

# Overview of Performance Tools on Titan

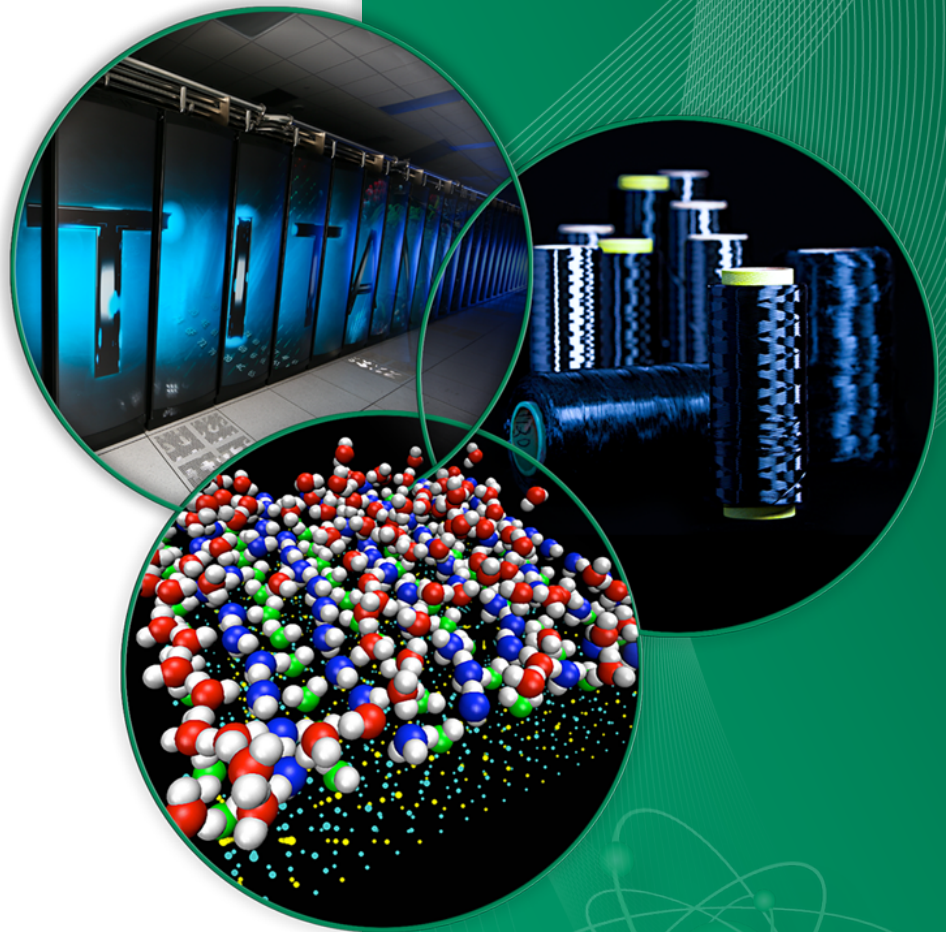
Michael J. Brim, Ph.D.

Computer Science & Mathematics Division /  
National Center for Computational Sciences

Oak Ridge National Laboratory

Argonne Training Program on  
Extreme-Scale Computing

Monday, August 5, 2013



# Talk Agenda

- Introduction
  - quick review of Titan architecture & programming models
  - performance tool taxonomy
  - the art of performance analysis
- Performance tools available on Titan
- Tips for using performance tools on Titan
  - integrating performance tools into application build/run
  - managing performance data using Spider and HPSS
- Resources for additional information

# Titan Architecture

- Cray XK7
  - 18,688+ compute nodes
    - 16-core AMD Opteron 6274 @ 2.2GHz
    - 32GB DDR3 RAM
    - NVIDIA Kepler K20 GPU: 14 SM with 6GB RAM
  - Gemini Interconnect: 3-D Torus
- Spider File System (Lustre)
  - user and project scratch space
  - only file system accessible to compute nodes
- High-Performance Storage System (HPSS)
  - long-term tape backup storage



Advancing the Era of Accelerated Computing

<http://www.olcf.ornl.gov/titan/>

# Titan Programming Models

## Types of Parallelism

- Pure MPI
- Hybrid MPI + OpenMP
- Hybrid MPI + pthreads
- All of the above + GPU
  - use CUDA Proxy to enable multiple threads/processes access to GPU
    - export CRAY\_CUDA\_PROXY=1

