Persistent Memory Programming

Adrian Jackson
EPCC
a.jackson@epcc.ed.ac.uk
@adrianjhpc

Javier Conejero
BSC

Christian Herold
TU Dresden

Aims

- Understand persistent memory hardware and software
- Understand the different aspects that can impact performance, and the share nature of the resources
- Learn how to program persistent memory
- Understand profiling and analysing persistent memory usage
- Get hands on with persistent memory hardware

Real Aims

- Understand data movement and application data requirements
- Understanding I/O and data movement is hard
- Thinking about different ways you can do I/O or storing data
- Understand some tools that can help you

Format

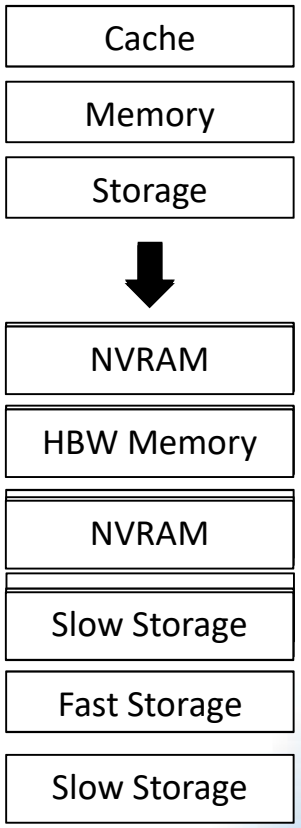
- Lectures and practicals
- Slides and exercise material available online:
  - https://github.com/adrianjhpc/iotutorial
- Exercises will be done on remote machine
  - NEXTGenIO prototype
  - We will give you accounts on these
- Some programming for those who want to, for others can just run programs
- Some setting up of profiling tools

Timetable

- 09.00 Persistent memory hardware
- 09.30 Persistent memory performance practical
- 10.00 Persistent memory software
- 10.45 Break
- 11.30 Persistent memory programming practical
- 11.30 Alternative methods of exploiting persistent memory
- 12.00 Profiling and debugging persistent memory applications
- 12.30 Summary and Finish

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
- NVRAM main memory
  - High capacity, ultra fast storage
  - Low energy (when at rest) but still slower than DRAM

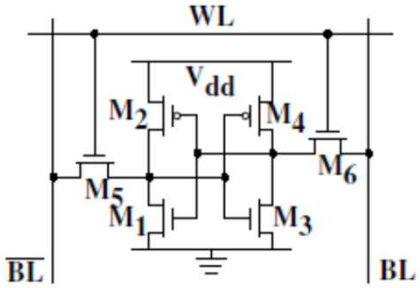


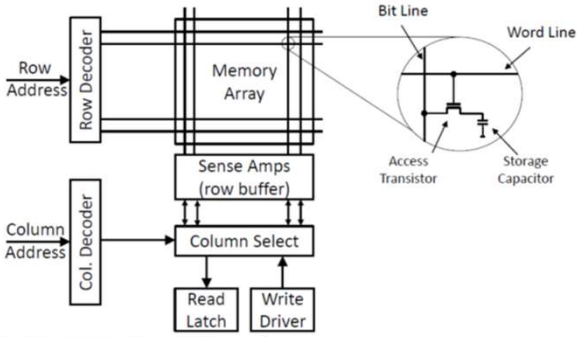
Non-volatile memory

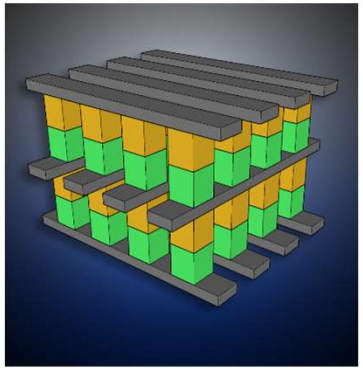
- Non-volatile RAM
  - 3D XPoint technology
  - STT-RAM
- Much larger capacity than DRAM
  - Hosted in the DRAM slots, controlled by a standard memory controller
- Slower than DRAM by a small factor, but significantly faster than SSDs
- STT-RAM
  - Read fast and low energy
  - Write slow and high energy
    - Trade off between durability and performance
    - Can sacrifice data persistence for faster writes

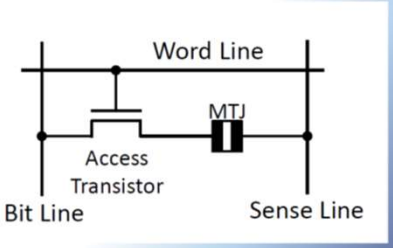
SRAM vs NVRAM

- SRAM used for cache
- High performance but costly
  - Die area
  - Energy leakage
- DRAM lower cost but lower performance
  - Higher power/refresh requirement
- NVRAM technologies offer
  - Much smaller implementation area
  - No refresh/ no/low energy leakage
  - Independent read/write cycles









