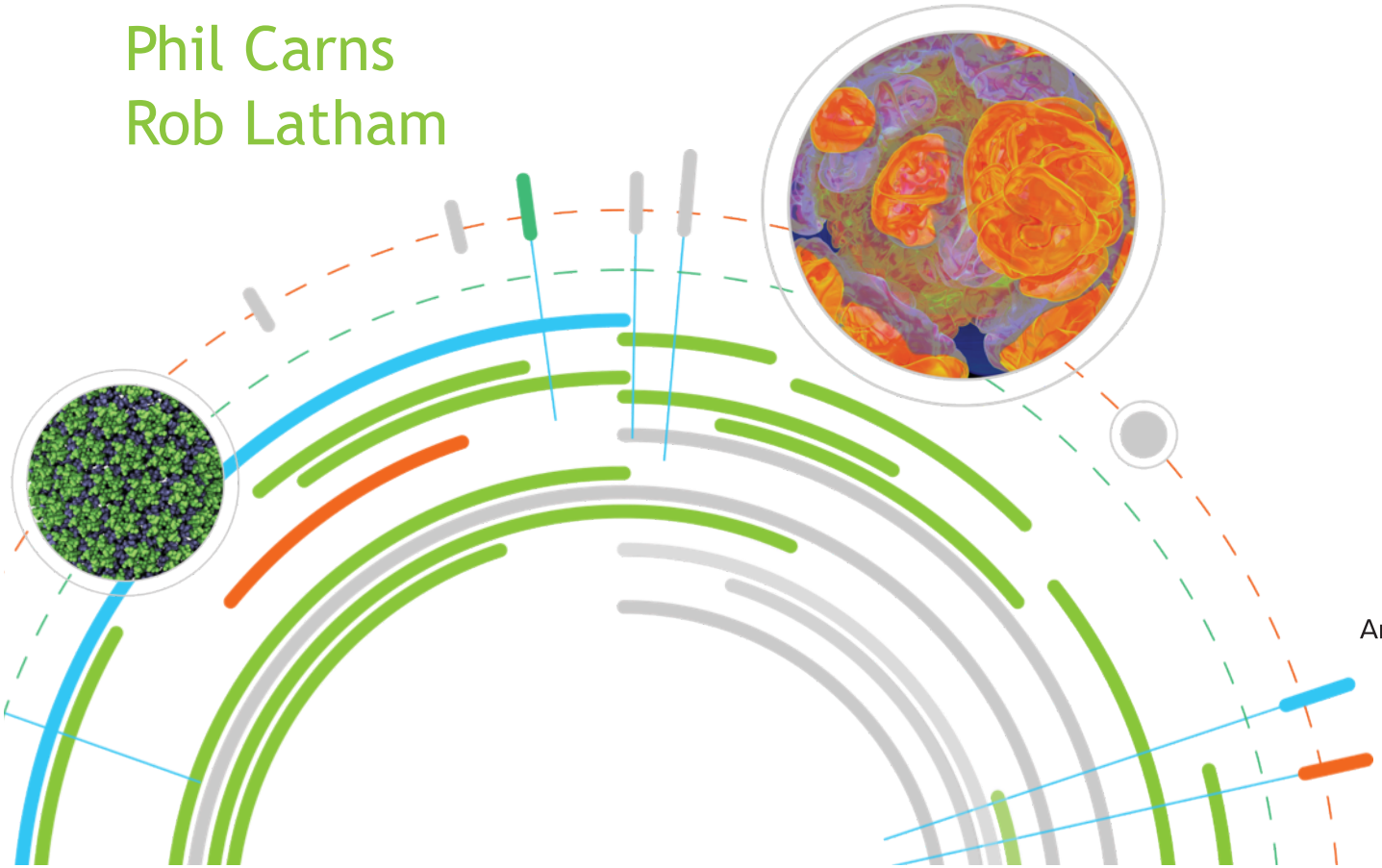
Parallel I/O on Mira

Mira Boot Camp 2015
Kevin Harms – harms@alcf.anl.gov

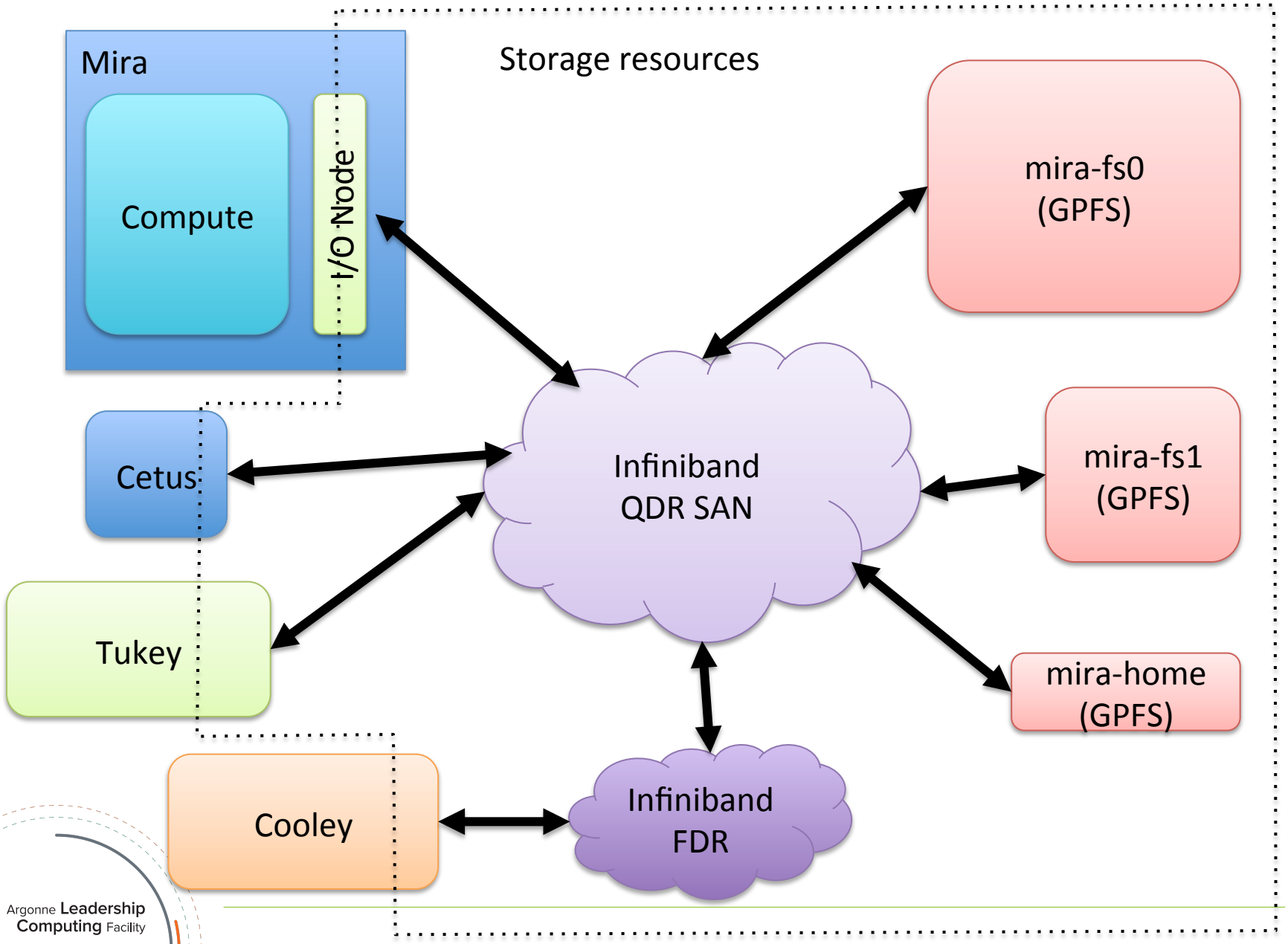
Venkat Vishwanath
Phil Carns
Rob Latham



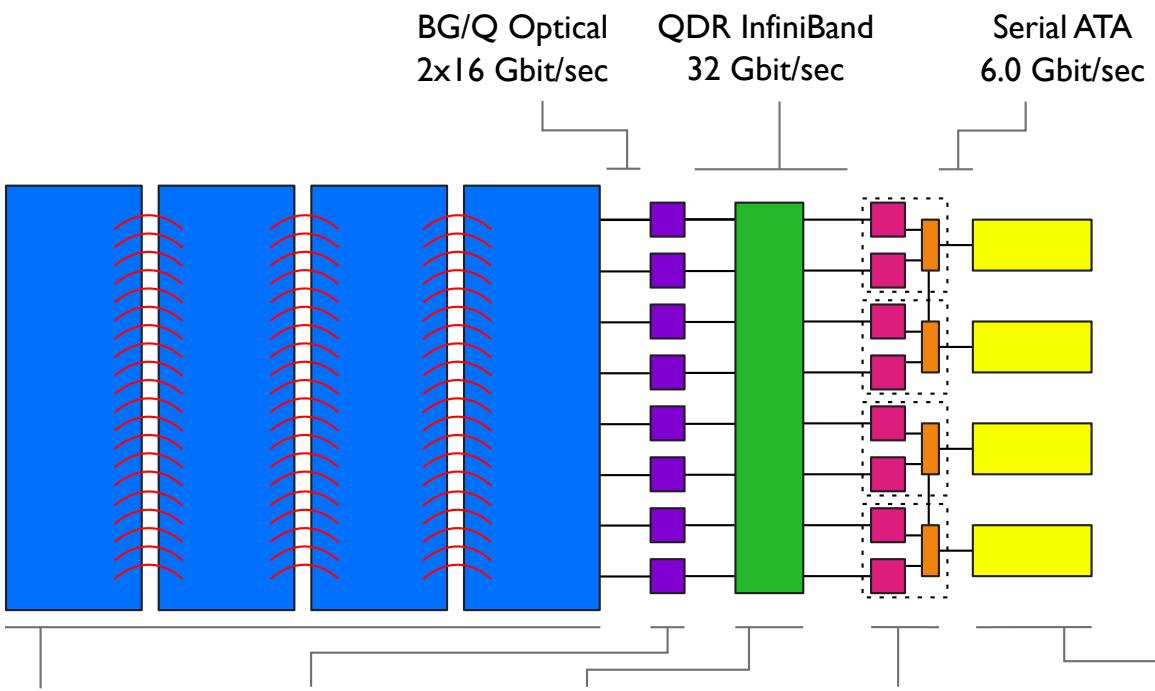
Argonne **Leadership Computing** Facility

Argonne NATIONAL LABORATORY

Mira I/O Infrastructure Overview



Mira I/O Infrastructure



Compute nodes run applications and some I/O middleware.

768K cores with 1 Gbyte of RAM each

Gateway nodes run parallel file system client software and forward I/O operations from HPC clients.

384 16-core PowerPC A2 nodes with 16 Gbytes of RAM each

Commodity network primarily carries storage traffic.

