

# Hands-on: BullX DLC *Froggy* Using TAU with NPB-MZ-MPI / BT

Sameer Shende <u>sameer@cs.uoregon.edu</u> Performance Research Laboratory University of Oregon http://tau.uoregon.edu



# TAU tutorial exercise objectives

- Familiarise with usage of TAU tools
  - complementary tools' capabilities & interoperability
- Prepare to apply tools productively to your applications(s)
- Exercise is based on a small portable benchmark code
  - unlikely to have significant optimisation opportunities
- Optional (recommended) exercise extensions
  - analyse performance of alternative configurations
  - investigate effectiveness of system-specific compiler/MPI optimisations and/or placement/binding/affinity capabilities
  - investigate scalability and analyse scalability limiters
  - compare performance on different HPC platforms

■ ...

# Local Installation (Froggy BullX DLC)

- Setup preferred program environment compilers
  - Default set Intel Compilers with Intel MPI
  - GCC+OpenMPI and Intel + BullxMPI also available

```
% source /applis/site/env.bash
```

- % module load intel-devel
- % module use /home/PROJECTS/pr-vi-hps-tw18/opt/mf

```
% module load tau
```

 Copy tutorial sources to your working directory, ideally on a parallel file system (scratch: /scratch/\$USER)

% cd /scratch/\$USER)
% tar zxvf /home/PROJECTS/pr-vi-hps-tw18/tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI

# **NPB-MZ-MPI Suite**

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - Available from:

# http://www.nas.nasa.gov/Software/NPB

- 3 benchmarks in Fortran77
- Configurable for various sizes & classes
- Move into the NPB3.3-MZ-MPI root directory

% <b>ls</b>					
bin/	common/	jobscript/	Makefile	README.install	SP-MZ/
BT-MZ/	config/	LU-MZ/	README	README.tutorial	sys/

- Subdirectories contain source code for each benchmark
  - plus additional configuration and common code
- The provided distribution has already been configured for the tutorial, such that it's ready to "make" one or more of the benchmarks and install them into a (tool-specific) "bin" subdirectory

### **Building an NPB-MZ-MPI Benchmark**



 Type "make" for instructions

#### **Building an NPB-MZ-MPI Benchmark**

```
% make bt-mz CLASS=C NPROCS=8
make[1]: Entering directory `BT-MZ'
make[2]: Entering directory `sys'
cc -o setparams setparams.c -lm
make[2]: Leaving directory `sys'
../sys/setparams bt-mz 8 C
make[2]: Entering directory `../BT-MZ'
Mpiifort -c -O3 -q -openmp
                            bt.f
                                     [...]
Mpiifort -c -O3 -q -openmp mpi setup.f
cd ../common; mpiifort -c -O3 -q -openmp
                                                 print results.f
cd ../common; mpiifort -c -O3 -g -openmp
                                                 timers.f
Mpiifort -03 -g -openmp -o ../bin/bt-mz_C.8 bt.o
initialize.o exact solution.o exact rhs.o set constants.o adi.o
rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o solve subs.o
 z_solve.o add.o error.o verify.o mpi_setup.o ../common/print_results.o
 ../common/timers.o
make[2]: Leaving directory `BT-MZ'
Built executable .../bin/bt-mz C.8
make[1]: Leaving directory `BT-MZ'
```

- Specify the benchmark configuration
   benchmark name: bt-mz, lu-mz, sp-mz
  - the number of MPI processes: NPROCS=8
  - the benchmark class
     (S, W, A, B, C, D, E):
     CLASS=C

#### Shortcut: % make suite

# NPB-MZ-MPI / BT (Block Tridiagonal Solver)

- What does it do?
  - Solves a discretized version of the unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Performs 200 time-steps on a regular 3-dimensional grid
- Implemented in 20 or so Fortran77 source modules
- Uses MPI & OpenMP in combination
  - 8 processes each with 4 threads should be reasonable for 2 compute nodes
  - bt-mz\_C.8 should take around 40 seconds

#### **NPB-MZ-MPI / BT Reference**

```
% cd bin
% cp ../jobscript/froggy/run.oar .
% less run.oar
% oarsub -S ./run.oar
% cat test <job id>
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones: 16 x 16
Iterations: 200 dt: 0.000100
Number of active processes:
                               8
Total number of threads: 32 (
                                     4.0 threads/process)
Time step
            1
Time step
            20
 [...]
Time step 180
Time step 200
Verification Successful
BT-MZ Benchmark Completed.
Time in seconds = 30.14
```

