DARSHAN

Characterize IO performance

Cristian Simarro
Peter Towers

Special thanks to Cray

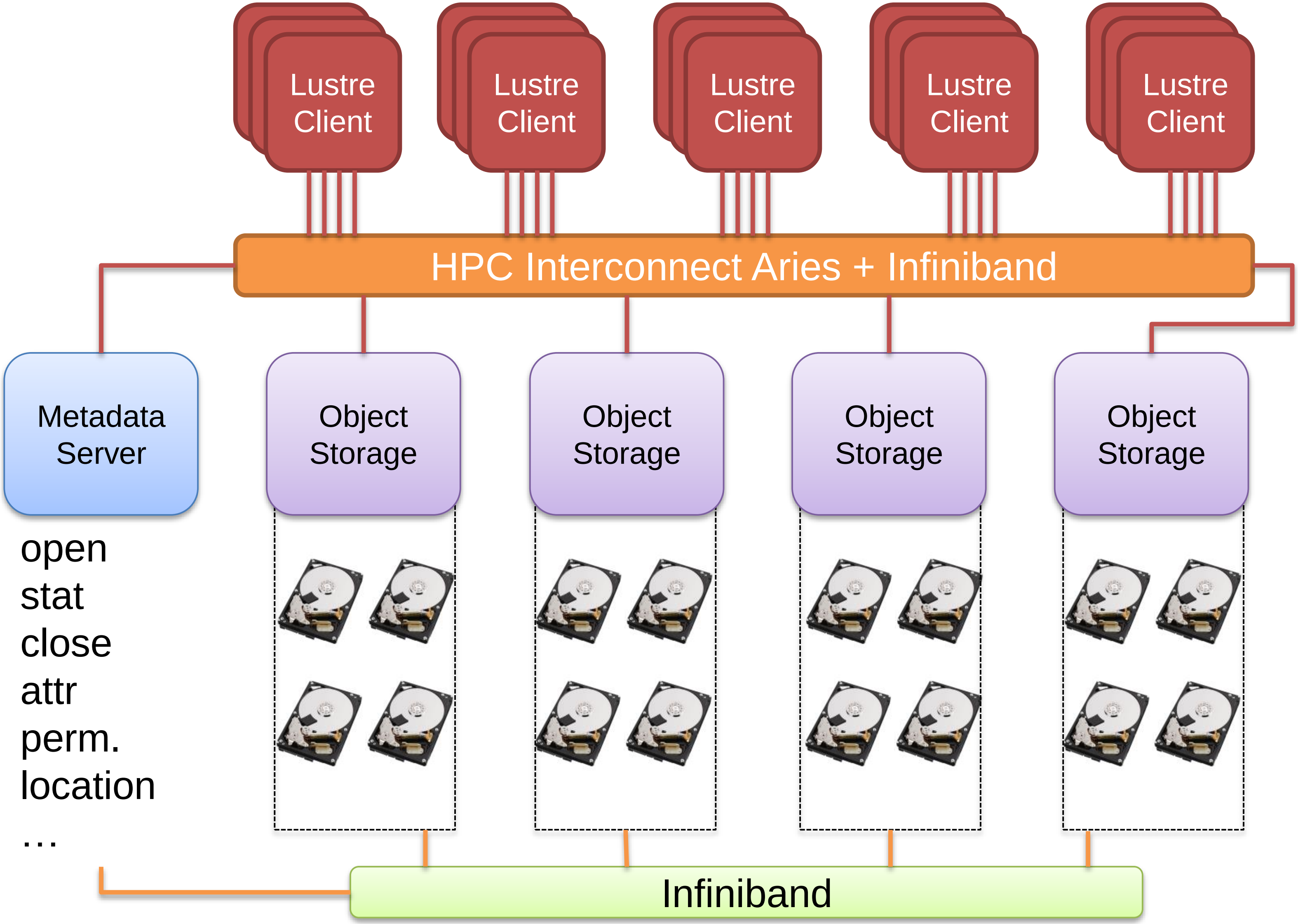
Cristian.Simarro@ecmwf.int
Peter.Towers@ecmwf.int

ECMWF

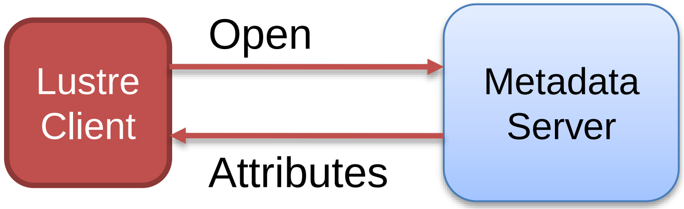
Index

- Lustre summary
- HPC I/O
  - Different I/O methods
- Darshan
  - Introduction
  - Goals
  - Considerations
  - How to use it
  - Job example
  - Log files
- I/O Recommendations

Lustre filesystem in ECMWF

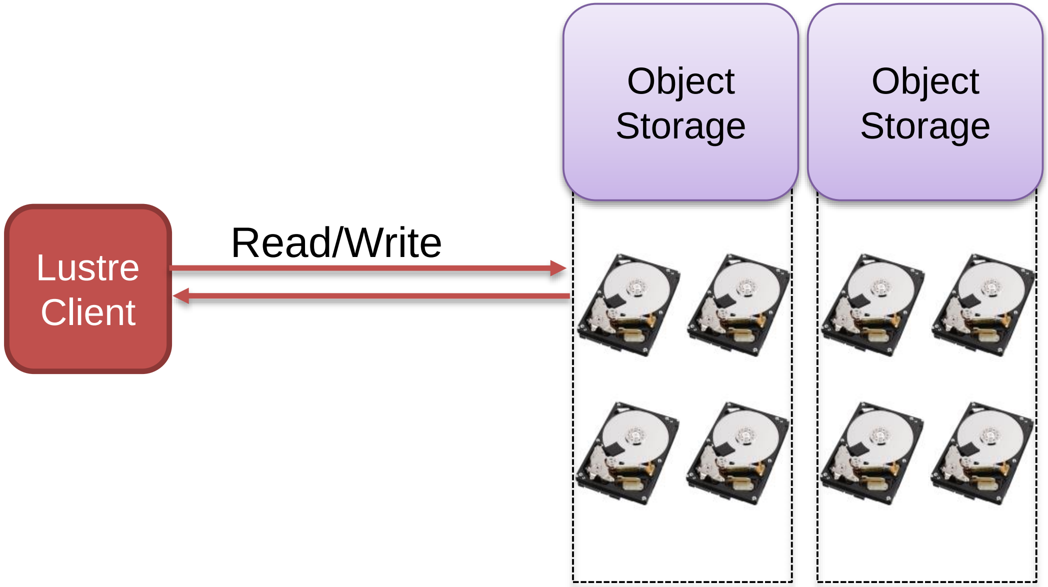


Lustre workload



The node asks to the metadata:

