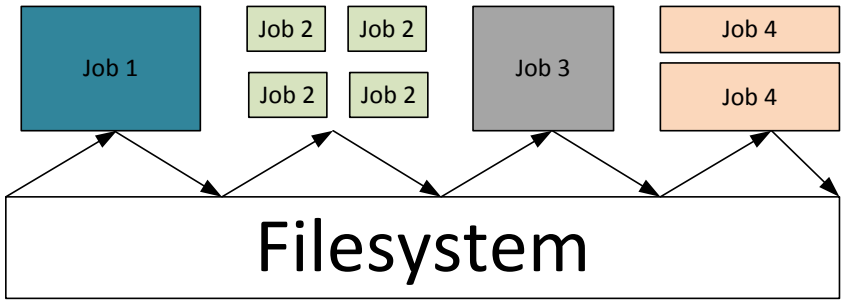
- New usage models
  - Resident data sets
    - Sharing preloaded data across a range of jobs
    - Data analytic workflows
    - How to control access/authorisation/security/etc….?
  - Workflows
    - Producer-consumer model



    - Remove filesystem from intermediate stages

Using distributed storage

- Workflows
  - How to enable different sized applications?



  - How to schedule these jobs fairly?
  - How to enable secure access?

The challenge of distributed storage

- Enabling all the use cases in multi-user, multi-job environment is the real challenge
  - Heterogeneous scheduling mix
  - Different requirements on the SCM
  - Scheduling across these resources
  - Enabling sharing of nodes
  - Not impacting on node compute performance
  - etc….
- Enabling applications to do more I/O
  - Large numbers of our applications don't heavily use I/O at the moment
  - What can we enable if I/O is significantly cheaper?

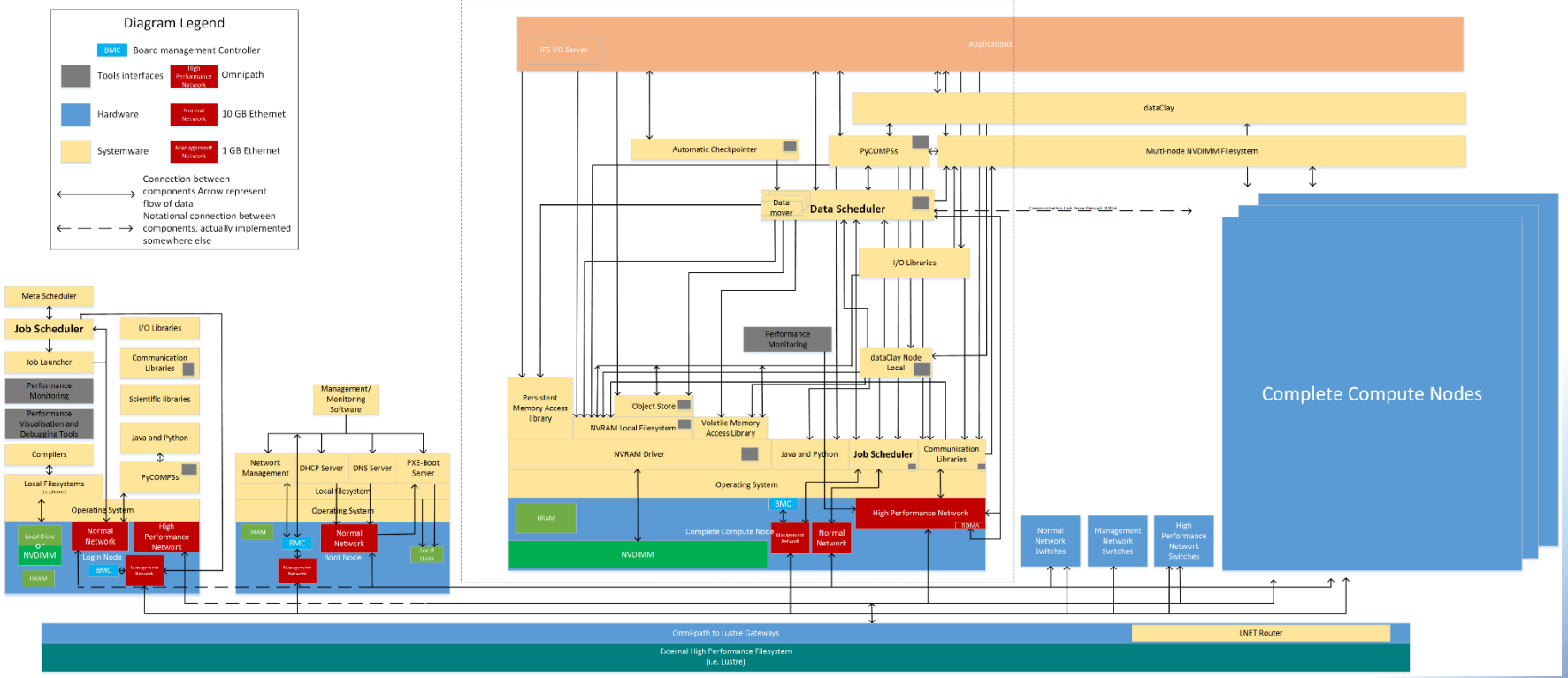
Potential solutions

- Large memory space
- Burst buffer
- Filesystem across NVRAM in nodes
- HSM functionality
- Object store across nodes
- Checkpointing and I/O libraries
- Much of the above require active systemware
  - Integration with the job scheduler
  - Data scheduler for “automatic” data movement
  - Namespace provision and distributed -> single dataview management

