Understanding and Improving I/O Performance of HPC systems

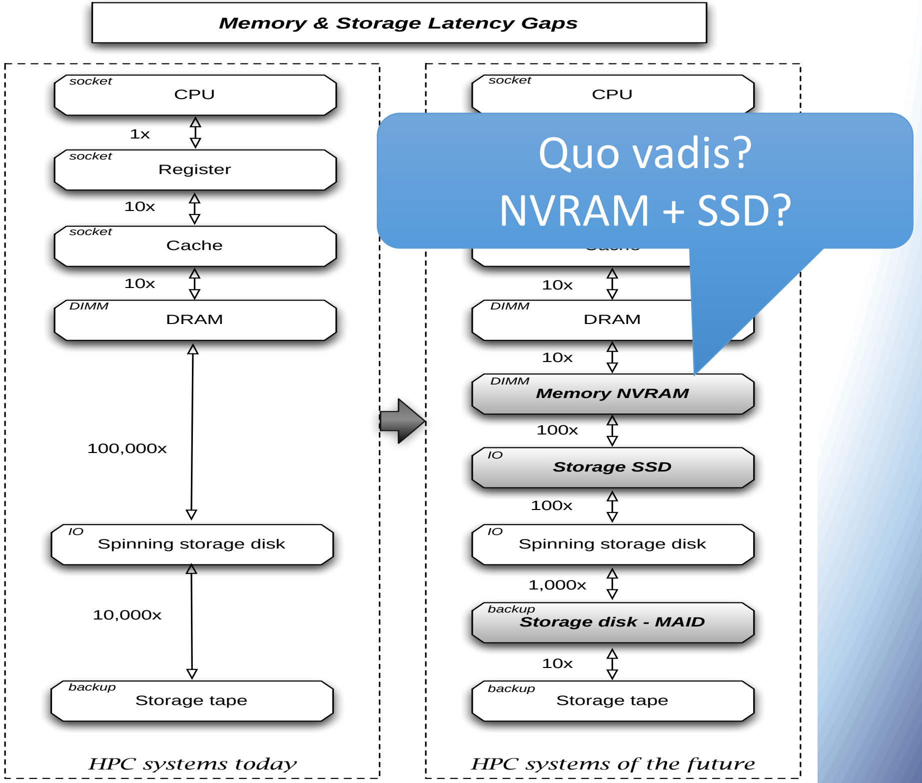
Muhammad Sarim Zafar (sarim.zafar@tu-dresden.de)

ZIH, Technische Universität Dresden

ISC17, Frankfurt, June 18th 2017

New Members in Memory Hierarchy

- New memory technology
- Changes the memory hierarchy we have
- Impact on applications e.g. simulations?
- I/O performance is one of the critical components for scaling applications

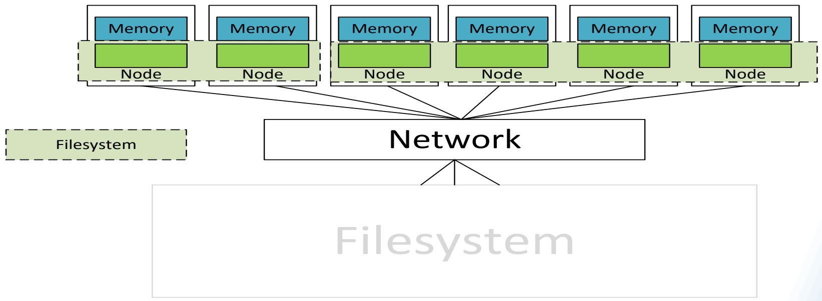


Three Usage Models

- The "memory" usage model
  - Extension of the main memory
  - Data is *volatile* like normal main memory
- The "storage" usage model
  - Classic *persistent* block device
  - Like a very fast SSD
- The "application direct" usage model
  - Maps *persistent* storage into address space
  - Direct CPU load/store instructions

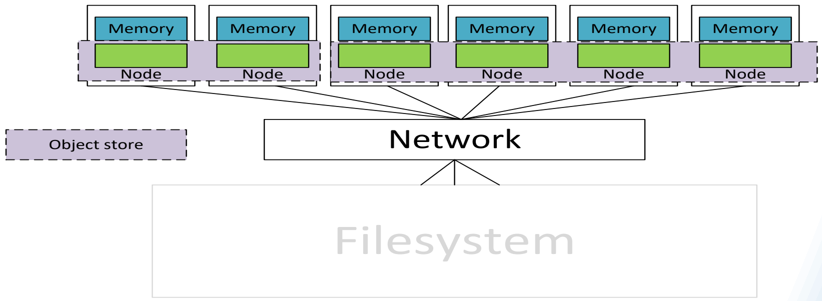
Using Distributed Storage

- Distributed global file system
  - No changes to apps
  - Support for NVRAM
- Required functionality
  - Preload and post load filesystems
  - Create and share files amongst the jobs
  - Works across nodes
  - Support multiple filesystems across cluster
- I/O Performance
  - Sum of memory layers available



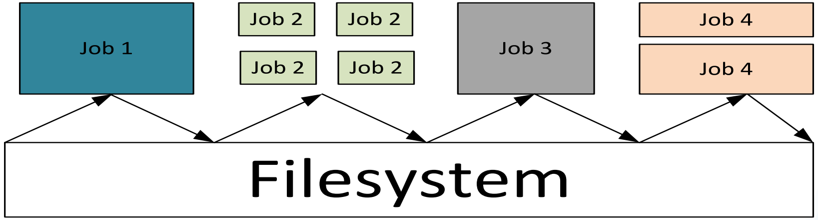
Using an Object Store

- Needs changes at the application layer
  - Needs same functionality as global filesystem
- I/O Performance
  - Mapping to objects
  - Different type of abstraction (APIs etc.)
  - Different kind of Instrumentation



On the larger perspective

- Towards Workflows
  - Workflows
    - Producer-consumer model
  - Resident data sets
    - Sharing preloaded data across a range of jobs
    - Data analytic workflows
      - E.g. Total IO operations performed by the workflow
      - E.g. Number of IO operations performed on DRAM and NVRAM
  - I/O Performance
    - Data merging/integration between the jobs
    - Remove file system from intermediate stages