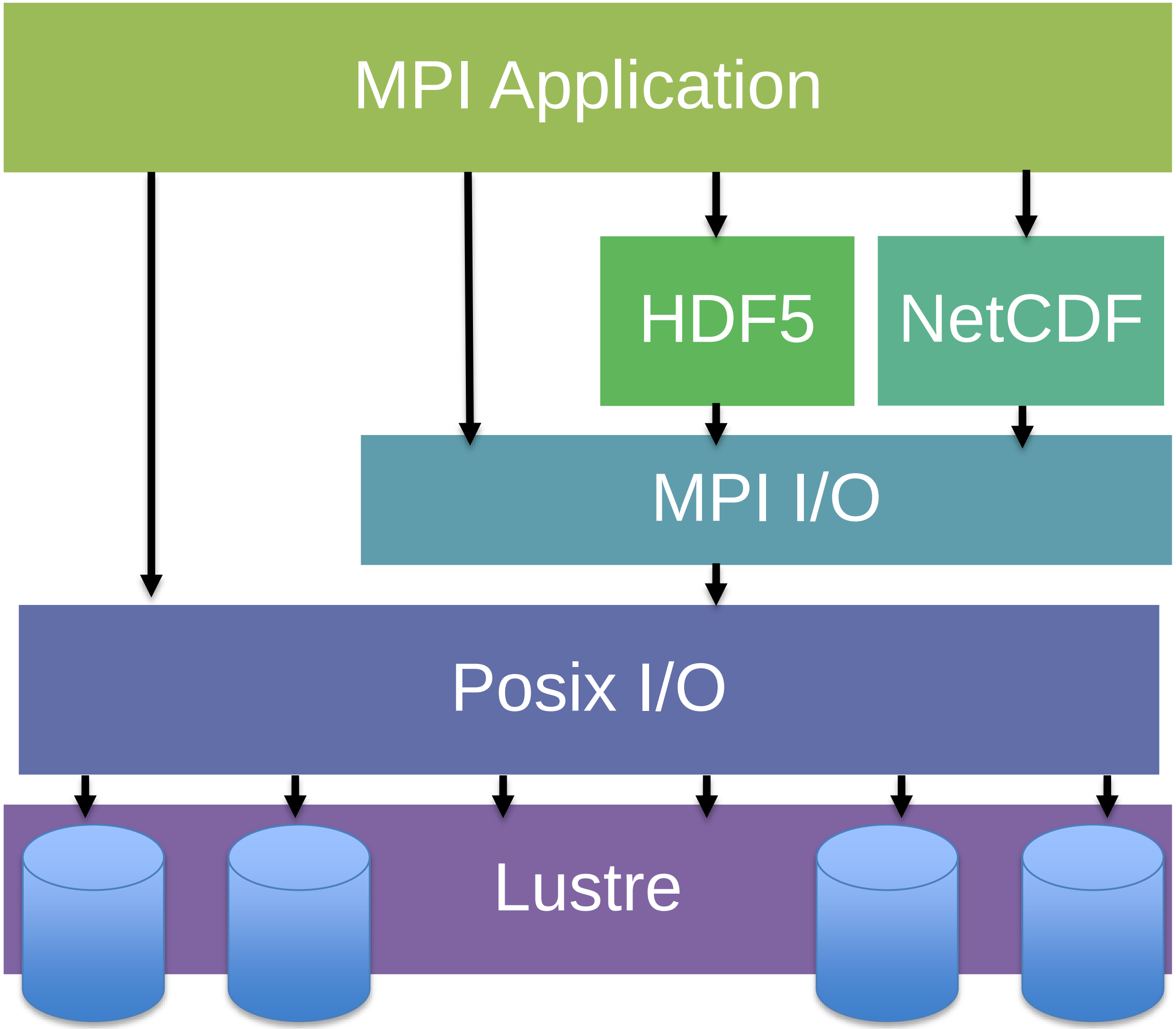
- If read, where is the file
- If write, random Object Storage



Once the node knows where, the communication begins.

All the data transfer is done directly from now on for this file.

