

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications The 11<sup>th</sup> DOE ACTS Workshop Berkeley, CA – Aug 20, 2010

Sameer Shende Performance Research Lab University of Oregon http://tau.uoregon.edu







Acknowledgments: PRL, UO



- Wyatt Spear
- Dr. Chee Wai Lee
- Scott Biersdorff
- Alan Morris
- Suzanne Millstein
- Dr. Rob Yelle
- Carl Woeck
- William Voorhess

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## TAU Performance System<sup>®</sup>

- <u>*T*</u>uning and <u>*A*</u>nalysis <u>*U*</u>tilities (15+ year project)
- Performance problem solving framework for HPC
  - Integrated, scalable, flexible, portable
  - Target all parallel programming / execution paradigms
- Integrated performance toolkit (open source)
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
- Broad application use (NSF, DOE, DOD, ...)

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## **TAU Performance System**

- <u>http://tau.uoregon.edu/</u>
- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid
- Integration in complex software, systems, applications

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







## What is TAU?

- TAU is a performance evaluation tool
- It supports parallel profiling and tracing
- Profiling shows you how much (total) time was spent in each routine
- Tracing shows you *when* the events take place in each process along a timeline
- TAU uses a package called PDT for automatic instrumentation of the source code
- Profiling and tracing can measure time as well as hardware performance counters from your CPU
- TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
  - paraprof is TAU's 3D profile browser
- To use TAU's automatic source instrumentation, you need to set a couple of environment variables and substitute the name of your compiler with a TAU shell script

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







## **Using TAU: A brief Introduction**

- TAU supports several measurement options (profiling, tracing, profiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique stub makefile and library that is generated when you configure it
- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:
  - % setenv TAU\_MAKEFILE \$TAU/Makefile.tau-mpi-pdt
  - % setenv TAU\_OPTIONS '-optVerbose ...' (see tau\_compiler.sh -help)
  - And use tau\_f90.sh, tau\_cxx.sh or tau\_cc.sh as Fortran, C++ or C compilers:

% mpif90 foo.f90

changes to

% tau\_f90.sh foo.f90

- Execute application and analyze performance data:
  - % pprof (for text based profile display)
  - % paraprof (for GUI)

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications







## **Direct Observation: Events**

- Event types
  - Interval events (begin/end events)
    - measures performance between begin and end
    - metrics monotonically increase
  - Atomic events
    - used to capture performance data state
- Code events
  - Routines, classes, templates
  - Statement-level blocks, loops
- User-defined events
  - Specified by the user
- Abstract mapping events

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





### **Inclusive and Exclusive Profiles**

- Performance with respect to code regions
- Exclusive measurements for region only
- Inclusive measurements includes child regions



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## Interval Events, Atomic Events in TAU

NODE 0;CON	ITEXT 0;THRE	AD 0:		X xterm				
%Time E	xclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name	-	
100.0 93.2 5.9 4.6 1.2 0.8 0.0 0.0 0.0 0.0	$\begin{array}{c} 0.187\\ 1.030\\ 0.879\\ 51\\ 13\\ 9\\ 0.137\\ 0.086\\ 0.002\\ 0.001\\ \end{array}$	$ \begin{array}{r} 1,105\\ 1,030\\ 65\\ 51\\ 13\\ 9\\ 0.137\\ 0.086\\ 0.002\\ 0.001\\ \end{array} $	1 40 40 120 1 120 40 1 1	44 0 320 0 0 0 0 0 0 0 0	1105659 1030654 1637 1277 111 9328 1 2 2 1	<pre>int main(int, char * MPI_Init() void func(int, int) MPI_Barrier() MPI_Recv() MPI_Finalize() MPI_Send() MPI_Bcast() MPI_Comm_size() MPI_Comm_rank()</pre>	- **) C C	Interval event e.g., routines (start/stop)
USER EVENT	S Profile :							
NumSamples	a MaxValue	e MinValue	MeanValue	Std. Dev.	Event Name		-	Atomic events
365 365 40 ⊟	5.138E+04 5.138E+04 40	44.39 2064 40	3.09E+04 3.115E+04 40	1.234E+04 1.21E+04 0	Heap Memorı Heap Memorı Message si:	y Used (KB) : Entry y Used (KB) : Exit ze for broadcast <b>&amp;</b> 27,1	- - 1% <u>*</u>	(trigger with value)

0

1

#### % setenv TAU\_CALLPATH\_DEPTH % setenv TAU\_TRACK\_HEAP

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### Atomic Events, Context Events







## Context Events (default)

NODE 0	CONTEXT 0;THR	EAD 0:				X xterm				
%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive usec/call	Name				
100.0 92.6 6.7 0.1 0.1 0.0 0.0 0.0 0.0	0.357 1.031 72 8 1 0.608 0.136 0.095 0.001 0	1.114 1.031 74 8 1 0.608 0.136 0.095 0.001 0	1 1 40 1 120 40 120 40 1 1 1	44 0 320 0 0 0 0 0 0 0 0	1114040 1031066 1885 8002 12 15 15 2 1 0	<pre>int main(int, MPI_Init() void func(int, MPI_Finalize() MPI_Barrier() MPI_Barrier() MPI_Scast() MPI_Comm_size() MPI_Comm_rank()</pre>	<pre>char **: , int) C ) () ()</pre>	) C		
USER EVE	NTS Profile	NODE 0, CON	TEXT 0, THRE	AD 0						
1 1 1 1 2	365       5.139E+0         1       44.3         1       206         1       206         1       5.139E+0         1       5.139E+0         40       5.139E+0         40       5.139E+0         40       5.139E+0         20       5.139E+0         365       5.139E+0	4 44.39 4 44.39 3 2068 5 2066 4 5.139E+04 3 57.58 4 2069 4 3098 4 1.13E+04 4 1.13E+04 4 1.13E+04 4 2065	3.091E+04 44.39 2068 2066 5.139E+04 57.58 3.011E+04 3.134E+04 3.134E+04 3.134E+04 3.134E+04 3.134E+04	1.234E+04 0 0 0 1.228E+04 1.227E+04 1.187E+04 1.187E+04 1.187E+04 1.21E+04	Heap Memory Heap Memory	Used (KB) : E Used (KB) : E	Entry : : Entry : :	<pre>int main(int, char **) C int main(int, char **) C =&gt; MPI_Comm_rank() int main(int, char **) C =&gt; MPI_Comm_size() int main(int, char **) C =&gt; MPI_Finalize() int main(int, char **) C =&gt; MPI_Init() int main(int, char **) C =&gt; void func(int, void func(int, int) C =&gt; MPI_Barrier() void func(int, int) C =&gt; MPI_Bcast() void func(int, int) C =&gt; MPI_Recv() void func(int, int) C =&gt; MPI_Send() 3,7</pre>	int) C	Context event = atomic event
	% seter	v TAU	_CALL	PATH_	_DEPTI	H	2			+ executing
	% seter	v TAU	_TRAC	CK_HE	AP		1			context
			oftware							11th DOE ACTS Workshop
						Applicatio				
			ACTS	2					VERSITY	OF OREGON

COLLECTION

## A New Approach: tau\_exec

- Runtime instrumentation by pre-loading the measurement library
- Works on dynamic executables (default under Linux)
- Substitutes I/O, MPI and memory allocation/ deallocation routines with instrumented calls
- Track interval events (e.g., time spent in write()) as well as atomic events (e.g., how much memory was allocated) in wrappers
- Accurately measure I/O and memory usage

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







### Issues

- Heap memory usage reported by the mallinfo() call is not 64-bit clean.
  - 32 bit counters in Linux roll over when > 4GB memory is used
  - We keep track of heap memory usage in 64 bit counters inside TAU
- Compensation of perturbation introduced by tool
  - Only show what application uses
  - Create guards for TAU calls to not track I/O and memory allocations/ de-allocations performed inside TAU
- Provide broad POSIX I/O and memory coverage

