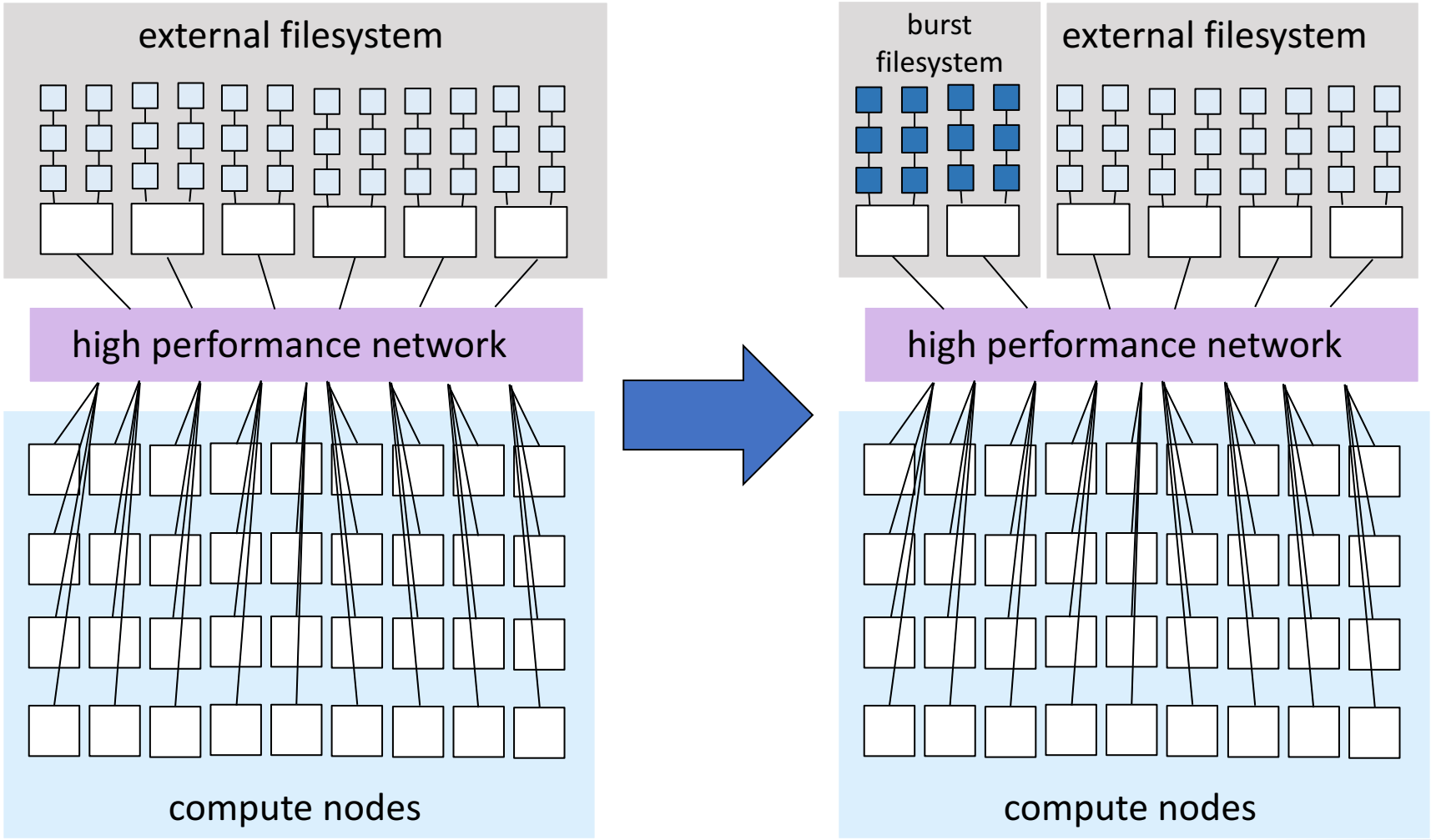

# Non-Volatile Memory for Next Generation I/O

Dr Michèle Weiland
m.weiland@epcc.ed.ac.uk

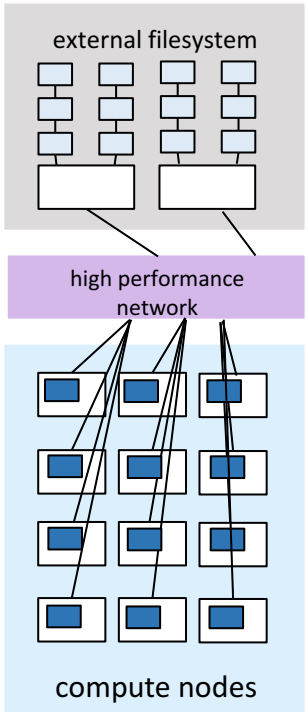
# Current trends & approaches

# Burst buffer



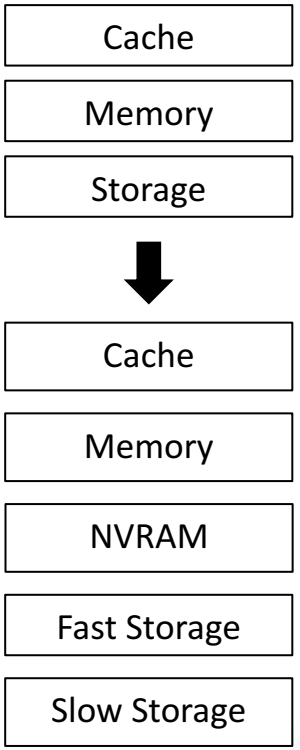
# Moving beyond burst buffer

- Non-volatile is coming to the node rather than the filesystem
- Argonne Theta machine has 128GB SSD in each compute node




# Non-volatile memory

- Non-volatile RAM
  - 3D XPoint technology is one example
- Much larger capacity than DRAM
  - Hosted in the DRAM slots (DIMM form factor), controlled by a standard memory controller
- Slower than DRAM by a small factor, but significantly faster than SSDs



# The NEXTGenIO approach

# NEXTGenIO key facts

- FETHPC Research & Innovation Action
- Active for 18 months so far
- 8 partners, covering
  - Hardware
  - HPC centres and users
  - Software
  - Tools developers

