Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

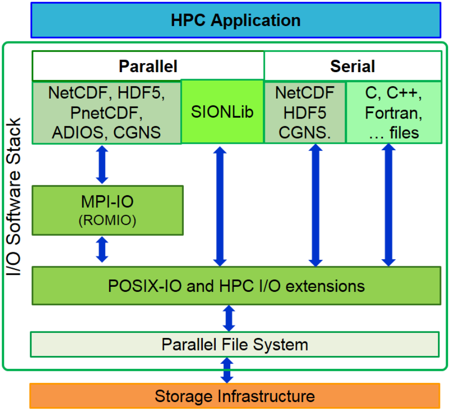
PRACE PATC Course: Advanced Topics in HPC
Topic: I/O Profiling with the Darshan Tool

Sandra Mendez, PhD - HPC Group, LRZ
sandra.mendez@lrz.de

Outline

I/O Performance Analysis

- One of the main performance limiting factor for applications is the storage systems.
- The I/O problem could be in I/O Software Stack or the configuration of the underlying I/O system.
- Understanding the I/O behavior of parallel scientific applications is critical for improving the performance on HPC systems.



I/O Profiling and Tracing tools

Profiling vs. Tracing

I/O Profiling Tools

- Characterize I/O performance of HPC applications by counting I/O-related events.
- Less intrusive and useful in identifying potential I/O bottlenecks in the performance.
- Not provide enough information for a detailed understanding of I/O.

I/O Tracing Tools

- Save individual event records with precise timestamps and per process
- Log the timing of each I/O function calls and their arguments.
- Represent as Timeline

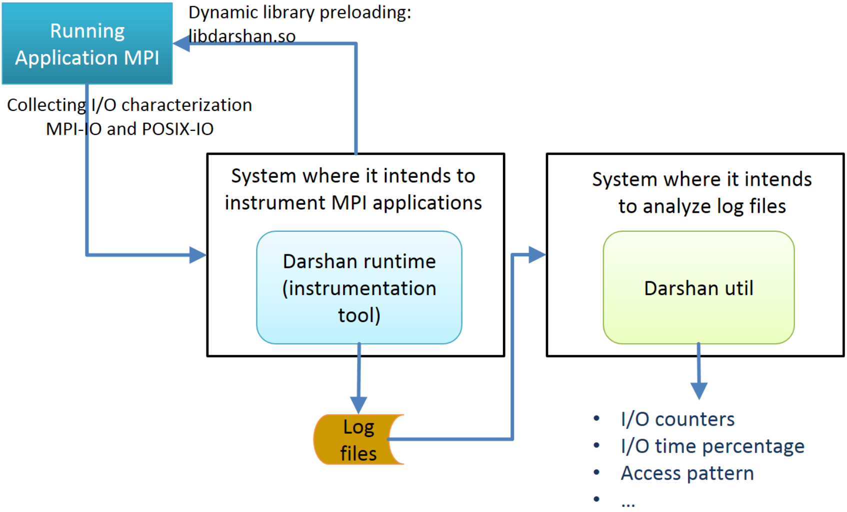
Darshan Tool

Darshan Tool

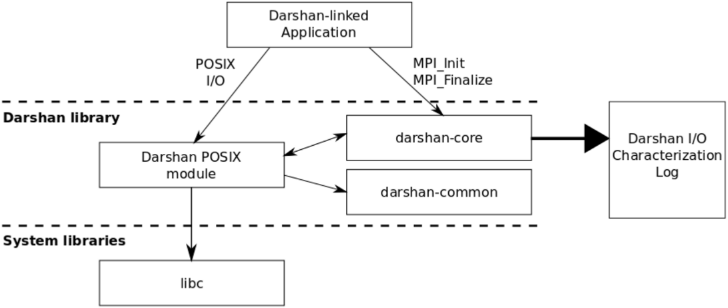
Darshan is a lightweight, scalable I/O characterization tool that transparently captures I/O access pattern information from production applications

- Developed by the Argonne Leadership Computing Facility (ANL). http://www.mcs.anl.gov/darshan
- Profile I/O (C and Fortran) calls including: POSIX and MPI-IO (Limited to HDF5 and PnetCDF)
- Darshan does not provide information about the I/O activity along the runtime.
- It uses LD_PRELOAD mechanism to wrap the I/O calls

Darshan Components



Darshan 3.x - runtime component



- intercepting I/O functions of interest from a target application;
- extracting statistics, timing information, and other data characterizing the application's I/O workload;
- compressing I/O characterization data and corresponding metadata;
- logging the compressed I/O characterization to file for future evaluation

How to use Darshan (1)?

- Using LD_PRELOAD
  Fortran Program

```
export LD_PRELOAD=libfmpich.so:$DARSHAN_INTALL/lib/libdarshan.so
```

  For C/C++ Program

```
export LD_PRELOAD=$DARSHAN_INTALL/lib/libdarshan.so
```

- Setting DARSHAN_LOGPATH_ENV if it was configured with:

```
--with-log-path-by-env=DARSHAN_LOGPATH_ENV.
```

- Once executed application, a log file is generated in DARSHAN_LOGPATH_ENV if the execution finishes without errors.
- Analyzing the *.darshan file using darshan-util. The log file can be analyzed in another system.