## GPU Programming

- High-level directives
  - PGI Accelerator
  - OpenACC
    - community standard inspired by PGI Accelerator
  - OpenHMPP (CAPS Enterprises)
    - hybrid multicore parallel processing directives
- Low-level GPU programming
  - CUDA, OpenCL for C
  - PGI CUDA Fortran

# Performance Tool Taxonomy

- Source Code Static Analysis

- automatic analysis to identify optimization opportunities (e.g., data access or comm. patterns)

- Runtime Profilers

- periodic sampling of application and/or hardware state
  - application function call stack, processor/GPU hardware counters
- statistical performance results
- low overhead given reasonable sampling period

- Runtime Tracers

- record events or time/resources used by:
  - all functions
  - library interfaces (e.g., MPI, OpenMP, CUDA)
  - user-specified functions or code regions
- temporal + spatial performance results
- higher overhead than profiling

# The Art of Performance Analysis

1. Fix all known bugs!!
2. **Generate Hypotheses**: source code analysis
  - identify potential performance bottlenecks
    - e.g., compute, memory, disk, network, load imbalance
  - static analysis tools can sometimes help
3. **Hypothesis Testing**: profiling
  - shows high-level compute/communication behavior
  - guides focus for further inspection
4. **Focused analysis**: tracing
  - instrument hot-spots, detailed communication analysis
5. Use insight to optimize source. Rinse. Repeat

# Performance Tools on Titan

	Profiling	Tracing	GPU Support	Multi-node Support
NVIDIA Profiling Tools	✓		✓	
CrayPat	✓	✓	✓	✓
TAU	✓	✓	✓	✓
Vampirtrace + Vampir		✓	✓	✓
HPCToolkit	✓		✓	✓
Open SpeedShop	✓	✓	✓	✓

Currently Available

Available Soon

# NVIDIA Profiling Tools

- Typical usage:

```
module load cudatoolkit
```

```
# use Command Line Profiler
```

```
export COMPUTE_PROFILE=1
```

```
export COMPUTE_PROFILE_CSV=1
```

```
aprun -nl <exe> <args>
```

**OR**

```
# use nvprof
```

```
aprun -nl nvprof <prof-opts> <exe> <args>
```

```
# run NVIDIA Visual Profiler
```

```
[setup X11 forwarding]
```

```
nvvp
```

```
[ File => Import Nvprof/CSV Profile ]
```

- User Manual

<http://docs.nvidia.com/cuda/profiler-users-guide/index.html>

## nvprof Event Profile: SortingNetworks (CUDA SDK)

===== Profiling result:

	Invocations	Avg	Min	Max	Event Name
Device 0					
Kernel: bitonicSortShared(unsigned int*, unsigned int*, unsigned int*, unsigned int*, unsigned int, unsigned int)					
	5	8923304	5297467	13029281	active_cycles
	5	16384	16384	16384	warps_launched
Kernel: bitonicSortShared1(unsigned int*, unsigned int*, unsigned int*, unsigned int*)					
	10	12625641	12619053	12630504	active_cycles
	10	16384	16384	16384	warps_launched
Kernel: bitonicMergeShared(unsigned int*, unsigned int*, unsigned int*, unsigned int*, unsigned int, unsigned int)					
	55	2166104	2160057	2173376	active_cycles
	55	16384	16384	16384	warps_launched
Kernel: bitonicMergeGlobal(unsigned int*, unsigned int*, unsigned int*, unsigned int*, unsigned int, unsigned int, unsigned int, unsigned int)					
	220	934841	914484	955065	active_cycles
	220	16384	16384	16384	warps_launched



# CrayPat

- Typical usage:

```
module load perftools
```

```
# build, keeping object & archive files
```

```
make clean && make
```

```
# rebuild and run with automatic analysis
```

```
pat_build -O apa <exe>
```

```
aprun <options> <exe>+pat <args>
```

```
pat_report -T -o patrpt.txt <exe>+pat*.xf
```

```
# rebuild and run with focused analysis
```

```
pat_build -O <exe>+pat*.apa
```

```
aprun <options> <exe>+apa <args>
```

```
pat_report -T -o aparpt.txt <exe>+apa*.xf
```

```
# visualize analysis data with  
Apprentice2
```

```
[ setup X11 forwarding ]
```

```
app2 <exe>+apa*.ap2
```