QDR Infiniband Federated Switch

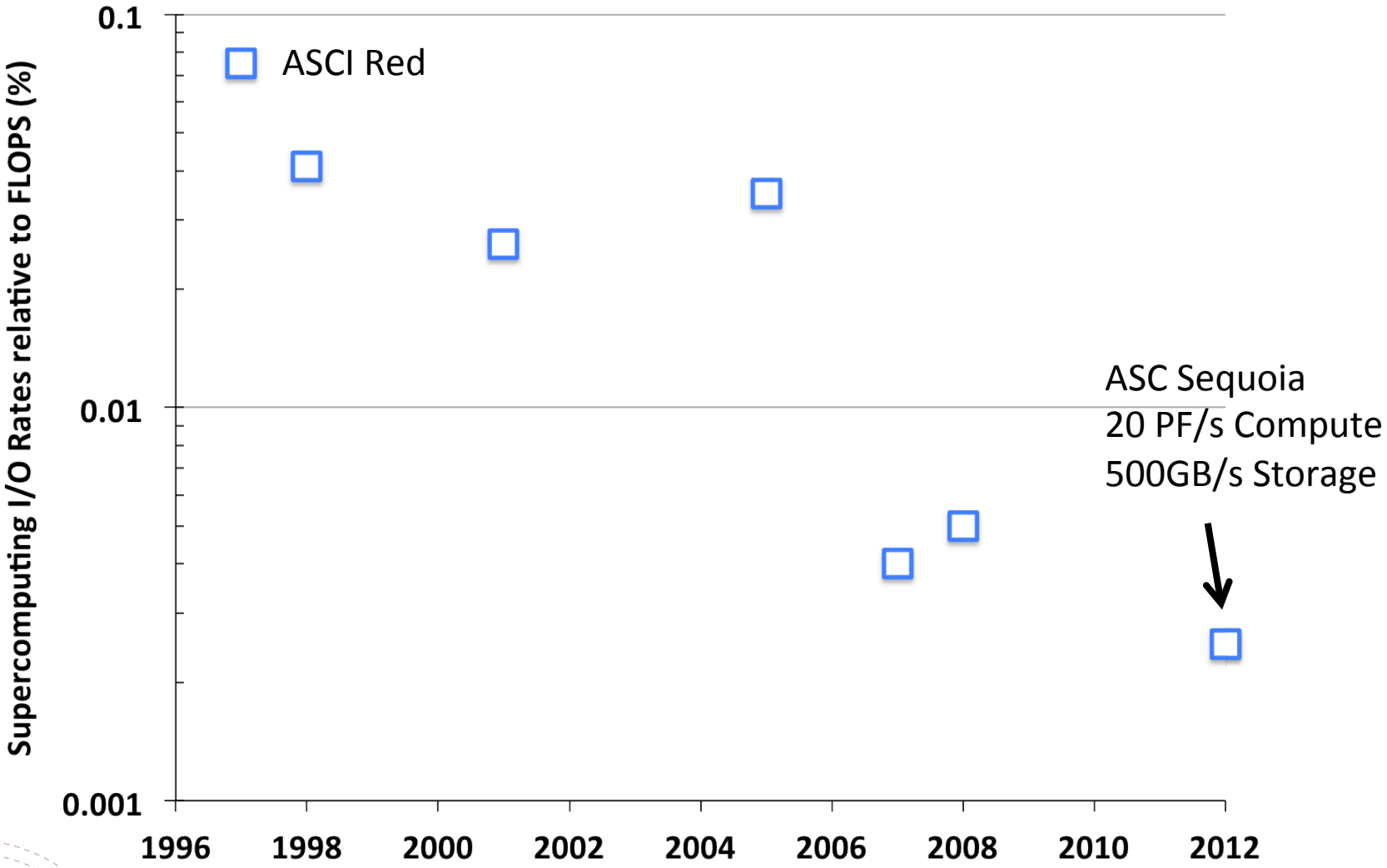
Storage nodes run parallel file system software and manage incoming FS traffic from gateway nodes.

SFA12KE hosts VM running GPFS servers

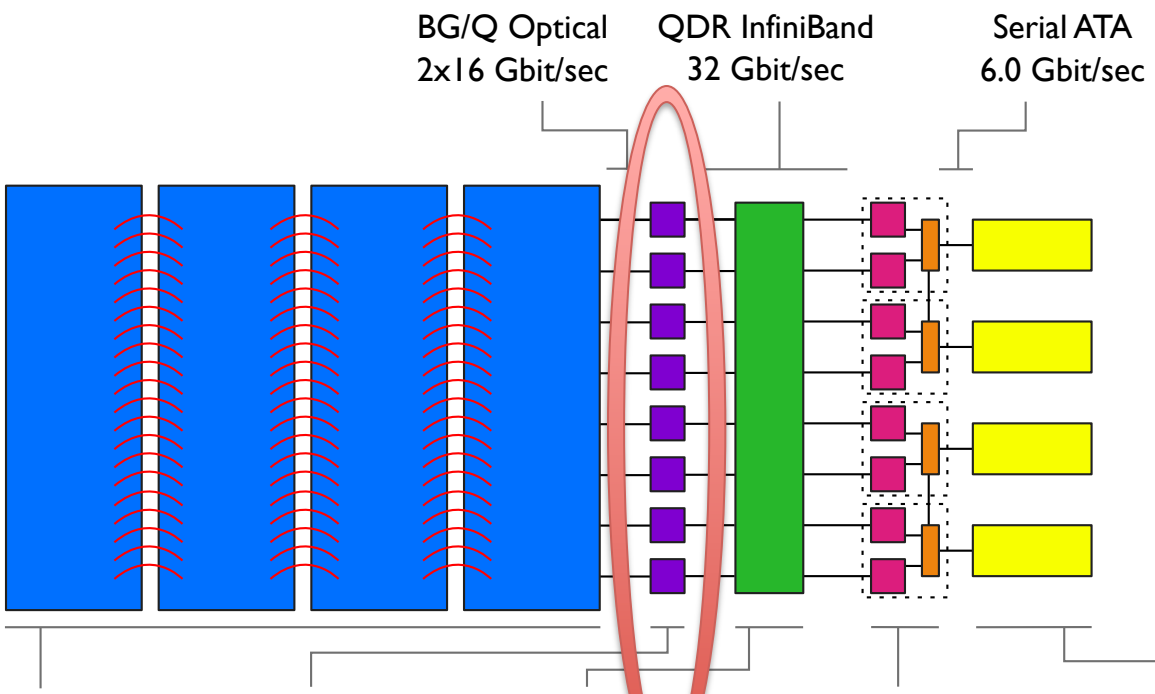
Enterprise storage controllers and large racks of disks are connected via InfiniBand.

32 DataDirect SFA12KE; 560 3 Tbyte drives + 32 200 GB SSD; 16 InfiniBand ports per pair

Storage vs Computation Trend



Mira I/O Hardware (ION)



Compute nodes
run applications and some I/O middleware.

768K cores with 1 Gbyte of RAM each

Gateway nodes
run parallel file system client software and forward I/O operations from HPC clients.

384 16-core PowerPC A2 nodes with 16 Gbytes of RAM each

Commodity network
primarily carries storage traffic.

QDR Infiniband Federated Switch

Storage nodes
run parallel file system software and manage incoming FS traffic from gateway nodes.