I/O characterization



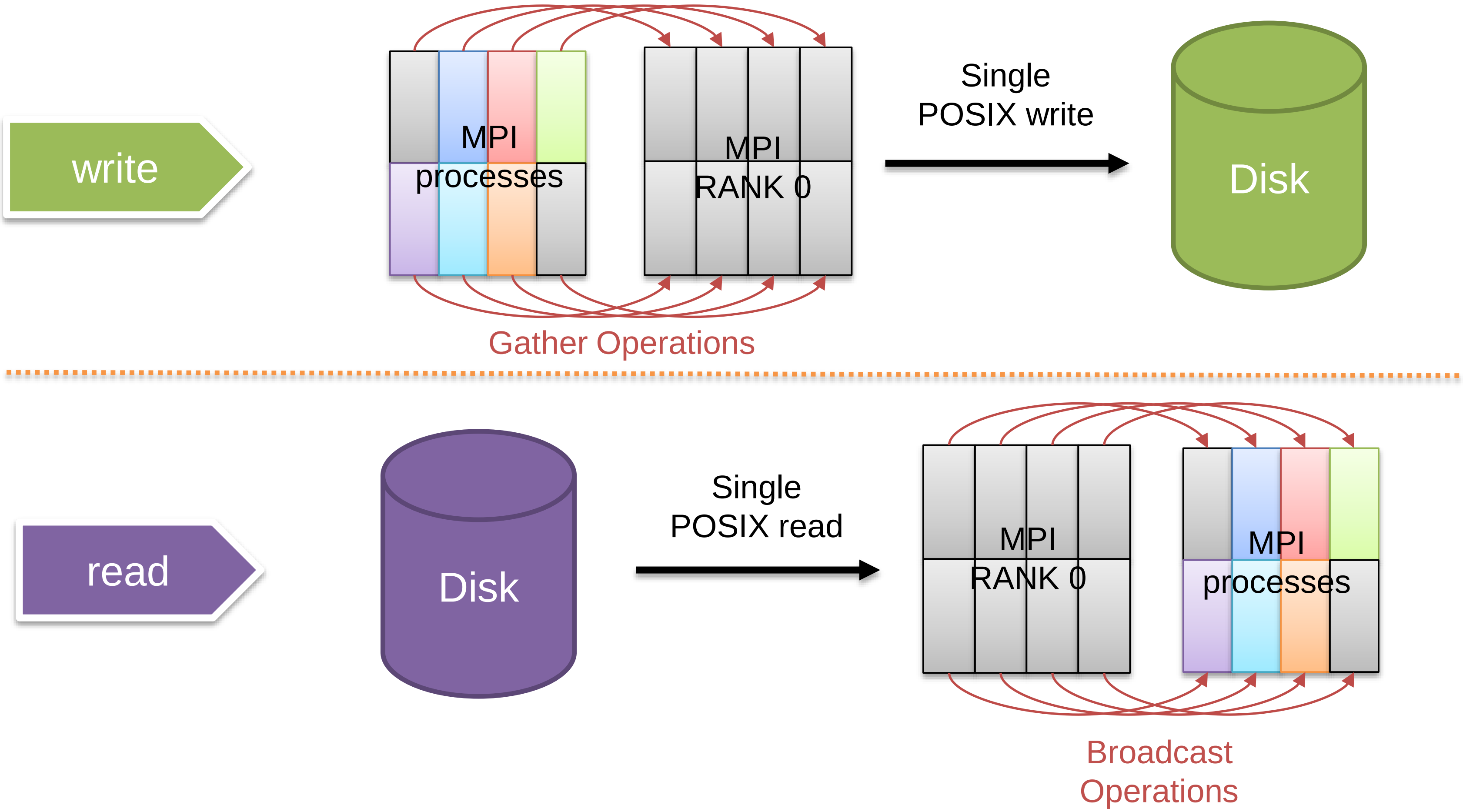
Different HPC I/O methods
Posix

- Portable Operating System Interface
- API + shell and utilities interfaces compatible UNIX
- Simplest mechanism to write data on disk

- Two different strategies can be used

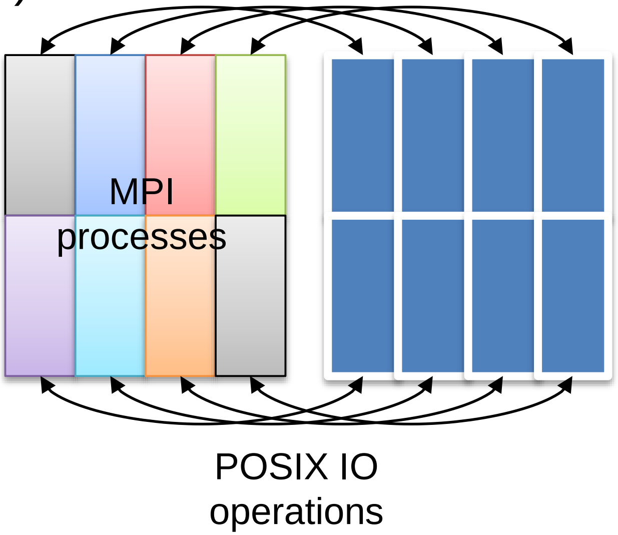
Different HPC I/O methods
Posix 1

Single POSIX call + MPI call (small files)

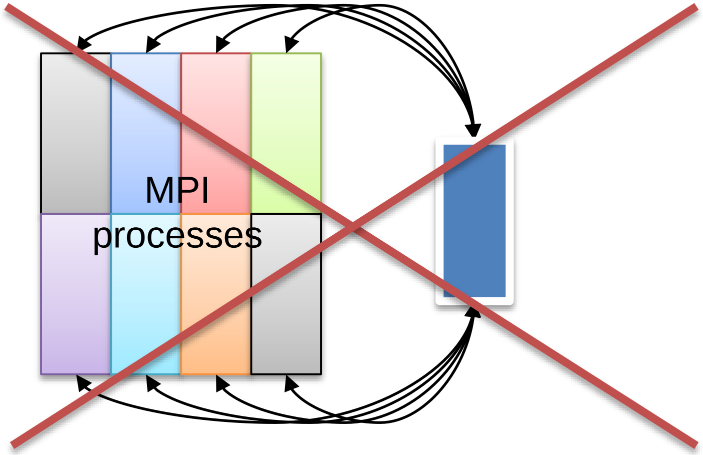


Different HPC I/O methods
Posix 2

Multiple (different) POSIX files



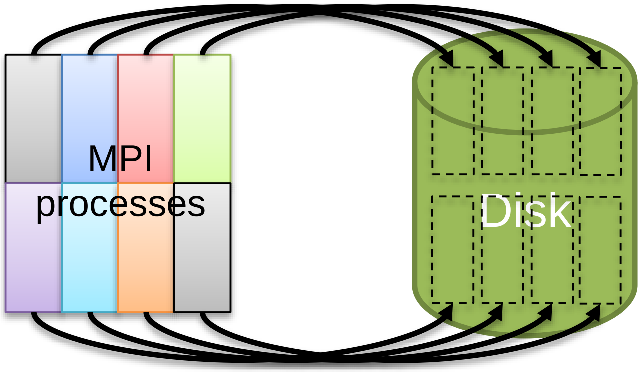
Avoid Multiple POSIX operations from several parallel tasks to the same file
(read)



Different HPC I/O methods
MPI-IO

- Same behaviour for HDF5

- Is built on MPI data types + collective communications
- Stripe
- Allows an application to write into both
  - distinct files
  - or the same file from multiple MPI processes



HPC I/O considerations

- WRITE
  - Single writer multiple files -> scalability problems
  - Multiple writers multiple files -> metadata bottleneck
  - Multiple writers single file
    - If no stripe -> bottleneck OST
    - Use parallel tools (MPI-IO, HDF5, pnetCDF…)
    - Group tasks to write (reduction)
      - Use 1 IO task to collect and write per group/node…
- READ
  - Avoid different tasks reading same file
    - Use 1 read + broadcast
  - Avoid unnecessary metadata operations

You need to experiment to find the best I/O method!!