- User Manual

[https://www.olcf.ornl.gov/kb\\_articles/software-craypat/](https://www.olcf.ornl.gov/kb_articles/software-craypat/)

xterm

Table 1: Profile by Function Group and Function

Time%	Time	Imb. Time	Imb. Time%	Calls	Group Function PE=HIDE
100.0%	107.147732	--	--	150933.3	Total
96.8%	103.748076	--	--	57476.0	USER
55.3%	59.229094	0.182782	0.3%	1.0	main
38.3%	41.050389	0.038665	0.1%	45888.0	rmatmult3
3.2%	3.468573	0.454085	11.9%	11586.0	rbndcom
0.0%	0.000020	0.000002	8.0%	1.0	exit
2.5%	2.731736	--	--	3255.0	MPI_SYNC
1.4%	1.540552	1.379348	89.5%	347.0	MPI_Bcast(sync)
1.1%	1.190828	0.657905	55.2%	2898.0	MPI_Allreduce(sync)
0.0%	0.000189	0.000036	18.8%	6.0	MPI_Allgather(sync)
0.0%	0.000167	0.000090	53.5%	4.0	MPI_Reduce(sync)
0.6%	0.667557	--	--	89999.3	MPI
0.2%	0.263234	0.232756	48.4%	19882.5	MPI_Waitany
0.1%	0.143674	0.011333	7.5%	2898.0	MPI_Allreduce
0.1%	0.102331	0.023518	19.3%	17052.5	MPI_Isend
0.1%	0.090642	0.264080	76.8%	2902.0	MPI_Waitall
0.0%	0.043943	0.002554	5.7%	29729.0	MPI_Wtime
0.0%	0.013601	0.004758	26.8%	17052.5	MPI_Irecv
0.0%	0.005903	0.002615	31.7%	347.0	MPI_Bcast
0.0%	0.002034	0.000710	26.7%	42.8	MPI_Send
0.0%	0.001650	0.000861	35.4%	35.2	MPI_Wait
0.0%	0.000194	0.000018	8.6%	6.0	MPI_Allgather
0.0%	0.000157	0.001197	91.3%	42.8	MPI_Recv
0.0%	0.000139	0.000010	7.1%	1.0	MPI_Finalize
0.0%	0.000045	0.000041	48.9%	4.0	MPI_Reduce
0.0%	0.000004	0.000000	5.2%	1.0	MPI_Comm_group
0.0%	0.000003	0.000001	16.1%	1.0	MPI_Init
0.0%	0.000002	0.000000	9.4%	1.0	MPI_Comm_size
0.0%	0.000002	0.000000	11.5%	1.0	MPI_Comm_rank
0.0%	0.000000	0.000012	100.0%	0.0	MPI_Wtick

- Typical usage:

```
module load tau java
```

```
# use tau compiler wrappers
```

```
export CC='tau_cc.sh'
```

```
export CXX='tau_cxx.sh'
```

```
export F90='tau_f90.sh'
```

```
# build the application
```

```
make clean && make
```

```
# run instrumented executable
```

```
export TAU_TRACK_MESSAGE=1 # track MPI msgs
```

```
export TAU_COMM_MATRIX=1 # track comm ranks
```

```
jobtrace=/tmp/work/$USER/tau/$PBS_JOBID
```

```
export TAU_TRACE=1 # if tracing
```

```
export {PROFILEDIR|TRACEDIR}=$jobtrace
```

```
aprun <options> <exe> <args>
```

```
# process results
```

```
cd $jobtrace && tau_treemerge.pl
```