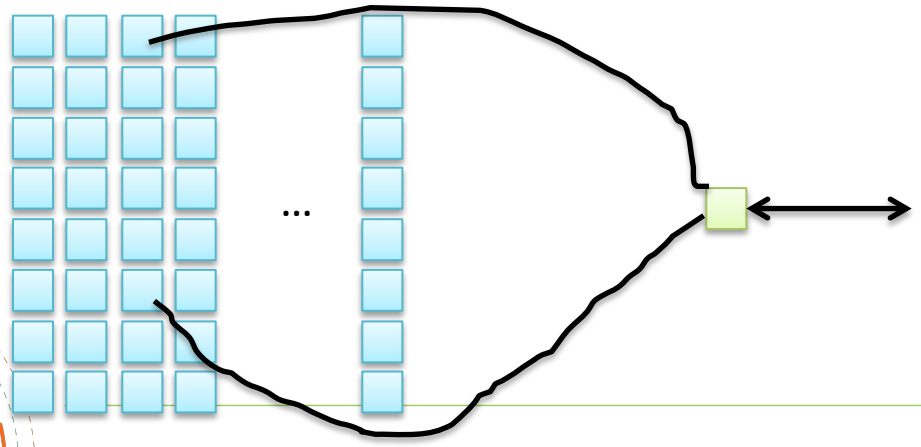
SFA12KE hosts VM running GPFS servers

Enterprise storage
controllers and large racks of disks are connected via InfiniBand.

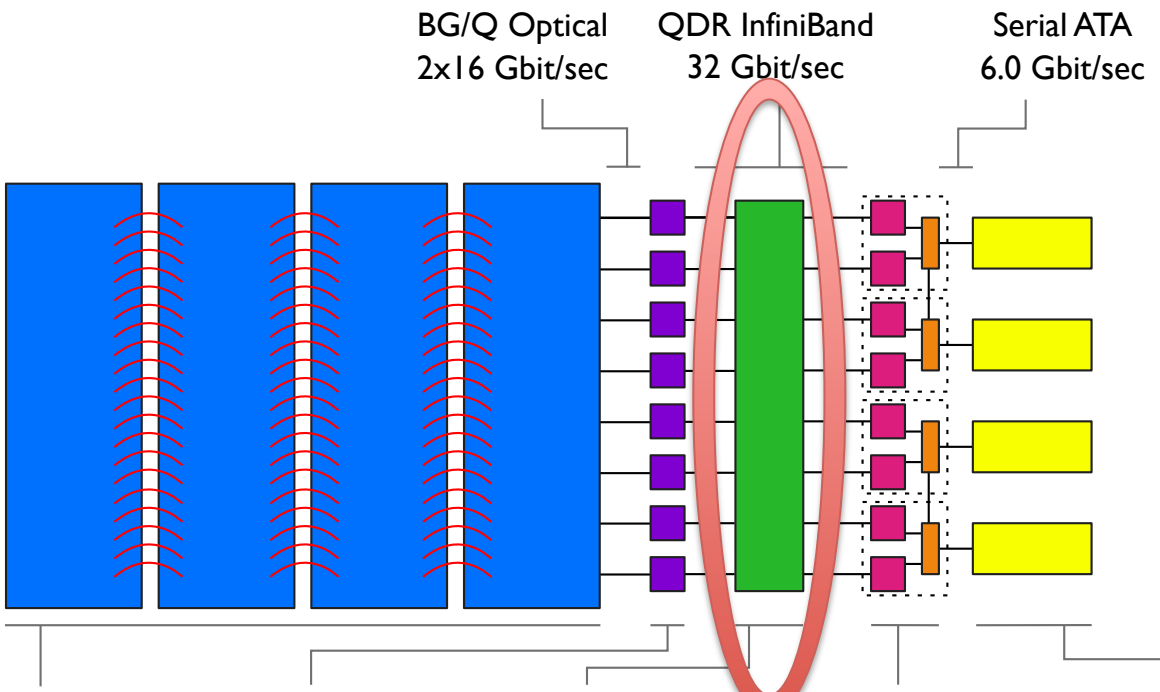
32 DataDirect SFA12KE; 560 3 Tbyte drives + 32 200 GB SSD; 16 InfiniBand ports per pair

Mira I/O Hardware (ION)

- More or less the same hardware as a compute node
- PowerPC A2 with 16 cores running at 1600 MHz
- 16 GB of DDR3
- (2) 2 GB/s bi-directional optical links which connect to BG/Q Torus
- (1) 4 GB/s bi-directional Infiniband QDR link which connects to SAN
- Runs the I/O forwarding daemon
- Runs the GPFS parallel file system client
- Dedicated resource when running on at least 512 nodes
- One ION for every 128 compute nodes
  - 49152 Compute nodes -> 384 I/O nodes



Mira SAN (Storage Area Network)



Compute nodes run applications and some I/O middleware.

768K cores with 1 Gbyte of RAM each

Gateway nodes run parallel file system client software and forward I/O operations from HPC clients.

384 16-core PowerPC A2 nodes with 16 Gbytes of RAM each

Commodity network primarily carries storage traffic.

QDR Infiniband Federated Switch

Storage nodes run parallel file system software and manage incoming FS traffic from gateway nodes.

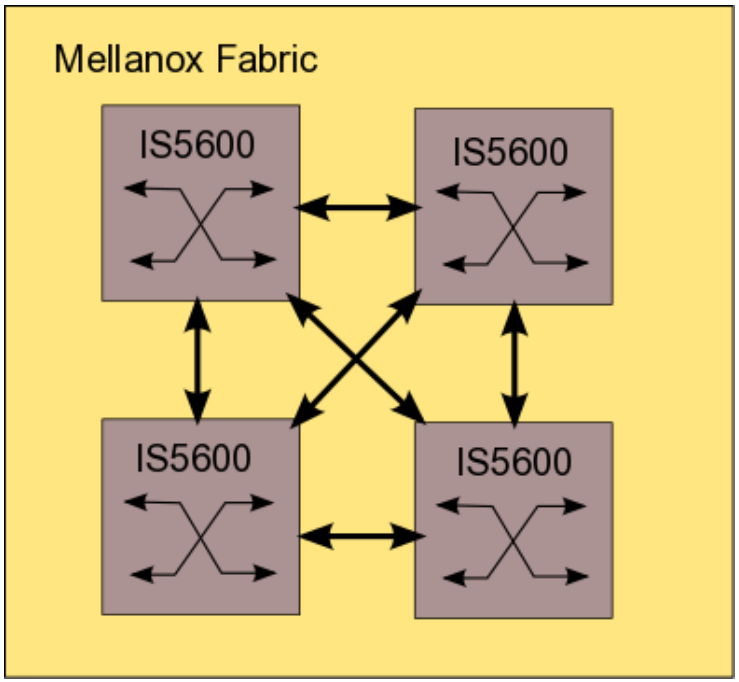
SFA12KE hosts VM running GPFS servers

Enterprise storage controllers and large racks of disks are connected via InfiniBand.