NEXTGenIO - key facts

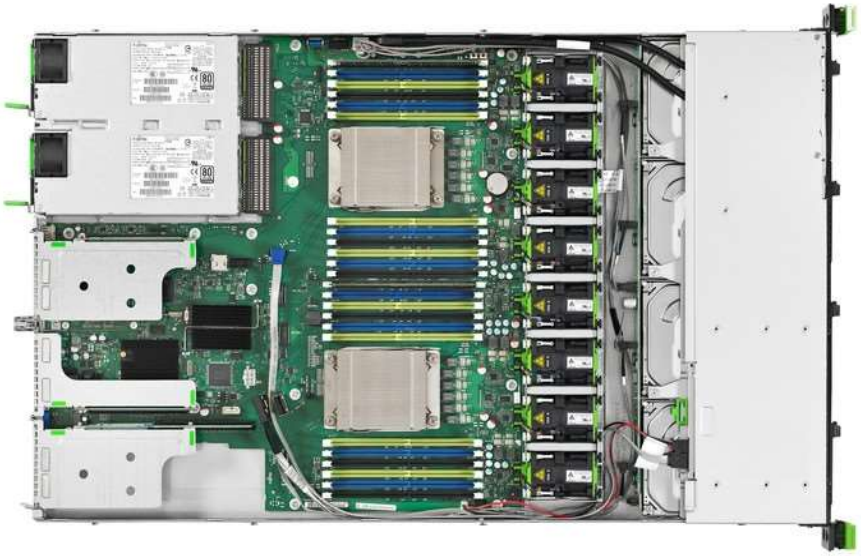
- Research & Innovation Action
- 48 month duration
- €8.1 million
- Approx. 50% committed to hardware development
- Prototype system part of the project