# Our 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

# Systemware

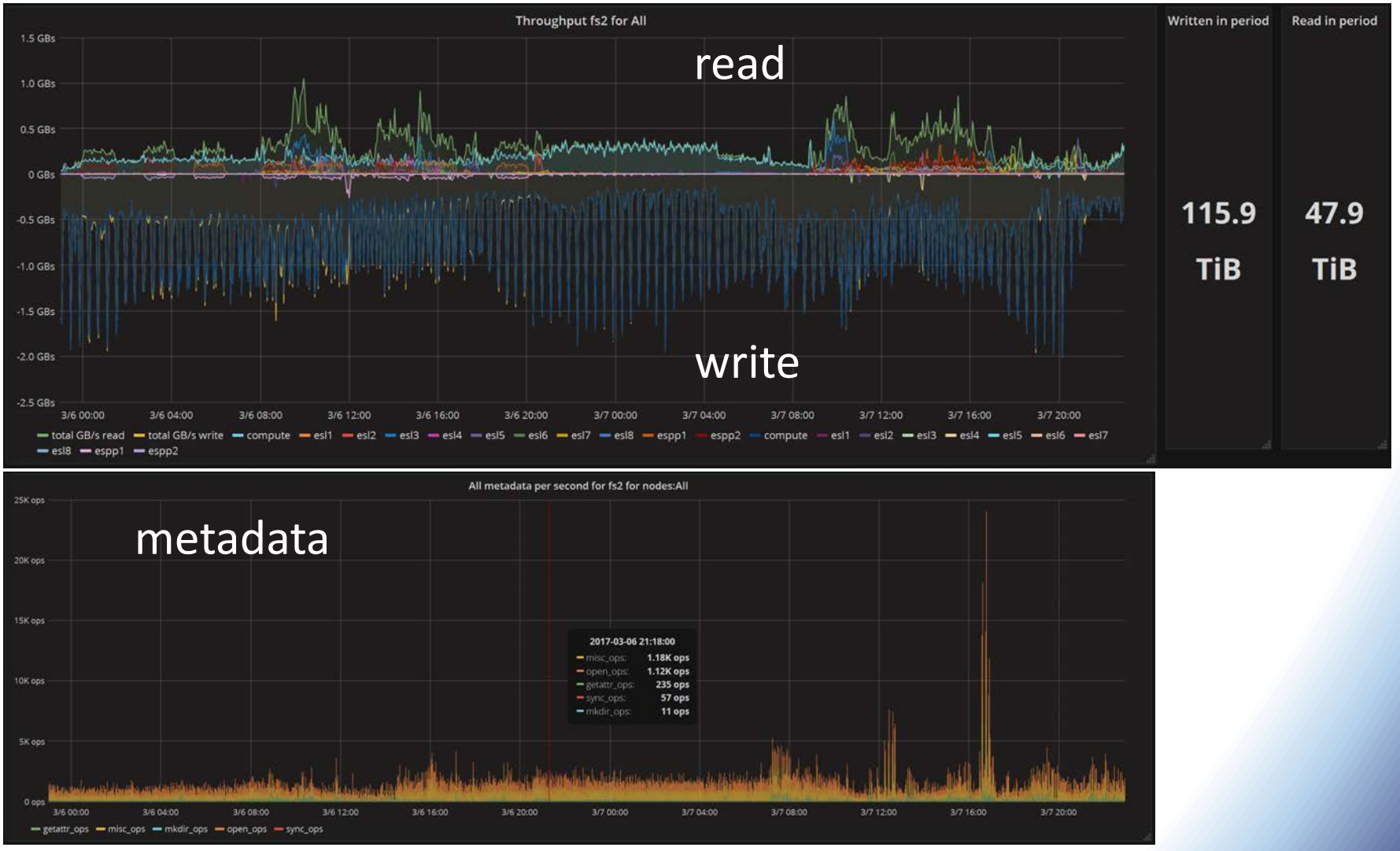

- System software must understand extra level present in the memory hierarchy
- Work on adapting job scheduler (SLURM)
- Development of a data scheduler
- Object stores as alternatives to file systems
  - DAOS (Distributed Application Object Storage)
  - dataClay
- Multi-node NVRAM file system
  - echoFS