In Context of: Vampir & Score-P

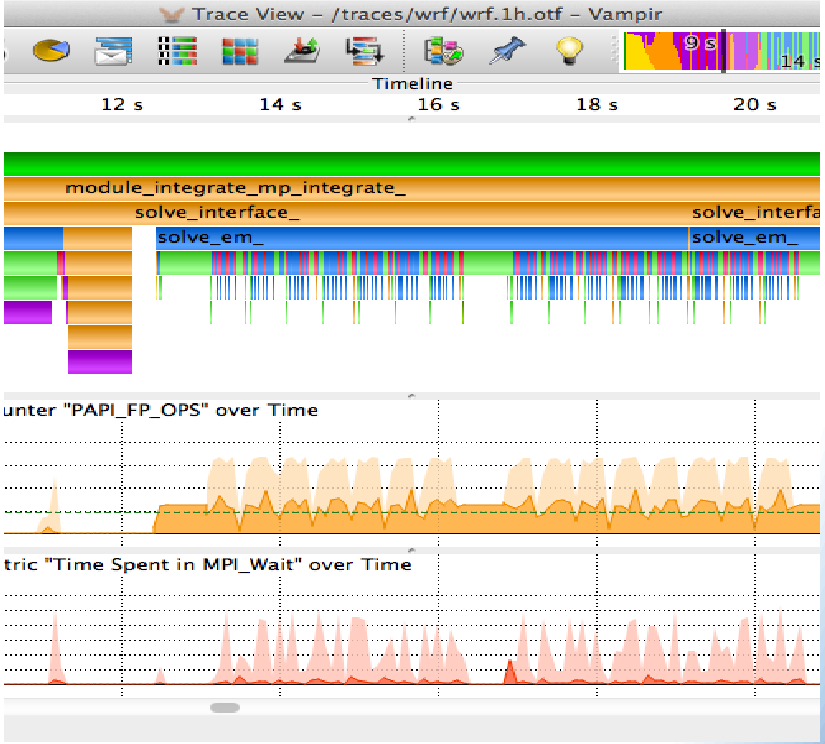
Score-P

Scalable performance measurement infrastructure for parallel codes

VAMPIR

New Tool Key Objectives

- Analysis tools need to
  - Reveal performance interdependencies in I/O and memory hierarchy
  - Support workflow visualization(Workload modelling)
  - Exploit NVRAM to store data for themselves



Exploiting I/O Layers

- I/O layers
  - E.g. Distributed FS
    - Client side – Easy to access
    - Server side – Complex
  - Kernel – IO : rusage etc
  - POSIX – fopen etc
  - MPI-I/O
  - HDF5
  - NetCDF
  - PNetCDF

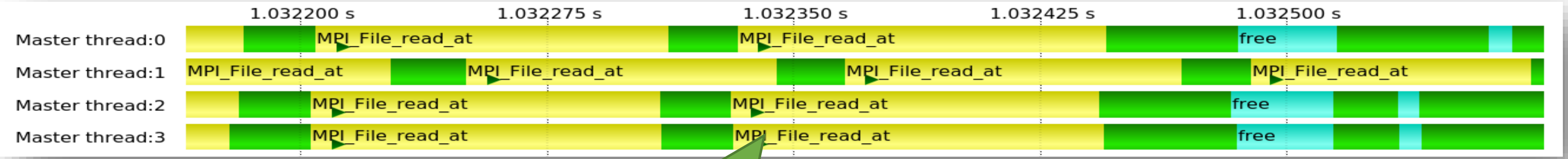
I/O performance data

- Event data AND aggregated numbers
  - Open/Create/Close operations (meta data)
  - Data transfer operations
- Client side instrumentation
  - procfs via PAPI-Components counters, Score-P (user space)
- Server side instrumentation (FS dependent)
  - Data base (custom made, system space)
  - Mapping to nodes/applications

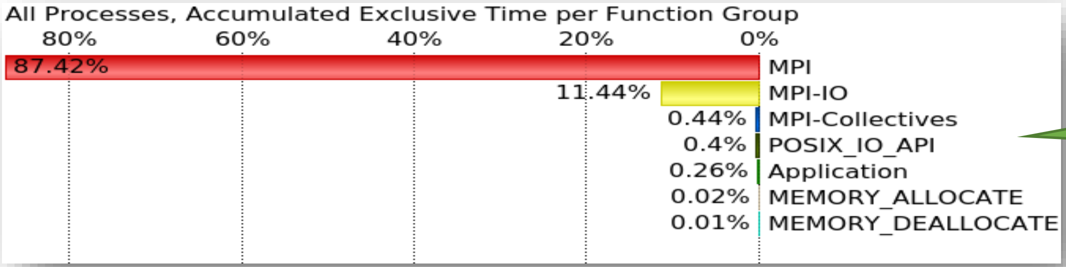
In the presence of NVMs…

- Information
  - Memory size (requested, usable)
  - High Water Mark metric
  - Size and number of elements in memory
- Individual NVRAM load/stores
  - Remain out of scope (e.g. memory mapped files)
- NVRAM health status
  - Not measurable at high frequencies