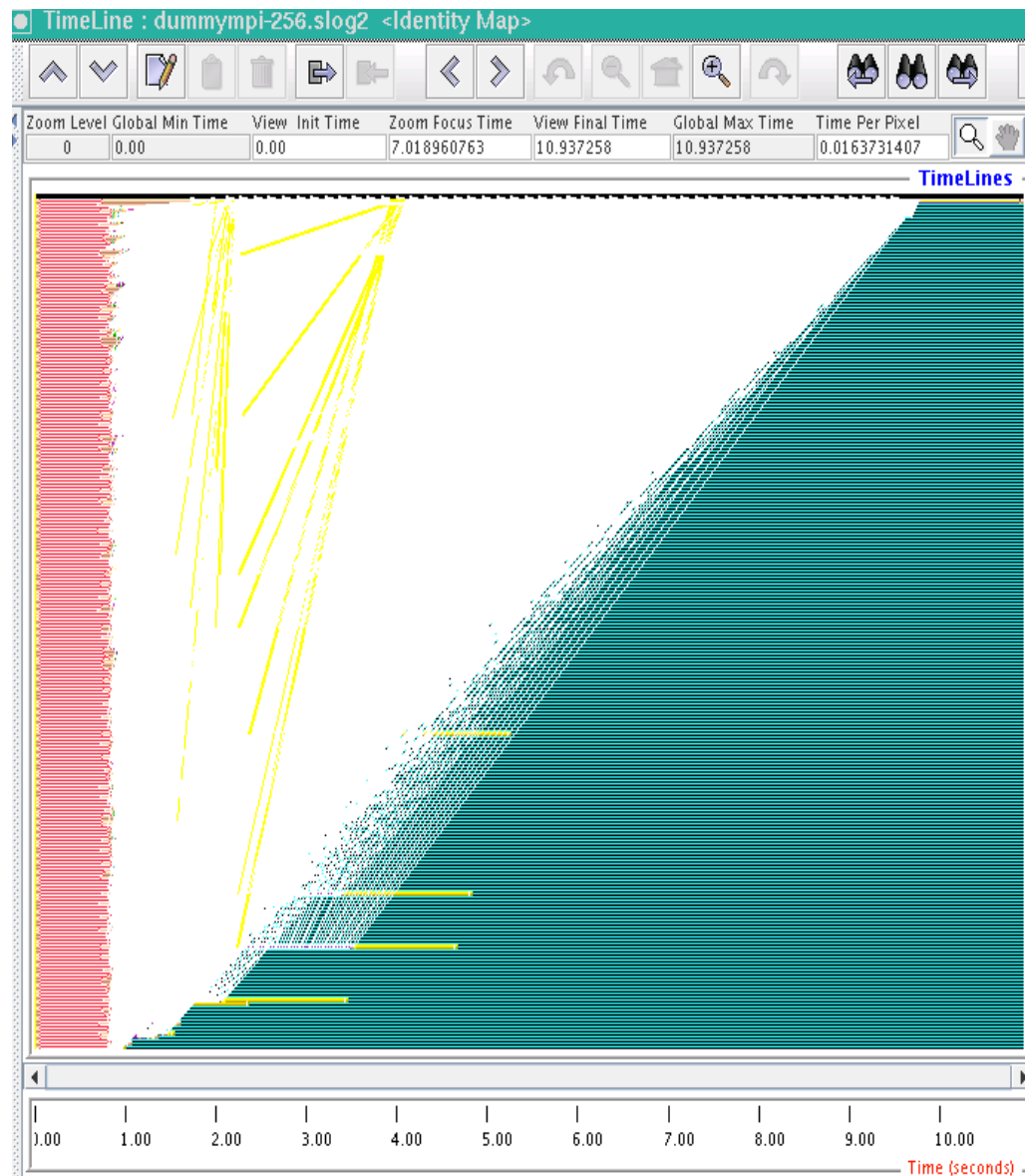
```
# view results
```

```
[ setup X11 forwarding ]
```

```
paraprof
```

- User Manual

<http://www.cs.uoregon.edu/research/tau/tau-usersguide.pdf>



# Vampirtrace + Vampir

- Typical usage:

```
module load vampirtrace
```

```
# use Vampirtrace compiler wrappers
```

```
export CC='vtcc -vt:hyb'
```

```
export CXX='vtcxx -vt:hyb'
```

```
export F90='vtf90 -vt:hyb'
```

```
# use compiler OR TAU instrumentation
```

```
export VT_INST={compinst|tauinst}
```

```
# build the application
```

```
make clean && make
```

```
# run instrumented executable
```

```
jobtrace=/tmp/work/$USER/vtrace/$PBS_JOBID
```

```
export VT_PFORM_GDIR=$jobtrace
```

```
export VT_FILE_UNIQUE=1 # prevent node collisions
```

```
aprun <options> <exe> <args>
```