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## tau\_exec: Usage

♥////////	sameer on l2: /usr/cta/pet/pkgs/tau2							
<u>F</u> ile <u>E</u> dit <u>V</u>	ew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp							
> tau_	exec	<b>^</b>						
Usage:	tau_exec [options] [] <exe> <exe options=""></exe></exe>							
Options	:							
	-v verbose mode							
	-qsub Use qsub mode (see below)							
	-io track I/O							
	-memory track memory							
	-T <disable,icpc,mpi,pdt,profile,pthread,python,serial> : specify TAU option</disable,icpc,mpi,pdt,profile,pthread,python,serial>							
	-XrunTAUsh- <options> : specify TAU library directly</options>							
Notes:								
Defaults if unspecified: -T MPI								
MPI is assumed unless SERIAL is specified								
Example:								
mpirun -np 2 tau_exec -io ./ring								
	,							
qsub mode:								
Original:								
(	qsub -n 1mode smp -t 10 ./a.out							
WI	n IAU:							
	au_exec -qsub -10 -memory qsub -n 1mode smp -t 10 ./a.c	Jut						
_ > ■								
4		▼						





### tau\_exec

- Uninstrumented execution
  - % mpirun –np 256 ./a.out
- Track MPI Performance
  - % mpirun –np 256 tau\_exec ./a.out
- Track I/O Performance (MPI enabled by default)
  - % mpirun –np 256 tau\_exec –io ./a.out
- Track Memory
  - % setenv TAU\_TRACK\_MEMORY\_LEAKS 1
  - % mpirun –np 256 tau\_exec –memory ./a.out
- Track I/O and Memory
  - % mpirun –np 256 tau\_exec –io –memory ./a.out