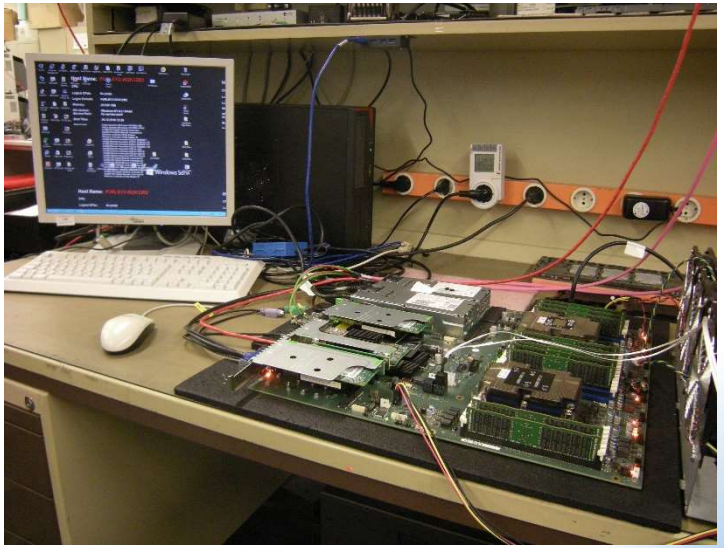
Project objectives

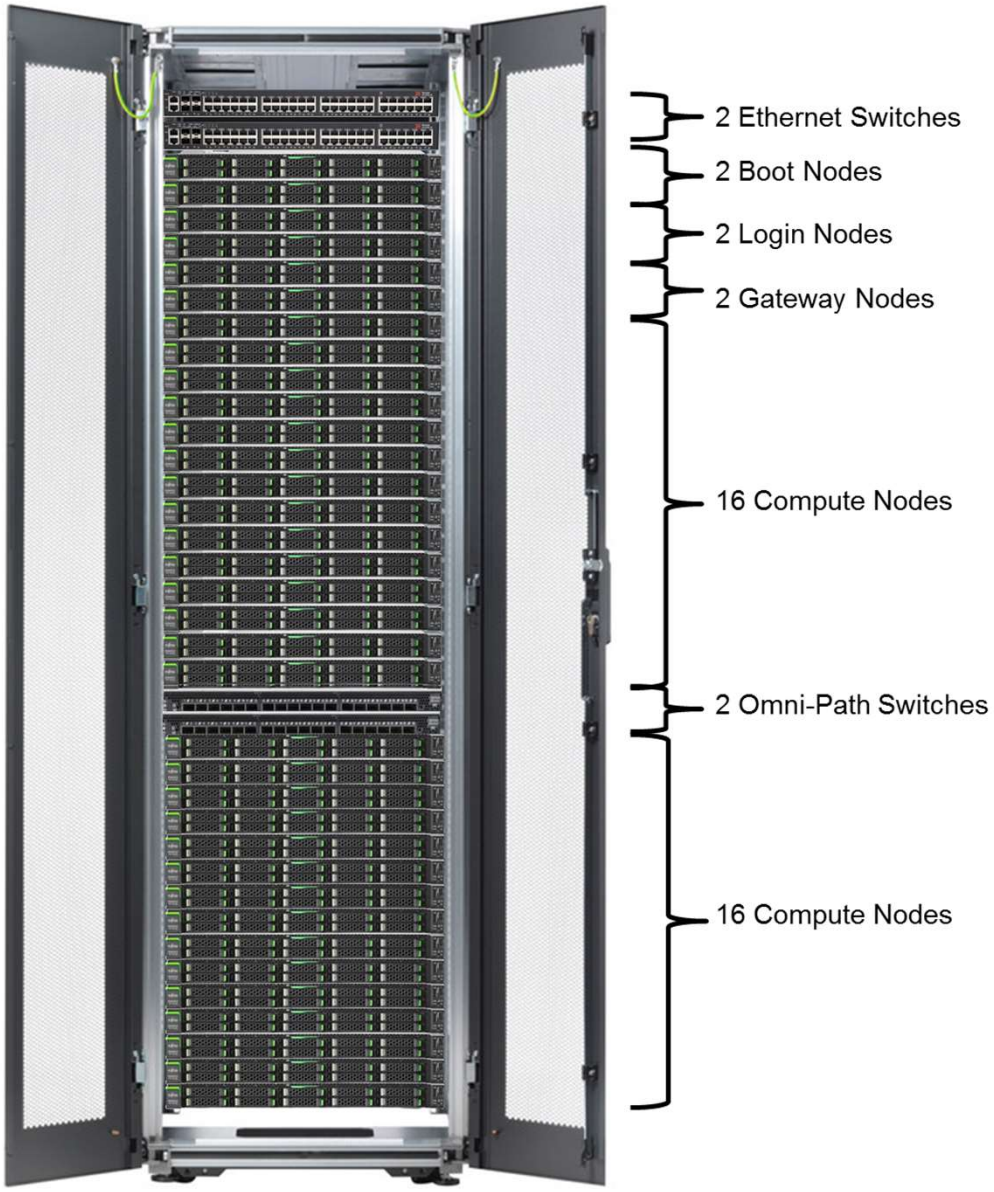
- Hardware platform prototype
  - Demonstrating the prototype's broad applicability for both HPC and data centric applications
- Exascale I/O investigation
  - Understanding how best to exploit NVRAM
- Systemware development:
  - Producing the necessary software to enable Exascale application execution on the hardware platform
- Application co-design
  - Understanding individual application I/O profiles and typical I/O workloads on shared systems running multiple different applications