I/O Operations over Time

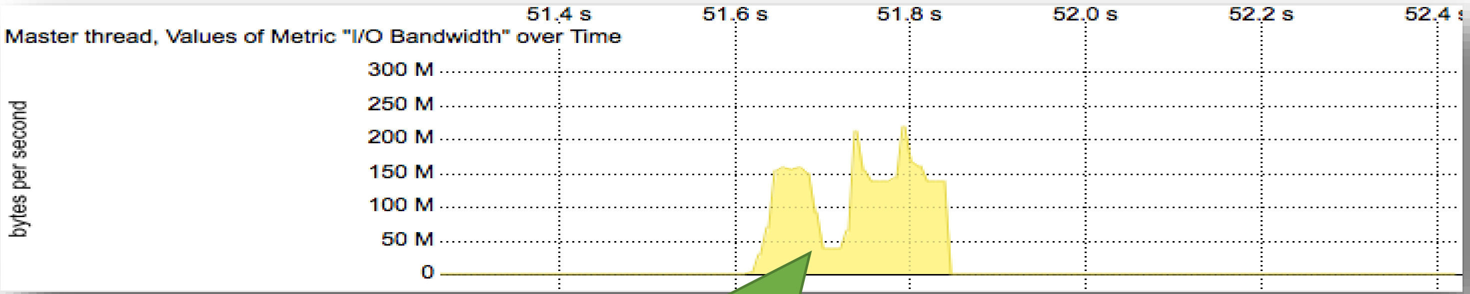


Individual I/O Operation



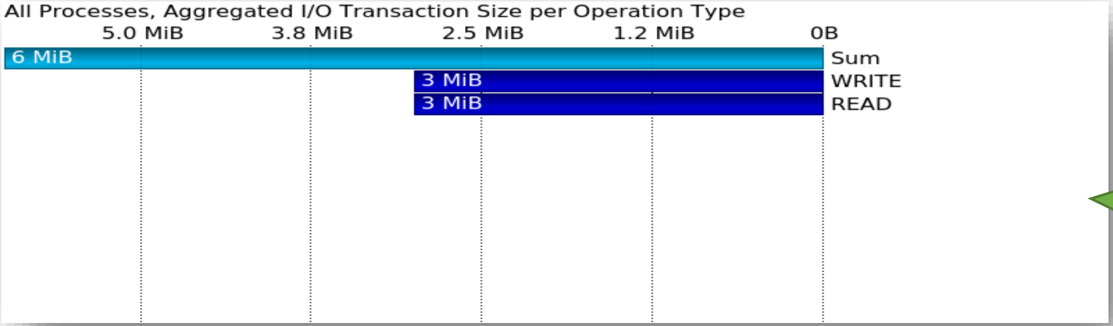
I/O Runtime Contribution

I/O Data Rate over Time



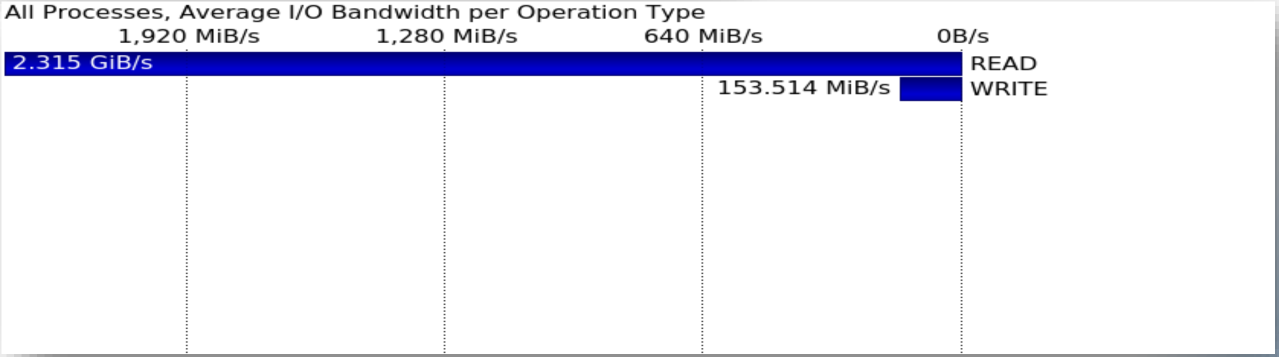
I/O Data Rate of single thread

I/O Summaries with Totals

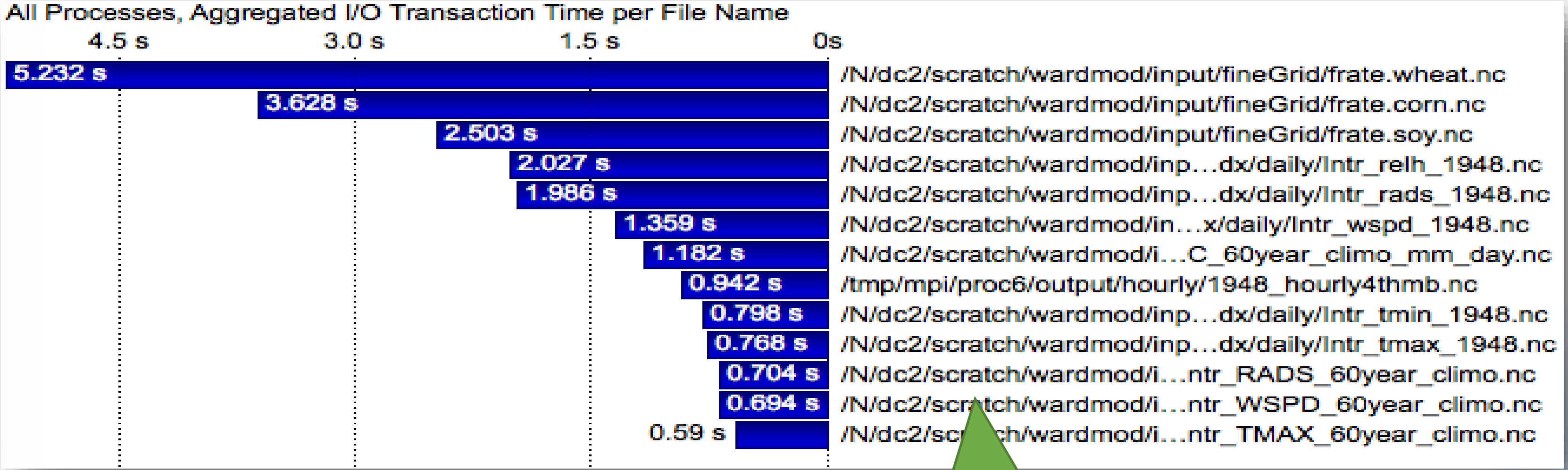


Other Metrics:

- IOPS
- I/O Time
- I/O Size
- I/O Bandwidth

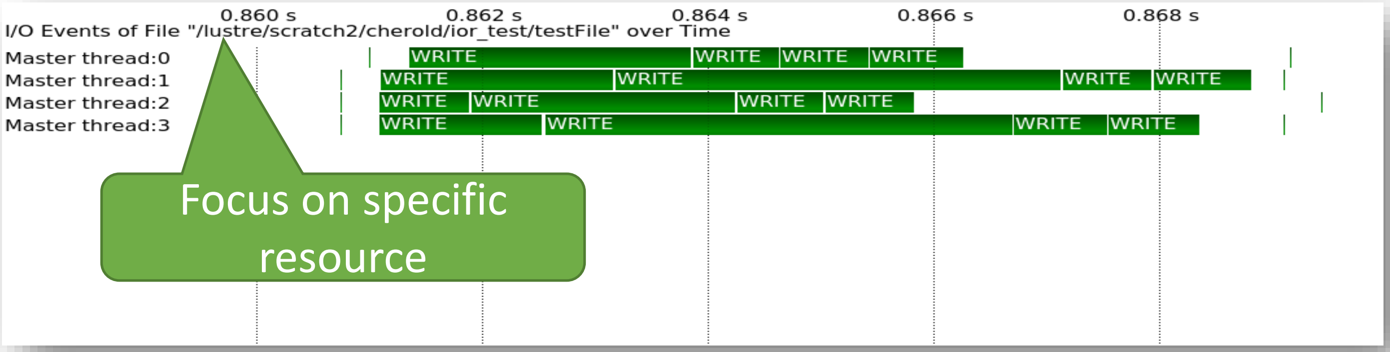


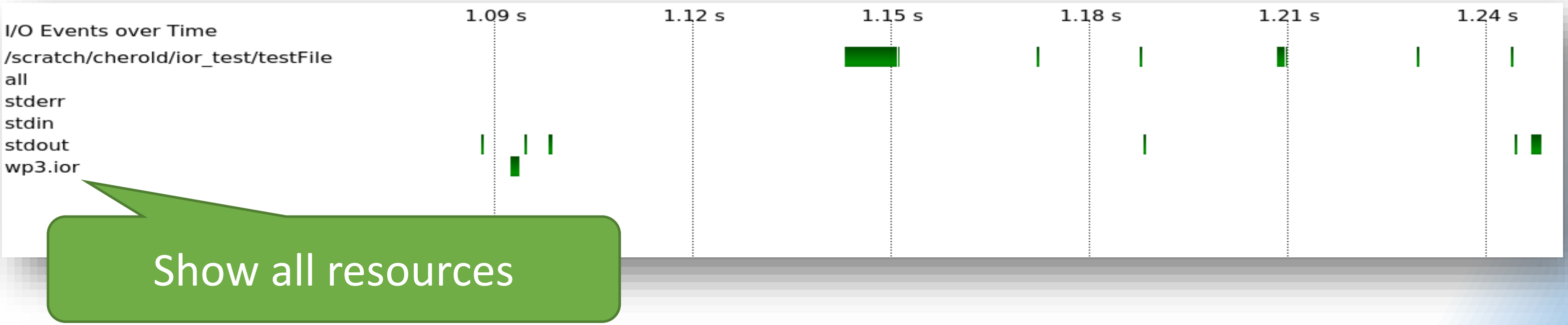
I/O Summaries per File



Aggregated data for specific resource

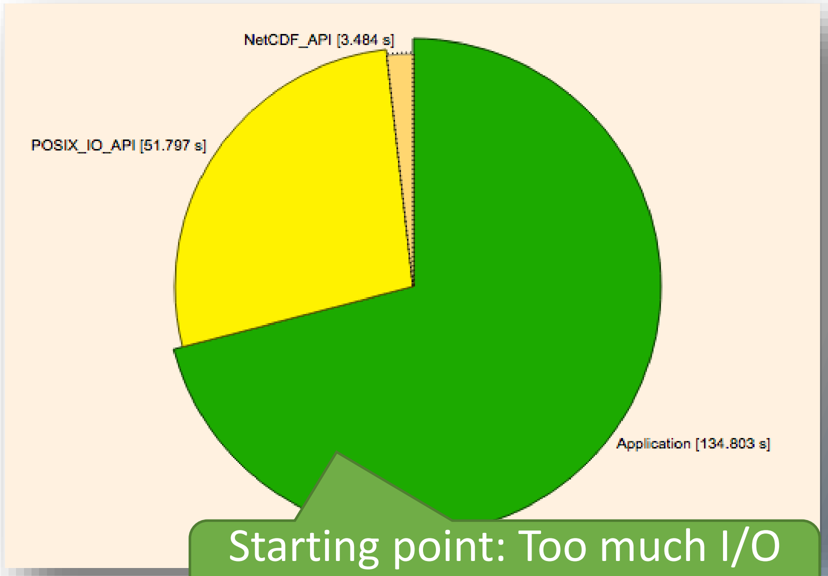
I/O Operations per File





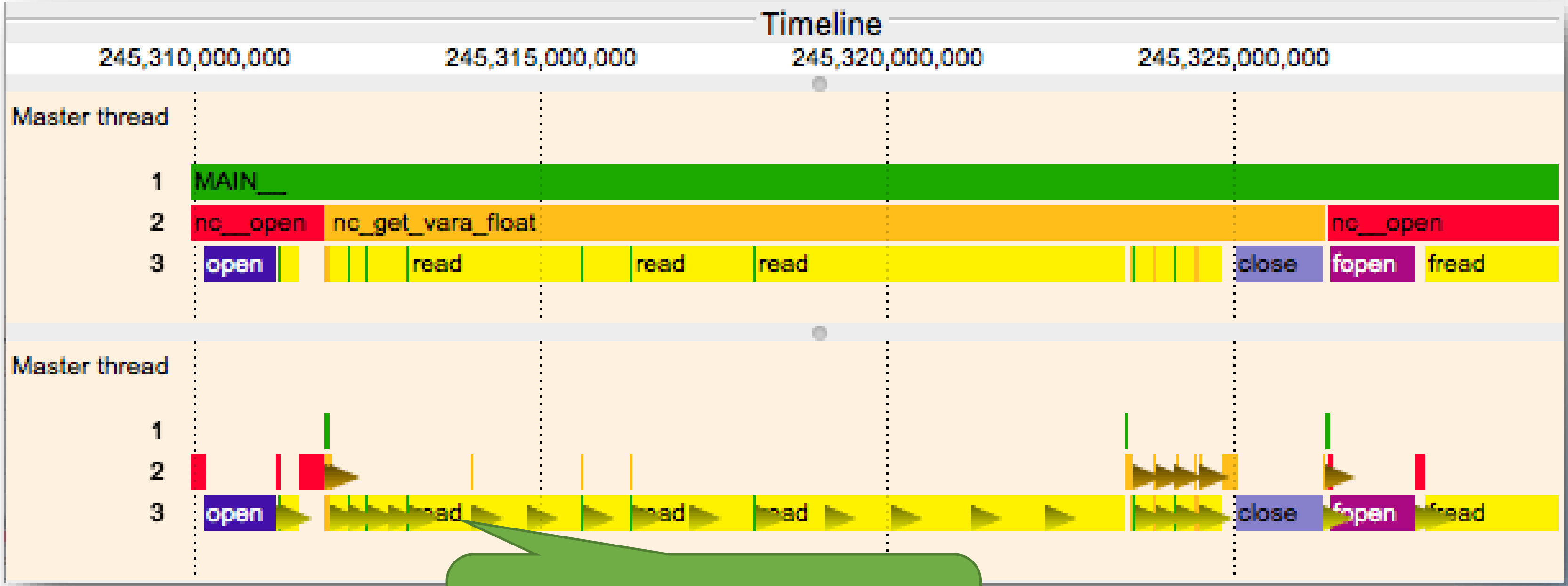
Example

- Bringing the Lustre system I/O down
  - with a single (serial) application
- Higher I/O demand than IOR benchmark
- Why?