☛ **Key goal:** Platform must be usable “as is” for legacy applications

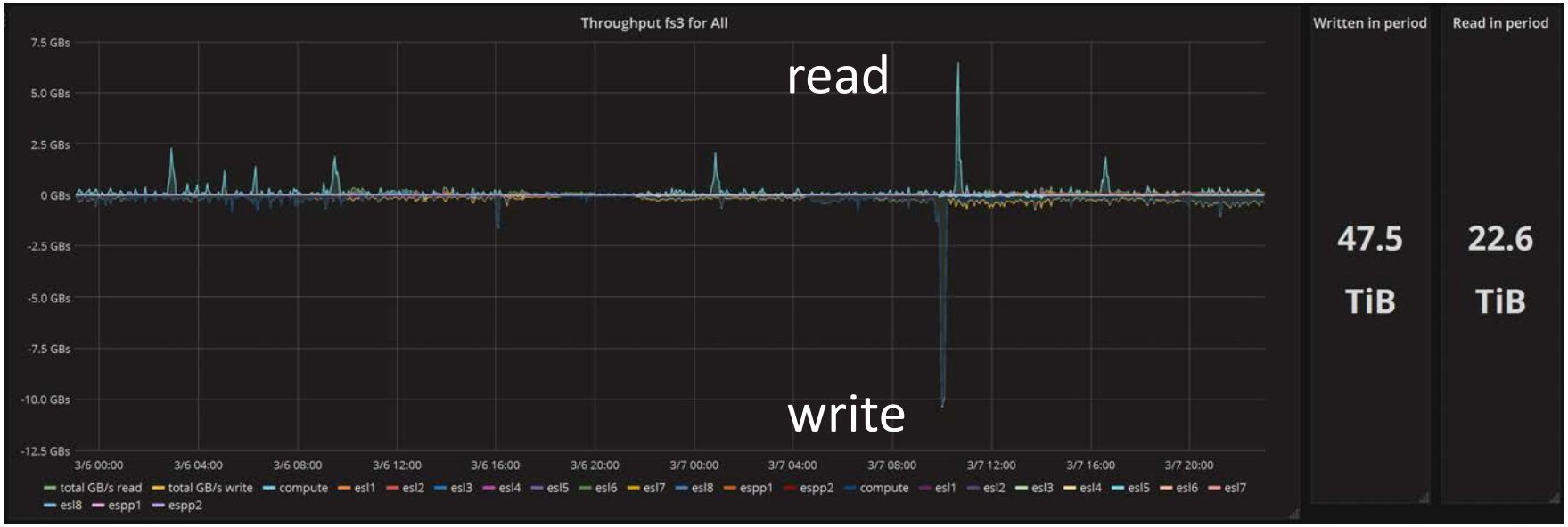
# Workloads & I/O

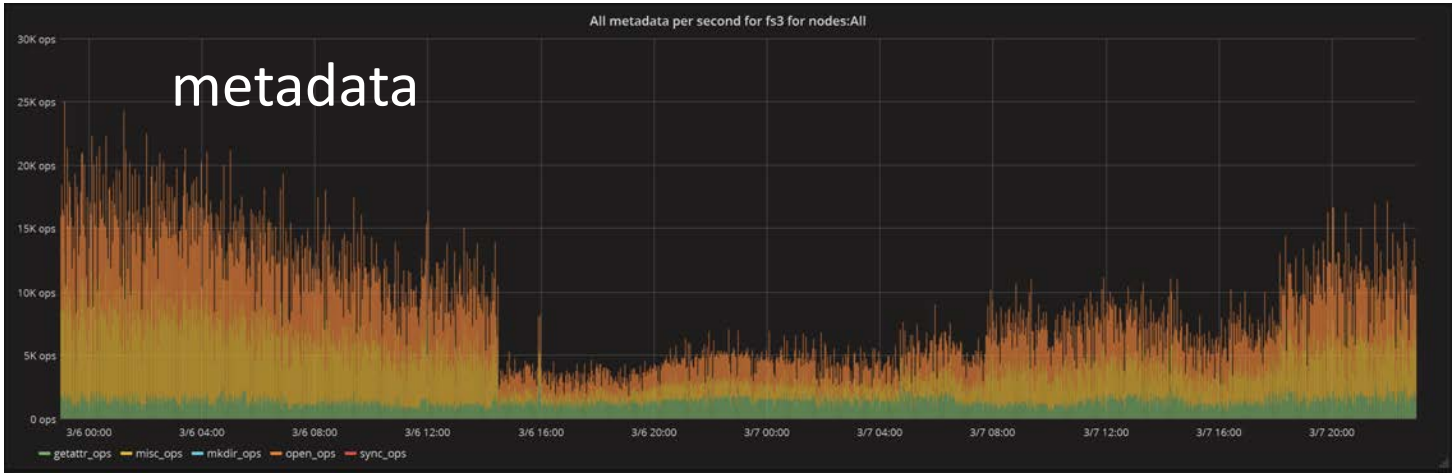
- Try and understand how different I/O behaviour and scheduling policies will impact job throughput
  - Three different workloads
    - Generic → EPCC
    - Special purpose → ECMWF
    - Commercial → Arctur
- I/O Workload Simulator
  - Create benchmark of synthetic jobs generated from real workloads → to be deployed on HPC system
  - Create simulator of workload schedule to test impact of policies and I/O performance → to be deployed on laptop