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





#### tau\_exec: A tool to simplify Memory, I/O evaluation

#### 0 0

X xterm

```
> cd ~/workshop-point/matmult
> mpif90 matmult.f90 -o matmult
> mpirun -np 4 ./matmult
> # To use tau_exec to measure the I/O and memory usage:
> mpirun -np 4 tau_exec -io -memory ./matmult
> # To measure memory leaks and get complete callpaths
> setenv TAU TRACK MEMORY LEAKS 1
> setenv TAU_CALLPATH_DEPTH 100
> mpirun -np 4 tau_exec -io -memory ./matmult
> paraprof
> # Right click on a given rank (e.g, "node 2") and choose "Show Context Event
> # Window" and expand the ".TAU Application" node to see the callpath
> # To use a different configuration (e.g., Makefile.tau-papi-mpi-pdt)
> setenv TAU_METRICS TIME:PAPI_FP_INS:PAPI_L1_DCM
> mpirun -np 4 tau_exec -io -memory -T papi,mpi,pdt ./matmult
> # Using tau_exec with DyninstAPI:
> tau_run matmult -o matmult.i
> mpirun -np 4 tau_exec -io -memory ./matmult.i
> tau_run -XrunTAUsh-papi-mpi-pdt matmult -o matmult.i
> mpirun -np 4 tau_exec -io -memory -T papi,mpi,pdt ./matmult.i
> paraprof
                                                                                h.
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## **Environment Variables in TAU**

Environment Variable	Default	Description				
TAU_TRACE	0	Setting to 1 turns on tracing				
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling				
TAU_TRACK_MEMORY_LEAKS	0	Setting to 1 turns on leak detection				
TAU_TRACK_HEAP or TAU_TRACK_HEADROOM	0	Setting to 1 turns on tracking heap memory/headroom at routine entry & exit using context events (e.g., Heap at Entry: main=>foo=>bar)				
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)				
TAU_SYNCHRONIZE_CLOCKS	1	Synchronize clocks across nodes to correct timestamps in traces				
TAU_COMM_MATRIX	0	Setting to 1 generates communication matrix display using context events				
TAU_THROTTLE	1	Setting to 0 turns off throttling. Enabled by default to remove instrumentation in lightweight routines that are called frequently				
TAU_THROTTLE_NUMCALLS	100000	Specifies the number of calls before testing for throttling				
TAU_THROTTLE_PERCALL	10	Specifies value in microseconds. Throttle a routine if it is called over 100000 times and takes less than 10 usec of inclusive time per call				
TAU_COMPENSATE	0	Setting to 1 enables runtime compensation of instrumentation overhead				
TAU_PROFILE_FORMAT	Profile	Setting to "merged" generates a single file. "snapshot" generates xml format				
TAU_METRICS	TIME	Setting to a comma separted list generates other metrics. (e.g., TIME:linuxtimers:PAPI FP OPS:PAPI NATIVE <event>)</event>				

High Performance Software Tools to Fast-Track

11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



**O** UNIVERSITY OF OREGON

## Memory Leaks in MPI

\varTheta 🔿 🔿 TAU: ParaProf: Context E	vents for thread: n,c	,t, 0,0,0 - samarc_	obe_4p_iomem_cp.pp	ok		
Name 🛆	Total	MeanValue	NumSamples	MaxValue	MinValue	Std. Dev.
<ul> <li>.TAU application</li> </ul>						
MPI_Finalize()						
free size	23,901,253	22,719.822	1,052	2,099,200	2	186,920.9
malloc size	5,013,902	65,972.395	76	5,000,000	2	569,732.8
MEMORY LEAK!	5,000,264	500,026.4	10	5,000,000	3	1,499,991
▼ read()	8					
Bytes Read	4	4	1	4	4	
READ Bandwidth (MB/s) <file="pipe"></file="pipe">		0.308	1	0.308	0.308	
Bytes Read <file="pipe"></file="pipe">	4	4	1	4	4	
READ Bandwidth (MB/s)		0.308	1	0.308	0.308	
▼ write()						
WRITE Bandwidth (MB/s)		0.635	102	12	0	1.4
Bytes Written <file=" dev="" infiniband="" rdma_cm"=""></file=">	24	24	1	24	24	
Bytes Written	1,456	14.275	102	28	4	5.1
WRITE Bandwidth (MB/s) <file=" dev="" infiniband="" uverbs0"=""></file=">		0.528	97	12	0.089	1
Bytes Written <file="pipe"></file="pipe">	64	16	4	28	4	
WRITE Bandwidth (MB/s) <file=" dev="" infiniband="" rdma_cm"=""></file=">		1.714	1	1.714	1.714	
Bytes Written <file=" dev="" infiniband="" uverbs0"=""></file=">	1,368	14.103	97	24	12	4.
WRITE Bandwidth (MB/s) <file="pipe"></file="pipe">		2.967	4	5.6	0	2.0
▼ writev()						
WRITE Bandwidth (MB/s)		4.108	2	7.667	0.549	3.
Bytes Written	297	148.5	2	230	67	8
WRITE Bandwidth (MB/s) <file="socket"></file="socket">		4.108	2	7.667	0.549	3.
Bytes Written <file="socket"></file="socket">	297	148.5	2	230	67	8
▼ readv()						
Bytes Read	112	28	4	36	20	
READ Bandwidth (MB/s) <file="socket"></file="socket">		25.5	4	36	10	11.0
Bytes Read <file="socket"></file="socket">	112	28	4	36	20	
READ Bandwidth (MB/s)		25.5	4	36	10	11.
MPI Comm free()						
free size	10,952	195.571	56	1,024	48	255.3
► read()						
► MPI Type free()						
► MPI Init()						
v fopen64()						
free size	231,314	263,456	878	568	35	221.3
MEMORY LEAK!	1,105.956	1,868.169	592	7.200	32	3.078.
malloc size	1,358,286	901.318	1,507	7,200	32	2,087.7
► OurMain()					1770.	-1-3111
▶ fclose()						



**O** UNIVERSITY OF OREGON

### Instrumentation Issues

- Dynamic Instrumentation using DyninstAPI [U. Wisconsin, Madison, and U. Maryland]
- Pre-execution instrumentation
- Shell script spawned the task on the node and instrumented it
- As the number of processors increased, more time was wasted:
  - transferring un-instrumented executables to the compute nodes,
  - Instrumenting the application binary
- Solution: Binary re-writing!

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### **Re-writing Binaries**

- Support for both static and dynamic executables
- Specify the list of routines to instrument/exclude from instrumentation
- Specify the TAU measurement library to be injected
- Simplify the usage of TAU:
  - To instrument:
    - % tau\_run a.out –o a.inst
  - To perform measurements, execute the application:
    - % mpirun –np 4 ./a.inst
  - To analyze the data:
    - % paraprof

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### Using TAU with DyninstAPI : tau\_run

X livetau@paratools01:~ /home/livetau% cd ~/tutorial /home/livetau/tutorial% # Build an uninstrumented bt NAS Parallel Benchmark /home/livetau/tutorial% make bt CLASS=W NPROCS=4 /home/livetau/tutorial% cd bin /home/livetau/tutorial/bin% # Run the instrumented code /home/livetau/tutorial/bin% mpirun -np 4 ./bt\_W.4 /home/livetau/tutorial/bin% /home/livetau/tutorial/bin% # Instrument the executable using TAU with DyninstAPI /home/livetau/tutorial/bin% /home/livetau/tutorial/bin% tau\_run ./bt\_W.4 -o ./bt.i /home/livetau/tutorial/bin% rm -rf profile.\* MULT\* /home/livetau/tutorial/bin% mpirun -np 4 ./bt.i /home/livetau/tutorial/bin% paraprof /home/livetau/tutorial/bin% /home/livetau/tutorial/bin% # Choose a different TAU configuration /home/livetau/tutorial/bin% ls \$TAU/libTAUsh libTAUsh-depthlimit-mpi-pdt.so\* libTAUsh-papi-pdt.so\* libTAUsh-mpi-pdt.so\* libTAUsh-papi-pthread-pdt.so\* libTAUsh-mpi-pdt-upc.so\* libTAUsh-param-mpi-pdt.so\* libTAUsh-mpi-python-pdt.so\* libTAUsh-pdt.so\* libTAUsh-papi-mpi-pdt.so\* libTAUsh-pdt-trace.so\* libTAUsh-phase-papi-mpi-pdt.so\* libTAUsh-papi-mpi-pdt-upc.so\* libTAUsh-papi-mpi-pdt-upc-udp.so\* libTAUsh-pthread-pdt.so\* libTAUsh-papi-mpi-pdt-vampirtrace-trace.so\* libTAUsh-python-pdt.so\* libTAUsh-papi-mpi-python-pdt.so\* /home/livetau/tutorial/bin% ls \$TAU/libTAUsh-/home/livetau/tutorial/bin% /home/livetau/tutorial/bin% tau\_run -XrunTAUsh-papi-mpi-pdt-vampirtrace-trace bt\_W.4 -o bt.vpt /home/livetau/tutorial/bin% setenv VT\_METRICS PAPI\_FP\_INS:PAPI\_L1\_DCM /home/livetau/tutorial/bin% mpirun -np 4 ./bt.vpt /home/livetau/tutorial/bin% vampir bt.vpt.otf & /home/livetau/tutorial/bin%





hop 2010

1

## **TAU Performance System Components**



# **TAU Instrumentation / Measurement**



## **TAU Instrumentation**

- Flexible instrumentation mechanisms at multiple levels
  - Source code
    - manual (TAU API, TAU Component API)
    - automatic
      - C, C++, F77/90/95 (Program Database Toolkit (*PDT*))
      - OpenMP (directive rewriting (*Opari*), *POMP* spec)
  - Object code
    - pre-instrumented libraries (e.g., MPI using *PMPI*)
    - statically-linked and dynamically-linked
  - Executable code
    - dynamic instrumentation (pre-execution) (*DynInstAPI*)
    - virtual machine instrumentation (e.g., Java using *JVMPI*)
    - Python interpreter based instrumentation at runtime
  - Proxy Components

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications







# **TAU Analysis**



## ParaProf Profile Analysis Framework



### Parallel Profile Visualization: ParaProf



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### **PerfDMF** Architecture



### **Building Bridges to Other Tools**





 $\mathbf{O}$  | university of oregon

## **Direct Performance Observation**

- Execution actions of interest exposed as events
  - In general, actions reflect some execution state
    - presence at a code location or change in data
    - occurrence in parallelism context (thread of execution)
  - Events encode actions for performance system to observe
- Observation is direct
  - Direct instrumentation of program (system) code (probes)
  - Instrumentation invokes performance measurement
  - Event measurement: performance data, meta-data, context
- Performance experiment
  - Actual events + performance measurements
- Contrast with (indirect) event-based sampling

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







### **TAU Instrumentation Approach**

- Support for standard program events
  - Routines, classes and templates
  - Statement-level blocks
  - Begin/End events (Interval events)
- Support for user-defined events
  - Begin/End events specified by user
  - Atomic events (e.g., size of memory allocated/freed)
  - Flexible selection of event statistics
- Provides static events and dynamic events
- Enables "semantic" mapping
- Specification of event groups (aggregation, selection)
- Instrumentation optimization

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications







## **TAU Event Interface**

- Events have a type, a group association, and a name
- TAU events names are character strings
  - Powerful way to encode event information
  - Inefficient way to communicate each event occurrence
- TAU maps a new event name to an event ID
  - Done when event is first encountered (get event handle)
  - Event ID is used for subsequent event occurrences
  - Assigning a uniform event ID a priori is problematic
- A new event is identified by a new event name in TAU
  - Can create new event names at runtime
  - Allows for dynamic events (TAU renames events)
  - Allows for context-based, parameter-based, phase events

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







## Using TAU: A brief Introduction

- TAU supports several measurement options (profiling, tracing, ulletprofiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique • stub makefile and library that is generated when you configure it
- To instrument source code using PDT lacksquare
  - Choose an appropriate TAU stub makefile in <arch>/lib:
  - % setenv TAU\_MAKEFILE /usr/local/packages/tau/i386\_linux/lib/Makefile.tau-mpi-pdt % setenv TAU\_OPTIONS '-optVerbose ...' (see tau\_compiler.sh -help)
  - And use tau\_f90.sh, tau\_cxx.sh or tau\_cc.sh as Fortran, C++ or C compilers:

% mpif90 foo.f90

changes to

% tau f90.sh foo.f90

- Execute application and analyze performance data:
  - % pprof (for text based profile display)

% paraprof (for GUI) High Performance Software Tools to Fast-Track







### **TAU Measurement Configuration**

% cd /usr/local/packages/tau/i386\_linux/lib; ls Makefile.\* Makefile.tau-pdt Makefile.tau-mpi-pdt Makefile.tau-opari-openmp-mpi-pdt Makefile.tau-mpi-scalasca-epilog-pdt Makefile.tau-mpi-vampirtrace-pdt Makefile.tau-multiplecounters-mpi-papi-pdt Makefile.tau-multiplecounters-papi-mpi-openmp-opari-pdt Makefile.tau-pthread-pdt...

#### • For an MPI+F90 application, you may want to start with:

Makefile.tau-mpi-pdt

- Supports MPI instrumentation & PDT for automatic source instrumentation
- % setenv TAU\_MAKEFILE
  /usr/local/packages/tau/i386\_linux/lib/Makefile.tau-mpipdt
- % tau\_f90.sh matrix.f90 -o matrix

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







## **Usage Scenarios: Routine Level Profile**

- Goal: What routines account for the most time? How much?
- Flat profile with wallclock time:

Metric: P\_VIRTUAL\_TIME Value: Exclusive Units: seconds



**Development of Scalable and Sustainable Applications** 



Berkeley, California – Aug 17-20, 2010



### Generating a flat profile with MPI

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application




### **TAU Measurement Configuration –**

- ./configure –pdt=/usr/local/packages/pdtoolkit-3.16 -mpi Configure using PDT and MPI
- ./configure -papi=/usr/local/packages/papi-4.0.0
   -pdt=<dir> -mpi ; make clean install
  - Use PAPI counters (one or more) with C/C++/F90 automatic instrumentation.
     Also instrument the MPI library.
- Typically configure multiple measurement libraries using installtau
- Past configurations are stored in TAU's .all\_configs file and .installflags
- Each configuration creates a unique <arch>/lib/Makefile.tau<options> stub makefile. It corresponds to the configuration options used. e.g.,
  - /usr/local/packages/i386\_linux/lib/Makefile.tau-mpi-pdt
  - /usr/local/packages/i386\_linux/lib/Makefile.tau-mpi-papi-pdt

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







# **Compile-Time Environment Variables**

Optional parameters for TAU\_OPTIONS: [tau\_compiler.sh\_help] -optVerbose Turn on verbose debugging messages -optCompInst Use compiler based instrumentation Do not revert to compiler instrumentation if source -optNoCompInst instrumentation fails. Turn on debugging memory allocations/ -optDetectMemoryLeaks de-allocations to track leaks -optKeepFiles Does not remove intermediate .pdb and .inst.\* files -optPreProcess Preprocess Fortran sources before instrumentation -optTauSelectFile="" Specify selective instrumentation file for tau instrumentor -optLinking="" Options passed to the linker. Typically \$(TAU MPI FLIBS) \$(TAU LIBS) \$(TAU CXXLIBS) Options passed to the compiler. Typically -optCompile="" \$(TAU MPI INCLUDE) \$(TAU INCLUDE) \$(TAU DEFS) -optPdtF95Opts="" Add options for Fortran parser in PDT (f95parse/gfparse) -optPdtF95Reset="" Reset options for Fortran parser in PDT (f95parse/gfparse) -optPdtCOpts="" Options for C parser in PDT (cparse). Typically \$(TAU MPI INCLUDE) \$(TAU INCLUDE) \$(TAU DEFS) -optPdtCxxOpts=""

Options for C++ parser in PDT (cxxparse). Typically \$(TAU\_MPI\_INCLUDE) \$(TAU\_INCLUDE) \$(TAU\_DEFS)

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### **Compiling Fortran Codes with TAU**

- If your Fortran code uses free format in .f files (fixed is default for .f), you may use: % setenv TAU\_OPTIONS '-optPdtF95Opts="-R free" -optVerbose '
- To use the compiler based instrumentation instead of PDT (source-based):
   % setenv TAU\_OPTIONS '-optCompInst -optVerbose'
- If your Fortran code uses C preprocessor directives (#include, #ifdef, #endif):
   % setenv TAU\_OPTIONS '-optPreProcess -optVerbose -optDetectMemoryLeaks'
- To use an instrumentation specification file:
  % setenv TAU\_OPTIONS '-optTauSelectFile=mycmd.tau -optVerbose -optPreProcess'
  % cat mycmd.tau
  BEGIN\_INSTRUMENT\_SECTION

  memory file="foo.f90" routine="#"
  # instruments all allocate/deallocate statements in all routines in foo.f90
  loops file="\*" routine="#"
  io file="abc.f90" routine="FOO"
  END INSTRUMENT SECTION

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### **Usage Scenarios: Loop Level Instrumentation**

- Goal: What loops account for the most time? How much?
- Flat profile with wallclock time with loop instrumentation:

Metric: GET_T	ME_OF_DAY
Value: Exclusiv	e
Units: microsec	onds
1729975.333	Loop: MULTIPLY_MATRICES [{matmult.f90} {31,9}-{36,14}]
	443194 MPI_Recv()
	81095 🔚 MAIN
	49569 MPI_Bcast()
	45669 🗧 Loop: MAIN [{matmult.f90} {86,9}-{106,14}]
	12412 MPI_Send()
	8959   Loop: INITIALIZE [{matmult.f90} {17,9}-{21,14}]
	8953   Loop: INITIALIZE [{matmult.f90} {10,9}-{14,14}]
	5609.2   MPI_Finalize()
	2932.667 MULTIPLY_MATRICES
	2577.667   Loop: MAIN [{matmult.f90} {117,9}-{128,14}]
	2091.8   MPI_Barrier()
	1875.667   Loop: MAIN [{matmult.f90} {112,9}-{115,14}]
	1833   Loop: MAIN [{matmult.f90} {71,9}-{74,14}]
	107   Loop: MAIN [{matmult.f90} {77,9}-{84,14}]
	30   INITIALIZE
	14.25   MPI_Comm_rank()
	1   MPI_Comm_size()





----

#### Generating a loop level profile

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                              /lib/Makefile.tau-mpi-pdt
% setenv TAU OPTIONS `-optTauSelectFile=select.tau -optVerbose'
% cat select tau
 BEGIN INSTRUMENT SECTION
  loops routine="#"
 END INSTRUMENT SECTION
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% make F90=tau f90.sh
(Or edit Makefile and change F90=tau f90.sh)
% mpirun -np 4 ./a.out
% paraprof --pack app.ppk
 Move the app.ppk file to your desktop.
% paraprof app.ppk
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### **Usage Scenarios: Compiler-based Instrumentation**

• Goal: Easily generate routine level performance data using the compiler instead of PDT for parsing the source code



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





### **Use Compiler-Based Instrumentation**

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





# Usage Scenarios: Calculate mflops in

- Goal: What MFlops am I getting in all loops?
- Flat profile with PAPI\_FP\_INS/OPS and time with loop instrumentation:

Metric: PAPI\_FP\_INS / GET\_TIME\_OF\_DAY Value: Exclusive Units: Derived metric shown in microseconds format

770.699 Loop: MULTIPLY\_MATRICES [{matmult.f90} {31,9}-{36,14}] Loop: INITIALIZE [{matmult.f90} {10,9}-{14,14}] 223.39 Loop: INITIALIZE [{matmult.f90} {17,9}-{21,14}] 223.24 Loop: MAIN [{matmult.f90} {71,9}-{74,14}] 171.855 Loop: MAIN [{matmult.f90} {112,9}-{115,14}] 170.862 122.96 Loop: MAIN [{matmult.f90} {117,9}-{128,14}] MULTIPLY MATRICES 37.549 21.367 INITIALIZE 13.795 Loop: MAIN [{matmult.f90} {86,9}-{106,14}] 11 MPI Comm size() 8.935 Loop: MAIN [{matmult.f90} {77,9}-{84,14}] 1.131 | MPI Send() 0.794 MPI Comm rank() 0.647 | MPI Bcast() 0.355 | MPI Recv() 0.171 MPI Barrier() 0.115 | MPI Finalize() 0.023 MAIN

High Performance Software roots to Fast-mack Development of Scalable and Sustainable Application Berkeley, California – Aug 17-20, 2010





### Generate a PAPI profile with 2 or more

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                              /lib/Makefile.tau-papi-mpi-pdt
% setenv TAU OPTIONS `-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
 BEGIN INSTRUMENT SECTION
 loops routine="#"
 END INSTRUMENT SECTION
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% make F90=tau f90.sh
(Or edit Makefile and change F90=tau f90.sh)
% setenv TAU METRICS TIME:PAPI FP INS:PAPI L1 DCM
% mpirun -np 4 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
  Choose Options -> Show Derived Panel -> "PAPI FP INS", click "/", "TIME", click "Apply"
    choose.
```