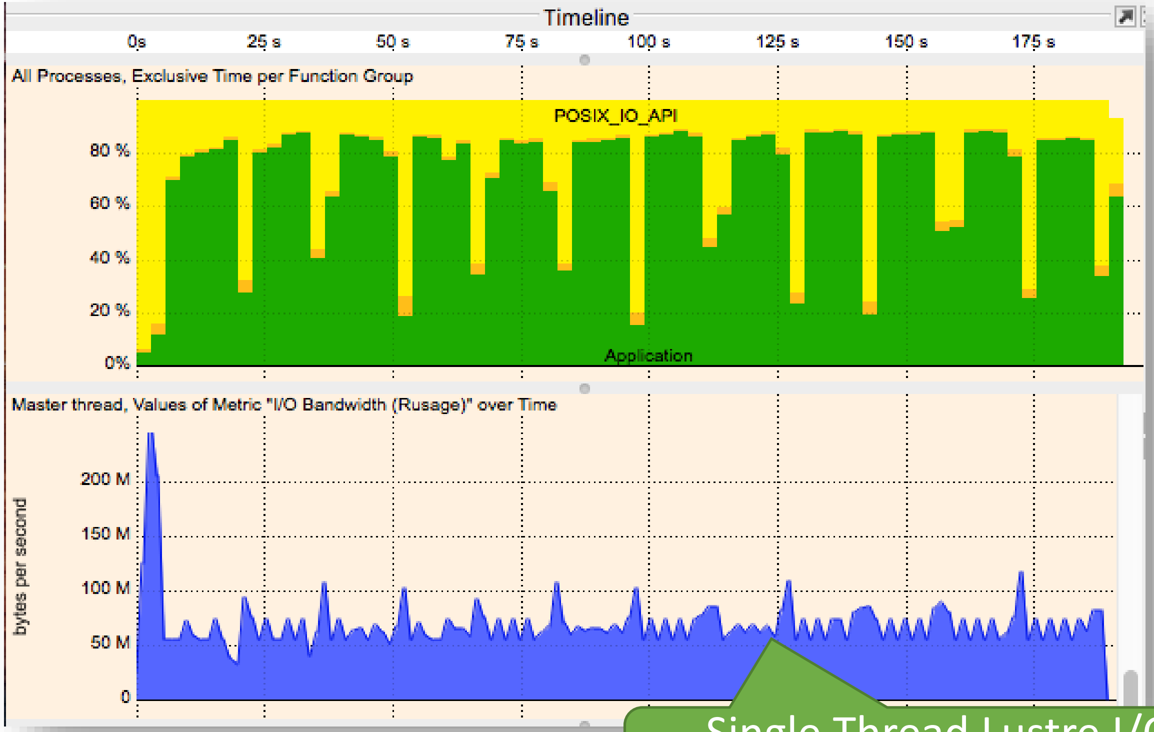
Starting point: Too much I/O (Bandwidth) on Lustre FS!!!

Details Make a Difference



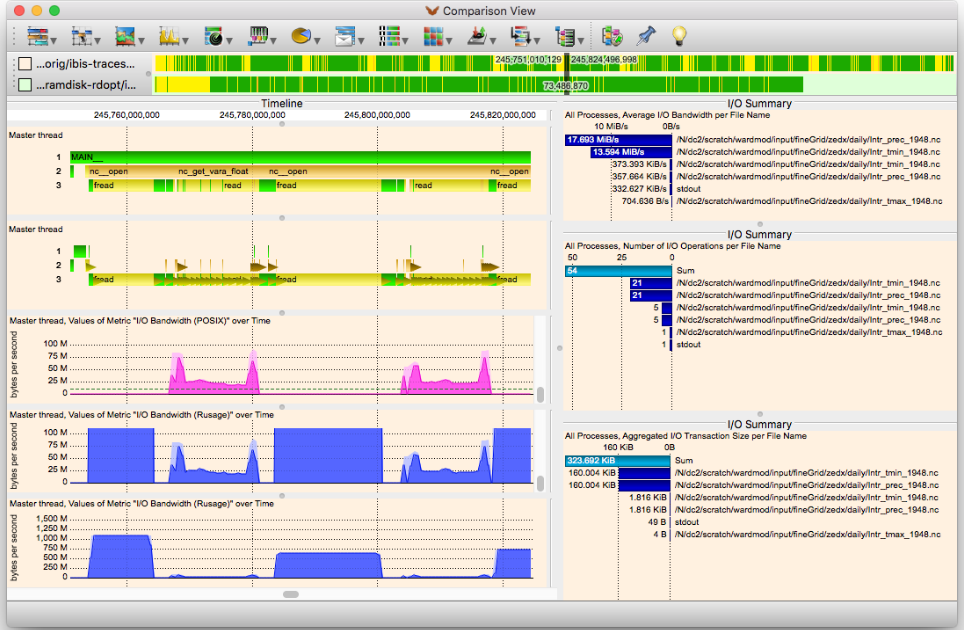
...15! POSIX read operations

Coarse Grained Time Series Reveal Some Clue, but...