DARSHAN

Introduction
Goals
Considerations
How to use it
Job example
Log files

Introduction

- **Darshan** is a scalable HPC I/O characterization tool
- Developed by (ANL)
  - http://www.mcs.anl.gov/darshan
- Profile I/O (C and Fortran) calls including:
  - **POSIX**
  - MPI-IO
  - HDF5
  - PnetCDF
- It uses **LD_PRELOAD** mechanism to **wrap** the IO calls
- Based on version 2.3.1-pre1 and patched for ECMWF
- We have created a summary tool

Goals

- Allow ***member state users*** to characterize and improve the IO of their applications
- Allow ***HPC support and admins*** to gain insight about the IO behavior of the applications
- Guidance to ***researchers*** to tune the directions of HPC IO of the product generation and models

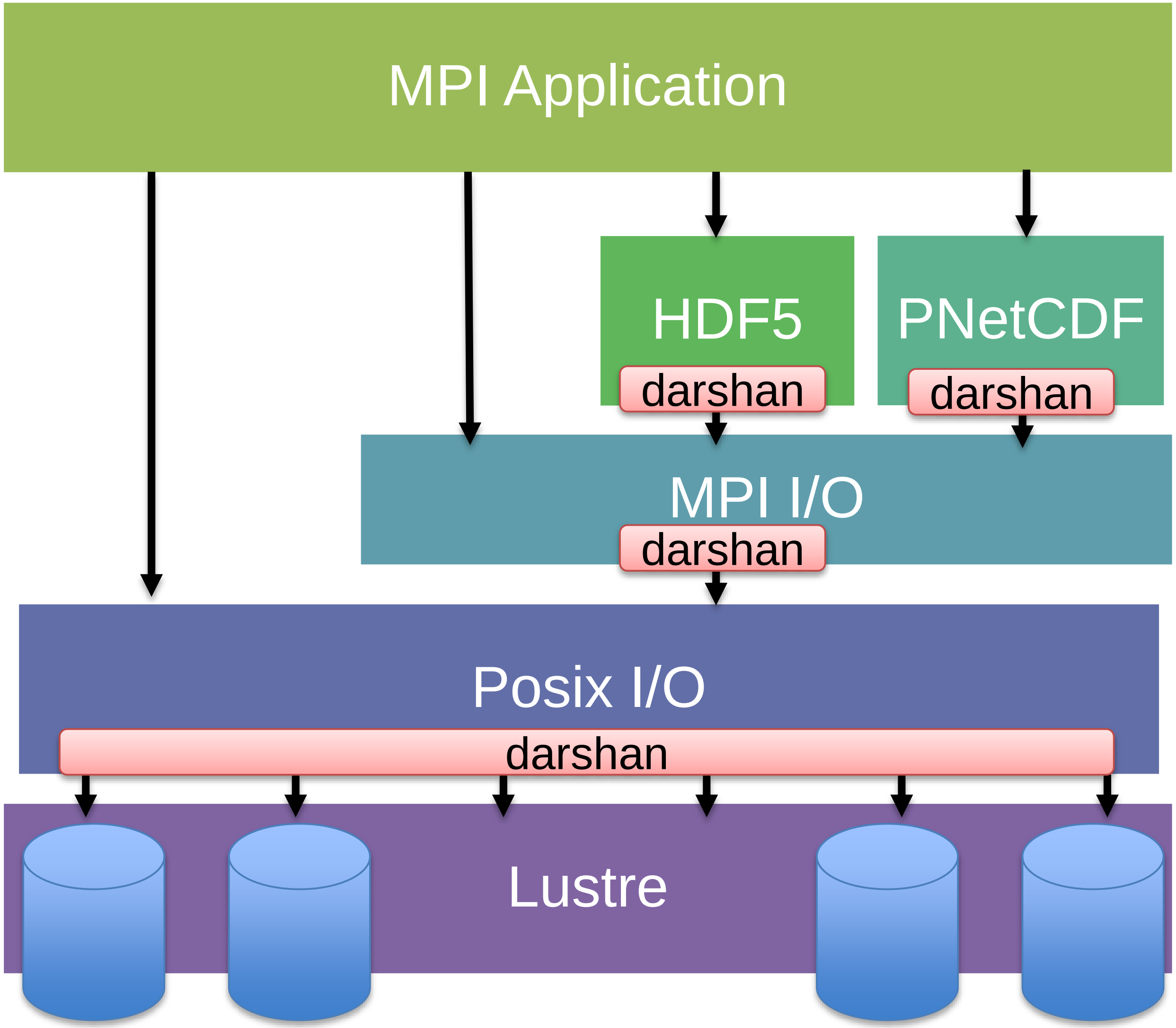
Requirements

- It has to be as transparent as possible
  - Scalable
  - “Automatic”
- User-friendly summary tools to inspect the results

Considerations

- Darshan is not a IO tracer, it reports statistics, counters and timings for the IO
- The information is gathered at the MPI_Finalize
  - The program **must** contain MPI_Init() and MPI_Finalize() calls
- Incompatibility with system() call. It has been disabled
- Selective system directories not profiled by default
  - /usr/, /proc/, /etc/ …
  - They can be activated manually
- mmap is not profiled because of Cray incompatibility
- We recommend to "module unload atp" before running with Darshan

Darshan wrappers



```
functionA(args):
   timer1
   _real_functionA(…)
   timer2
   darshan_log(function,T1,T2)
```

Workload

- Compile the MPI program
- Run the application with
  - module unload atp
  - module load darshan
- Look for the **Darshan log file**
  - Normally in the directory from the job was submitted
  - or setting DARSHAN_LOG_DIR=
- Use darshan tools to analyse the log
  - IOsummary.py
  - darshan-job-summary.pl
  - darshan-parser.pl

Job example

```
#!/bin/bash
#PBS -N DSH_TEST
#PBS -q np
#PBS -l EC_total_tasks=48
#PBS -l EC_threads_per_task=1
#PBS -l EC_hyperthreads=2
#PBS -l walltime=01:00:00

cd $SCRATCH/...
module unload atp
module load darshan
export DARSHAN_LOG_DIR=$SCRATCH/darshan-logs
mkdir -p $DARSHAN_LOG_DIR

####export DARSHAN_EXCLUDE_DIRS="/etc/,/proc/"

aprun -N $EC_tasks_per_node -n $EC_total_tasks -d
$EC_threads_per_task -j $EC_hyperthreads <mpi_program>
```