Development of Scalable and Sustainable Applications







### **Derived Metrics in ParaProf**

000		🔀 TAU: ParaProf Manager						
File Options Help								
Applications	TriplField	Value						
E Standard Applications	Trial ID	0						
Joindard Applications (CPII Cores 6								
Cal Default App	CPU MHz	2600.093						
V Default Exp	CPU Type	Six-Core AMD Opteron/tm) Processor 8435						
	CPU Vendor	AuthenticAMD						
	CWD	/root/f90						
APILLI_DOM	Cache Size	512 KB						
FARI_FF_IND	Executable	/root/f90/ring.i						
	File Type Index							
	File Type Name	Tau profiles						
	Hostname	b2e4s5.coming.com						
	Local Time	2010-04-18T02:27:58-04:00						
	MPI Processor Name	b2e4s5.coming.com						
	Memory Size	66006592 kB						
	Node Name	b2e4s5.coming.com						
	OS Machine	x86_64						
	OS Name	Linux						
	OS Release	2.6.18-128.el5.perfctr						
	OS Version	#1 SMP Mon Jun 29 12:32:22 PDT 2009						
	Starting Timestamp	1271572078767980						
	TAU Architecture	x86_64						
	TAU Config	-c++=g++ -cc=gcc -fortran=gfortran -mpilib=/usr/nic/dev/intel/impi/3.2.2.006/lib64 -mpiinc=/usr/nic/dev/intel/impi/3.2.2.0	06/					
	TAU Makefile	/usr/corning/apps/paratools/tau-2.19.1/x86_64/lib/Makefile.tau-inteImpi-papi-mpi-pdt						
	TAU Version	tau-2.19.1						
	TAU_CALLPATH	off						
	TAU_CALLPATH_DEPTH	2						
	TAU_COMM_MATRIX	off						
	TAU_COMPENSATE	off						
	TAU_PROFILE	on						
	TAU_PROFILE_FORMAT	profile						
	TAU_THROTTLE	on						
	TAU_THROTTLE_NUMCALLS	100000						
	TAU_THROTTLE_PERCALL	10						
	TAU_TRACE	off						
	TAU_TRACK_HEADROOM	off						
	TAU_TRACK_HEAP	off						
	TAU_TRACK_MESSAGE	off						
	Timestamp	1271572078835079						