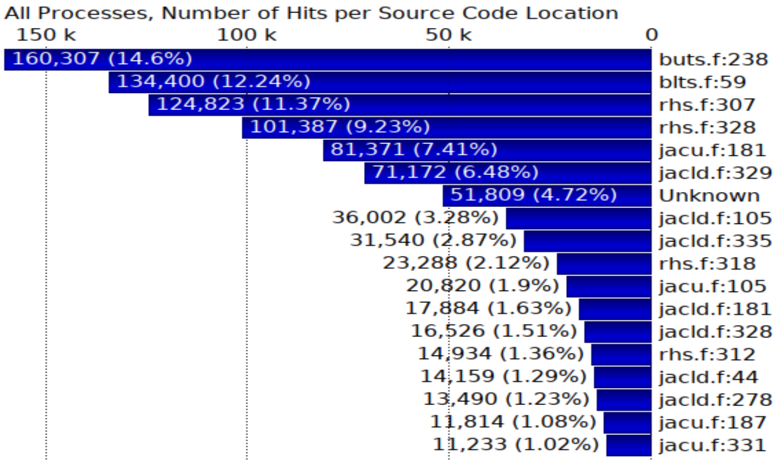


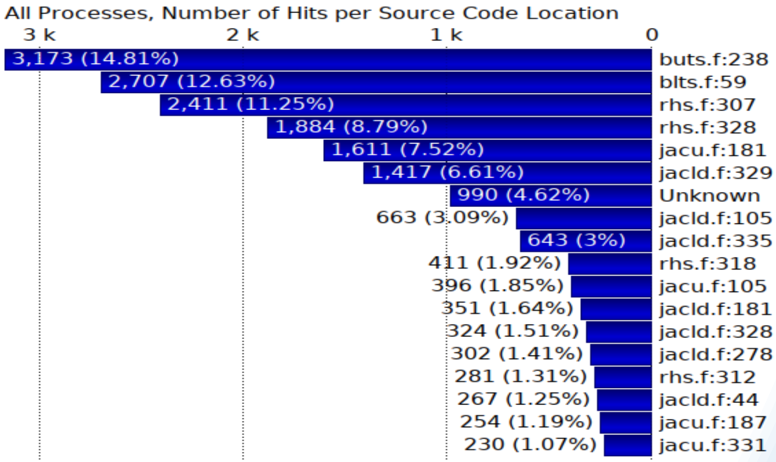
Single Thread Lustre I/O Bandwidth over time