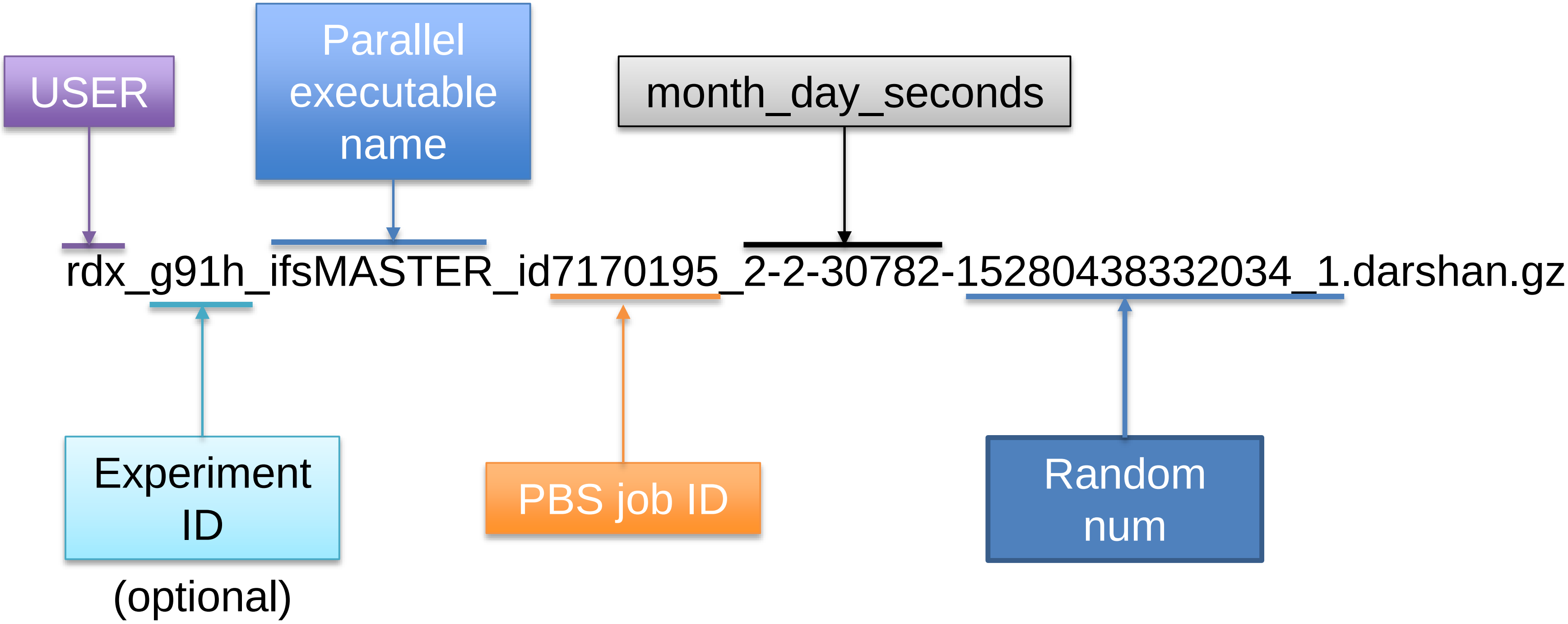
Job output

```
…
## INFO OUT: #PBS -l EC_tasks_per_node=48
## INFO OUT: #PBS -l EC_total_tasks=96
## INFO OUT: #PBS -l EC_hyperthreads=2
## INFO OUT: #PBS -q np
## INFO OUT: #PBS -l walltime=02:00:00
## INFO


INFO: activating darshan, log will be placed here in
/scratch/us/uscs/apps/MPIIO/darshan-logs


longest_io_time        = 828.979162 seconds
total_number_of_bytes = 103079215104
transfer rate          = 118.584404 MB/s
…
```