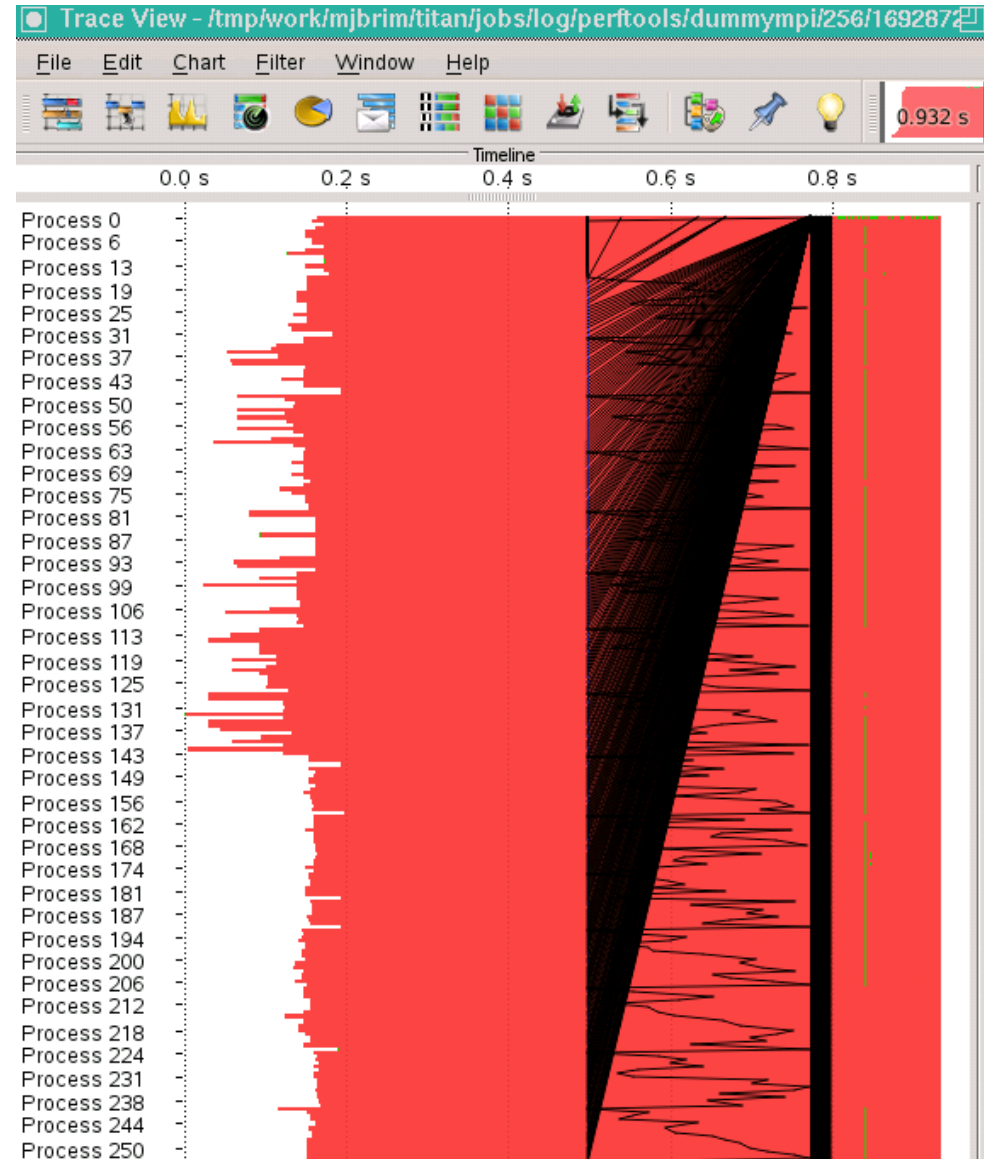
```
# view trace
```

```
[ see OLCF Vampir usage, 2nd link below ]
```