Darshan in IvyMUC (1)

- To make use of Darshan in its version 3.x, the module appropriate must be loaded.

```
module load darshan
```

- Set up the variable FORTRAN_PROG in “true” if the program is a Fortran program and false if it is not.

```
FORTRAN_PROG=false
```

- Set up environment variable DARSHAN_JOBID to environment variable name that contain the job identifier of the job manager.
- Set up Darshan log path.

```
export LOGPATH_DARSHAN_LRZ=`darshan-logpath.sh`
```

Darshan in IvyMUC (2)

Current Version

- Default version 3.1.4

Example

```
module use -a /lrz/sys/share/modules/extfiles
module load darshan/3.1.4_SLES12
######### Darshan Variables ########
FORTRAN_PROG=false
export LD_PRELOAD=`darshan-user.sh $FORTRAN_PROG`
export LOGPATH_DARSHAN_LRZ=`darshan-logpath.sh`
######### End Darshan Variables ########
```

Darshan Utils

- Using Darshan analysis tools. See online documentation: http://www.mcs.anl.gov/research/projects/darshan/docs/darshan-util.html
- Key tools:
  - darshan-job-summary.pl: creates pdf file with graphs useful for initial analysis
  - darshan-summary-per-file.sh: creates a separate pdf file for each file opened by the application
  - darshan-parser: dumps all information into ascii (text) format

```
>darshan-parser --help
Usage: darshan-parser [options] <filename>
--all   : all sub-options are enabled
--base  : darshan log field data [default]
--file  : total file counts
--file-list  : per-file summaries
--file-list-detailed  : per-file summaries with additional detail
--perf  : derived perf data
--total : aggregated darshan field data
```

IOR Benchmark

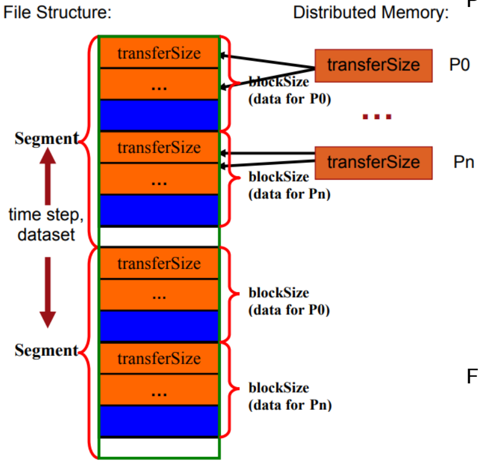


Figure: IOR benchmark Spatial Pattern (Source "Using IOR to Analyze the I/O performance for HPC Platforms".

Parameters:

- -a set the api to one of: POSIX, MPIIO, HDFS5 or NCMPI.
- -s number of segments (a segment is a contiguous chunk of data accessed by multiple clients each writing/reading their own contiguous data; comprised of blocks accessed by multiple tasks),
- -b block-size (contiguous bytes written per task),
- -t transfersize (size in bytes of a single data buffer to be transferred in a single I/O call)
- -i number of repetitions of the whole test.

File Size:

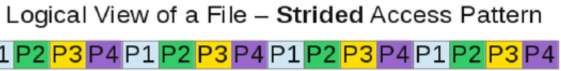
- Shared File: $\#MPI\_Procs \times s \times b$
- 1 File per Process: $s \times b$

Analyzing a strided Pattern(1)

I/O Benchmark

IOR can be used for testing performance of parallel file systems using various I/O libraries: MPI-IO, POSIX-IO, HDF5 and PnetCDF.

- MPI processes = 64, request size = 1 MiB, 16 MPI processes per Compute Node, a MPI process per core, 1GiB of data per process.
- I/O Pattern



- IOR Configuration for MPIIO

```
IOR-MPIIO-ibmmpi -a MPIIO -s 1024 -b 1m -t 1m
```

Analyzing a strided Pattern(2)

Experiments

1. Using independent I/O operations (**IOR-Ind-Strided**)
2. Using collective I/O operations by default (**IOR-Coll-Strided**)
3. Using collective I/O operations (**IOR-Coll-Strided-Hint**) by setting:

```
export ROMIO_HINTS=romio-hints
```

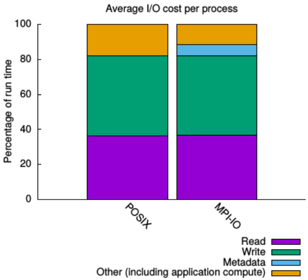
Where romio_hints contains:

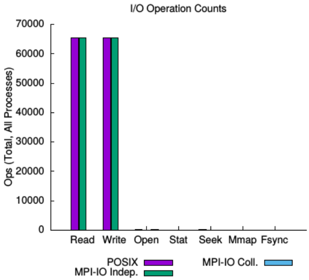
```
romio_cb_read enable
romio_cb_write enable
```

IOR-Ind-Strided MPI-IO(1)

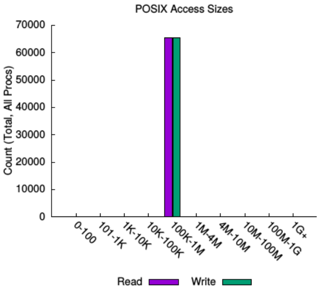
jobid: 1414202	uid: 3366230	nprocs: 64	runtime: 39 seconds

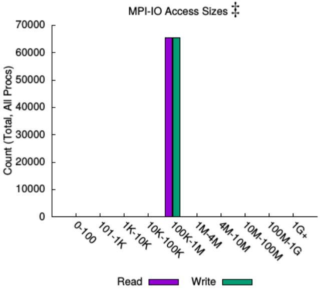
I/O performance *estimate* (at the MPI-IO layer): transferred 131072.0 MiB at 3456.22 MiB/s





IOR-Ind-Strided MPI-IO(2)





Most Common Access Sizes

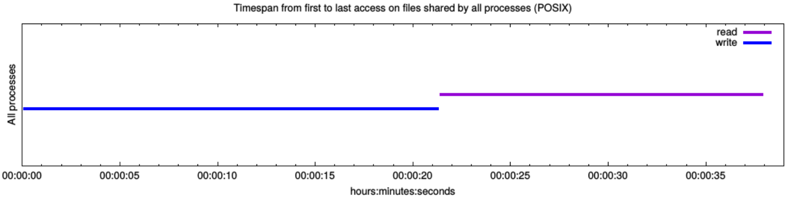
	access size	count
POSIX	1048576	131072
MPI-IO ‡	1048576	131072

‡ NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

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

type	number of files	avg. size	max size
total opened	1	64G	64G
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	64G	64G
created files	1	64G	64G

IOR-Ind-Strided MPI-IO(3)



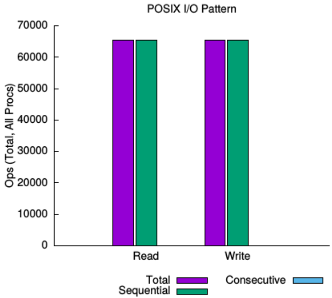
Average I/O per process (POSIX)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	0	0
Independent metadata	0	N/A
Shared reads	14.213397015625	1024
Shared writes	17.769705890625	1024
Shared metadata	0.047554546875	N/A

Data Transfer Per Filesystem (POSIX)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/gpfs	65536.00000	1.00000	65536.00000	1.00000

IOR-Ind-Strided MPI-IO(4)



sequential: An I/O op issued at an offset greater than where the previous I/O op ended.
consecutive: An I/O op issued at the offset immediately following the end of the previous I/O op.

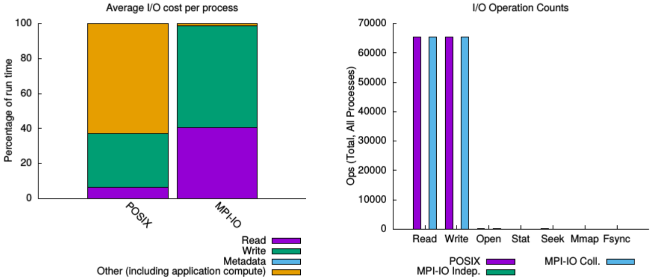
Variance in Shared Files (POSIX)

File Suffix	Processes	Fastest			Slowest			σ	
		Rank	Time	Bytes	Rank	Time	Bytes	Time	Bytes
...IIO/testFile	64	19	29.098880	2.0G	15	34.476520	2.0G	1.72	0

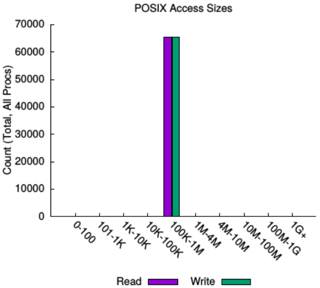
IOR-Coll-Strided MPI-IO(1)

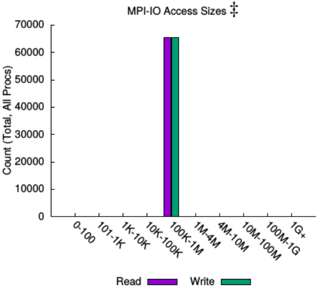
jobid: 1389652	uid: 3366230	nprocs: 64	runtime: 71 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 131072.0 MiB at 1868.17 MiB/s



IOR-Coll-Strided MPI-IO(2)