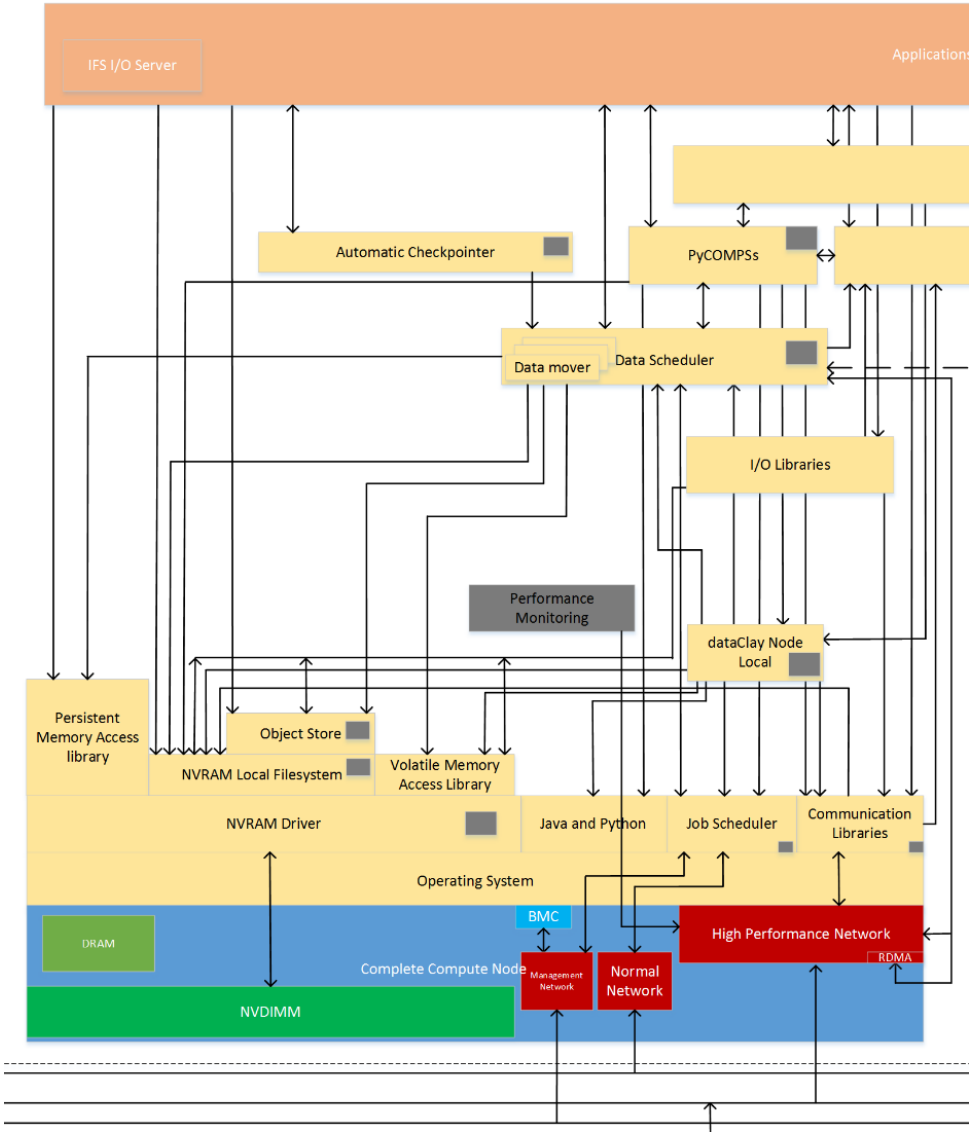
Software

- Systemware software
  - SLURM – job scheduler
  - Data scheduler
  - DAOS and dataClay – object stores as alternatives to file systems
  - echoFS – multi-node NVRAM file system
- Profiling tools
  - Arm Map
  - ScoreP / Vampir
- Applications