- User Manuals

[https://www.olcf.ornl.gov/kb\\_articles/software-vampirtrace/](https://www.olcf.ornl.gov/kb_articles/software-vampirtrace/)

[https://www.olcf.ornl.gov/kb\\_articles/software-vampir/](https://www.olcf.ornl.gov/kb_articles/software-vampir/)



# HPCToolkit

- Typical usage:

```
module load hpctoolkit
```

```
# build application using hpclink & hpcstruct
```

```
export LINK="hpclink cc|CC|ftn"
```

```
make clean && make
```

```
hpcstruct -I
```

```
# run instrumented application
```

```
jobtrace=/tmp/work/$USER/hpctk/$PBS_JOBID
```

```
export HPCRUN_OUT_PATH=$jobtrace
```

```
export HPCRUN_PROCESS_FRACTION="0.1" # 10%
```

```
aprun <opts> <exe> <args>
```

```
# process results
```

```
profdir="$jobtrace/results"
```

```
profopts="-M stats -S <exe>.hpcst -o $profdir"
```

```
aprun <opts> hpcprof-mpi $profopts $jobtrace
```

```
# view results
```

```
[ setup X11 forwarding ]
```

```
hpcviewer # for viewing profiles
```

```
hpctraceviewer # for viewing profiles
```

- User Manual

<http://hpctoolkit.org/manual/HPCToolkit-users-manual.pdf>

The screenshot shows the hpcviewer application interface. At the top, there's a title bar "hpcviewer: irs.PGI.hpctk\_default" and a menu bar "File View Window Help". Below the menu bar is a toolbar with icons for main.c, checkpara.c, docmd.c, run.c, xirs.c, and MatrixSolve.c. The main window displays C code with line numbers 266 to 277. Line 269 is highlighted: `for ( iblk = 0 ; iblk < my_nblk ; iblk++ ) {`. Below the code is a toolbar with icons for Calling Context View, Callers View, Flat View, and other navigation tools. A table shows the call stack with columns for Scope, PAPI\_TOT\_CYC:Sum (l), and PAPI\_. The table lists various scopes and their corresponding cycle counts and percentages.

Scope	PAPI_TOT_CYC:Sum (l)	PAPI_
435: run	5.35e+11 68.4%	
loop at run.c: 32	5.35e+11 68.4%	
inlined from run.c: 14	5.35e+11 68.4%	
17: xirs	5.35e+11 68.4%	
loop at xirs.c: 59	5.35e+11 68.4%	
loop at xirs.c: 142	5.34e+11 68.3%	
167: tdiff	2.72e+11 34.8%	
74: rdiff	2.66e+11 34.1%	
58: MatrixSolveDriver	2.57e+11 32.9%	
75: MatrixSolveCG	2.57e+11 32.8%	
loop at MatrixSolve.c: 210	2.56e+11 32.7%	
loop at MatrixSolve.c: 269	7.87e+10 10.1%	
loop at MatrixSolve.c: 218	7.78e+10 9.9%	

At the bottom right, there's a status bar showing "43M of 89M" and a trash icon.

# Open|SpeedShop

- Typical usage:

```
module load openspeedshop
```

```
# build application using osslink
```

```
collector="pcsamp"
```

```
export LINK="osslink -c $collector cc"
```

```
make clean && make
```

```
# run instrumented application
```

```
jobtrace=/tmp/work/$USER/openss/$PBS_JOBID
```

```
export OPENSs_RAWDATA_DIR=$jobtrace
```

```
aprun <opts> <exe> <args>
```

```
# process results with ossutil
```

```
ossutil $jobtrace
```

```
mv X.0.openss $jobtrace/expdb.openss
```

```
# view results
```

```
[ setup X11 forwarding ]
```

```
openss
```