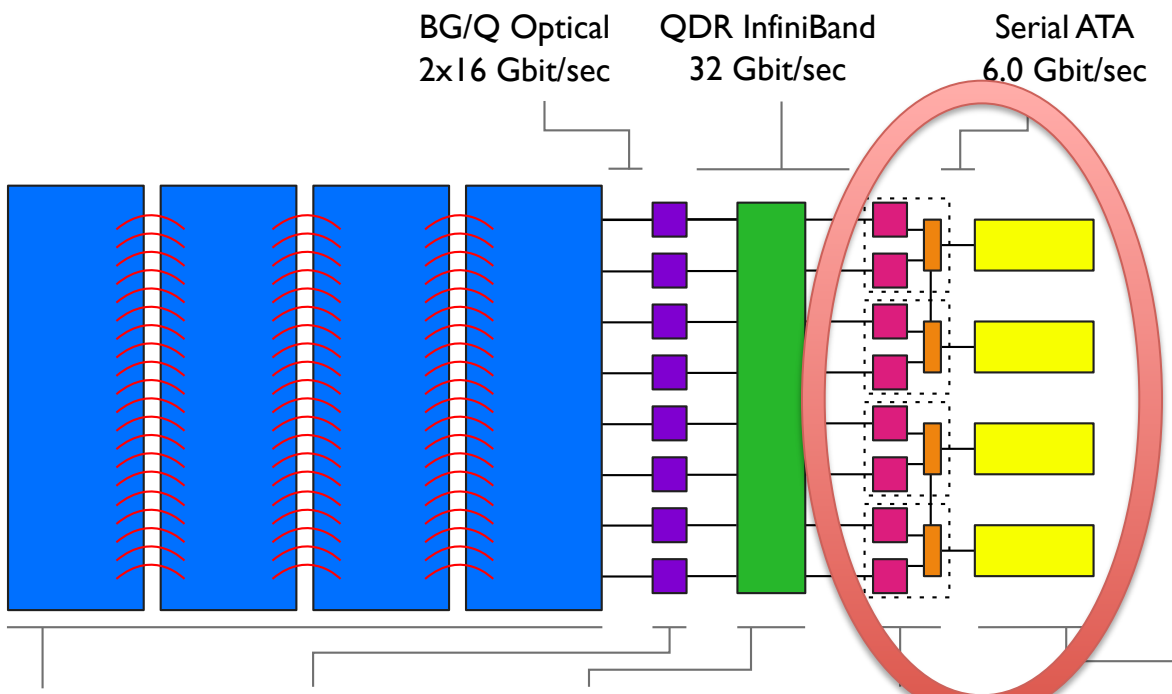
32 DataDirect SFA12KE; 560 3 Tbyte drives + 32 200 GB SSD; 16 InfiniBand ports per pair

Mira SAN (Storage Area Network)

- SAN connects Mira, Cetus, Tukey and Cooley all to Mira file systems
- Shared resource
- (4) Mellanox IS5600 Infiniband QDR switches
  - 3,024 ports



Mira SAN (Storage Area Network)



Compute nodes run applications and some I/O middleware.

768K cores with 1 Gbyte of RAM each

Gateway nodes run parallel file system client software and forward I/O operations from HPC clients.

384 16-core PowerPC A2 nodes with 16 Gbytes of RAM each

Commodity network primarily carries storage traffic.

QDR Infiniband Federated Switch

Storage nodes run parallel file system software and manage incoming FS traffic from gateway nodes.

SFA12KE hosts VM running GPFS servers

Enterprise storage controllers and large racks of disks are connected via InfiniBand.

32 DataDirect SFA12KE; 560 3 Tbyte drives + 32 200 GB SSD; 16 InfiniBand ports per pair

Mira I/O Hardware (storage systems)

- **mira-fs0** - *project file system*
  - (16) DDN SFA12Ke systems
    - 8,960 SATA disks
    - 512 SSDs
  - 19 PB of formatted storage
  - ~240 GB/s (read or write)
- **mira-fs1** – *project file system*
  - (6) DDN SFA12Ke systems
    - 3,360 SATA disks
    - 12 SSDs
  - 7 PB of formatted storage
  - ~90 GB/s
- **mira-home** – *home file system*
  - (3) DDN SFA12Ke systems (5 drawer)
    - 1 PB of formatted storage

SATA – RAID6 (8+2)
SSD – RAID10

Mira I/O Forwarding

- Storage system as a whole is optimized for scale (# processors) and throughput (# of GB)
  - Implies that small operations or individual latencies may be worse
- The Blue Gene /Q uses a technique call I/O forwarding
- Certain system calls issued on the compute node are sent to the I/O node for processing (mainly calls related to I/O)
- The system calls are replayed on the I/O node and the results returned to the compute node
- This architecture is used in order to simplify interactions with the parallel file system
  - 49152 compute nodes -> 384 I/O nodes

Notes

- Can make typically low-overhead system calls costly
  - Transmitting system call over network to be processed at ION
  - Example: *lseek*
  - Minimize these type of calls which perform a small amount of work per call

Mira - GPFS

- IBM's GPFS is used for all parallel file systems on Mira
- Fully supports POSIX semantics
- Client-side and server-side caching
- Metadata is replicated on all file systems
- Quotas are enabled
  - *myquota* (home)
  - *myprojectquotas* (project)
  - You will get an error if you overrun your quota (-EQUOTA)