# ARCHER workload



# ARCHER workload



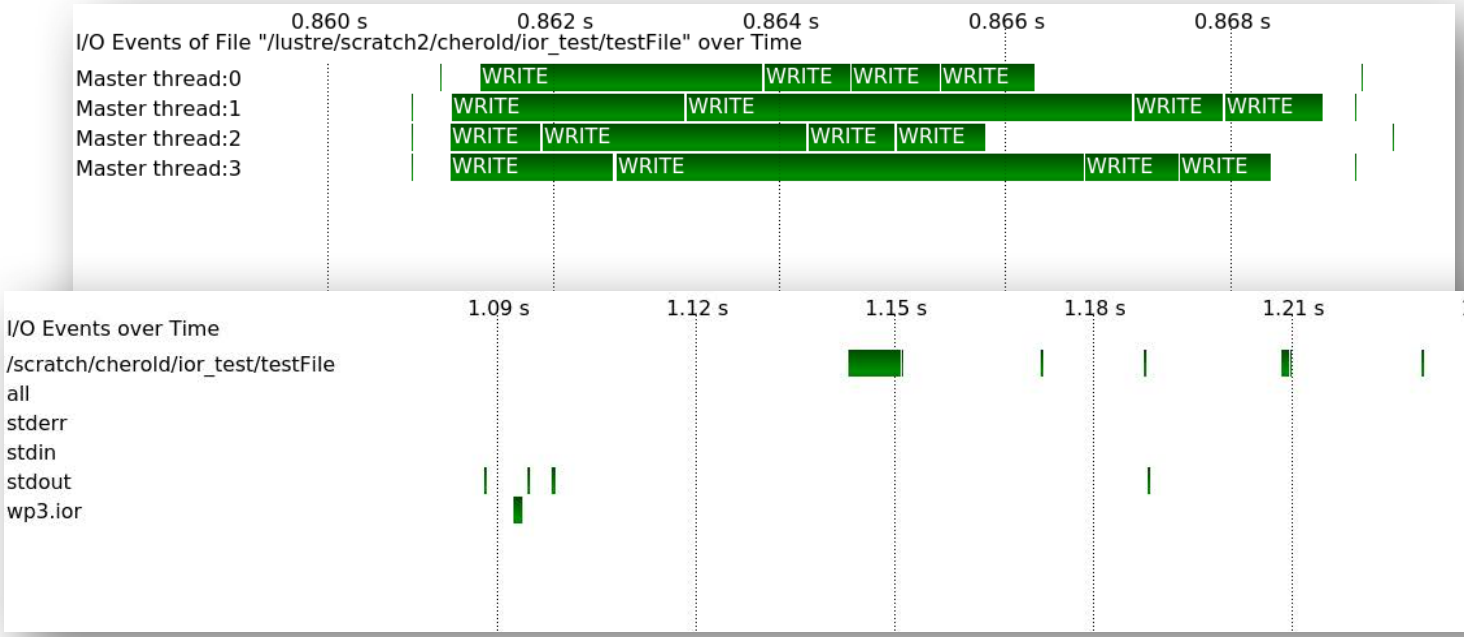



# Tools support

allinea MAP

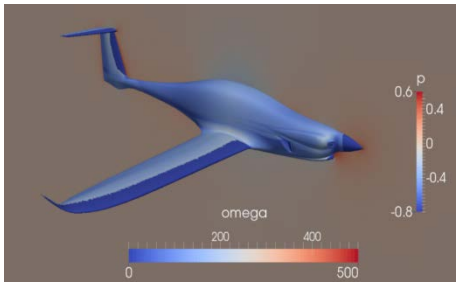
nextgenio

- Profiling and debugging tools need to be able to understand implications of an additional (potentially persistent) memory layer

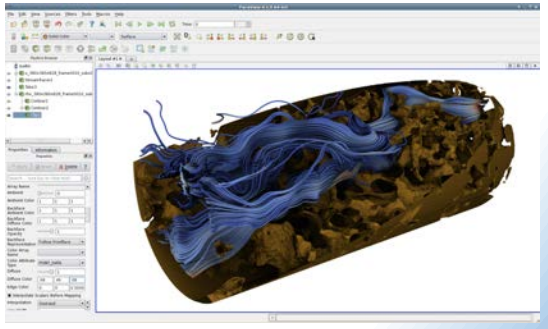


# Applications

- Traditional HPC
  - OpenFOAM → CFD
  - CASTEP → Chemistry
  - IFS → weather forecasting
  - MONC → cloud modelling




- Novel uses
  - OSPRay → ray tracing, rendering engine
  - Halvade → genome sequencing
  - Tiramisu → deep learning (based on Caffe)
  - K-means → ML

