# The TAU Performance System: Advances in Performance Mapping

### Sameer Shende University of Oregon





John von Neumann - Institut für Computing

Zentralinstitut für Angewandte Mathematik



## Outline

- $\Box$  Introduction
- Motivation for performance mapping
- □ SEAA model
- □ Examples:
  - O POOMA II
  - 0 Uintah
- Conclusions

### Motivation

- Complexity
- Layered software
- Multi-level instrumentation
- Entities not directly in source
- Mapping
- User-level abstractions



### Hypothetical Mapping Example

### Particles distributed on surfaces of a cube





### Hypothetical Mapping Example Source

```
Particle* P[MAX]; /* Array of particles */
int GenerateParticles() {
 /* distribute particles over all faces of the cube */
 for (int face=0, last=0; face < 6; face++) {</pre>
   /* particles on this face */
   int particles on this face = num(face);
   for (int i=last; i < particles on this face; i++) {</pre>
     /* particle properties are a function of face */
     P[i] = ... f(face);
   last+= particles on this face;
```

### Hypothetical Mapping Example (continued)

```
int ProcessParticle(Particle *p) {
 /* perform some computation on p */
int main() {
 GenerateParticles();
 /* create a list of particles */
 for (int i = 0; i < N; i++)
   /* iterates over the list */
   ProcessParticle(P[i]);
```

How much time is spent processing face *i* particles?
 What is the distribution of performance among faces?

## No Performance Mapping versus Mapping

- Typical performance tools report performance with respect to routines
- Do not provide support for mapping

	, <i>c,t</i> 0,0,0	profile			•
File	<u>V</u> alue	<u>O</u> rder	<u>M</u> ode	<u>U</u> nits	<u>H</u> elp
98.00	%		2	n,c,t 0,0,0 ProcessParticles() 2.00% [] GenerateParticles() main()	
				close	

Performance tools with SEAA mapping can observe performance with respect to scientist's programming and problem abstractions



n	,c,t 0,0,0	profile			• 🗋
<u>F</u> ile	<u>¥</u> alue	<u>O</u> rder	<u>M</u> ode	<u>U</u> nits	<u>H</u> elp
79.39	%	10	0.00% 6.50% 2.00% 2.00% 0.06% 0.05%	n,c,t 0,0,0 Cost of processing face #6 Cost of processing face #5 Cost of processing face #1 Cost of processing face #4 GenerateParticles() Cost of processing face #2 Cost of processing face #3 main()	
				close	

### Semantic Entities/Attributes/Associations

- □ New dynamic mapping scheme SEAA
  - $\ensuremath{\circ}$  Entities defined at any level of abstraction
  - $\boldsymbol{\circ}$  Attribute entity with semantic information
  - ${\rm O}$  Entity-to-entity associations
- □ Two association types:
  - Embedded extends data structure of associated object to store performance measurement entity
  - External creates an external look-up table using address of object as the key to locate performance measurement entity

### Tuning and Analysis Utilities (TAU)

- Performance system framework for scalable parallel and distributed high-performance computing
- General complex system computation model
   o nodes / contexts / threads
  - O Multi-level: system / software / parallelism
  - $\odot$  Measurement and analysis abstraction
- Integrated toolkit for performance instrumentation, measurement, analysis, and visualization
  - Portable performance profiling/tracing facility

### **TAU Performance System Architecture**



### Multi-Level Instrumentation in TAU

- □ Uses multiple instrumentation interfaces
- Shares information: cooperation between interfaces
- □ Targets a common performance model
- Taps information at multiple levels
   o source (manual annotation)
  - o preprocessor (PDT, OPARI/OpenMP)
  - o compiler (instrumentation-aware compilation)
  - O library (MPI wrapper library)
  - o runtime (DyninstAPI[U.Wisc, U.Maryland])
  - 0 virtual machine (JVMPI [Sun])

### Program Database Toolkit (PDT)



### Performance Mapping in TAU

Supports both embedded and external associations:



## TAU Mapping API

- □ Source-Level API
  - TAU\_MAPPING(statement, key);
     TAU\_MAPPING\_OBJECT(funcIdVar);
     TAU\_MAPPING\_LINK(funcIdVar, key);
  - TAU\_MAPPING\_PROFILE (funcIdVar);
     TAU\_MAPPING\_PROFILE\_TIMER(timer, funcIdVar);
     TAU\_MAPPING\_PROFILE\_START(timer);
     TAU\_MAPPING\_PROFILE\_STOP(timer);

## Mapping in POOMA II

- POOMA [LANL] is a C++ framework for Computational Physics
- Provides high-level abstractions:
  - Fields (Arrays), Particles, FFT, etc.
- Encapsulates details of parallelism, datadistribution
- Uses custom-computation kernels for efficient expression evaluation [PETE]
- Uses vertical-execution of array statements to re-use cache [SMARTS]

### POOMA II Array Example

Binclude "Pooma/Arrays.h"

```
#include <iostream.h>
```

```
// The size of each side of the domain.
const. int. N = 3*1024:
```

#### lint main(

```
int
char *
```

// argument count argv[] // argument list

```
3(
```

// Initialize Pooma. Pooma::initialize(argc, argv):

```
// The array we'll be solving for
Array(2) A(\bar{N}, N), B(N,N), C(\bar{N},N), D(N,N), E(N,N);
```

argc,

// Must block since we're doing some scalar code (see Tutorial 4). Pooma::blockAndEvaluate();

```
A = 1.0:
B = 2.0:
C = 3.0:
D = 4.0:
E = 5.0:
A = B + C + D:
C = E - A + 2.0 * B:
D = A + C:
C = D + A - B:
A = 2.0 * D + E:
E = 1.5 * B - A :
```

Pooma::blockAndEvaluate():

```
cout << "D(1,1) = " << D(1,1) << endl:
cout << "D(9,9) = " << D(9,9) << endl;
```

```
// Clean up Pooma and report success.
Pooma::finalize():
return 0:
```

n Multidimensional array statements  $\Box$  A=B+C+D:

POOMA, PETE and SMARTS



# Using Synchronous Timers

BACY •	<i>n,c</i>	;,t 0,0,0 pr	ofile		•
<u>F</u> ile <u>C</u> onfigure <u>H</u> elp	<u>F</u> ile	<u>V</u> alue	<u>O</u> rder	<u>M</u> ode	Units Help
Functions         mean       Image: Comparison of the second s	99.81%	6		0.14% 0.02%	n,c,t 0,0,0 void Pooma::blockAndEvaluate() int main(int, char **) bool Pooma::finalize(bool) Inform &Inform::Inform(const char *, Inf bool Pooma::initialize(const Pooma::Op C = E - A + 2.0 * B; Pooma::Options &Pooma::Options::Opt
n,c,t 0,0,1 profile         File       Value       Order       Mode       Units         n,c,t 0,0,1       run ExpressionKernel <array< td="">         14.89%       run ExpressionKernel<array< td="">         14.51%       run ExpressionKernel<array< td="">         14.20%       run ExpressionKernel<array< td="">         10.80%       run ExpressionKernel<array< td="">         9.69%       run ExpressionKernel<array< td="">         9.16%       run ExpressionKernel<array< td="">         0.04%       schedule_private() void ()         _startoff() void (Thread *)</array<></array<></array<></array<></array<></array<></array<>	<2, View<2, View<2, View<2, View<2, View<2, View<2, View<2, View	w0 <array w0<array w0<array w0<array w0<array w0<array< th=""><th>• <u>H</u>elp 2, d 2, d 2, d 2, d 2, d 2, d 2, d 2, d</th><th></th><th>A = 1.0; bool Pooma::finalize() void Pooma::debugLevel(int) Inform &amp;Inform::Inform(const char *, sto Inform::ID_t Inform::open(Inform::Conte void Inform::setup(const char *) Inform void Inform::setOutputLevel(Inform::Le bool Pooma::initialize(int &amp;, char **&amp;, bc Pooma::Scheduler_t &amp;Pooma::schedule A = B + C + D; C = D + A - B; E = 1.5 * B - A ; A = 2.0 * D + E ; void Pooma::-::cleanup_s() D = A + C; Pooma::Options &amp;Pooma::Options::Opt B = 2.0</th></array<></array </array </array </array </array 	• <u>H</u> elp 2, d 2, d 2, d 2, d 2, d 2, d 2, d 2, d		A = 1.0; bool Pooma::finalize() void Pooma::debugLevel(int) Inform &Inform::Inform(const char *, sto Inform::ID_t Inform::open(Inform::Conte void Inform::setup(const char *) Inform void Inform::setOutputLevel(Inform::Le bool Pooma::initialize(int &, char **&, bc Pooma::Scheduler_t &Pooma::schedule A = B + C + D; C = D + A - B; E = 1.5 * B - A ; A = 2.0 * D + E ; void Pooma::-::cleanup_s() D = A + C; Pooma::Options &Pooma::Options::Opt B = 2.0
close					close