 Copy jobscript and launch as a hybrid MPI+OpenMP application

Hint: save the benchmark output (or note the run time) to be able to refer to it later

### **NPB-MZ-MPI / BT Execution with TAU**

% oarsub −S ./tau.oar % . /applis/site/env.bash % module load intel-devel % module use /home/PROJECTS/pr-vi-hps-tw18/opt/mf % module load tau % cat test\_<jobid> Number of zones:  $16 \times 16$ Iterations: 200 dt: 0.000100 Number of active processes: 8 Use the default load factors with threads Total number of threads: 32 ( 4.0 threads/process) Calculated speedup = 31.98 Time step 1 Time step 20 ••• BT-MZ Benchmark Completed. Class C Size = 480x 320x 28 Iterations 200 = Time in seconds = 31.85

 Changes: Load the TAU module and use tau\_exec –ompt before the name of the executable.

Compare the uninstrumented time to the TAU execution time to assess overhead.

# VI-HPS

#### tau\_exec

\$ tau_e	xec		
Usage:	tau_exec [opti	ons] [] <exe> <exe options=""></exe></exe>	Tau_exec preloads
Options	:		the TAU wrapper
	-v	Verbose mode	
	-s	Show what will be done but don't actually do anything (dryrun)	libraries and
	-qsub	Use qsub mode (BG/P only, see below)	
	-10	Track I/O	performs
	-memory	Track memory allocation/deallocation	
	-memory_debug	Enable memory debugger	measurements.
	-cuda	Track GPU events via CUDA Track CDU events via CUDTI (Algo goo ony variable TAU CUDTI ADI)	
	-cupti	Track CDU events via COPII (AISO SEE ENV. Variable IAU_COPII_API)	
	-openado	Track GPU events via OpenACC (gurrently DGI only)	
	-openace	Track OpenMD events via OMDT interface	
	-ompc -armai	Track APMCI events via DAPMCI	
		Fnable event-based sampling	
	-ebs period=	counts Sampling period (default 1000)	
	-ebs_perroe=	counters Counter (default itimer)	
	-11m	Enable Unified Memory events via CUPTI	
	-T <disable.(< td=""><td>NULICPC.MPT.OMPT.OPENMP.PAPT.PDT.PROFILE.PTHREAD.SCOREP.SERIAL&gt; : Specify TAU t</td><td>ags</td></disable.(<>	NULICPC.MPT.OMPT.OPENMP.PAPT.PDT.PROFILE.PTHREAD.SCOREP.SERIAL> : Specify TAU t	ags
	-loadlib= <fil< td=""><td>e.so&gt; : Specify additional load library</td><td></td></fil<>	e.so> : Specify additional load library	
	-XrunTAUsh-<	options> : Specify TAU library directly	
	-qdb	Run program in the gdb debugger	
	5	No need	to recompile the application!
Notes:			
	Defaults i	f unspecified: -T MPI	
	MPI is ass	numed unless SERIAL is specified	

#### tau\_exec Example (continued)

```
Example:
    mpirun -np 2 tau exec -T icpc,ompt,mpi -ompt ./a.out
    mpirun -np 2 tau exec -io ./a.out
Example - event-based sampling with samples taken every 1,000,000 FP instructions
    mpirun -np 8 tau exec -ebs period=1000000 -ebs source=PAPI FP INS ./ring
Examples - GPU:
    tau_exec -T serial, cupti - cupti ./matmult (Preferred for CUDA 4.1 or later)
    tau exec -openacc ./a.out
   tau_exec -T serial -opencl ./a.out (OPENCL)
    mpirun -np 2 tau exec -T mpi, cupti, papi -cupti -um ./a.out (Unified Virtual Memory in CUDA 6.0+)
qsub mode (IBM BG/Q only):
    Original:
      qsub -n 1 --mode smp -t 10 ./a.out
    With TAU:
      tau exec -qsub -io -memory -- qsub -n 1 ... -t 10 ./a.out
Memory Debugging:
    -memory option:
     Tracks heap allocation/deallocation and memory leaks.
    -memory debug option:
     Detects memory leaks, checks for invalid alignment, and checks for
      array overflow. This is exactly like setting TAU TRACK MEMORY LEAKS=1
      and TAU_MEMDBG_PROTECT_ABOVE=1 and running with -memory
```

 tau\_exec can enable event based sampling while launching the executable using the -ebs flag!

