Measuring I/O with TAU

Kevin Huck

CO-PIs : Allen D. Malony & Sameer Shende

khuck@cs.uoregon.edu

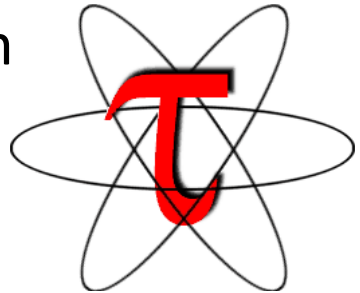
SCIDAC4 INSTITUTE
RAPIDS
RESOURCE & APPLICATION PRODUCTIVITY THROUGH COMPUTATION, INFORMATION, AND DATA SCIENCE

UNIVERSITY OF OREGON

U.S. DEPARTMENT OF ENERGY | Office of Science

TAU Performance System®

- Tuning and Analysis Utilities (20+ year effort)
- Comprehensive performance profiling and tracing
  - Integrated, scalable, flexible, portable
  - Targets all parallel programming/execution paradigms
- Integrated performance toolkit
  - Instrumentation, measurement, analysis, visualization
  - Widely ported performance profiling/tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)



I/O Measurement with TAU

- Several approaches to I/O measurement, situation dependent
- POSIX wrapper - static or dynamic wrapping of API calls
- MPI name-shifting (MPI->PMPI) interception for MPI I/O
- Sampling
- Library API wrapping : HDF5, (P)netCDF
- Permanent instrumentation : ADIOS, ADIOS2
- OS/HW monitoring - /proc/net/dev and /proc/self/io

POSIX includes counters & metadata

Broken down by filename

Name △	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
Bytes Written	400,000,000	10,000	40,000	40,000	40,000	0
Bytes Written <file=out.dat>	400,000,000	10,000	40,000	40,000	40,000	0
Write Bandwidth (MB/s)		10,000	2,222.222	816.327	1,660.024	208.113
Write Bandwidth (MB/s) <file=out.dat>		10,000	2,222.222	816.327	1,660.024	208.113
write()						
Write Bandwidth (MB/s) <file=out.dat>		10,000	2,222.222	816.327	1,660.024	208.113
Write Bandwidth (MB/s)		10,000	2,222.222	816.327	1,660.024	208.113
Bytes Written <file=out.dat>	400,000,000	10,000	40,000	40,000	40,000	0
Bytes Written	400,000,000	10,000	40,000	40,000	40,000	0

...and by calling context

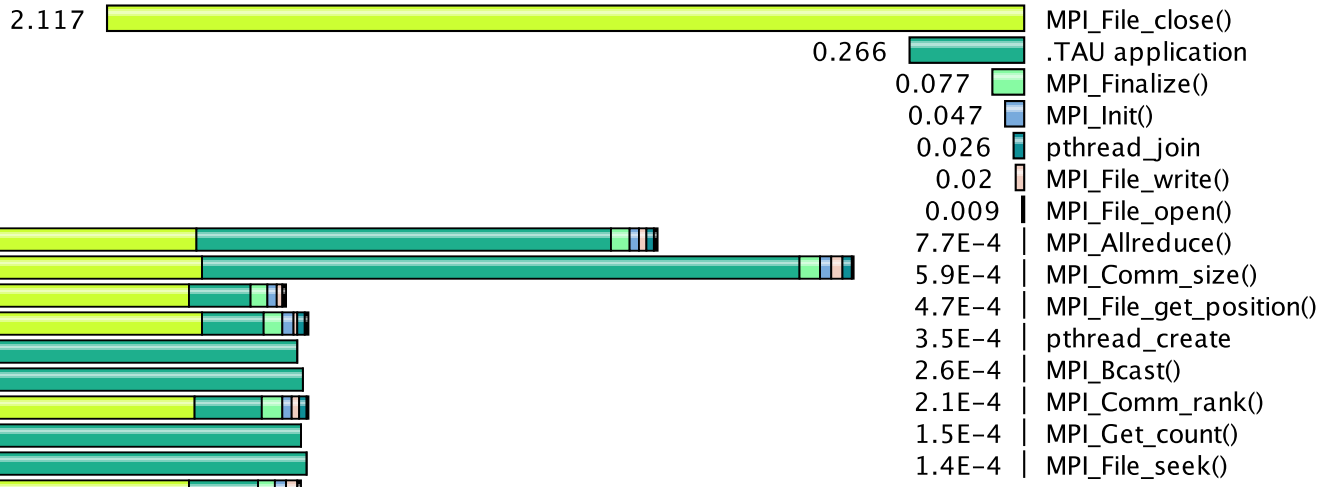
Metadata for file

Timestamp	1529437165126767
UTC Time	2018-06-19T19:39:25Z
pid	107899
posix open[0]	{"event-type":"output","name":".TAU application","time":"1529437165127306","node-id":"0","thread-id":"0","filename":"out.dat"}
tid	107899
username	khuck

MPI I/O Example

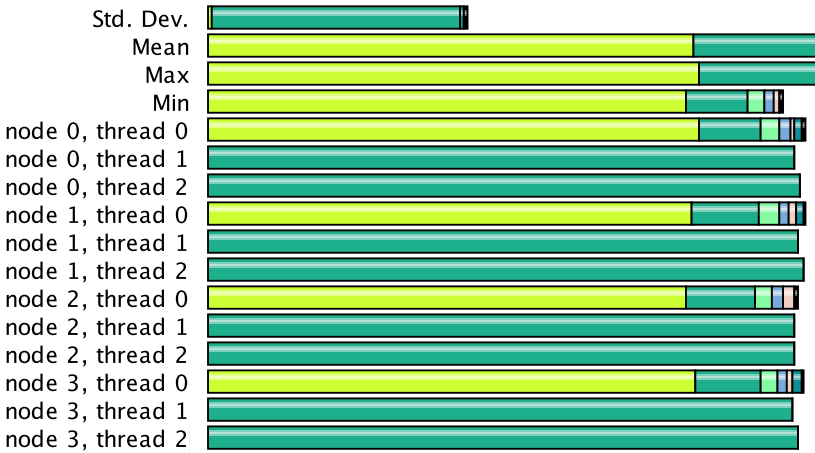
Metric: TIME
Value: Exclusive
Units: seconds