Name	Type	Blocksize	Capacity	Speed
mira-fs0	project	8 MB	19 PB	240 GB/s
mira-fs1	project	8 MB	7 PB	90 GB/s
mira-home	home	256 K	1 PB	-

myprojectquotas

```
[harms@miralac1 benchmarks]$ myprojectquotas

mira-fs0 : Current Project Quota information for projects you're a member of:

Name                          Type      Filesystem        GB_Used       GB_Quota           Grace
================================================================================================
Acceptance                    Project       mira-fs0        7491.93        7400.00          7 days
IBMGSSTest                    Project       mira-fs0           0.00           0.01            none
Maintenance                   Project       mira-fs0       14074.71      102400.00            none
Performance                   Project       mira-fs0       88247.11      102400.00            none
Stability_Harness             Project       mira-fs0        1842.53        1842.53            none


mira-fs1 : Current Project Quota information for projects you're a member of:

Name                          Type      Filesystem        GB_Used       GB_Quota           Grace
================================================================================================
ALCF_Getting_Started          Project       mira-fs1           0.00        1024.00            none
CONVERGE-BGQ-LDRD             Project       mira-fs1         947.45           0.01         expired
IBM-performance               Project       mira-fs1         154.19        1024.00            none
Maintenance                   Project       mira-fs1          81.38        5120.00            none
OpenFOAM-ALCF                 Project       mira-fs1         604.85        1024.00            none
VERIFI_Workshop               Project       mira-fs1        1495.54           0.01         expired
```

Libraries

- ALCF offers several compiled I/O libraries
  - HDF5
  - NetCDF
  - pNetCDF
  - Adios
  - Look under /soft/libraries/
- These libraries offer capabilities to make managing large parallel I/O easier
- HDF5 Example
  - provides ability to access data as arrays
  - Access data by variable names
- Quincey Koziol will give a detailed presentation on HDF5

MPI-IO

- Mira has great support for MPI-IO
  - Leveraged by major I/O libraries like HDF5 or pNetCDF
- Aggregates I/O requests in to larger request sizes
- Handles alignment on block boundaries
- Leverages Mira 5D Torus network
- Set this environment variable for better performance
  - --env BGLOCKLESSMPIO_F_TYPE=0x47504653
  - Disables extra locking within Blue Gene ADIO layer

Notes

- MPI-IO scales up well but at full machine scales it is possible to run into issues with running out of memory
  - This is related to memory usage needed for various Alltoallv calls and derived data types
  - Workarounds exist if you run into these issues

MPI-IO continued

- Advanced Options

- MPICH-3.1.1
  - Has new enhancements that improve I/O performance on BG
  - Not an IBM supported version, so you need to build it yourself
  - http://www.mpich.org/2014/06/04/mpich-3-1-1-released/

- Variables for workarounds
  - Generally will result in worse performance
  - --envs BGMPIO_COMM=1
  - --envs PAMID_SHORT=0
  - --envs PAMID_DISABLE_INTERNAL_EAGER_TASK_LIMIT=1

Files

- Creating and opening files has overhead (metadata operations)
- Using thousands of files for a dataset is probably ok, using tens of thousands may become problematic
- Suggestion is to use one to hundreds of files
- Creating files in a single directory in parallel becomes problematic because of locking of the directory

Notes

- If you insist on file-per-process
  - Pre-create the files before the job runs
  - *or* Use a unique (pre-created) directory per file
  - *or* Create all the files on 1 rank first, then reopen the files on the other ranks

Files - Examples

```
[harms@miralac1 test]$ time python create.py
real    1m10.544s
user    0m1.129s
sys     0m7.092s
[harms@miralac1 test]$ ls file.* | wc -l
131072
```

From IOR

```
 access              = file-per-process
        pattern             = segmented (1 segment)
        ordering            = sequential offsets
        clients             = 49152 (1 per node)
        repetitions         = 1
        xfersize            = 8 MiB
        blocksize           = 2 GiB
        aggregate filesize = 98304 GiB

Commencing write performance test.
Wed Sep 18 01:38:07 2013

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s)   iter
------    ---------  ---------- ---------  --------   --------   --------   ----
write     138040     2097152    8192       308.73     420.48     0.026884   0
```

Argonne Leadership Computing Facility

Files – code example

```
//   Create and Preallocate the File(s) On Master
if ( 0 == m_rank)
{
     retval = MPI_File_open(MPI_COMM_SELF, (char *)m_partFileName[i],
                            MPI_MODE_WRONLY | MPI_MODE_CREATE,
                            MPI_INFO_NULL,&m_fileHandle[i]);
    assert(retval == MPI_SUCCESS);

     // Preallocate File
     MPI_File_set_size(m_fileHandle[i], m_partFileSize);
    MPI_File_close (&m_fileHandle[i]);
}

MPI_Barrier (MPI_COMM_WORLD);

//   Open the File for Writing
retval = MPI_File_open(MPI_COMM_WORLD, (char *)m_partFileName,
                       MPI_MODE_WRONLY, MPI_INFO_NULL, &m_fileHandle);

assert(retval == MPI_SUCCESS);
```

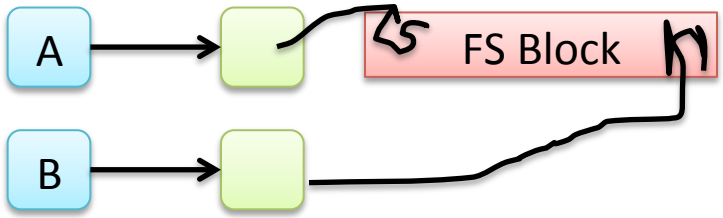
Create file on single rank and open on the other ranks

Alignment

- A critical element for success is performing block aligned I/O when using shared files
- The GPFS project file systems are all 8 MB
- Unaligned access will be punished due to GPFS locking