+ - \* / = ( )

High Performance Software Tools to Fast-Track

Apply



11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



Clear

### **Comparing Effects of Multi-Core Processors**



UNIVERSITY OF OREGON



#### Mflops Sorted by Exclusive Time



### **Generating Callpath Profiles**

- Goal: To expose the calling sequence. E.g., what routine calls an MPI\_Barrier()? Where?
- Callpath profile for a given callpath depth:

000	🔀 n,c,t, 0,0,0 – callpath-all/scaling/flash/taudata/disk2/mnt/
File Options Windows	: Help
Metric Name: Time	
value Type: exclusive	
26 474%	
26.474%	FLASH => FVOLVE => HYDRO: HYDRO 3D => MODIII FHYDROSWFFP: HYDRO SWFFP
24.556%	
24.556%	FLASH = > FVOLVE = > HYDRO::HYDRO 3D = > MODILLEHYDROSWEEP::HYDRO SWEEP = > MODILLEHYDRO 1D::HYDRO 1D
	14.351% MODULEINTRFC
	14.351% FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEHYDRO_1D::HYDRO_1D => MODULEINTRFC::INT
	4.501% MODULEE0S3D:E0S3D
	4.427% MPL_Ssendo
	3.678% EASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEEOS3D::EOS3D
	3.536% MPL_Allreduce¢
	2.727% MPL_Waitall
	2.242% MODULEUPDATE_SOLN:UPDATE_SOLN
	2.242% ELASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MODULEUPDATE_SOLN:UPDATE_SOLN
	2.059% AMR_GUARDCELL_CC_SRL
	1.703% FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_
	1.56% HLASH => EVOLVE => HYDRO::HYDRO::HYDRO::HYDRO::HYDRO::WEEP::HYDRO::SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_
	1.400% TLASH => EVOLVE => MESH_UPDATE_CRU_REPINEMENT => MESH_REPINE_DEREPINE => AMR_REPINE_DEREPINE => AMR_MORTON_URDER => A
	1.501% TEASH => IMESTEP => MIT_AUTEQUILEQ
	1.077% TLASH => EVOLVE => HYDRO::HYDRO 3D => MODIII FHYDROSWFEP::HYDRO SWFEP => ARIINDANCE RESTRICT
	1.064% DBASETREE::DBASENEIGHBORBLOCKLIST
	1% FLASH => EVOLVE => HYDRO::HYDRO.3D => MODULEHYDROSWEEP::HYDRO.SWEEP => MESH.GUARDCELL => AMR.RESTRICT => AMR.RESTRICT
	0.987% FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_FLUX_CONSERVE => AMR_FLUX_CONSERVE_UDT
	0.96% 🗌 FLASH => EVOLVE => HYDR0::HYDR0_3D => MODULEHYDROSWEEP::HYDR0_SWEEP => MESH_GUARDCELL => AMR_GUARDCELL_C_T0_F => A
	0.916% 📕 MPL_Barriero
	0.807% FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => MESH_GUARDCELL => TOT_BND => DBASETREE::DBAS
	0.806% AMR_PROLONG_UNK_FUN
	0.735% AMR_DIAGONAL_PATCH
	0.699% UDIFFUSE
	0.699% I FLASH => EVOLVE => HYDRO::HYDRO_3D => MODULEHYDROSWEEP::HYDRO_SWEEP => DIFFUSE
	0.671% [] AMR_RESTRICT_RED
	U6/1% II FLASH => EVOLVE => HYDROCHYDRO.3D => MODULEHYDROSWEEP: HYDROSWEEP => MESH_FLUX_CONSERVE => AMR_FLUX_CONSERVE_UD
	U.537% IFLASH => EVULVE => HURVCHTURV_3U => MUDULEHTURVSWEH: HTURCSWEH => MBH_GUARDCELL => AMR_GUARDCELL_SRL => AMR_ 0.638% IFLASH => EVULVE => MURVCHURDATE COND DEFINITIONET => MMR_DEFINITIONET => MRF_DURVEHT
	U.033% IFLASH => EVULVE => MESH_UPU/ATE_GNU_KENIKENEN => MARK_GNU_KENIKENEN => MPL_SAMERY
	$0.01/0$ (Trash => EVOLVE => HIDROCHIDRO_3D => MODULEHIDROSWEED-SHIDROSWEEP => MESH (MARDUELL => AMA (MARDUELL \[1]] (T_1 => A
	0.330% BTC301 - / EVOXE - / HIDKO, HIDKO, 50 - / MODULENDROSWEEF, HIDROSWEEF = / MESH_GUARDCELL = / AMK_GUARDCELL_TO_F = / A
	0.500% [ TAT_DITAL 0.454%   FLASH => FVOLVE => MESH UPDATE CRID REFINEMENT => MARK CRID REFINEMENT => MODULEFOS3D*FOS3D

High Performance Software roots to Fast-Track

COLLECTION

Development of Scalable and Sustainable Applications

11" DUE ACTS Workshop



Berkeley, California – Aug 17-20, 201



### Callpath Profile

• Generates program callgraph



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### Generate a Callpath Profile

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





### Usage Scenario: Detect Memory Leaks

	TAU: ParaProf: Mean Context Events - mem.ppk							
ile Windows Help								
Name 🛆	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.			
MAIN [{matrix.f90} {141,7}-{146,22}]								
MATRICES::ALLOCATE_MATRICES [{matrix.f90} {10,7}-{13,38}]								
MEMORY LEAK! malloc size <file=matrix.f90, line="11" variable="C,"></file=matrix.f90,>	- 1	8,000,000	8,000,000	8,000,000				
malloc size <file=matrix.f90, line="11" variable="A,"></file=matrix.f90,>	1	8,000,000	8,000,000	8,000,000				
malloc size <file=matrix.f90, line="11" variable="B,"></file=matrix.f90,>	1	8,000,000	8,000,000	8,000,000				
malloc size <file=matrix.f90, line="11" variable="C,"> MATRICES::DEALLOCATE_MATRICES [{matrix.f90} {14,7}-{17,40}]</file=matrix.f90,>	1	8,000,000	8,000,000	8,000,000				
– free size <file=matrix.f90, line="15" variable="A,"></file=matrix.f90,>	1	8,000,000	8,000,000	8,000,000				
free size <file=matrix.f90, line="15" variable="B,"></file=matrix.f90,>	1	8,000,000	8,000,000	8,000,000				
	сшърк							
le Options Windows Help ame: MEMORY LEAK! malloc size <file=matrix.f90, line="11" variable="C,"> : MAIN [{m</file=matrix.f90,>	atrix.f90}{141.	7}-{146,22}]	= > MATRICES:	:ALLOCATE_MAT				
Ie         Options         Windows         Help           ame:         MEMORY LEAK!         malloc size <file=matrix.f90, line="11" variable="C,"> : MAIN [{m           matrix.f90}{10,7}-{13,38}]         alue Type:         Max Value           3000000         3000000         3000000           3000000         3000000         3000000</file=matrix.f90,>	atrix.f90} {141,	7}-{146,22}]	= > MATRICES:	:ALLOCATE_MAT	Mean n,c,t 0,0,0 n,c,t 1,0,0 n,c,t 2,0,0 n,c,t 3,0,0 Std. Dev.			