Most Common Access Sizes

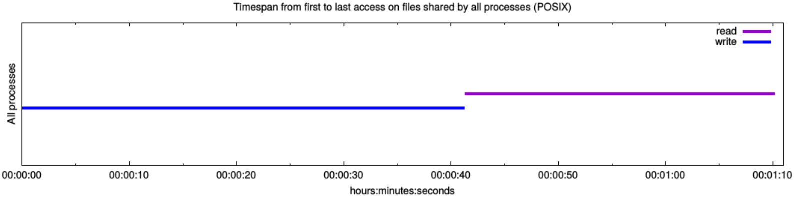
	access size	count
POSIX	1048576	131072
MPI-IO ‡	1048576	131072

‡ NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

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

type	number of files	avg. size	max size
total opened	1	64G	64G
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	64G	64G
created files	1	64G	64G

IOR-Coll-Strided MPI-IO(3)



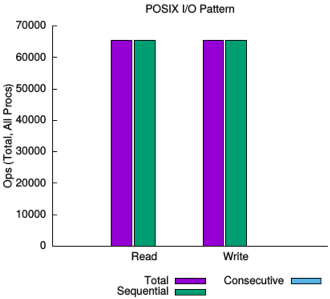
Average I/O per process (POSIX)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	0	0
Independent metadata	0	N/A
Shared reads	4.52837978125	1024
Shared writes	21.705210609375	1024
Shared metadata	0.025636546875	N/A

Data Transfer Per Filesystem (POSIX)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/gpfs	65536.00000	1.00000	65536.00000	1.00000

IOR-Coll-Strided MPI-IO(4)



sequential: An I/O op issued at an offset greater than where the previous I/O op ended.
consecutive: An I/O op issued at the offset immediately following the end of the previous I/O op.

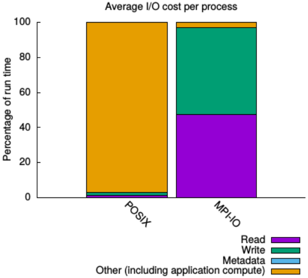
Variance in Shared Files (POSIX)

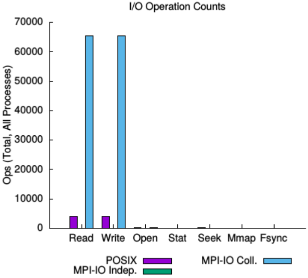
File Suffix	Processes	Fastest			Slowest			σ	
		Rank	Time	Bytes	Rank	Time	Bytes	Time	Bytes
...IIO/testFile	64	51	10.173873	2.0G	12	36.705539	2.0G	8.87	0

IOR-Coll-Strided-Hint MPI-IO(1)

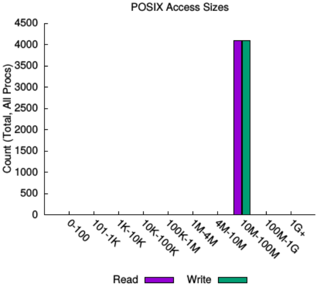
jobid: 1389655	uid: 3366230	nprocs: 64	runtime: 27 seconds

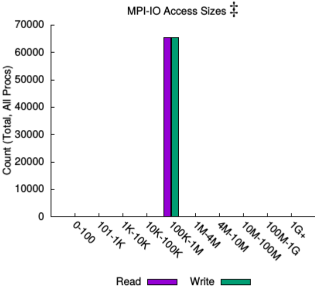
I/O performance *estimate* (at the MPI-IO layer): transferred 131072.0 MiB at 5001.21 MiB/s





IOR-Coll-Strided-Hint MPI-IO(2)





Most Common Access Sizes

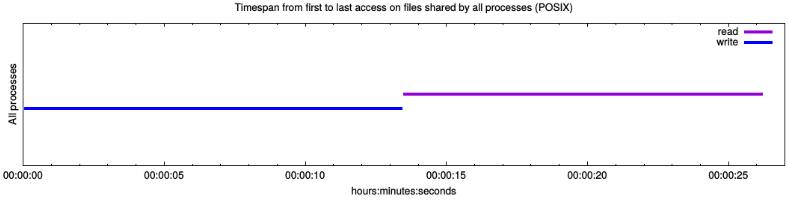
	access size	count
POSIX	16777216	8192
MPI-IO ‡	1048576	131072

‡ NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

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

type	number of files	avg. size	max size
total opened	1	64G	64G
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	64G	64G
created files	1	64G	64G

IOR-Coll-Strided-Hint MPI-IO(3)



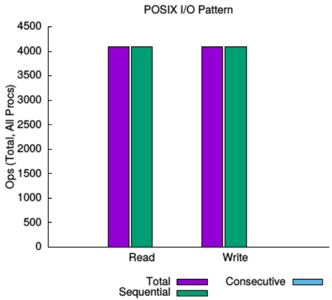
Average I/O per process (POSIX)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	0	0
Independent metadata	0	N/A
Shared reads	0.33923846875	1024
Shared writes	0.434844890625	1024
Shared metadata	0.021452546875	N/A

Data Transfer Per Filesystem (POSIX)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/gpfs	65536.00000	1.00000	65536.00000	1.00000

IOR-Coll-Strided-Hint MPI-IO(4)



sequential: An I/O op issued at an offset greater than where the previous I/O op ended.
consecutive: An I/O op issued at the offset immediately following the end of the previous I/O op.