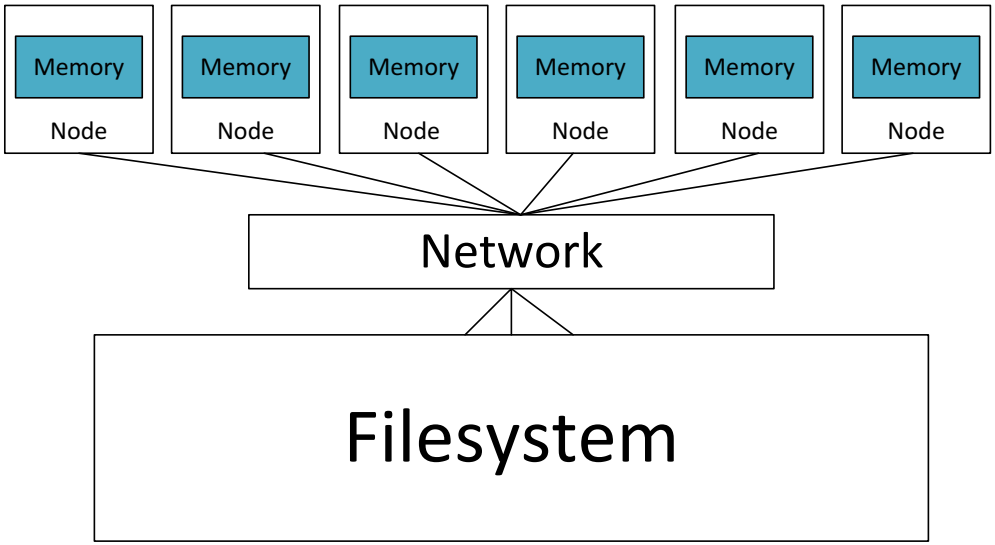


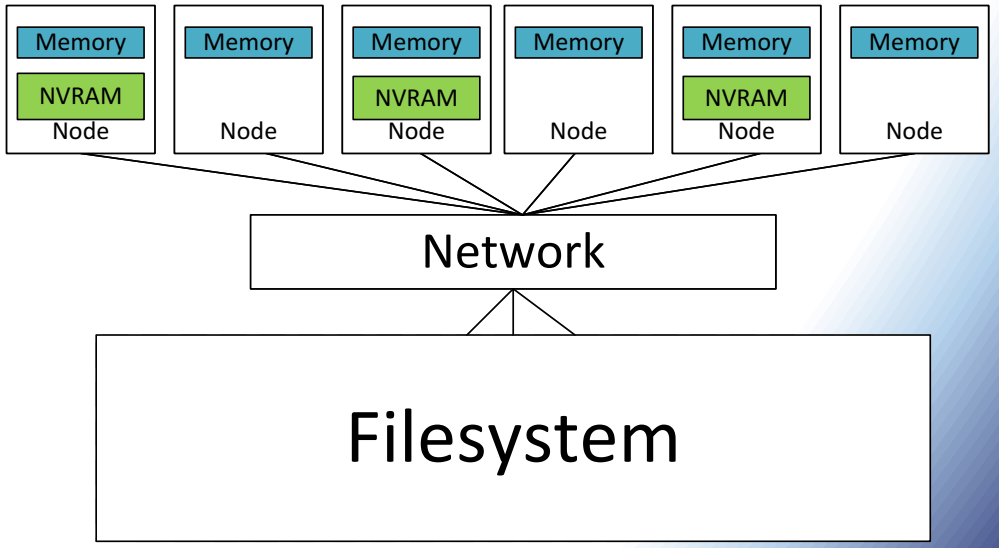
# Usage models

# NVRAM usage models


- The “memory” usage model allows for the extension of the main memory
  - The data is volatile like normal DRAM based main memory
- 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” 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