# TAU Analysis Tools: paraprof

	TAU: ParaProf	Vlanager	
	Applications	TrialField	Value
Launch paraprof	Standard Applications	Name	bt_ompt.ppk
		Application ID	0
		Experiment ID	0
	V Contraction Contraction Contraction	Trial ID	0
% paraprof	🔻 🤪 bt_ompt.ppk	CPU Cores	8
o Parapror	∠ S TIME	CPU MHz	2600.000
	Default (idhc:h2:/lisers/sameer/ParaProf/perfdmf/perfdmf:AUTO_SERVER=TRUE)	CPU Type	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
		CPU Vendor	GenuineIntel
		CWD	/scratch/sameer/NPB3.3-MZ-MPI/bin
		Cache Size	20480 КВ
		Command Line	./bt-mz_C.8
		Executable	/scratch/sameer/NPB3.3-MZ-MPI/bin/bt-mz_C.8
		File Type Index	0
		File Type Name	ParaProf Packed Profile
Metric		Hostname	frog9
		Local Time	2015-05-18T00:37:38+02:00
		MPI Processor Name	frog9
		Memory Size	65944056 kB
		Node Name	frog9
		OMP_CHUNK_SIZE	1
		OMP_DYNAMIC	off
		OMP_MAX_THREADS	4
		OMP_NESTED	off
		OMP_NUM_PROCS	4
		OMP_SCHEDULE	UNKNOWN
		OS Machine	x86_64
		OS Name	Linux
		OS Release	2.6.32-279.5.2.bl6.Bull.33.x86_64
		OS Version	#1 SMP Sat Nov 10 01:48:00 CET 2012

#### Paraprof main window



### Paraprof main window



### Paraprof main window



Each routine occupies its own space. Can see the extent of imbalance across all threads. \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* × × × × × × × × VIRTUAL INSTITUTE – HIGH PRODUCTIVITY SUPERCOMPUTING

#### Paraprof node window (function barchart window)

	TAU: ParaProf: node 0, thread 1 - bt_ompt.ppk		
	Metric: TIME Value: Exclusive Units: seconds		
Exclusive time spent in each code region (OpenMP loop) is shown here for MPI rank 0 thread 1	8.214 8.038 7.899 3.549 2.223 0.006 0.007 0.		
Exclusive time spent in each code region (OpenMP loop) is shown here for MPI rank 0 thread 1	3.549 0.001		

### **Paraprof 3D visualization window**



# Using Event Based Sampling (EBS) in TAU

#### Edit tau.oar to uncomment line with tau\_exec –ebs and comment previous line

```
% vi tau.oar
# Use tau_exec to launch the binary
mpiexec.hydra -genvall -n $NPROCS tau_exec -T mpi,ompt,pdt,papi,icpc -ompt $EXE
# Then launch it with -ebs enabled to get profiles that contain event based samples
# Please uncomment the lines below to get callpaths that contain the samples and show
# the calleer-callee relationships.
#export TAU_CALLPATH=1
#export TAU_CALLPATH=10
#mpiexec.hydra -genvall -n $NPROCS tau_exec -T mpi,ompt,pdt,papi,icpc -ompt -ebs $EXE
```

% cd bin
% oarsub -S tau.oar
% oastat -u \$USER
% paraprof

Still no modification to binary or source code. No need to recompile !

VICTOR COMPUTING

### **Paraprof Thread Statistics Table**



0

Co

inte

sam

### **Statement Level Profiling with TAU**

	0.0	IX TAU: ParaProf: Source Browser: /scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/x_solve.f
	353 354 355 356 357 358 359 359 359 359	<pre>call matmul_sub(lhs(1,1,aa,i),</pre>
	361 362 363 364 365 366 366 366 368 368 369	<pre>C</pre>
Courses	370 371 372 373	C
Source	374 375	c B(isize) = B(isize) - C(isize-1)*A(isize)
location where	376 377 379 380 381	C
mples are taken.	383 384 385 386 386	C
Compute ntensive -	388 389 390 391 392	<pre>c back solve: if last cell, then generate U(isize)=rhs(isize) c else assume U(isize) is loaded in un pack backsub_info c so just use it c after call u(istart) will be sent to next cell c-</pre>
region.	393 394 95 397	do i=isize-1,0,-1 do m=1,BLOCK_SIZE do n=1,BLOCK_SIZE
	398 399 400 401 402	<pre>rns(m,1,j,k) = rns(m,1,j,k) &gt; - lhs(m,n,cc,i)*rhs(n,i+1,j,k) enddo enddo enddo</pre>