Variance in Shared Files (POSIX)

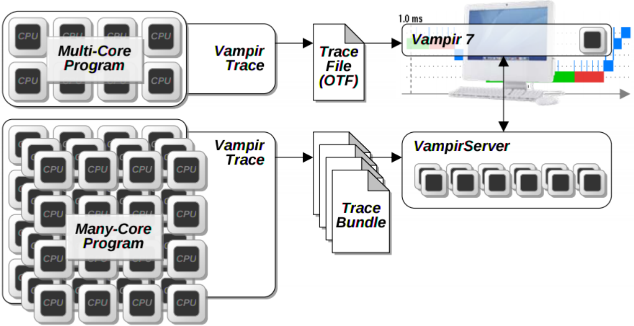
File Suffix	Processes	Fastest			Slowest			σ	
		Rank	Time	Bytes	Rank	Time	Bytes	Time	Bytes
...INT/testFile	64	9	0.009473	0	0	16.058937	32G	3.05	8.32e+09

Darshan Summary

- Instrumentation is inserted at build time (for static executables) or at run time (for dynamic executables).
- Captures POSIX I/O, MPI-IO, and limited HDF5 and PNetCDF functions.
- Provide a framework to create modules for gathering I/O data from a specific system component (which could be from an I/O library, platform-specific data, etc.)
- Minimal application impact
  - Low memory consumption,
  - Reduces, compresses, and aggregates data at MPI_Finalize() time,
  - Instrumentation enabled via software modules, environment variables, or compiler scripts,
  - No source code or makefile changes and
  - No file system dependencies.

Vampir Tool

Vampir Architecture



VampirTrace

VampirTrace consists of a tool set and a runtime library for instrumentation and tracing of software applications. It is particularly tailored to parallel and distributed High Performance Computing (HPC) applications.

Usage order of the Vampir

- Instrument your application with VampirTrace
- Run your application with an appropriate test set
- Analyze your trace file with Vampir
  - Small trace files can be analyzed on your local workstation
    1. Start your local Vampir
    2. Load trace file from your local disk
  - Large trace files should be stored on the cluster file system
    1. Start VampirServer on your analysis cluster
    2. Start your local Vampir
    3. Connect local Vampir with the VampirServer on the analysis cluster
    4. Load trace file from the cluster file system

http://www.vi-hps.org/upload/material/tw08/vi-hps-tw08-Vampir_Overview.pdf

I/O Calls

- Building VampirTrace with I/O tracing support. Recording calls to I/O functions of the standard C library.
- The following functions are intercepted by VampirTrace:

```
close creat creat64 dup dup2 fclose fcntl fdopen fgetc fgets
flockfile fopen fopen64 fprintf fputc fputs fread fscanf fseek
fseeko fseeko64 fsetpos fsetpos64 ftrylockfile funlockfile
fwrite getc gets lockf lseek lseek64 open open64 pread pread64
putc puts pwrite pwrite64 read readv rewind unlink write writev
```

- Setting the environment variable VT_IOTRACE to yes.
- Set the environment variable VT_IOLIB_PATHNAME to the alternative one.
- Environment variable VT_IOTRACE_EXTENDED.

http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/vampirtrace

How to use VampirTrace (1)?

- Instrumentation (automatic with compiler wrappers)

```
vtf77 [-g] -c <further options> myprog.f
```

Fortran 90 and higher

```
vtf90 [-g] -vt:f90 mpif90 -c <further options> myprog.f90
```

C

```
vtcc [-g] -vt:cc mpicc -c <further options> myprog.c
```

C++

```
vtcxx [-g] -c -vt:cxx mpiCC <further options> myprog.cpp
```

- Application Execution Wrapper (vtrun)

```
mpirun -np 4 vtrun ./a.out
```

How to use VampirTrace (2)?

- Environment Variables

```
export VT_PFORM_LDIR=/gpfs/work/pr28fa/di98het/vampir-tmp
export VT_FILE_UNIQUE='yes'
export VT_IOTRACE='yes'
```

- Viewing the results (serial Vampir)

  After execution of your tracing run, you will find a file filename.otf as well as a number of files *.events.z.

  For small traces:

```
vampir <filename>.otf
```

  For large traces:

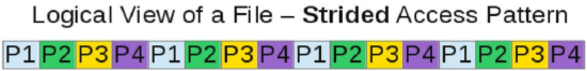
```
vampirserver start -n <tasks>
```

Analyzing a strided Pattern (1)

I/O Benchmark

IOR can be used for testing performance of parallel file systems using various I/O libraries: MPI-IO, POSIX-IO, HDF5 AND PnetCDF.