Example

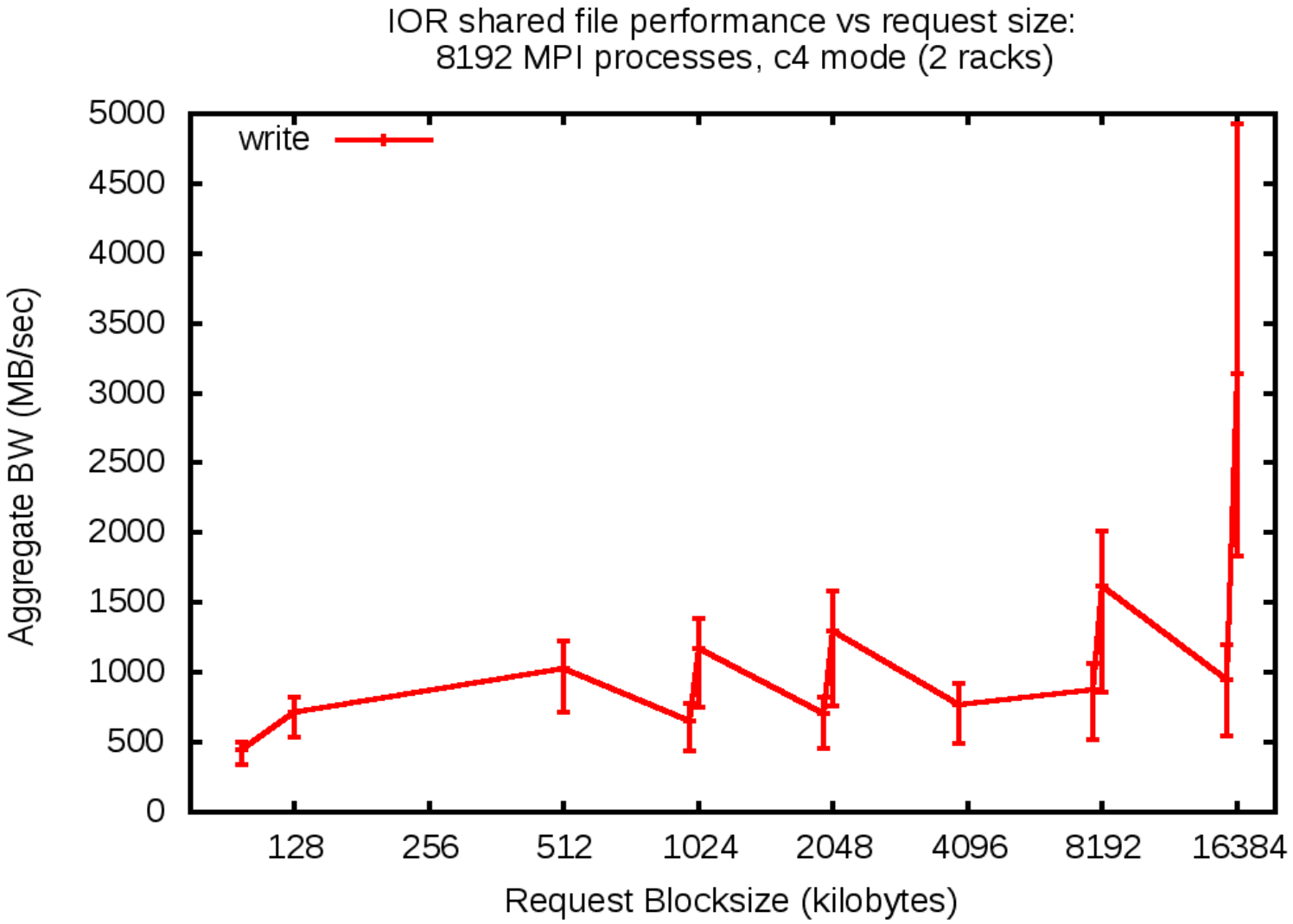
- MPI rank A and B happen to use two different I/O nodes
- Rank A writes the first MB of an 8 MB block
  - The GPFS client for rank A must acquire the lock for this fs block
- Rank B writes the last MB of an 8 MB block
  - The GPFS client for rank B tries to acquire the block for this block but must wait because it is in use
- Parallel I/O becomes serial for this workload



Request Size

- Another factor for good performance is your I/O request size
- Larger request sizes amortize the cost of operations and allow efficient transfers by the system software
- Multiples of the files system block size will work best!
  - 8, 16, 32 MB
- STREAM I/O (*fopen*, *fread*, *fwrite*, *fclose*) does buffering at the compute node level, so it will tolerate smaller requests sizes because internally it does larger requests 128K (by default)

Request Size / Alignment - Examples



Tuning

- When to tune?
  - If your I/O is less than or equal to 5% of your runtime then you are well off
  - If your I/O is 15-30%, you probably want to look at what's going on
  - If your I/O is over 30%, you need to make changes
- What to tune?
  - How "parallel" is your I/O? rank 0 only does I/O?
  - Look at file count
  - Look at request size
- Check scaling
  - I/O easily breaks at scale, you definitely want to check your solution works for your intended run size
    - Even if you ran successfully at another site, you should still check your relative performance here

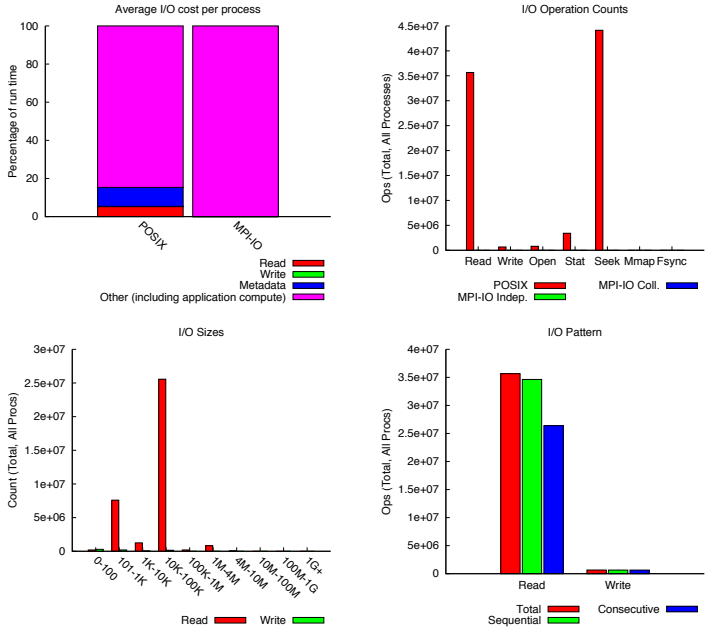
Performance Tools

- Darshan
  - https://www.alcf.anl.gov/user-guides/darshan
  - An open-source tool developed for statistical profiling of I/O
  - Designed to be lightweight and low overhead
    - Finite memory allocation for statistics (about 2MB) done during MPI_Init
    - Overhead of 1-2% total to record I/O calls
    - Variation of I/O is typically around 4-5%
    - Darshan does not create detailed function call traces
  - No source modifications
    - Uses PMPI interfaces to intercept MPI calls
    - Use ld wrapping to intercept POSIX calls
    - Can use dynamic linking with LD_PRELOAD instead
  - Stores results in single compressed log file