### **Detect Memory Leaks**

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                          /lib/Makefile.tau-mpi-pdt
% setenv TAU OPTIONS `-optDetectMemoryLeaks -optVerbose'
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% make F90=tau f90.sh
(Or edit Makefile and change F90=tau f90.sh)
% setenv TAU CALLPATH DEPTH 100
% mpirun -np 4 ./a.out
% paraprof --pack app.ppk
 Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Context Event Window -> Select thread -> select...
   expand tree)
(Windows -> Thread -> User Event Bar Chart -> right click LEAK
-> Show User Event Bar Chart)
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





old O | university of oregon

### Instrument a Python program

#### • Goal: Generate a flat profile for a Python program



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### Instrumenting a Python program

% cat foo.py #!/usr/bin/env python Original import numpy ra=numpy.random code: la=numpy.linalg size=2000 a=ra.standard\_normal((size,size)) b=ra.standard\_normal((size,size)) c=la.linalg.dot(a,b) print c *Create a wrapper:* % cat wrapper.py #!/usr/bin/env python # setenv PYTHONPATH \$PET\_HOME/pkgs/tau-2.17.3/ppc64/lib/bindings-gnu-python-pdt import tau def OurMain(): import foo tau.run('OurMain()') High Performance Software Tools to Fast-Track



UNIVERSITY OF OREGON

### Generate a Python Profile

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                /lib/Makefile.tau-python-pdt
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% cat wrapper.py
  import tau
  def OurMain():
      import foo
  tau.run(`OurMain()')
Uninstrumented:
% ./foo.pv
Instrumented:
% export PYTHONPATH= <taudir>/i386 linux/<lib>/bindings-python-pdt
(same options string as TAU MAKEFILE)
% export LD LIBRARY PATH=<taudir>/i386 linux/lib/bindings-python-pdt:
$LD LIBRARY PATH
% ./wrapper.py
Wrapper invokes foo and generates performance data
% pprof/paraprof
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### Usage Scenarios: Mixed Python+F90+C

• Goal: Generate multi-level instrumentation for Python+MPI+C+F90+C++ ...

	TAU: ParaProf: Mean Data – py.ppk
Metric: Time	
Value: Exclusive percent	
39.466%	void SAMINT::timestep(double, double) [{SAMINT.C} {72,1}-{78,1}]
	6.563% int pyMPI_Main_with_communicator(int, int *, char ***, MPI_Comm) C [{pyMPI_main.c} {23,1}-{80,1}]
	5.18% <pre>status</pre> 5.18%
	4.91% void pyMPI_distutils_init(PyObject **) C [{pyMPI_distutils.c} {275,1}-{400,1}]
	3.845% <pre><module> [{/usr/local/PE1/pkgs/python-2.5.1/lb/python2.5/site-packages/numpy/core/numeric.py} {1}]</module></pre>
	3.633% void pyMPL site(void) C [[pyMPL site.C] [23,1]-[121,1]]
	$3.00\%$ = <pre>Conductive (/usr/local/ret/pkgs/python=2.5.1/ib/python=2.5/site-packages/numpy/core/_inite_py[25]</pre>
	3.1% Macharmin(l/usr/local/PE1/pkgs/python=2.5.1/lb/python2.5/site-packages/numpy/hb/machar.py} [50]
	2.2018parse [r/ds//bda/ref/pkgs/pytion=2.51/lb/pytion2.5/sie_parse.py[303]
	1 187% any [//usr/local/PET/pkg/python-2 5 1/lib/python2 5 //site_ongkanes/putpet/pyt 505]
	1.034% <pre>cmterity://document/or/or-c++-f90/example/samerrun.ov/{5}</pre>
	0.982% cmodule> [/usr/local/PET/pks/sython-2.5.1/lb/python2.5/site-packages/numpy/ init .pv} {17}]
	0.941% Tokenizer:: next [/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre parse.py] [188]
	0.929% append
	0.906% <pre>cmodule&gt;[{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/ctypes/_initpy}[4]]</pre>
	0.895% 🗍 any
	0.888% 📱 <module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/_initpy} {1}]</module>
	0.884% 🛽 len
	0.863% 🔋 <module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/index_tricks.py} {3}]</module>
	0.806% 📱 void pyMPl_pickle_init(PyObject **) C [{pyMPl_pickle.c} {43,1}-{61,1}]
	0.739% 📗 <module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/random/_initpy} {2}]</module>
	0.67% [] SubPattern::getwidth [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} [146]]
	0.658% <pre>// calle&gt; [//usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/testing/numpytest.py] {1]</pre>
	0.643% []optimize_charset [{usr/local/PE1/pkg5/python-2.5.1/lib/python2.5/sre_compile.py]{213}]
	0.564% i Tokenizer::get[{/usr/local/PE/jpkg5/pytton-2.5.1/lb/pytton2.5/sre_parse.py}{20/}]
	0.522% (modules [[/usr/local/PE1/pkgs/pythone.cs.1/http/pythone.cs/inspect.py] {24]]
	0.542.6 Since (https://www.sinceri/py/py-c+++15/samint.py///j/
	0.537%   <module> [//bs//bca//EF//bks//ython-25/1/bb/ython25//ite-nackages/numpy/testing/_initnyt[2]]</module>
	$0.532\%$ [ <module> [[//sers3/camer/nv/nv-c++-f9n/samer/nv] {9]]</module>
	0.475% d <module> [//usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/ctypeslib.py] [1]</module>
	0.462% add newdoc [[/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numov/lib/function base.py} [1154]]
	0.453% <module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/lib/function_base.py} {1}]</module>
	0.451% parse_sub [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse_py} {307}]
	0.378%   _compile_charset [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_compile.py} {184}]
	0.367%   abs
	0.365% amarc::runStep [{/users3/sameer/py/py-c++-f90/samarc.py} {105}]
	0.365% <pre><pre>0.365%</pre> <pre><pre>0.365%</pre> <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
	0.296%   SubPattern:len [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/sre_parse.py} {132}]
	0.28%   <module> [{/usr/local/PET/pkgs/python-2.5.1/lib/python2.5/site-packages/numpy/core/arrayprint.py} [4]]</module>
	0.2/%   MPI_Finalize()
	0.225%   Tokenizer::match [{/usr/locai/PE1/pkgs/python-2.5.1/lib/python2.5/sre_parse.py[201]]
	U.213% [_mK_DITTAB [[/UST/IOCA]/FL/PKgS/Python-2.5.1/IID/python2.5/Sre_compile.py][264]]
	0.199%   Kini usiri) 0.199K   Mini usiri)

High Performance Software Tools to Fast-Track





### Generate a Multi-Language Profile

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                /lib/Makefile.tau-python-mpi-pdt
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% setenv TAU OPTIONS `-optShared -optVerbose...'
(Python needs shared object based TAU library)
% make F90=tau f90.sh CXX=tau cxx.sh CC=tau cc.sh (build libs, pyMPI w/TAU)
% cat wrapper.py
  import tau
 def OurMain():
      import App
  tau.run(`OurMain()')
Uninstrumented:
% mpirun -np 4 pyMPI ./App.py
Instrumented:
% export PYTHONPATH= <taudir>/i386 linux/<lib>/bindings-python-mpi-pdt
(same options string as TAU MAKEFILE)
% export LD LIBRARY PATH=<taudir>/i386 linux/lib/bindings-python-mpi-pdt:
$LD LIBRARY PATH
% mpirun -np 4 <pkgs>/pyMPI-2.5b0-TAU/bin/pyMPI
                           (Instrumented pyMPI with wrapper.py)
./wrapper.py
```



#### **Tracing Measurement**





 $\mathbf{O}$  | university of oregon

### **Tracing Analysis and Visualization**





**O** UNIVERSITY OF OREGON

# **Usage Scenarios: Generating a Trace**

- Goal: Identify the temporal aspect of performance. What happens in my code at a given time? When?
- Event trace visualized in Vampir/Jumpshot



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application 11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



UNIVERSITY OF OREGON

# **VNG Process Timeline with PAPI**



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application 11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



 $\mathbf{O}$  | university of oregon

### Vampir Counter Timeline Showing I/O BW



ACTS



### Generate a Trace File

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux/
lib/Makefile.tau-mpi-pdt
% export TAU TRACE=1
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% make F90=tau f90.sh
(Or edit Makefile and change F90=tau f90.sh)
% mpirun -np 4 ./a.out
% tau treemerge.pl
(merges binary traces to create tau.trc and tau.edf files)
JUMPSHOT:
% tau2slog2 tau.trc tau.edf -o app.slog2
% jumpshot app.slog2
  OR
VAMPIR:
% tau2otf tau.trc tau.edf app.otf -n 4 -z
(4 streams, compressed output trace)
% vampir app.otf
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





**O** UNIVERSITY OF OREGON

### **Usage Scenarios: Evaluate Scalability**

- Goal: How does my application scale? What bottlenecks occur at what core counts?
- Load profiles in PerfDMF database and examine with PerfExplorer



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





### **Usage Scenarios: Evaluate Scalability**



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application 11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



UNIVERSITY OF OREGON

# **Performance Regression Testing**



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications 11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



 $\mathbf{O}$  | university of oregon

#### Evaluate Scalability using PerfExplorer Charts

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                          /lib/Makefile.tau-mpi-pdt
% set path=(/usr/local/packages/tau/i386 linux/bin $path)
% make F90=tau f90.sh
(Or edit Makefile and change F90=tau f90.sh)
% mpirun -np 1 ./a.out
% paraprof --pack 1p.ppk
% mpirun -np 2 ./a.out ...
% paraprof --pack 2p.ppk ... and so on.
On your client:
% perfdmf configure --create-default
(Chooses derby, blank user/passwd, yes to save passwd, defaults)
% perfexplorer configure
(Yes to load schema, defaults)
% paraprof
(load each trial: DB -> Add Trial -> Type (Paraprof Packed Profile) ->
   OK) OR use perfdmf loadtrial.
% perfdmf loadtrial -a "NWChem" -x "Scaling on i386 linux" -n "32p"
   32p.ppk
Then,
% perfexplorer
(Select experiment, Menu: Charts -> Speedup)
```