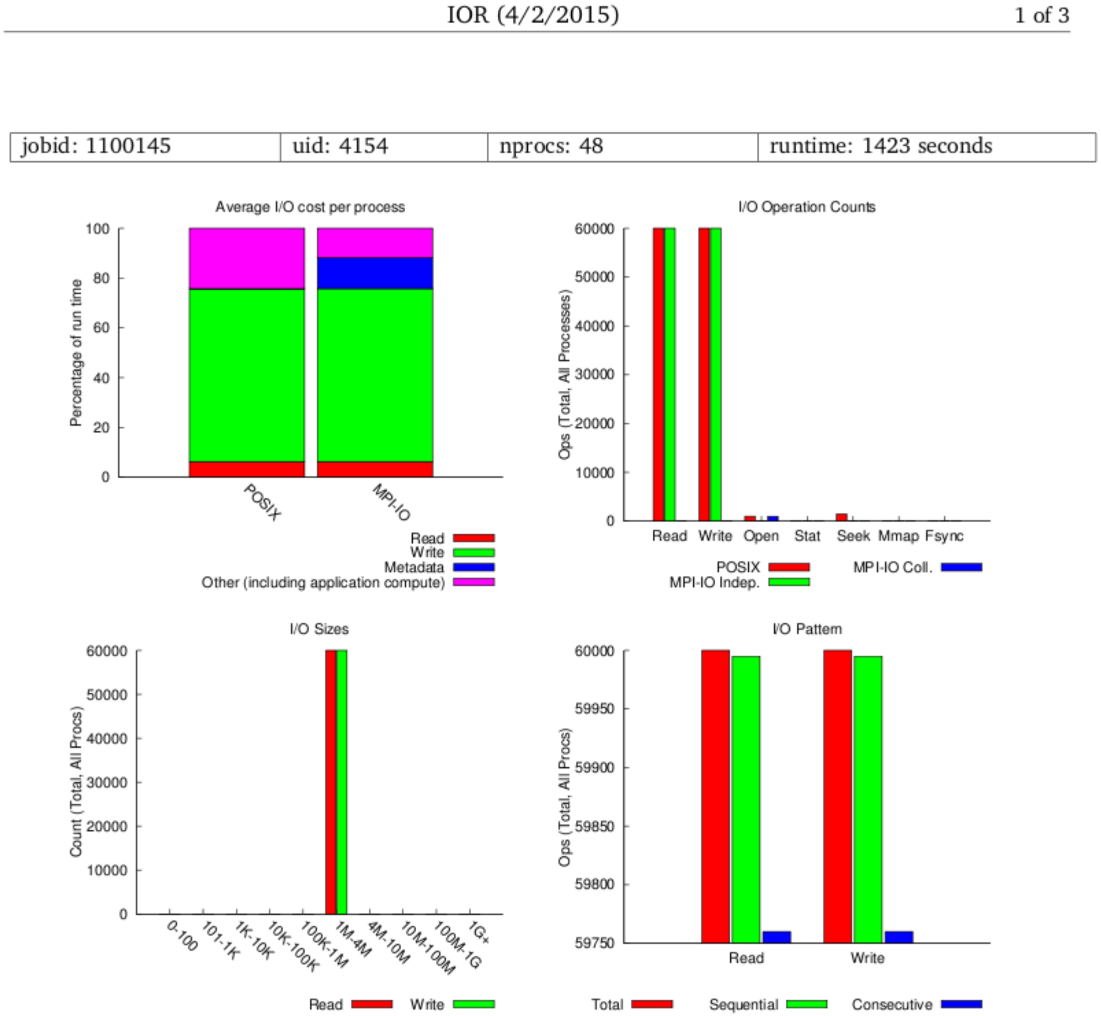
Darshan log file



Reading the Darshan Log File (.pdf)

- darshan-job-summary.pl <darshan_log.gz>

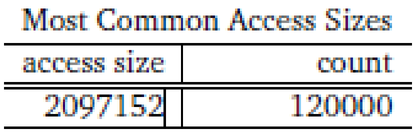
Just 1 file:
testFile
48 processes
12GB write
12GB read



IOR (4/2/2015) 1 of 3

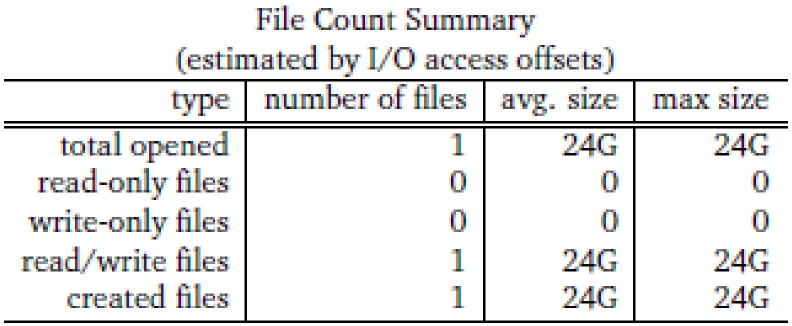
jobid: 1100145	uid: 4154	nprocs: 48	runtime: 1423 seconds

Reading the Darshan Log File (.pdf)



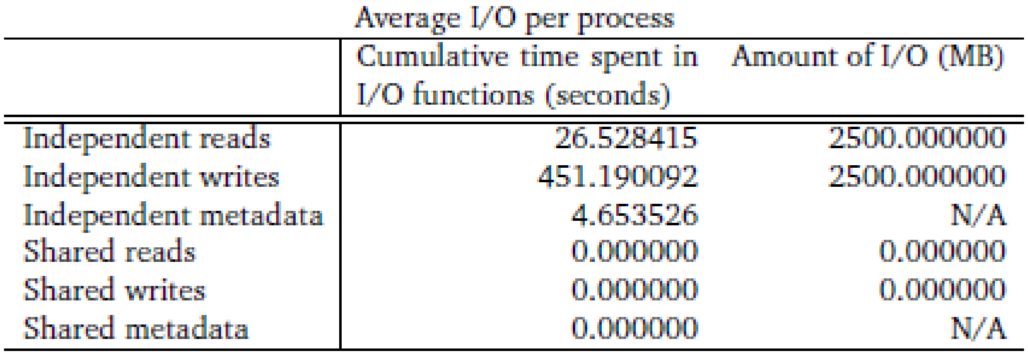
Most Common Access Sizes

access size	count
2097152	120000



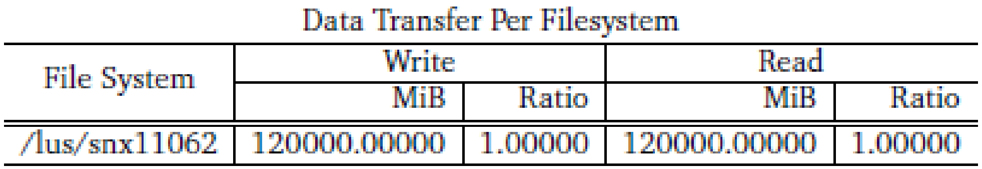
File Count Summary
(estimated by I/O access offsets)

type	number of files	avg. size	max size
total opened	1	24G	24G
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	24G	24G
created files	1	24G	24G



Average I/O per process

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	26.528415	2500.000000
Independent writes	451.190092	2500.000000
Independent metadata	4.653526	N/A
Shared reads	0.000000	0.000000
Shared writes	0.000000	0.000000
Shared metadata	0.000000	N/A



Data Transfer Per Filesystem

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/lus/snx11062	120000.00000	1.00000	120000.00000	1.00000

Reading the Darshan LogFile (IOsummary.py)

- ECMWF Python script to retrieve useful information in text format.
- Tailored to retrieve different information
  - Per file/shared file
  - Per MPI rank
  - Different summaries
- You can see different operation timings:
  - Metadata
  - Read
  - Write

Reading the Darshan LogFile (IOsummary.py)

```
usage: IOsummary.py <file_darshan.gz>

Arguments:
    -a  this enable all the reports
    -f  enable report each rank all files (default 10 per rank)
    -t  enable report aggregated per MPI rank
    -s  enable summary of all IO operations
    -i  enable print list of all shared files
    -j  enable summary of shared files
    -p  enable report for shared files
    -h  shows this help message

Extra arguments:
     --extended             shows all the files per rank
                                    (default: 10)
     --threshold=N.N        will change the default threshold to N.N seconds
                                    (default 5.0 seconds)
                            this means that the table will show all the
files which Meta + Read + Write time is lower than N.N
     --ntasks=N             minimum number of tasks to consider a file
shared
                                    (default: 4)
     --systemfiles          this special flag will enable the report of
system files a.k.a. /etc/, /usr/, /proc/... if you have
                            asked to report without excluding these dirs
```