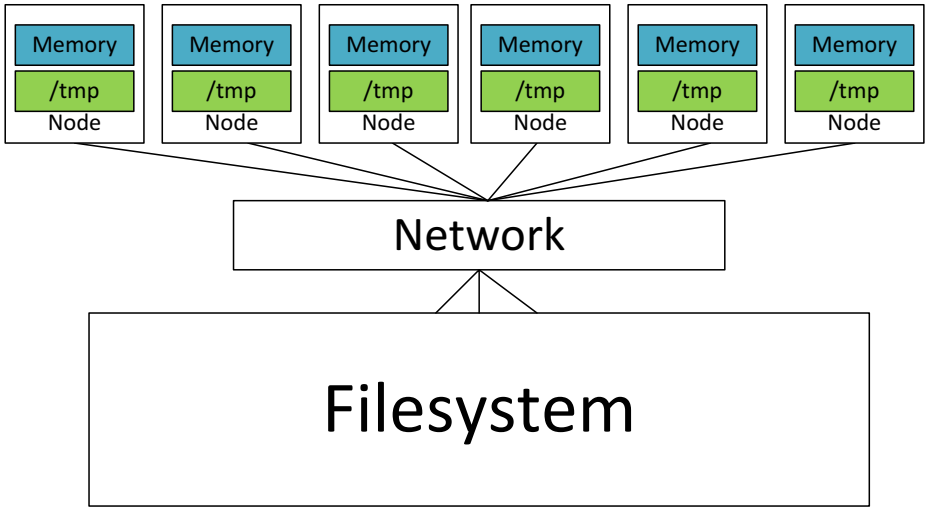
# Exploiting distributed storage



# Using distributed storage

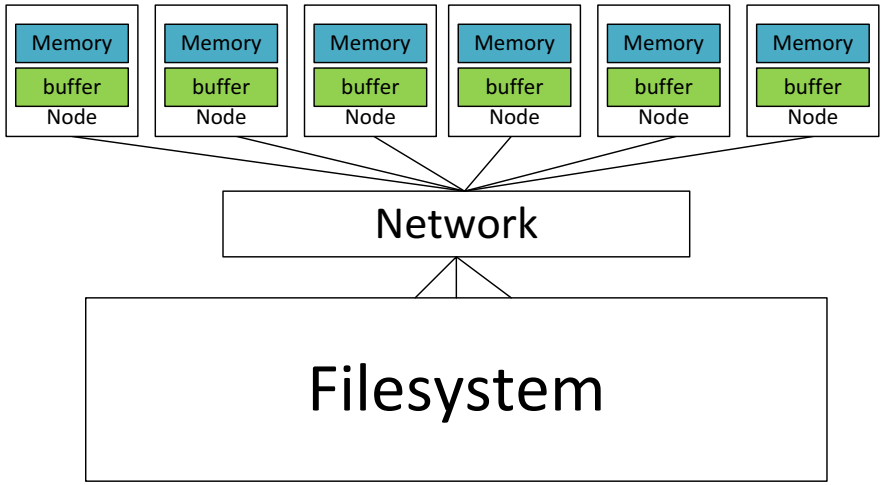
- Without changing applications
  - Large memory space/in-memory database etc…
  - Local filesystem




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

# Using distributed storage

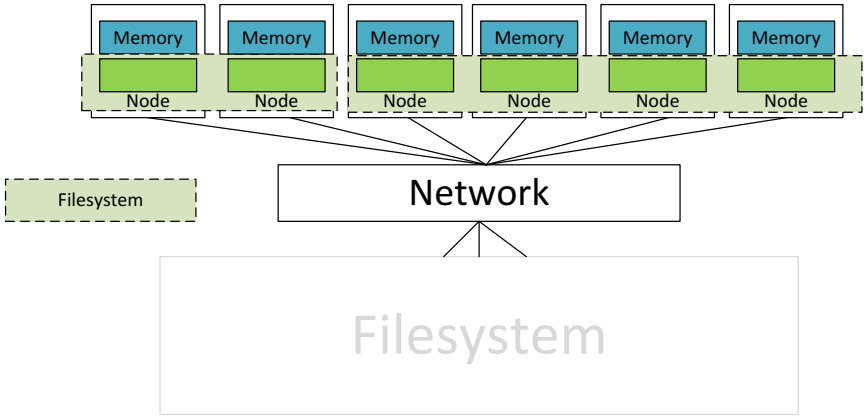
- Without changing applications
  - Filesystem buffer