Rank 0, thread 0:



All ranks and threads:

Metric: TIME
Value: Exclusive



Library API wrapping

- Auto-instrumentation technology in TAU can be used to generate a library wrapper to measure all API calls (static or dynamic linking)

```
# Generate the PDB file from the header
$ cxxparse hdf5.h `tau_cc.sh -tau:showincludes`
$ cat select.tau
BEGIN_INCLUDE_LIST
"#H5#"
END_INCLUDE_LIST
$ tau_wrap hdf5.h.pdb hdf5.h -o wr.cpp -w libhdf5.so -r
libhdf5.so -c++ -f ./select.tau
$ cd hdf5_wrapper && make
```

Permanent Instrumentation

- ADIOS library uses asynchronous threads to perform fast, staged I/O (API thread ‘hands off’ requests to background threads)
- Wrapping the API is insufficient - only times the API call
- The ADIOS 1.13 and ADIOS2 libraries have been instrumented for use with TAU*
- Just link with TAU or run with tau_exec to instantiate symbols

** TAU-specific API will be replaced with generic version...soon*

Example: Test Proxy Kernel

- Simulation runs thousands of timesteps
- At the end of each timestep, each rank writes out hundreds/thousands of variables (2-,3-D global arrays)
- Each variable is small (10x10 block of the array)
- Data is appended to output of previous timestep
- Executed with 400 MPI ranks on Summit, 20 ranks per node
- ADIOS BP3 writes show moderate scaling issues
- Proxy application skips computation, just does I/O

ADIOS BP3: MPI and Metadata

Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child ...
.TAU application	0.001	16.09	1	1
main	2.133	[illegible]	1	82.368
Write	0	12.349	20	20
writeADIOS	0.067	[illegible]	20	20,749.4
MPI_Allreduce()	9.374	9.374	40	0
BP3Writer::EndStep	0	2.039	20	40
BP3Writer::Flush	0.001	[illegible]	20	[illegible]
BP3Writer::WriteCollectiveMetadataFile	0.001	1.454	20	[illegible]
AggregateCollectiveMetadataIndices	[illegible]	1.451	20	180.15
AggregateCollectiveMetadataIndices	0.001	6.171	20	40
AggregateCollectiveMetadataIndices	0.024	1.596	20	8,000
MPI_Gatherv()	1.394	1.396	20	38.572
AggregateCollectiveMetadataIndices	0.506	0.506	20	0
MPI_Gather()	0.026	0.026	20	5.418
operator()	0.003	0.006	20	80
MPI_Comm_rank()	0	0	80	0
MPI_Comm_size()	0	0	40	0
write()	0.586	0.586	20	0
open()	0.058	0.058	20	0
close()	0.008	0.008	20	0
MPI_Comm_rank()	0	0	20	0
BP3Writer::WriteData	0.006	0.498	20	21
BP3Writer::PerformPuts	0.053	0.087	20	20,000
MPI_Barrier()	0.608	0.608	20	0
fwrite()	0.159	0.159	51,700	[illegible]
IO::GetAvailableVariables	0.102	0.135	20	[illegible]
putADIOSArray	0.052	0.104	10,000	20,000
fillArray	0.017	0.017	10,000	0

Two hotspots

Mean profile, 400 ranks

ADIOS BP4: Much better

Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child ...
.TAU application	0	5.06	1	1
main	2.008	5.06	1	82.368
Write	0	2.302	20	20
writeADIOS	0.066	2.302	20	20,749.4
MPI_Allreduce()	1.441	1.441	40	0
MPI_Barrier()	0.344	0.344	20	0
BP4Writer::EndStep	0	0.215	20	
BP4Writer::Flush	0.001	0.114	20	
BP4Writer::WriteCollectiveMetadataFile	0.005	0.091	20	
MPI_Gatherv()	0.078	0.078	20	
write()	0.043	0.043	40	
MPI_Gather()	0.008	0.008	20	
BP4Writer::PopulateMetadataIndexFileContent	0	0	20	0
MPI_Comm_rank()	0	0	100	0
MPI_Comm_size()	0	0	40	0
BP4Writer::WriteData	0	0.022	20	20
BP4Writer::PerformPuts	0.068	0.101	20	20,000
fwrite()	0.199	0.199	51,700	0
IO::GetAvailableVariables	0.101	0.135	20	19,000
putADIOSArray	0.063	0.081	10,000	10,000
fillArray	0.016	0.016	10,000	0
fopen64()	0.004	0.004	20	0
defineADIOSArray	0.002	0.003	500	500
write()	0.001	0.001	20	0
fclose()	0.001	0.001	20	
BP4Writer::BeginStep	0	0	20	
MPI_Init()	0.233	0.373	1	1,166.248
openStream	0	0.288	1	1

Less data reduction, more efficient metadata aggregation

Mean profile, 400 ranks

Summary

- Several approaches to I/O measurement, situation dependent
- POSIX wrapper - static or dynamic wrapping of API calls
- MPI name-shifting (MPI->PMPI) interception for MPI I/O
- Sampling
- Library API wrapping : HDF5, (P)netCDF
- Permanent instrumentation : ADIOS, ADIOS2
- OS/HW monitoring - /proc/net/dev and /proc/self/io