Reading the Darshan LogFile

IOsummary.py <file_darshan>

```
#####################################################################
#######################JOB RESUME####################################
Executable: /fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ifsMASTER
Nprocs (-n):    288
JOB ID:         676372
Start Time:     Mon Jan 19 08:35:30 2015
End Time:       Mon Jan 19 08:41:49 2015
Run Time:       380

SHOW INFO:
  - Showing 10 most expensive IO files per task
  - Showing files with more than 5.0 seconds of Meta + Read + Write time
, you can change it using --threshold=N.N
  - Considering shared files those that have been accessed by 4 or more
ranks
#####################################################################
```

This can be changed

```
--extended
--threshold=N.N
--ntasks=N
```

Reading the Darshan LogFile (IOsummary.py)

Individual 1 task 1 file per row

```
IOsummary.py -f

Report File per task data
----------------------------------------------------------------------
(threshold is 5.0 seconds of Meta + Read + Write time, you can change it using --threshold=N.N)
(the table is just showing the 10 most expensive files per rank, use --extended to see them all)


 rank opens   stats   seeks       File    Meta    Read        MB       MB/s   Write         MB       MB/s Filename
                                  size    time    time                        time
    0     2       1       1        0.4    14.0     0.1       0.0        0.0     0.0        0.4      368.8 ECMA.iomap
    0     1       2       1       31.9     5.2     0.2      31.9      159.7       -          -          - errstat
   24     1       2       1       31.0     4.8     1.1      31.0       28.8       -          -          - radiance_body
   27     1       2       1       30.7     5.5     0.3      30.7       96.0       -          -          - radiance_body
   38     1       0       0    Unknown     7.6       -         -          -     0.0        7.7     4484.0 radiance
   39     1       0       0    Unknown     6.8       -         -          -     0.0        0.7     3863.6 radiance
   40     1       0       0    Unknown     6.8       -         -          -     0.1       31.8      626.1 radiance
   41     1       0       0    Unknown     6.8       -         -          -     0.0       21.2      603.2 radiance
   42     1       0       0    Unknown     6.8       -         -          -     0.0        2.0     4289.5 radiance
   43     1       0       0    Unknown     6.9       -         -          -     0.0        3.5     4392.7 radiance
   44     1       0       0    Unknown     6.8       -         -          -     0.0        9.7      561.5 radiance
   46     1       0       0    Unknown     6.9       -         -          -     0.0        5.2     4264.7 radiance
   48     1       0       0    Unknown     6.8       -         -          -     0.1       30.4      486.1 radiance
   50     1       0       0    Unknown     6.8       -         -          -     0.1       32.0      613.2 radiance
   51     1       0       0    Unknown     6.8       -         -          -     0.1       28.2      447.5 radiance
   52     1       0       0    Unknown     6.8       -         -          -     0.0       14.9      688.5 radiance
   54     1       0       0    Unknown     6.8       -         -          -     0.0       11.5      630.0 radiance
   55     1       0       0    Unknown     6.8       -         -          -     0.0        2.4     4488.4 radiance
   56     1       0       0    Unknown     6.8       -         -          -     0.0        2.7     4102.1 radiance
   61     1       0       0    Unknown     6.8       -         -          -     0.0        1.9     4496.8 radiance
   62     1       0       0    Unknown     6.8       -         -          -     0.0        0.7     2666.3 radiance
   63     1       0       0    Unknown     7.6       -         -          -     0.0        0.0     1632.3 radiance
   72     1       2       1       31.2     3.7     1.4      31.2       21.9       -          -          - poolmask
  117     1       2       1       31.9     5.3     0.2      31.9      191.0       -          -          - errstat
  118     1       2       1       31.8     5.3     0.1      31.8      360.3       -          -          - errstat
  147     1       2       1       31.0     5.5     2.2      31.0       14.4       -          -          - radiance_body
  177     1       2       1       31.9     4.7     1.0      31.9       30.8       -          -          - errstat
```

Reading the Darshan LogFile (IOsummary.py)

Individual 1 task N files per row

```
IOsummary.py -t

Report aggregated per MPI task
---------------------------------------------------------------------------------------------------
 rank opens   stats   seeks    Meta    Read          MB       MB/s  Write          MB       MB/s
                           time    time                         time
    0   542    1151    1653    32.2    72.1      6751.8       93.6    1.2       262.4      219.6
    1    37      83     421     2.0     0.6        38.1       63.6    0.1        63.3      469.6
    2    39      85     422     2.3     2.0        61.4       31.4    0.1        83.8      579.6
    3    38      87     365     2.1     0.8       143.0      168.4    0.1        76.1      589.1
    4    42      87     442     3.0     1.6        61.6       38.4    0.3       174.6      501.0
    5    40      85     422     2.0     1.0        65.9       65.4    0.2       125.7      582.8
    6    42      91     441     2.4     2.5       152.0       61.8    0.3       125.6      431.6
    7    39      83     421     2.6     1.1        39.4       35.3    0.2       125.5      517.7
    8    46      97     389     3.2     2.0       258.4      126.9    0.4       198.8      544.4
    9    38      81     420     2.7     0.6         7.6       12.3    0.2       126.2      542.1
   10    41      87     431     3.4     3.8        88.3       23.2    0.2       126.6      572.4
   11    38      81     428     2.5     0.6         7.6       12.5    0.2       135.9      545.5
   12    54     103     576     3.6     2.0       177.9       88.5    0.5       233.2      454.6
   13    40      83     429     2.8     0.6        38.9       61.1    0.3       179.2      553.3
   14    43      87     423     2.5     1.4        92.8       66.5    0.3       152.8      539.5
   15    40      85     422     2.5     3.0        70.6       23.6    0.3       136.9      531.0
   16    43      89     459     3.2     1.3        91.2       71.7    0.4       198.2      518.1
   17    43      91     425     2.6     1.3       161.4      124.3    0.2       130.6      541.1
   18    43      91     425     2.8     3.5       150.0       42.4    0.3       124.9      485.6
   19    38      81     436     2.7     0.5         7.9       17.0    0.2       124.9      511.3
   20    42      87     442     3.2     0.8        61.6       80.4    0.4       205.8      540.5
   21    42      89     424     2.6     1.6       124.8       77.2    0.2       125.5      543.4
   22    40      85     422     2.6     1.1        61.6       57.5    0.2       126.9      544.6
…
```

Reading the Darshan LogFile (IOsummary.py)