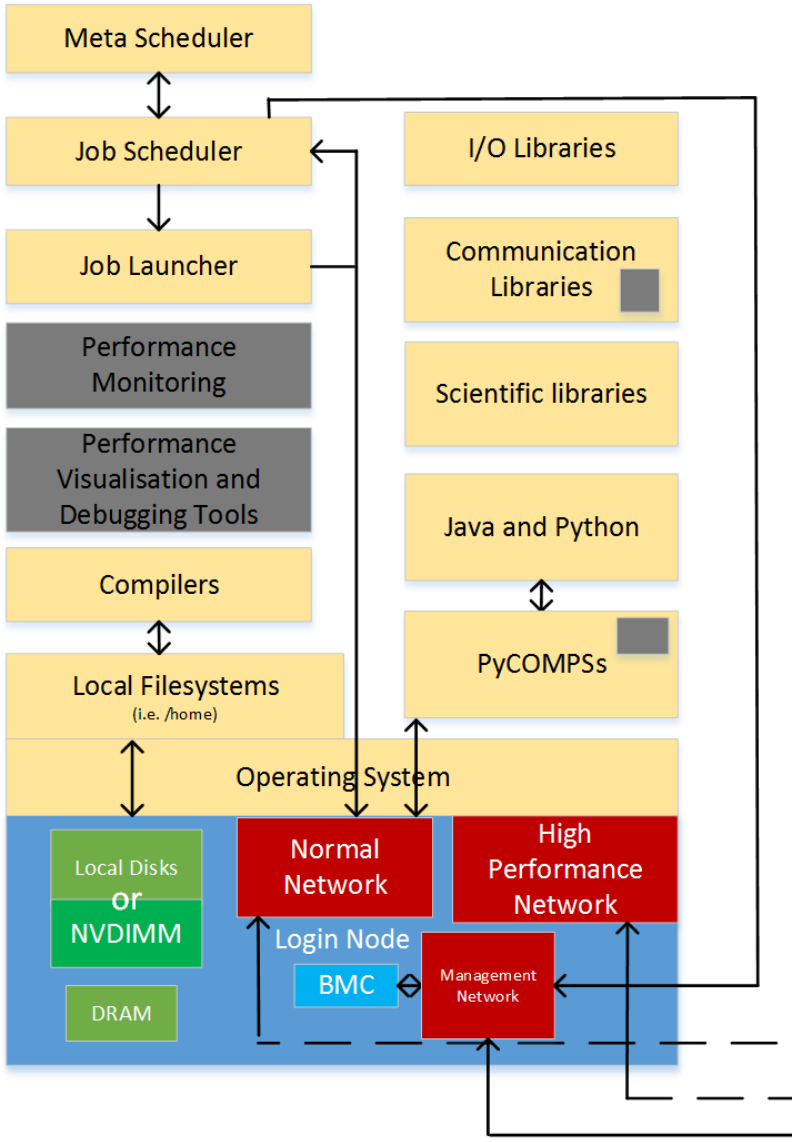
NEXTGenIO Systemware



Compute node systemware



User node systemware



Profiling tools

- Two profiling tools:
  - Map (Arm)
  - ScoreP / Vampir (TUD)
- Feedback has been provided to the developers on features that would be useful towards debugging and performance analysis in the prototype

Profiling tools: features

- Ability to see the activity timeline of a specific rank
- Ability to distinguish I/O to disk and I/O to NVRAM
- Reporting memory usage of NVRAMs
- Both tools will extract memory usage information in 1LM and 2LM modes
- Reporting background I/O transfer between disk and NVRAM (echoFS)
- On ARCHER
  - Lustre data statistics reported *per node*
  - System power usage (through IPMI) per node

Evaluation methodology: objectives

- Define and maintain a **suite of applications and testcases** that will be used to evaluate the NEXTGenIO technology
- Carry out systematic tests and **evaluation** as technology results become available
- Facilitate co-design by providing clear and constructive **feedback** to the technology work packages
- Clearly document the **benefits** of the NEXTGenIO technology, indicate its impact and sketch future lines of development

Evaluation methodology: apps

Combination of traditional and novel HPC applications

- **OpenFOAM**: CFD
- **CASTEP**: chemistry
- **MONC**: cloud modelling
- Halvade: genome sequencing
- OSPRay: ray-tracing
- IFS: weather forecasting
- K-means: machine learning
- Tiramisu: deep learning

Evaluation methodology: scenarios

1. Baseline measurements in today's systems:
   - Use of ARCHER (Cray XC30)
   - Use of ECMWF cluster for IFS
   - Use of Marenostrum for BSC applications (K-means and Tiramisu)
2. Measurements on the NEXTGenIO platform without NVRAM.
3. Measurements on the NEXTGenIO platform with NVRAM.

Summary

- Byte-Addressable Persistent Memory is here
  - Price and capacity remains to be seen, but initial indications are interesting (large, cheaper than DRAM on a per GB)
  - In-node persistent storage likely to come to (maybe some) HPC and HPDA systems shortly
  - Applications can program directly but….
  - …potentially systemware can handle functionality for applications, at least in transition period
- Interesting times
  - Convergence of HPC and HPDA (maybe)
  - Different data usage/memory access models may become more interesting
  - Certainly benefits for single usage machines, i.e. bioinformatics, weather and climate, etc…

Further reading

- http://www.nextgenio.eu
- Architectures for High Performance Computing and Data Systems using Byte-Addressable Persistent Memory http://arxiv.org/abs/1805.10041
- The project will have a system (~30 nodes) with this memory in it in early 2019
  - Get in touch if you're interested in access

EOF

Thanks for your attention!
Any questions?

Dr Juan F. R. Herrera
Applications Developer
EPCC – The University of Edinburgh
j.herrera [at] epcc.ed.ac.uk

The NEXTGenIO project received funding from the EU Horizon 2020 research and innovation programme under grant agreement No 671591.