- MPI processes = 64, request size = 1 MiB, 16 MPI processes per Compute Node, a MPI process per core, 1GiB of data per process.
- I/O Pattern

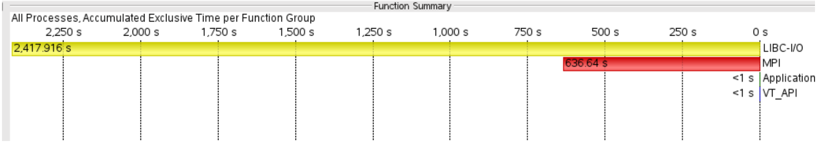


- IOR Configuration for MPIIO

```
IOR-MPIIO-ibmmpi -a MPIIO -s 1024 -b 1m -t 1m
```

IOR-Ind-Strided MPI-IO (1)

IOR-Ind-Strided MPI-IO (2)

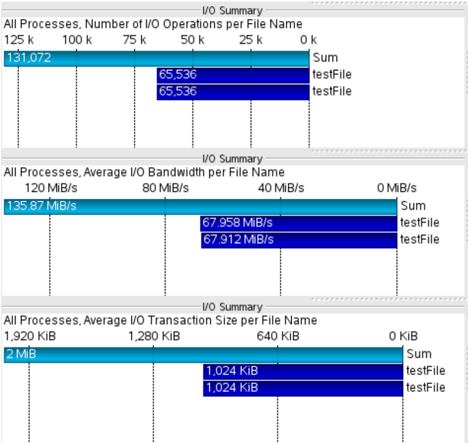


Call Tree

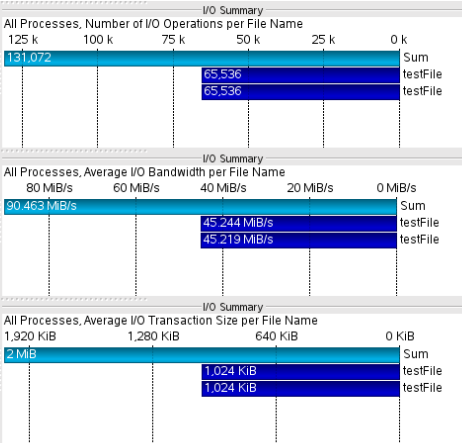
All Processes

Function	Min Number of Inv	Max Number of Inv	Min Inclusive Time	Max Inclusive T	Min Exclusive Time	Max Exclusive Time
user	1	1	47.383 s	47.867 s	8.801 ms	12.248 m
MPI_File_write_at	1,024	1,024	19.408 s	28.039 s	10.297 ms	13.556 m
pwrite64	1,024	1,024	19.396 s	28.025 s	19.396 s	28.025
MPI_File_read_at	1,024	1,024	11.713 s	18.801 s	8.605 ms	11.338 m
pread64	1,024	1,024	11.703 s	18.792 s	11.703 s	18.792
MPI_Barrier	7	7	7.040 ms	8.637 s	7.040 ms	8.637
MPI_File_open	4	4	25.793 ms	7.113 s	24.549 ms	7.100
open64	4	7	754.812 µs	14.939 ms	754.812 µs	14.939 m
close	0	2	0.000 s	26.437 µs	0.000 s	26.437 µ
read	0	1	0.000 s	564.437 ns	0.000 s	564.437 n
MPI_Init	1	1	0.446 s	0.931 s	0.423 s	0.914
MPI_File_get_size	2	2	46.303 ms	50.512 ms	12.884 µs	29.918 µ
lseek64	4	4	46.283 ms	50.495 ms	46.283 ms	50.495 m
MPI_File_delete	0	1	0.000 s	5.855 ms	0.000 s	19.066 µ
unlink	0	1	0.000 s	5.836 ms	0.000 s	5.836 m
MPI_Allreduce	6	6	361.056 µs	4.389 ms	361.056 µs	4.389 m
MPI_File_close	4	4	144.887 µs	1.701 ms	67.464 µs	159.490 µ
close	4	4	77.423 µs	1.574 ms	77.423 µs	1.574 m
MPI_Reduce	26	26	49.370 µs	979.173 µs	49.370 µs	979.173 µ
MPI_Bcast	6	6	33.365 µs	694.616 µs	33.365 µs	694.616 µ
MPI_Recv	0	126	0.000 s	520.018 µs	0.000 s	520.018 µ
MPI_Finalize	1	1	418.618 µs	438.032 µs	18.996 µs	29.641 µ
MPI_Comm_create	1	1	53.700 µs	160.573 µs	53.700 µs	160.573 µ

IOR-Ind-Strided MPI-IO (3)



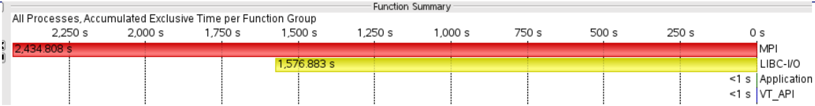
Read operations



Write Operations

IOR-Coll-Strided MPI-IO (1)