Hardware

- Motherboard developed at Fujitsu factory in Augsburg





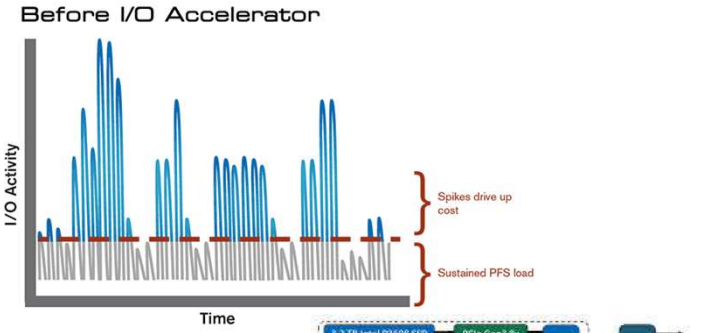

2 Ethernet Switches
2 Boot Nodes
2 Login Nodes
2 Gateway Nodes
16 Compute Nodes
2 Omni-Path Switches
16 Compute Nodes

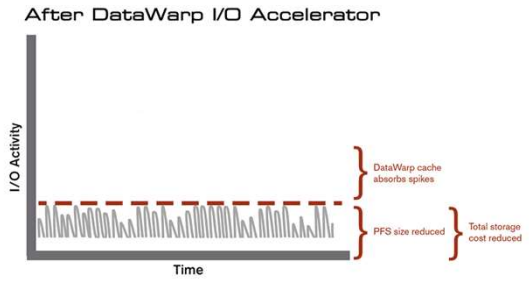
Systemware

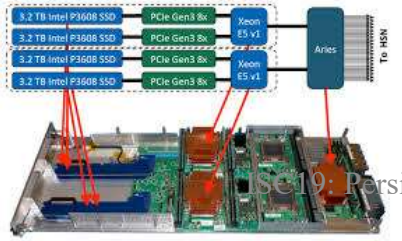
- Work on adapting job scheduler (SLURM)
- Development of a data scheduler
- Object stores as alternatives to file systems
  - DAOS (Distributed Application Object Storage)
  - dataClay
- Multi-node NVRAM file system
  - echoFS

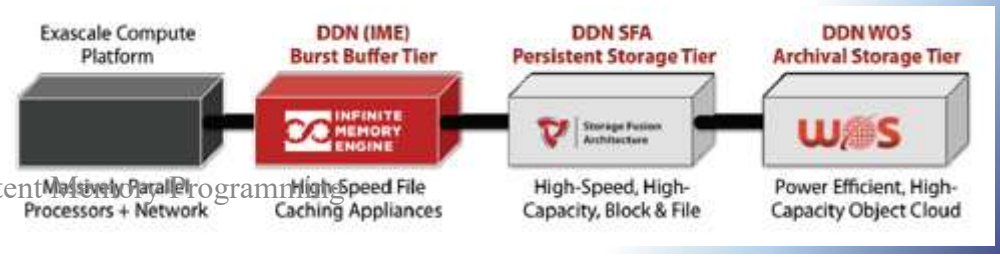
Burst Buffer

- 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)