# **Communication Matrix Display**

• Goal: What is the volume of inter-process communication? Along which calling path?





**O** UNIVERSITY OF OREGON

### **Communication Matrix**

```
% setenv TAU MAKEFILE /usr/local/packages/tau/i386 linux
                            /lib/Makefile.tau-mpi-pdt
    % set path=(/usr/local/packages/tau/i386 linux/bin $path)
    % make F90=tau f90.sh
    (Or edit Makefile and change F90=tau f90.sh)
    % export TAU COMM MATRIX=1
    % mpirun -np 4 ./a.out (setting the environment variables)
    % paraprof
    (Windows -> Communication Matrix)
High renormance sonware roots to ra
                                                                  ACTS Workshop
```



UNIVERSITY OF OREGON

### ParaProf: Communication Matrix Display



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





#### Measuring Performance of PGI GPGPU Accelerated Code

000	TAU: ParaProf: n,c,t 0,0,0 – mat1k.ppk							
Metric: TIME Value: Exclusive percent 68.044% 24.417% 3.206% 1.572%	pgi_cu_launch multiply_matrices (pgi_ke pgi_cu_init multiply_matrices [{mm2.f90 pgi_cu_download2 multiply_matrices va pgi_cu_upload2 multiply_matrices var=l	ernel_7,gx=32,gy=3 ){9}] r=a [{mm2.f90}{20}] b [{mm2.f90}{9}] c [{mm2.f90}]	2,gz=1,bx=16,by=1	6,bz=1) [{m	m2.f90}{15}]			
0.782%	$\begin{array}{c c c c c c c c c c c c c c c c c c c $							
$0.142\%$  pgi_cu_iauric multiply_matrices (pgi_kernet_2,gx=52,gy=52,gz=1,bx=16,by=16,bz=1) [{mm2.f90}{11}] 0.122%  pgi_cu_free multiply_matrices [{mm2.f90}] 0.122%  pgi_cu_free multiply_matrices [{mm2.f90}]								
0.12%  pgi_cu_alloc multiply_matrices [{mm2.f90}{9}] 0.017%   multiply_matrices [{mm2.f90} {5,0}]								
0.002%	pgi_ccu_module multiply_matrices [{mm2	2.f90}{9}]						
2.1E-4% 1.2E-4%	2.1E-4%  pgi_cu_module_function multiply_matrices [{mm2.f90}{11}] 1.2E-4%  pgi_cu_paramset multiply_matrices [{mm2.f90}]							
1.2E-4%	pgi_cu_module_function multiply_matrie	ces [{mm2.f90}{15}]						
					//			
🔴 🔿 🔿 TAU: I	ParaProf: Thread Statistics: n,c,t, 0,0,0 - mat1k.	ppk		_				
Name		Exclusive TIME $ abla$	Inclusive TIME	Calls	Child Calls			
pgi_cu_launch multiply_matrices (pgi_kernel_7,gx=32,gy=32,gz	=1,bx=16,by=16,bz=1) [{mm2.f90}{15}]	10.901	10.901	5	0			
pgi_cu_init multiply_matrices [{mm2.f90}{9}]	Show Source Code	3.912	3.912	5	0			
pgi_cu_download2 multiply_matrices var=a [{mm2.f90}{20}]	Show Function Bar Chart	0.514	0.514	5	0			
pgi_cu_upload2 multiply_matrices var=b [{mm2.f90}{9}]	Show Function Histogram	0.252	0.252	5	0			
pgi_cu_upload2 multiply_matrices var=c [{mm2.f90}{9}]	Assign Function Color  Perset to Default Color	0.252	0.252	5	0			
mymatrixmultiply [{mmdriv.f90} {1,0}]	Reset to Default Color	0.125	16.021	1	1			
pgi_cu_launch multiply_matrices (pgi_kernel_2,gx=32,gy=32,gz	0.023	0.023	5	0				
pgi_cu_free multiply_matrices [{mm2.f90}]		0.02	0.02	15	0			
pgi_cu_alloc multiply_matrices [{mm2.f90}{9}]		0.019	0.019	15	0			
multiply_matrices [{mm2.f90} {5,0}]		0.003	15.895	5	9			
pgi accelerator region		0.001	15.893	5	85			
pgi_cu_module multiply_matrices [{mm2.f90}{9}]		0	0	5	0			
pgi_cu_module_function multiply_matrices [{mm2.f90}{11}]		0	0	5	0			
ngi su naramaat multinlu matricas [[mm2 f00]]		0	0	10	0			




#### Scaling NAMD with CUDA (Jumpshot with TAU)



Development of Scalable and Sustainable Applications

rkelev California – Aug 17-20 2010



UNIVERSITY OF OREGON

## Parallel Profile Visualization: ParaProf





 $\mathbf{O}$  university of oregon

#### Scalable Visualization: ParaProf (128k cores)



## Scatter Plot: ParaProf (128k cores)



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## ParaProf (1m cores<sup>\*</sup>)

	TAU: ParaProf: 1m.ppk	// = = ×
File Options Windows Help		
Metric: BGP Timers Value: Exclusive		
node 1048532		
node 1048533		•
node 1048534		
node 1048535		
node 1048536		
node 1048537		
node 1048538		
node 1048539		
node 1048540		
node 1048541		
node 1048542		
node 1048543		
node 1048544		•
node 1048545		
node 1048546		•
node 1048547		•
node 1048548		•
node 1048549		
node 1048550		
node 1048551		
node 1048552		
node 1048553		
node 1048554		
node 1048555		
node 1048556		
node 1048557		
node 1048558		
node 1048559		•
node 1048560		
node 1048561		
node 1048562		•
node 1048563		*
node 1048564		
node 1048565		
node 1048566		•
node 1048567		•
node 1048568		
node 1048569		
node 1048570		
node 10485/1		
node 1048572		
node 1048573		
node 1048574		
node 1048575		

High Performance Software Tools to Fast-Track

11th DOE ACTS Workshop

Development of Scalable and Sustainable Applications \*1m core dataset generated by replicating a 32 k core dataset



**O** UNIVERSITY OF OREGON

## Histogram: ParaProf (1m cores\*)



#### \*1m core dataset generated by replicating a 32 k core dataset

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## Labs: LiveDVD

• Add one of

source /usr/local/packages/etc/point.bashrc

or

#### source /usr/local/packages/etc/point.cshrc

to the end of your **.login** file (for bash or csh/tcsh users respectively).