IOR-Coll-Strided MPI-IO (2)

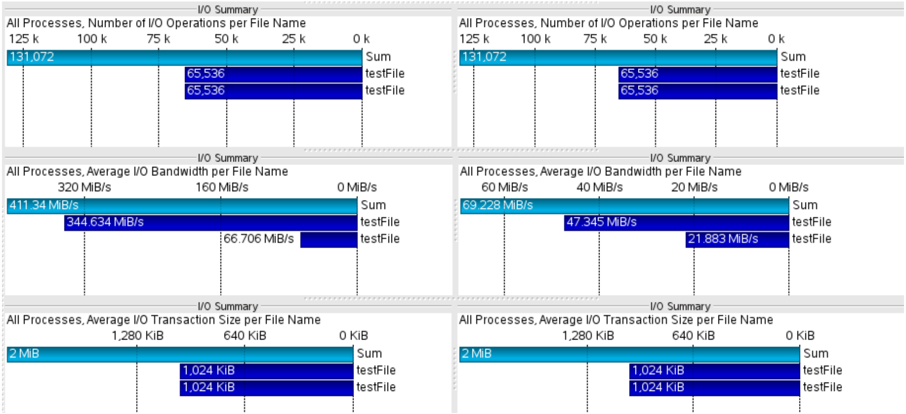


Call Tree

All Processes

Function	Min Number of Invo(	Max Number of Invo	Min Inclusive Time	Max Inclusive Time	Max Exclusive Time
user	1	1	62.610 s	62.723 s	8.029 ms
MPI_File_write_at_all	1,024	1,024	46.775 s	46.815 s	38.959 s
pwrite64	1,024	1,024	7.835 s	29.115 s	29.115 s
MPI_File_read_at_all	1,024	1,024	15.350 s	15.352 s	13.316 s
pread64	1,024	1,024	2.035 s	3.874 s	3.874 s
MPI_Init	1	1	0.391 s	0.505 s	0.481 s
MPI_Barrier	7	7	7.566 ms	48.867 ms	48.867 ms
MPI_File_open	4	4	22.761 ms	24.446 ms	23.467 ms
open64	4	7	363.637 µs	12.024 ms	12.024 ms
close	0	2	0.000 s	13.613 µs	13.613 µs
read	0	1	0.000 s	632.595 ns	632.595 ns
MPI_File_get_size	2	2	6.225 ms	11.562 ms	31.548 µs
lseek64	4	4	6.211 ms	11.550 ms	11.550 ms
MPI_File_delete	0	1	0.000 s	6.185 ms	19.702 µs
unlink	0	1	0.000 s	6.165 ms	6.165 ms
MPI_Allreduce	6	6	274.538 µs	5.644 ms	5.644 ms
MPI_Reduce	26	26	48.377 µs	1.041 ms	1.041 ms
MPI_Bcast	6	6	33.513 µs	920.286 µs	920.286 µs
MPI_File_close	4	4	124.202 µs	812.978 µs	119.296 µs
close	4	4	65.617 µs	722.969 µs	722.969 µs
MPI_Recv	0	126	0.000 s	503.195 µs	503.195 µs
MPI_Finalize	1	1	373.308 µs	398.729 µs	28.568 µs
MPI_Comm_create	1	1	52.639 µs	151.426 µs	151.426 µs
MPI_Send	0	2	0.000 s	39.714 µs	39.714 µs
MPI_Get_processor_name	2	2	6.213 µs	22.865 µs	22.865 µs
MPI_Group_range_incl	1	1	4.627 µs	11.119 µs	11.119 µs
MPI_Comm_free	1	1	5.402 µs	8.671 µs	8.671 µs
MPI_Comm_group	1	1	4.231 µs	7.554 µs	7.554 µs
fflush	0	3	0.000 s	3.167 µs	3.167 µs

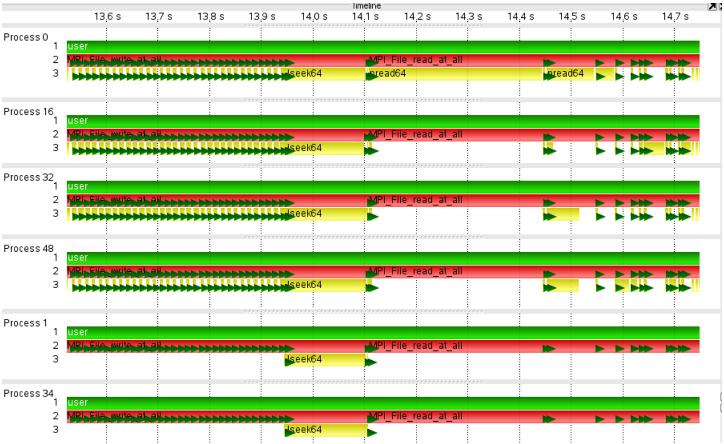
IOR-Coll-Strided MPI-IO (3)



Read operations Write Operations

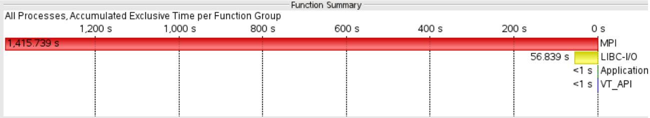
IOR-Coll-Strided-Hint MPI-IO (1)

IOR-Coll-Strided-Hint MPI-IO (2)



Zoom in 14s - Green triangles represent I/O events and yellow bars represent the I/O time.