Impact of Lustre Prefetching

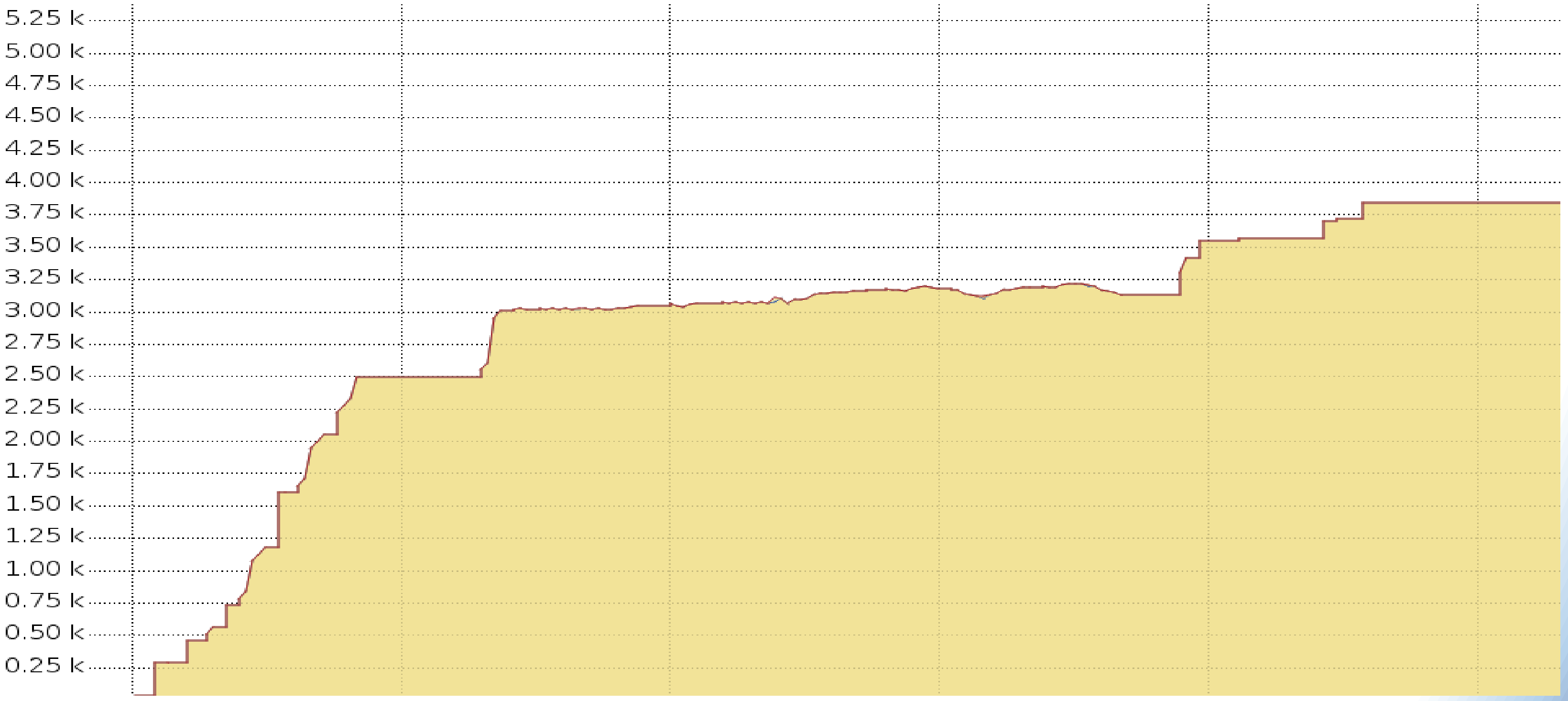


Future Stats



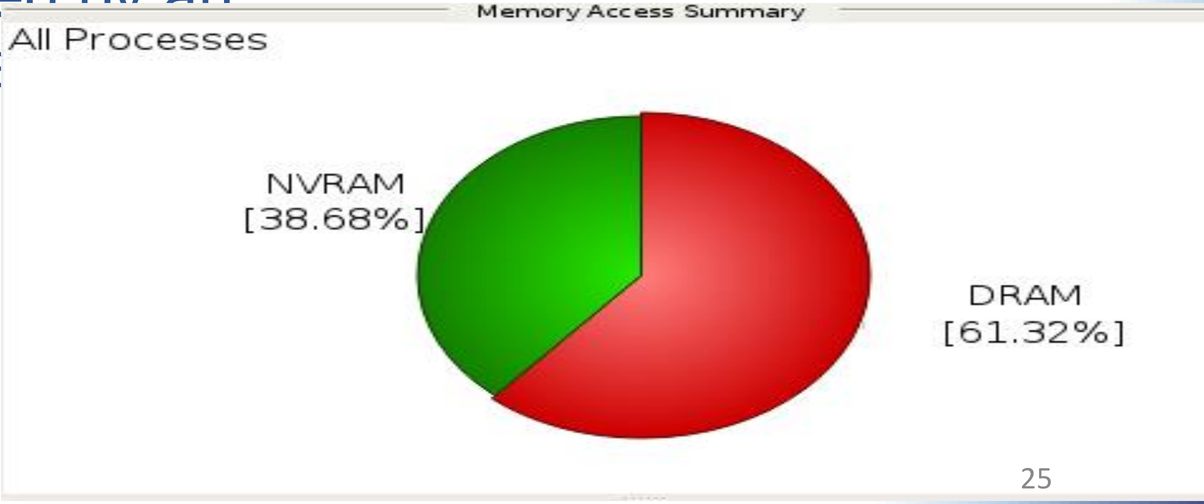


NVRAM allocation over time

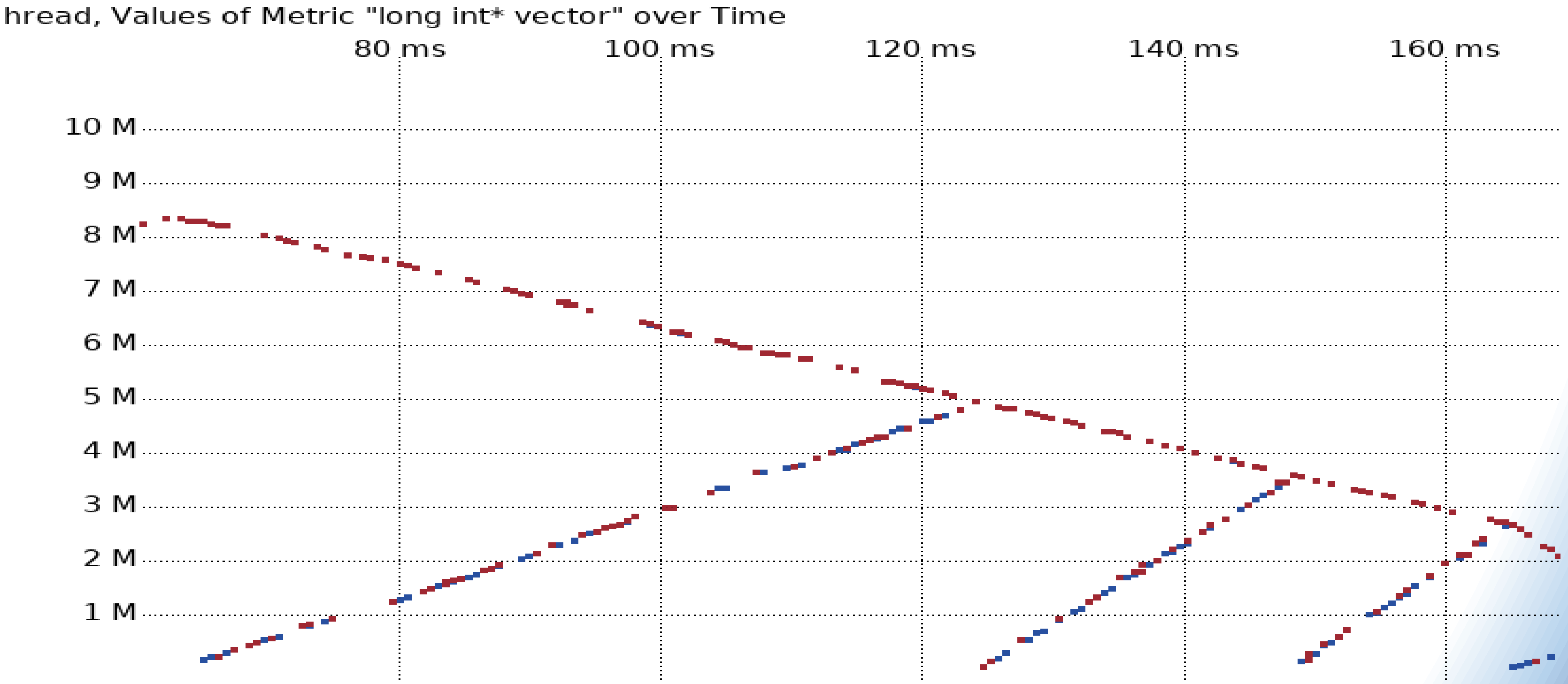


Visualization of memory access statistics

- Currently, Vampir has many different views presenting counter metrics
- Future: Absolute number of memory accesses performed by an application for diffe types of memory



Future Stats



Summary

- NEXTGenIO developing a full hardware ***and*** software solution
  - Solution includes Lustre
- Performance focus
  - Consider complete I/O stack
  - Incorporate new I/O paradigms
  - Study implications of NVRAM
- Reduce I/O costs
- New usage models for HPC and HPDA

Hands-on Slides

Objective

- Familiarize with the usage oge Score-P/Vampir tool suite
- Execises are based on a small portable IO benchmarking application.

- First stage: Instrument Benchio

```
# Connect
% ssh -X userid@abc.edu
```

```
# make separate dir
% mkdir scorep-vampir

# get the sources
% wget
http://www.archer.ac.uk/training/course-material/2016/09/160929_AdvMPI_EPCC/benchio.tar

# untar
```

Preparation

```
# Load modules
% module load cray-netcdf-hdf5parallel/4.4.0
% module load cray-hdf5-parallel/1.8.16
% module use /work/d131/d131/shared/vampir/modules
% module load vampir/io-dev
% module load scorep/cray
```

Building the benchmark

```
% vi Makefile
MF=      Makefile
# ARCHER
FC= ftn
FFLAGS=
LFLAGS=
# Ubuntu
#FC=    mpif90
#FFLAGS=-I/usr/include
#LFLAGS=-L/usr/lib -lnetcdff -L/usr/lib/x86_64-linux-gnu/ -lhdf5_fortran
EXE=    benchio
SRC= \
        benchio.f90 \
        mpiio.f90 \
        netcdf.f90 \
        hdf5.f90 \
        benchclock.f90
```

Clean and make, without instrumentation

Modify specification of compiler / linker

```
% vi
MF=       M    ile
# ARCHER
FC= scorep --mpp=mpi ftn
FFLAGS=
LFLAGS=
# Ubuntu
#FC=     mpif90
#FFLAGS=-I/usr/include
#LFLAGS=-L/usr/lib -lnetcdff -L/usr/lib/x86_64-linux-gnu/ -lhdf5_fortran
EXE=     benchio
SRC= \
        benchio.f90 \
        mpiio.f90 \
        netcdf.f90 \
        hdf5.f90 \
        benchclock.f90
```