Individual N tasks 1 file per row

```
IOsummary.py -p

Report of shared files IO
-----------------------------------------------------------------------------------------------------
(Considering shared files those that have been accessed by 4 or more ranks, you can change it using --ntasks=N)

 rank opens    stats    seeks     Meta     Read         MB         MB/s    Write         MB        MB/s
                                  time     time                            time
  288   289     1155        4      7.2    100.4      964.5          9.6      0.6       83.9       143.5 VARBC.cycle
  288   576     2304     3744     85.0      0.2        8.0         44.5        -          -           - wam_namelist
  288   288      864        0      8.6     72.0       27.7          0.4        -          -           -
ssmi_mean_emis_climato_05_cov_interpol
  288   288      864        0     73.2      0.0        0.5         10.2        -          -           - ascat_s0.cor
  288   288      864        0     65.7      0.0        0.2          6.1        -          -           - ers_s0.cor
  288   288      864        0     63.1      0.1        9.2         70.0        -          -           - ascat_sp.cor
  288   288      864        0     59.9      0.1        4.3         50.1        -          -           - ers_sp.cor
  288   288     2304        0     55.8      0.3      119.4        398.7        -          -           - wam_subgrid_2
  288   288     1152        0     51.7      0.1        0.1          1.1        -          -           - thin_reo3
  288   288     2304        0     34.4        -          -            -        -          -           - wam_subgrid_0
  288   288     2304        0     30.7        -          -            -        -          -           - wam_subgrid_1
  288   288     2304        0     28.3      0.3       98.6        365.7        -          -           - wam_grid_tables
   72    72       72        4     28.2      0.0        0.1          1.3      0.3        1.7         5.4 :4v:2100::::::12::.
  288   288        0        0     17.2      0.7      532.1        771.7        -          -           - fort.36
  288   576     1728   101088      9.3      2.1      515.4        250.5        -          -           - fort.4
  288     1      288        0      6.6      0.3       61.2        195.8        -          -           - specwavein
  288   288      864        0      5.4      0.1        1.6         24.8        -          -           - amv_p_and_tracking_error
  288     1      288        0      5.2      0.0        2.0         45.2        -          -           - sfcwindin
  288   288        0        0      1.6      3.3       11.0          3.3        -          -           - lowres_gg
  288     1      576        0      3.8      0.0        0.2         10.3        -          -           - uwavein
  288   289        1        0      2.5      0.2        1.6          7.3        -          -           - IOASSIGN.ifstraj_0
  288   288        0        0      0.8      0.6       49.3         83.3        -          -           - backgr_gg02
  288     1      288        0      0.8      0.4        0.2          0.6        -          -           - cdwavein
  288   288        0        0      0.7      0.2       20.1        110.6        -          -           - backgr_gg01
  288     2      288        0      0.4      0.3      294.6        899.2        -          -           - eda_spread_grib
  288   288        0        0      0.7      0.0        7.7        406.8        -          -           - backgr_gg00
  288   288        0        0      0.5      0.1        7.7         60.8        -          -           - main_gg
```

Reading the Darshan LogFile (IOsummary.py)

```
IOsummary.py -s

Summary of all IO
----------------------------------------------------------------------------------------------------
       3224 different files
       6656 read operations
       2643 write operations
      11171 opens
     111327 seeks
      26323 stats
       1150 files opened but no read/write action
       1435 files stat/seek but not opened

      674.7 read time
       75.0 write time
     1055.0 meta time
       16.7 stat/seek but no open time
      148.9 open but no read/write time

    45191.3 Mbytes read    at    67.0 MB/s
    38141.3 Mbytes written at   508.4 MB/s
```

Reading the Darshan LogFile (IOsummary.py)

```
IOsummary.py -j

Summary of shared files IO
---------------------------------------------------------------------------------------------------
(Considering shared files those that have been accessed by 4 or more ranks, you can change it using --ntasks=N)

         27 different files
       4907 read operations
         73 write operations
       6704 opens
     104840 seeks
      22540 stats
       1150 files opened but no read/write action
       1435 files stat/seek but not opened

      181.9 read time
        0.9 write time
      647.4 meta time
       16.7 stat/seek but no open time
      148.9 open but no read/write time

     2737.3 Mbytes read    at    15.0 MB/s
       85.5 Mbytes written at    96.2 MB/s
```

Reading the Darshan LogFile (IOsummary.py)

```
IOsummary.py -i

List of shared files
---------------------------------------------------------------------------------------------------
(Considering shared files those that have been accessed by 4 or more ranks, you can change it using --ntasks=N)

Ranks File
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/main_gg
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/wam_namelist
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/wam_grid_tables
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/fort.4
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ascat_sp.cor
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ers_sp.cor
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/amv_p_and_tracking_error
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/fort.36
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/backgr_gg01
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/backgr_gg00
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/backgr_gg02
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/VARBC.cycle
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/eda_spread_grib
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ssmi_mean_emis_climato_05_cov_interpol
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ers_s0.cor
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/uwavein
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/wam_subgrid_2
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/wam_subgrid_1
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/wam_subgrid_0
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/ascat_s0.cor
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/sfcwindin
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/cdwavein
   72 /fws2/lb/fdb/:rd:lwda:g:g91h:20140520::/:4v:2100::::::12::.
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/lowres_gg
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/thin_reo3
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/IOASSIGN.ifstraj_0
  288 /lus/snx11064/fws2/lb/work/rd/uscs/g91h/LWDA/2014052100/an/vardir/specwavein
```

I/O Recommendations

I/O Recomendations

- Try to minimize Metadata load
  - Create, Open, Close, get attributes …
  - Locks
- Individual application run may not see a problem
- Interactive commands may affect Metadata servers
- stat() is expensive! -> ls –l, shell <Tab>, find…
  - Access to Metadata Server and each OST owning a stripe
  - Avoid stripe small files
  - Lustre tools
    - lfs find, lfs df, lustre_rsync, etc…
- Avoid large directories
  - Sequential search each time metadata operation

I/O Recomendations

- Avoid unnecessary file operations
  - If you need read-only access, open with read-only
- Compilers may help I/O performance
- Ideally, 1 access to Metadata server and then direct access to OST
  - Write same file on same OST accesses by many -> lock
    - **Stripe**
  - Read data needed by all the tasks of large application
    - **1 Read + Aries network**

I/O Recomendations

- There is a Lustre API
  - man lustreapi
  - Can be used to set striping policy for files within an application
- Try to write aligned chunks of data **(1MB)**
- If very small file, maybe another filesystem?

Be nice to Lustre

Questions?

Lustre
Metadata
OST
MPI
Darshan
…

lustre®