IOR-Coll-Strided-Hint MPI-IO (3)

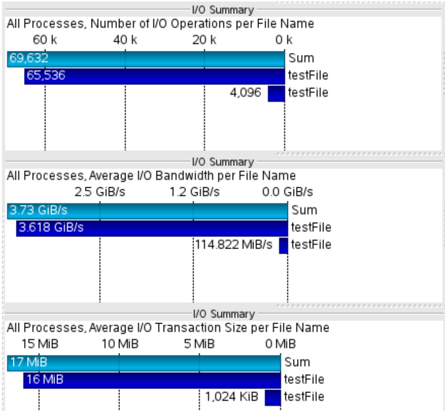


Call Tree

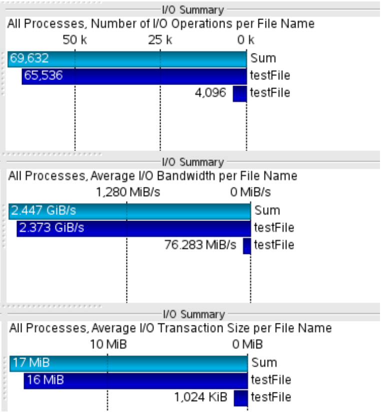
All Processes

Function	Min Number of In	Max Number of Inv	Min Inclusive Ti	Max Inclusive Ti	Max Exclusive Time
user	1	1	22.944 s	23.041 s	15.040 ms
MPI_File_write_at_all	1,024	1,024	13.421 s	13.426 s	13.426 s
pwrite64	0	1,024	0.000 s	8.205 s	8.205 s
MPI_File_read_at_all	1,024	1,024	8.913 s	8.920 s	8.920 s
pread64	0	1,024	0.000 s	4.642 s	4.642 s
MPI_Init	1	1	0.389 s	0.486 s	0.471 s
MPI_File_get_size	2	2	0.149 s	0.158 s	25.938 µs
lseek64	4	4	0.149 s	0.158 s	0.158 s
MPI_File_open	4	4	34.716 ms	37.116 ms	35.337 ms
open64	4	7	994.967 µs	21.461 ms	21.461 ms
read	0	1	0.000 s	221.539 µs	221.539 µs
close	0	3	0.000 s	22.466 µs	22.466 µs
MPI_Barrier	7	7	189.142 µs	9.490 ms	9.490 ms
MPI_Allreduce	6	6	297.113 µs	9.063 ms	9.063 ms
MPI_File_delete	0	1	0.000 s	7.696 ms	18.973 µs
unlink	0	1	0.000 s	7.677 ms	7.677 ms
MPI_File_close	4	4	127.275 µs	2.356 ms	121.384 µs
close	4	4	56.174 µs	2.270 ms	2.270 ms
MPI_Bcast	6	6	52.433 µs	663.996 µs	663.996 µs
MPI_Recv	0	126	0.000 s	472.305 µs	472.305 µs
MPI_Finalize	1	1	396.337 µs	420.071 µs	38.024 µs
MPI_Reduce	26	26	64.272 µs	385.992 µs	385.992 µs
MPI_Comm_create	1	1	54.670 µs	158.731 µs	158.731 µs
MPI_Send	0	2	0.000 s	50.346 µs	50.346 µs
MPI_Get_processor_name	2	2	6.089 µs	31.442 µs	31.442 µs
MPI_Group_range_incl	1	1	4.543 µs	29.453 µs	29.453 µs
MPI_Comm_free	1	1	5.496 µs	9.073 µs	9.073 µs
MPI_Comm_group	1	1	4.330 µs	7.699 µs	7.699 µs
MPI_Comm_size	2	2	1.508 µs	3.286 µs	3.286 µs

IOR-Coll-Strided-Hint MPI-IO (4)



Read operations



Write Operations

Comparing Traces

Vampir Summary

VampirTrace

- Convenient instrumentation and measurement infrastructure
- Hides complex details
- Highly configurable
  - Provides many options and switches for expert users
- Available under BSD-like license
  - Part of Open MPI >=1.3
  - Standalone download at www.tu-dresden.de/zih/vampirtrace

Vampir GUI and VampirServer

- Interactive trace visualization and analysis
- Intuitive browsing and zooming
- Vampir GUI available for Windows, Linux/Unix, Mac OS X
- Further information: www.vampir.eu

Thank you for your attention!