Right

here a

Code"

### **Paraprof Thread Statistics Table**

	TAU: ParaProf: Statistics for: node 2, thread 0 - bt_ebs.ppk				
	Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
	TAU application	1.754	36.26	1	88,049
	OpenMP_PARALLEL_REGION: L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {43,0}]	0.061	8.692	6,432	12,864
	OpenMP_IMPLICIT_TASK: L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {43,0}]	0.04	8.568	6,432	6,432
	OpenMP_LOOP: L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {43,0}]	8.528	8.528	6,432	0
	[CONTEXT] OpenMP_LOOP: L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {43,0}]	0	9.23	847	C
	SUMMARY] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f}]	3.67	3.67	340	C
	SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f}]	3.67	3.67	340	C
	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {419}]	0.22	0.22	21	C
	Show Source Column Show Source C	ar Chart 0.17	0.17	16	C
	SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {418}] Show Function H	istogram 0.16	0.16	12	C
	SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {123}] Assign Function	Color 0.11	0.11	11	C
	SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {193}]	0.08	0.08	5	C
	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {126}]	0.07	0.07	7	C
kight click	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {247}]	0.07	0.07	6	0
	SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {158}]	0.06	0.06	5	0
nere and	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {313}]	0.06	0.06	4	C
	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {230}]	0.06	0.06	4	0
choose	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {308]	0.05	0.05	3	0
"	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {191}]	0.05	0.05	3	0
"Show	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {81}]	0.05	0.05	4	C
•	[SAMPLE] L_z_solve_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f} {301}]	0.05	0.05	5	0
Source	SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_solve.t] [6/]	0.05	0.05	5	0
1.11.6	SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_solve.t] {1/5]	0.04	0.04	4	0
ode" for a	SAMPLE] L_Z_SOIVE43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_SOIVE.T] [89]	0.04	0.04	4	0
	[SAMPLE] L_Z_SOIVE_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_SOIVE.T} {55}]	0.04	0.04	4	0
sample	[SAMPLE] L_Z_SOIVE_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_SOIVE.f} {275}]	0.04	0.04	4	0
	■[SAMPLE] L_Z_SOIVE_43_par_region0_2_44 [{/scratch/sameer/NPB3.3-MZ-MPI/B1-MZ/Z_SOIVE.f} {129}]	0.04	0.04	4	0
	$ = [SAMPLE] L_Z_SOIVe_43_par_regionU_Z_44 [{/scratch/sameer/NPB3.3-MZ-MPI/BI-MZ/Z_SOIVe.f} {168}] $	0.04	0.04	4	0
	[SAMPLE] L_Z_SOIVE_43_par_regionU_2_44 [{/SCratch/Sameer/NPB3.3-MZ-MPI/B1-MZ/Z_SOIVE.T} {238}]	0.04	0.04	4	0

# **TAU Source Instrumentation**

- Edit config/make.def to adjust build configuration
  - Uncomment specification of compiler/linker: MPIF77 = tau\_f90.sh
- Make clean and build new tool-specific executable

```
% make clean
% make bt-mz CLASS=C NPROCS=8
Built executable ../bin.tau/bt-mz_C.8
```

 Change to the directory containing the new executable before running it with the desired tool configuration

```
% cd bin.tau
% cp ../jobscript/froggy/run.oar .
% oarsub -S run.oar
```

### NPB-MZ-MPI / BT: config/make.def

<pre># SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS. # # #</pre>	
# Configured for generic MPI with GCC compiler	
#OPENMP = -fopenmp # GCC compiler OPENMP = -openmp # Intel compiler	Default (no instrumentation)
 # # The Fortran compiler used for MPI programs #	
<pre>MPIF77 = mpiifort # Intel compiler</pre>	
<pre># Alternative variant to perform instrumentation #MPIF77 = tau_f90.sh -tau_makefile=\$(TAU)/Makefile.tau-[options]</pre>	
<pre># PREP is a generic preposition macro for instrumentation preparation #MPIF77 = \$(PREP) mpif77 -f77=ifort</pre>	Uncomment TAU's compiler wrapper to do source instrumentation with TAU

#### **Source Instrumentation with TAU**

% make clean % make suite % cd bin.tau % cp ../jobscript/froggy/run.oar . % oarsub -S ./run.oar % oarstat -u \$USER % paraprof

#### Instrumenting Source Code with PDT and Opari