- User Manual

<http://www.openspeedshop.org/wp/documentation/>

The screenshot displays the Open|SpeedShop application window. At the top is a teal title bar with the application name. Below it is a menu bar with 'File', 'Tools', and 'Help'. The main interface is divided into several panels. On the left, there's a 'Process Control' section with buttons for 'Run', 'Cont', 'Pause', and 'Update'. Below this is a 'Status' label followed by the text 'Process Loaded: Click on the "Run" button to begin the experiment.' To the right of the 'Process Control' section are two tabs: 'Stats Panel [1]' and 'ManageProcessesPanel [1]'. Below these tabs is a row of colorful icons representing different system components. To the right of the icons is the text 'Showing Hot Callpath Report:'. Below this is a section labeled 'Executables:' followed by the path '/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss\_hwtime' and 'Hosts:(32)'. The bottom half of the window is occupied by a table with four columns: 'Exclusive', 'Inclusive PAPI\_TOT\_CYC', '% of Total', and 'Call Stack Function (defining location)'. The table contains several rows of data, with the last row showing a value of 5.410929 in the '% of Total' column.

Exclusive	Inclusive PAPI_TOT_CYC	% of Total	Call Stack Function (defining location)
929340000000			@ 624 in meshgen_apply1d (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
929340000000		6.048926	@ 1410 in meshgen_pack (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			_start (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 226 in __libc_start_main (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 517 in monitor_main (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 435 in main (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 17 in run (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 167 in xirs (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 77 in tdiff (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 58 in rdiff (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 75 in MatrixSolveDriver (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 269 in MatrixSolveCG (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000			@ 14 in rmatmult (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)
831320000000		5.410929	@ 75 in rmatmult3 (/lustre/widow2/scratch/mjbrim/titan/apps/irs/bin/irs.PGI.oss_hwtime)

# Integrating Performance Tools

- Four common flavors of integration
  - source instrumentation: CrayPAT, TAU, Vampirtrace
  - compiler wrapping: TAU, Vampirtrace
  - linker wrapping: CrayPat, HPCToolkit, OpenSpeedShop
  - execution wrapping: NVIDIA, TAU, HPCToolkit, OpenSpeedShop
- Compiler wrapping
  - `$(TOOL_CC) $(TOOL_CC_OPTS) $(CC) $(CFLAGS) ...`
- Linker wrapping
  - `$(TOOL_LINK) $(TOOL_LINK_OPTS) $(LINK) $(LDFLAGS) ...`
- Execution wrapping
  - `aprun <opts> $TOOL_RUN $TOOL_RUN_OPTS <exe> <args>`



# Tool Integration - Makefile Tips

```
SRCS = module1.f90 module2.f90 app.f90
OBJS = $(addprefix $(OBJDIR)/, $(patsubst %.f90,%.o,$(SRCS)))
TAU_OBJS = $(addprefix $(OBJDIR)/tau/, $(patsubst %.f90,%.o,$(SRCS)))
```

Tip: use explicit objects to support tools that require compiler-wrapping

```
FFLAGS = -g -O2
```

Tip: add debug info for tools, but keep optimization. “-g -O2” for GCC or Intel, “-gopt” for PGI, “-G2” for Cray

```
default: $(BINDIR)/app.exe
hpctk: $(BINDIR)/app.exe.hpctk
tau: $(BINDIR)/app.exe.tau
tools: hpctk tau
```

Tip: create targets for tool executables

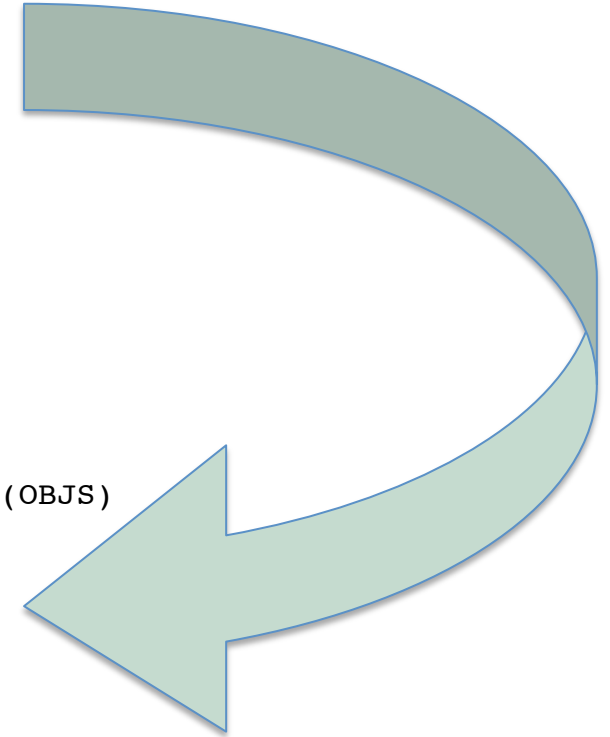
```
$(OBJDIR)/%.o : %.f90
    $(FTN) $(FFLAGS) -o $@ -c $<
$(BINDIR)/app.exe : $(OBJS)
    $(LINK) $(LDFLAGS) -o $@ $(OBJS)
```