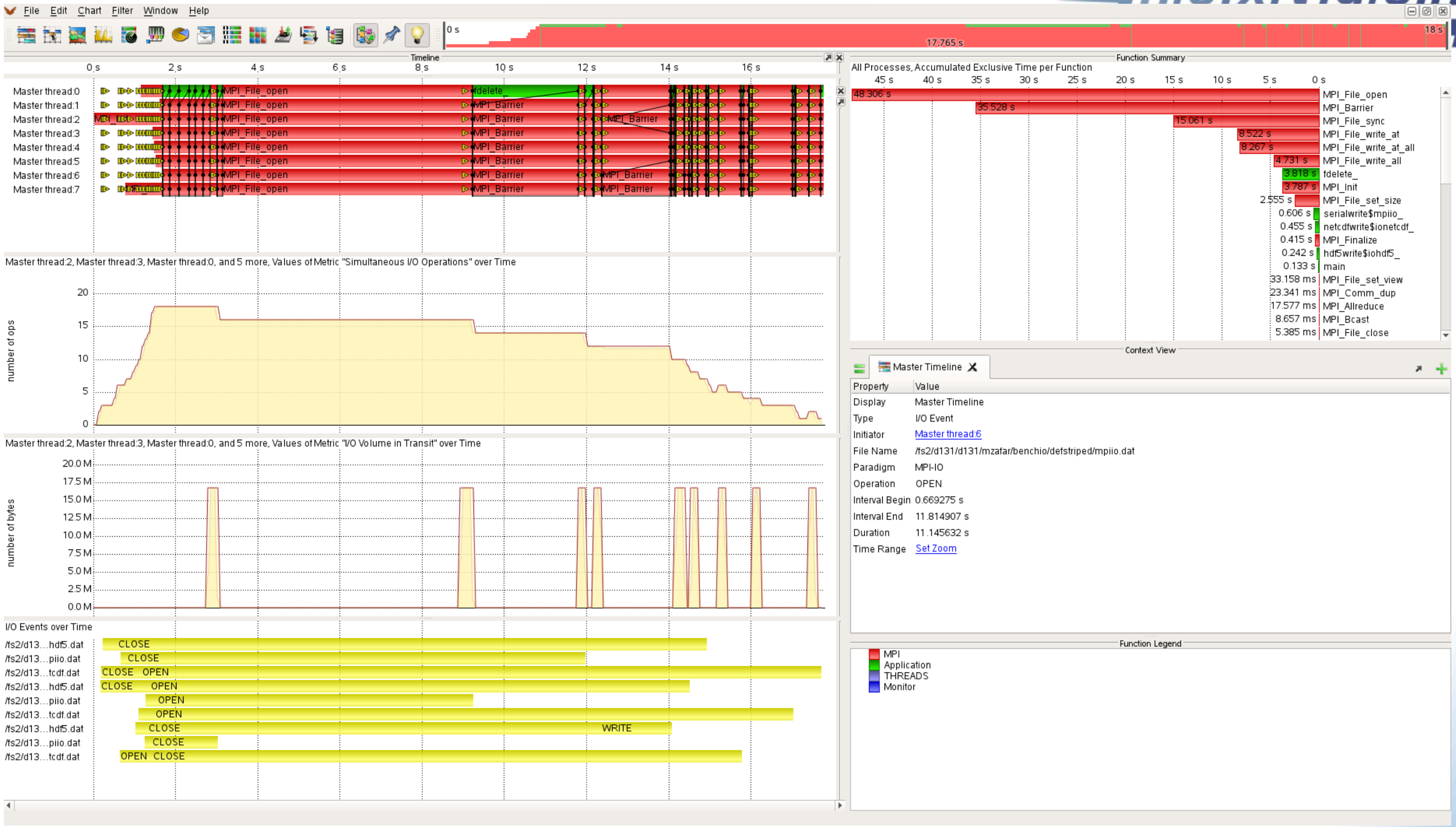
Specify the MPI paradigm for CRAY

Once finised, clean and make

Job submission

```
# By default, only profiling is enabled. Enable the tracing in
# job script benchio.pbs
#!/bin/bash --login

#PBS -A d131
#PBS -N benchio
#PBS -l select=3
#PBS -l walltime=00:05:00
#PBS -j oe

# change directory to where the job was submitted from

export SCOREP_ENABLE_TRACING=true

…
```

```
# Visualize Trace
% cd scorep-<traceid>
% vampir traces.otf2
```


File Edit Chart Filter Window Help
Timeline
Master thread:0
Master thread:1
Master thread:2
Master thread:3
Master thread:4
Master thread:5
Master thread:6
Master thread:7
MPI_File_open
MPI_Barrier
fdelete_
Master thread:2, Master thread:3, Master thread:0, and 5 more, Values of Metric "Simultaneous I/O Operations" over Time
number of ops
Master thread:2, Master thread:3, Master thread:0, and 5 more, Values of Metric "I/O Volume in Transit" over Time
number of bytes
I/O Events over Time
/fs2/d13...hdf5.dat
/fs2/d13...piio.dat
/fs2/d13...tcdf.dat
CLOSE
OPEN
WRITE
Function Summary
All Processes, Accumulated Exclusive Time per Function
48.306 s MPI_File_open
35.528 s MPI_Barrier
15.061 s MPI_File_sync
8.522 s MPI_File_write_at
8.267 s MPI_File_write_at_all
4.731 s MPI_File_write_all
3.818 s fdelete_
3.787 s MPI_Init
2.555 s MPI_File_set_size
0.606 s serialwrite$mpiio_
0.455 s netcdfwrite$ionetcdf_
0.415 s MPI_Finalize
0.242 s hdf5write$iohdf5_
0.133 s main
33.158 ms MPI_File_set_view
23.341 ms MPI_Comm_dup
17.577 ms MPI_Allreduce
8.657 ms MPI_Bcast
5.385 ms MPI_File_close
Context View
Master Timeline
Property Value
Display Master Timeline
Type I/O Event
Initiator Master thread:6
File Name /fs2/d131/d131/mzafar/benchio/defstriped/mpiio.dat
Paradigm MPI-IO
Operation OPEN
Interval Begin 0.669275 s
Interval End 11.814907 s
Duration 11.145632 s
Time Range Set Zoom
Function Legend
MPI
Application
THREADS
Monitor