On the LiveDVD, please see the ~/point-workshop directory

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## **Program Database Toolkit (PDT)**



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications

11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



 $\mathbf{O}$  | university of oregon

## **Automatic Source-level Instrumentation**



High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## Selective Instrumentation File

- Specify a list of events to exclude or include
- # is a wildcard in a routine name BEGIN\_EXCLUDE\_LIST Foo Bar D#EMM END\_EXCLUDE\_LIST

```
BEGIN_INCLUDE_LIST
int main(int, char **)
F1
F3
END_INCLUDE_LIST
High Performance Software Tools to Fast-Track
```

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## **Selective Instrumentation File**

- Optionally specify a list of files
- \* and ? may be used as wildcard characters **BEGIN\_FILE\_EXCLUDE\_LIST** f\*.f90
  - Foo?.cpp
  - END\_FILE\_EXCLUDE\_LIST
  - BEGIN FILE INCLUDE LIST
  - main.cpp
  - foo.f90

# END\_FILE\_INCLUDE\_LIST





# **TAU Integration with IDEs**

- High performance software development environments
  - Tools may be complicated to use
  - Interfaces and mechanisms differ between platforms / OS
- Integrated development environments
  - Consistent development environment
  - Numerous enhancements to development process
  - Standard in industrial software development
- Integrated performance analysis
  - Tools limited to single platform or programming language
  - Rarely compatible with 3rd party analysis tools
  - Little or no support for parallel projects

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application







## **TAU and Eclipse**





**O** UNIVERSITY OF OREGON

#### Choosing PAPI Counters with TAU in Eclipse

<ul> <li>Image: A start of the start of</li></ul>	Profile	*		-	×)
Create, manage, and run co	onfigurations		Counter	Definition	-
Create a configuration to launch a program to be instrumented and profiled by TAU.		1	PAPI_L1_DCM	Level 1 data cache misses	
			PAPI_L1_ICM	Level 1 instruction cache misses	
				Level 2 data cache misses	
	Name: Tammps-10Nov05withTAU		PAPI_L2_ICM	Level 2 instruction cache misses	
type filter text ■ Main @= Arguments The Environment H Parallel Analysis *2			PAPI_L1_TCM	Level 1 cache misses	
C/C++ Local Applic	PARI Countorr		PAPI_L2_TCM	Level 2 cache misses	
	MPI		PAPI_FPU_IDL	Cycles floating point units are idle	
lammps-10Nov0	Callpath Pro		PAPI_TLB_DM	Data translation lookaside buffer misses	
	Phase Base PAPI_L1_DCM		PAPI_TLB_IM	Instruction translation lookaside buffer misses	
			PAPI_TLB_TL	Total translation lookaside buffer misses	
	PAPI_L2_DCM		PAPI_L1_LDM	Level 1 load misses	
			PAPI_L1_STM	Level 1 store misses	
	OpenMP PAPI_L1_TCM		PAPI_L2_LDM	Level 2 load misses	
			PAPI_L2_STM	Level 2 store misses	
	PAPI PAPI FPU IDL nter	s	PAPI_STL_ICY	Cycles with no instruction issue	
			PAPI_HW_INT	Hardware interrupts	
			PAPI_BR_TKN	Conditional branch instructions taken	
			PAPI_BR_MSP	Conditional branch instructions mispredicted	
	Select Makefile		PAPI_TOT_INS	Instructions completed	
	PAPI L1 STM		PAPI_FP_INS	Floating point instructions	
	Selective Instru		PAPI_BR_INS	Branch instructions	
	None     Select All     Deselect All     Counter Descriptions		PAPI_VEC_INS	Vector/SIMD instructions	
	○ Internal		PAPI_RES_STL	Cycles stalled on any resource	
	O User Define		PAPI_TOT_CYC	Total cycles	
			PAPI_L1_DCH	Level 1 data cache hits	
			PAPI_L2_DCH	Level 2 data cache hits	
	Appl <u>y</u> Re <u>v</u> ert		PAPI_L1_DCA	Level 1 data cache accesses	
			PAPI_L2_DCA	Level 2 data cache accesses	
0	Profile		PAPI_L2_DCR	Level 2 data cache reads	
	<u> </u>		PAPI_L2_DCW	Level 2 data cache writes	*
0	Profile Close		PAPI_L2_DCR PAPI_L2_DCW	Level 2 data cache reads Level 2 data cache writes	₹,

#### % /usr/local/packages/eclipse/eclipse High Performance Software Tools to Fast-Track 11th DOE ACTS Work

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## VampirTrace and Vampir

- Introduction
- Event Trace Visualization
- Vampir & VampirServer
- The Vampir Displays
  - Timeline
  - Process Timeline with Performance Counters
  - Summary Display
  - Message Statistics
- VampirTrace
  - Instrumentation & Run-Time Measurement
- Conclusions

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Application





## VampirServer Architecture



## Vampir Displays

The main displays of Vampir:

- Global Timeline
- Process Timeline w/o Counters
- Statistic Summary
- Summary Timeline
- Message Statistics
- Collective Operation Statistics
- Counter Timeline
- Call Tree

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## Vampir Global Timeline Display



High Performance Software Tools to Fast-Track

11<sup>th</sup> DOE ACTS Workshop Berkeley, California – Aug 17-20, 2010



UNIVERSITY OF OREGON

## **Process Timeline Display**





## **Process Timeline with Counters**



## **Statistic Summary Display**

🔍 🦳 💻 Vampir - Sum	mary Chart - <	: <u>2 &gt; &lt;@a</u> 01>	)	
Name	Token	Value		
hypre_StructMatvecCompute	[299]	2:15.199		
hypre_StructAxpy	[ 306 ]	2:10.744		
hypre_StructInnerProd	[ 295 ]	1:14.087		
MPI_Finalize	[82]	1:04.179		
hypre_StructCopy	[ 297 ]	51.516 s		
MPI_Waitall	[ 163 ]	20.135 s		
hypre_StructVectorSetConstantVa	[ 303 ]	20.124 s		
hypre_StructScale	[ 308 ]	15.580 s		
MPI_Allreduce	[9]	13.283 s		
MPI_Isend	[ 115 ]	9.010 s		
hypre_StructMatrixSetBoxValues	[ 229 ]	8.455 s		
sync	[2]	5.654 s		
main	[ 184 ]	4.661 s		
hypre_CAlloc	[ 186 ]	2.050 s		
hypre_StructVectorSetBoxValues	[ 260 ]	1.827 s		
hypre_StructMatrixInitializeDat	[ 224 ]	0.738 s		
hypre_StructKrylovAxpy	[ 305 ]	0.668 s		
MPI_Init	[ 108 ]	0.436 s		
hypre_StructKrylovCopyVector	[296]	0.221 s		
hypre_StructKrylovMatvec	[298]	0.215 s		
hypre_PCGSolve	[ 293 ]	0.212 s		
MPI_Irecv	[ 113 ]	0,190 s		
hypre_BoxGetSize	[ 227 ]	0,182 s		
hypre_Free	[ 187 ]	0,169 s		
hypre_InitializeCommunication	[ 250 ]	0,160 s		
Sorted by Value Down			All Symbols: Exclu	usive Times

High Performance Software Tools to Fast-Track

Development of Scalable and Sustainable Applications







http://www.tu-dresden.de/zih/vampirtrace/,

get support via <u>vampirsupport@zih.tu-dresden.de</u>

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications



Berkeley, California – Aug 17-20, 2010



## Jumpshot

- <u>http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm</u>
- Developed at Argonne National Laboratory as part of the MPICH project
  - Also works with other MPI implementations
  - Installed on IBM BG/P
  - Jumpshot is bundled with the TAU package
- Java-based tracefile visualization tool for postmortem performance analysis of MPI programs
- Latest version is Jumpshot-4 for SLOG-2 format
  - Scalable level of detail support
  - Timeline and histogram views
  - Scrolling and zooming
  - Search/scan facility

High Performance Software Tools to Fast-Track Development of Scalable and Sustainable Applications





## Jumpshot



High Performance Software Tools to Last Track



UNIVERSITY OF OREGON

## Support Acknowledgements

- Department of Energy (DOE)
  - Office of Science
    - MICS, Argonne National Lab
  - ASC/NNSA
    - University of Utah ASC/NNSA Level 1
    - ASC/NNSA, Lawrence Livermore National Lat
- Department of Defense (DoD)
- NSF Software Development for Cyberinfrastructure (SDCI) •
- **Research Centre Juelich** •
- ANL, LBL, PNNL, LLNL, LANL, SNL
- **TU** Dresden
- ParaTools, Inc.





Office of

J.S. DEPARTMENT OF ENERGY

## **ParaTools**

**IÚLICH** 