- TAU
  - https://www.alcf.anl.gov/user-guides/tuning-and-analysis-utilities-tau
  - "-optTrackIO" in TAU_OPTIONS

Darshan Example

- Logs can be found in
  - /gpfs/mira-fs0/logs/darshan/mira/<year>/<month>/<day>
  - /gpfs/vesta-fs0/logs/darshan/vesta/<year>/<month>/<day>
- Known issue where no logs produced when using mpixlf90 linker
  - Usually can be worked around by using mpixlf77 for linking

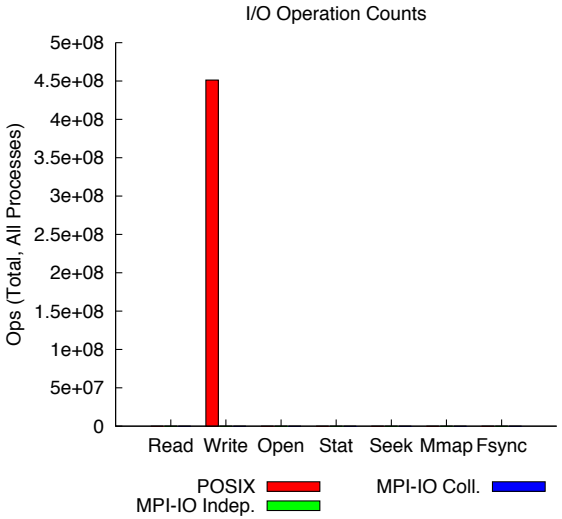


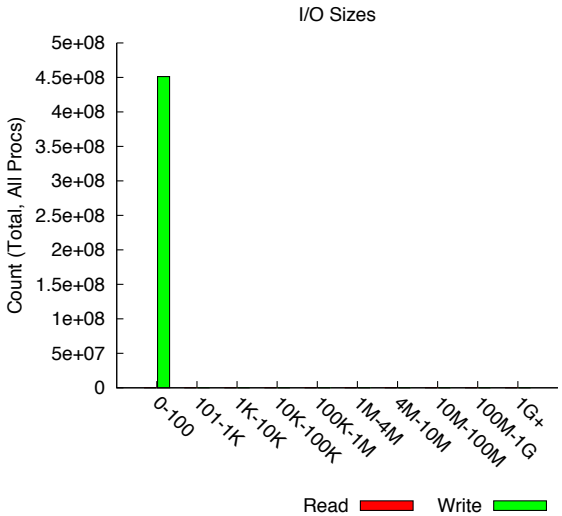
Most Common Access Sizes

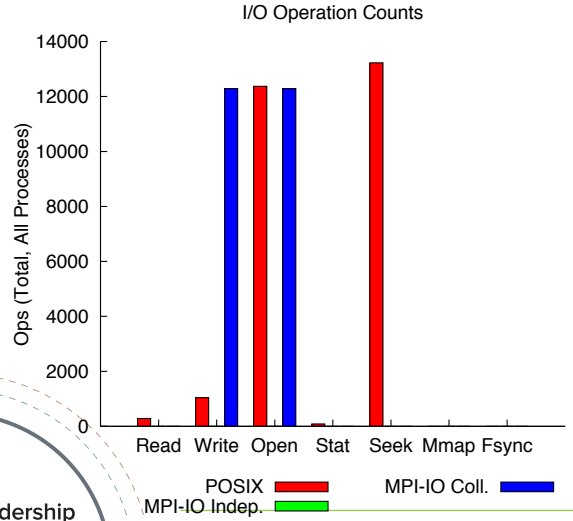
access size	count
65536	23066526
800	2378602
304	2172322
400	2029354

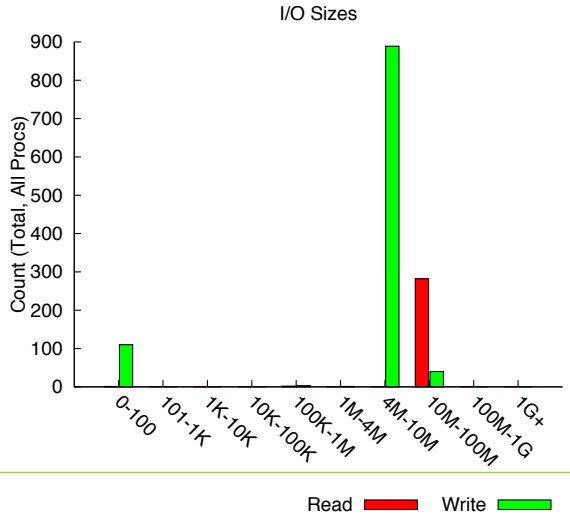
Darshan Example

- Example of an application where I/O worked well at 64 ranks on a traditional Linux cluster, problematic on Mira on 2048 ranks.









Best Practices

- Consider using a high-level I/O library
- Use a “small” number of files from 1 to hundreds
- Consider using MPI-IO and collective calls if not a high-level library
- Perform block aligned I/O
  - /projects file systems are 8 MB blocks
- I/O request sizes should be “large” where large is 8 MB
  - Helps to achieve the above and uses GPFS efficiently

Conclusion

- There is no one-size fits all I/O solution (unfortunate)
- The solution you build might be *somewhat* system specific
  - Meaning the exact tuning for Mira may be different than what you might use for Titan (also unfortunate)
- However
  - I/O is usually a tractable problem and doesn't require extensive resources to fix
  - Solutions can also be built that are flexible enough to be modified for different sites
  - High level I/O libraries are a good example
- ALCF Staff is available to help!