## # HPCToolkit: linker-wrapping + post-link analysis

```
$(BINDIR)/app.exe.hpctk : $(OBJS)
    hpclink $(HPCTK_OPTIONS) $(LINK) $(LDFLAGS) -o $@ $(OBJS)
    hpcstruct --compact -I $(PWD) -o $@.hpcstruct $@
```

## # TAU: compiler-wrapping

```
$(OBJDIR)/tau/%.o : %.f90
    tau_f90.sh $(TAU_OPTIONS) $(FFLAGS) -o $@ -c $<
$(BINDIR)/app.exe.tau : $(TAU_OBJS)
    tau_f90.sh $(TAU_OPTIONS) $(LDFLAGS) -o $@ $(TAU_OBJS)
```



# Performance Data Storage on Titan

- Runtime profiles and traces must be stored on Lustre
  - `/tmp/work/$USER`
  - no storage quota, but 14-day purge in effect
- `$HOME` has 10GB quota
  - performance data for large app runs can easily exceed that amount
  - best option is to backup to HPSS
- HPSS has 2TB (or 2,000 files) quota
  - only accessible using `hsi` and `htar` commands
  - [https://www.olcf.ornl.gov/kb\\_articles/transferring-data-with-hsi-and-htar/](https://www.olcf.ornl.gov/kb_articles/transferring-data-with-hsi-and-htar/)
- Use [dtn.ccs.ornl.gov](https://dtn.ccs.ornl.gov) for remote data transfers
  - [https://www.olcf.ornl.gov/kb\\_articles/employing-data-transfer-nodes/](https://www.olcf.ornl.gov/kb_articles/employing-data-transfer-nodes/)



# Tips for Managing Performance Data

- Create a new Lustre data directory as part of every job

```
datadir=/tmp/work/$USER/<app>/<tool>/$PBS_JOBID
```

```
mkdir -p $datadir
```

```
# tools often write one file per process, update Lustre striping
```

```
lfs setstripe -c 1 $datadir
```

- Add backup to the end of job scripts

```
hpssdir="<app>/<tool>"
```

```
print "Archiving job data in $hpssdir"
```

```
cd /tmp/work/$USER/<app>/<tool>
```

```
htar -P -cf ${hpssdir}/${PBS_JOBID}.tar $PBS_JOBID/
```

# Resources for additional information

- Titan User Guide
  - <https://www.olcf.ornl.gov/support/system-user-guides/titan-user-guide/>
- Titan Debugging and Performance Tools
  - <https://www.olcf.ornl.gov/support/system-user-guides/titan-user-guide/#379>
- Info on Cray XK7 Performance Counters
  - GPU: `module load perftools ; man accpc ; man accpc_k20`
  - CPU: `ssh titan-login1` and then `module load papi ; papi_avail`
- OLCF People
  - INCITE project liaisons provide application-specific guidance on algorithms, porting, and optimization
  - For software/system troubleshooting, contact OLCF User Assistance
    - email: `help (at) olcf.ornl.gov`
    - phone: +1 865 241-6536
  - For specific tools advice, feel free to contact me: `brimmj (at) ornl.gov`
- How to setup X11 forwarding
  - [https://www.olcf.ornl.gov/kb\\_articles/x11-forwarding/?nccssystems=Connect%20&%20Login](https://www.olcf.ornl.gov/kb_articles/x11-forwarding/?nccssystems=Connect%20&%20Login)
  - or, ask me about setting up VNC (better than X11 forwarding, surprisingly fast on Internet2)

# Questions & Feedback



[www.ornl.gov](http://www.ornl.gov)