- 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

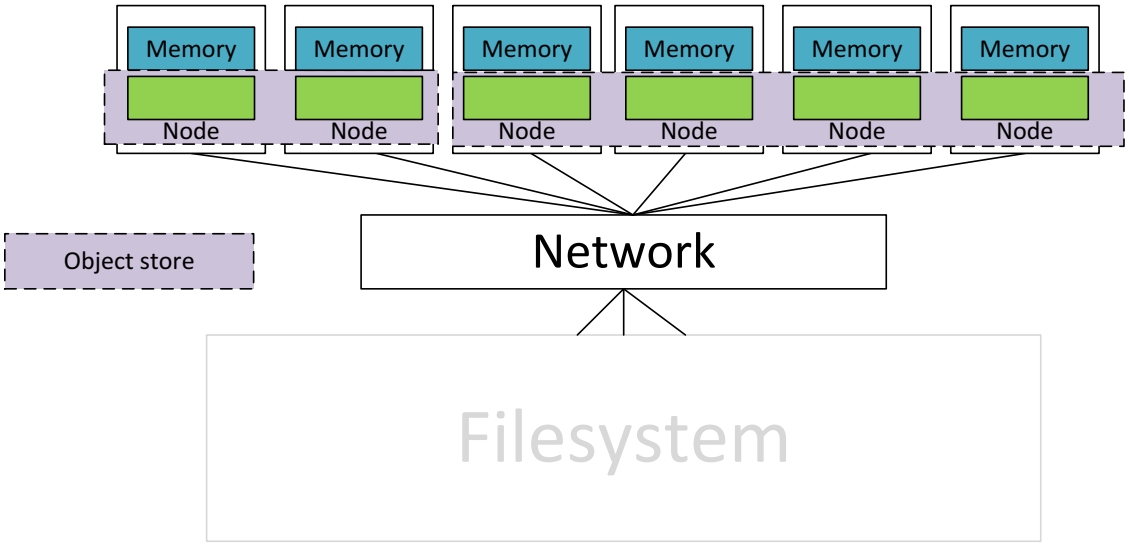
- Without changing applications
  - Global filesystem




  - 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



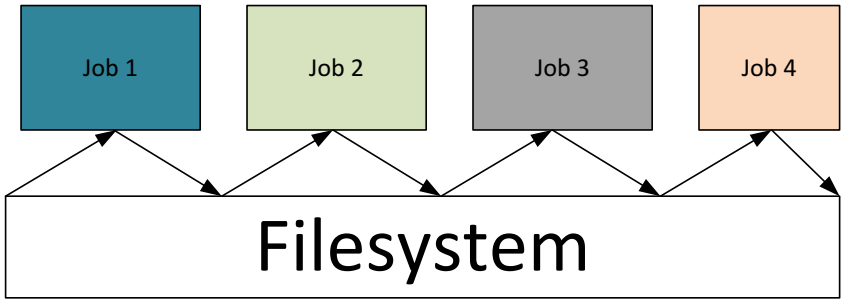

- Needs same functionality as global filesystem
- Removes need for POSIX, or POSIX-like functionality

# Using distributed storage


- Without changing applications
  - Automatic check-pointing
    - Resiliency
    - Local check-pointing without hitting the filesystem
  - Pause and restart
    - Just-in-time scheduling/high priority jobs
    - Waiting for something else to happen…

# Using distributed storage

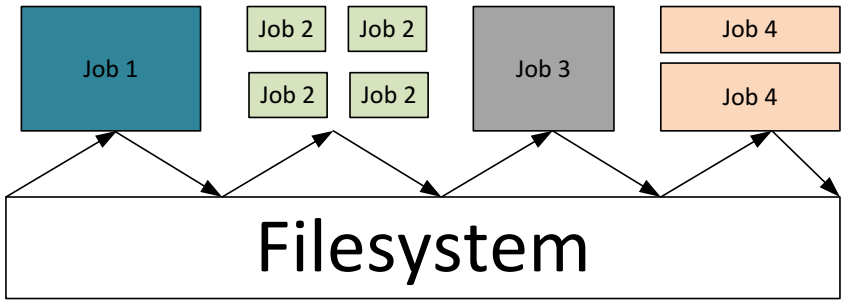
- New usage models
  - Resident data sets
    - Sharing preloaded data across a range of jobs
    - Data analytic workflows
    - How to control access/authorisation/security/etc….?
  - Workflows
    - Producer-consumer model




    - 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 NVRAM
  - Scheduling across these resources
  - Enabling sharing of nodes
  - Not impacting on node compute performance
- 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?

# *Questions?*