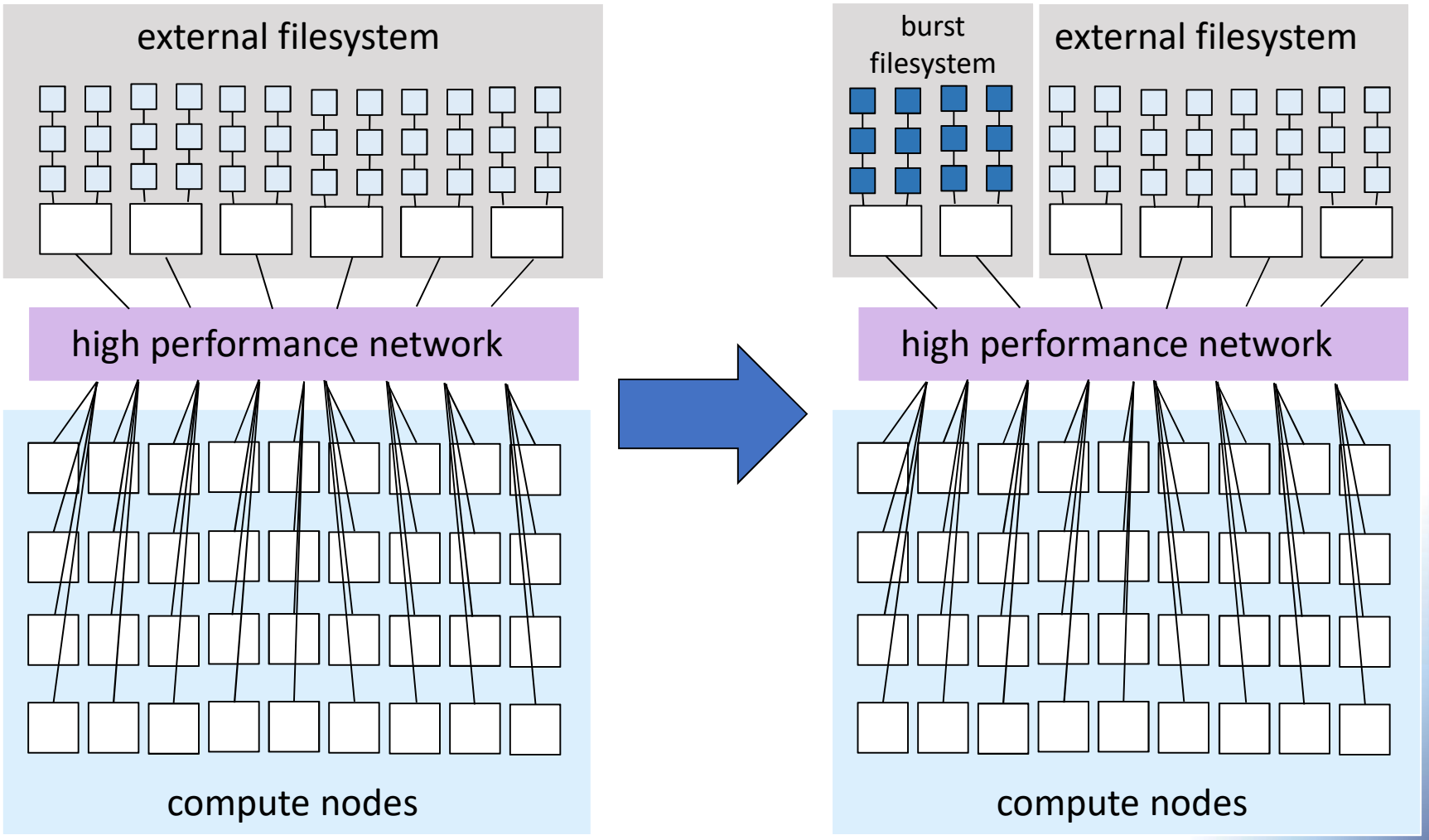






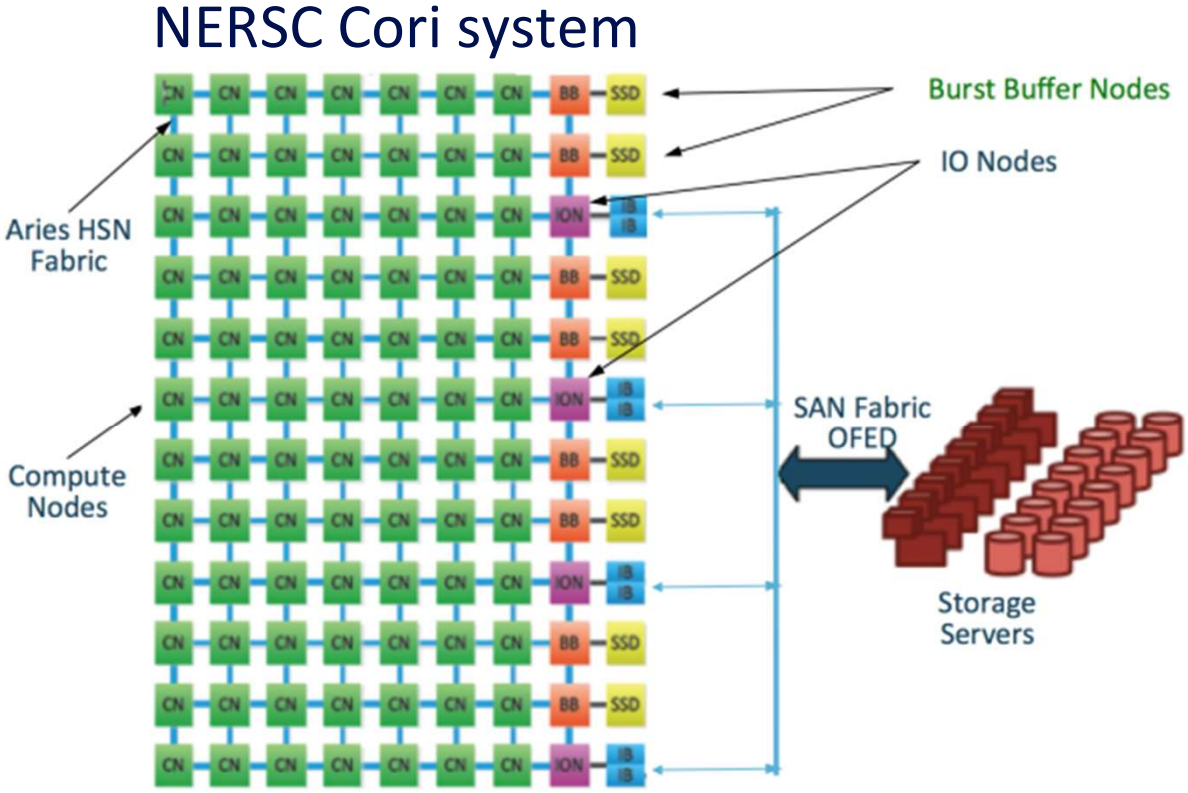


Burst buffer



Burst buffer

- Burst buffer is (generally) separate resource to filesystem
- Managing usage/allocation/data movement is user responsibility
- Storage has compute
- Resource is external
- Scheduling can be separate
  - Usage not required



Summary

- Please don’t hesitate to ask questions!
- We are doing practicals
  - But if you’re not confident in programming we have other options
- We are aiming at different experience levels so if it’s too easy/you know it already let us know