### Form of Expression Templates in POOMA

10.3 2.064 2.064 2064221 run Expr 0 essionKernel<Array<2, View0<Array<2, double, Brick>::This\_t>::NewT\_t, View0 <Array<2, double, Brick>::This\_t>::NewEngineTag\_t>, OpAssign, ConstArray<2,</pre> View0<ConstArray<2, MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultipl</pre> y, Scalar<double>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t >>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::T\_t, Expre ssionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<do uble>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>, Referenc e<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::Tree\_t>>::This\_t>::New T\_t, ViewO<ConstArray<2, MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMul tiply, Scalar<double>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLe af\_t>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::T\_t, E xpressionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scala r<double>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>, Refe rence<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::Tree\_t>>::This\_t>: :NewEngineTag\_t>, KernelTag<View0<Array<2, double, Brick>::This\_t>::Type\_t, View0<ConstArray<2, MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultipl</pre> y, Scalar<double>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t >>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::T\_t, Expre ssionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<do uble>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>, Referenc e<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf\_t>>>::Tree\_t>>::This\_t>::Typ e\_t>::Kernel\_t>

### Mapping Problem

- □ One-to-many upward mapping
- Traditional methods of mapping (ammortization/aggregation) lack resolution and accuracy!



## POOMA II Mappings

- Each work packet belongs to an ExpressionKernel object
- Each statement's form associated with timer in the constructor of ExpressionKernel
- ExpressionKernel class extended with embedded timer
- Timing calls and entry and exit of run() method start and stop per object timer

## **Results of TAU Mappings**

### Per-statement profile!

n,c,t 0,0,1 profile			•							
<u>F</u> ile <u>V</u> alue <u>O</u> rder	<u>M</u> ode	<u>U</u> nits	<u>H</u> elp							
	n,c,t 0,	0,1	A							
15.63%	C = E -	- A + 2.0 ^ B;								
12.09%	A = B ·	+ C + D;								
11.97%	C = D ·	+ A – B;								
11.37%	E = 1.5	5*B-A;								
11.10%	A = 2.0	)*D+E;								
9.71%	D = A -	+ C;								
5.67%	A = 1.0	);								
5.67%	D = 4.0	);								
5.60%	B = 2.0									
5.59%	C = 3.0	);								
5.57%	E = 5.0									
0.02%	sched	ule private() void ()								
	lterate	<fastasync>::execute() void ()</fastasync>								
	starte	off() void (Thread *)								
			11							
close										





□ Helps bridge the semantic-gap!

### Uintah

- □ U. of Utah, C-SAFE ASCI Level 1 Center
- Component-based framework for modeling and simulation of the interactions between hydrocarbon fires and high-energy explosives and propellants [Uintah]
- Work-packets belong to a higher-level task that a scientist understands
  - 0 e.g., "interpolate particles to grid"

### Without Mapping

BACY Task execution [MPIScheduler::execute()] profile • • [ Configure Help File Value Mode File Units Help Task execution [MPIScheduler::execute()] **Functions** 84.26% mean mean Function Legend • 🔳 79.91% n.c.t 0.0.0 MPI Type indexed() n.c.t 0.0.0 81.36% n.c.t 1.0.0 MPI Type size() n,c,t 1,0,0 81.91% n.c.t 2.0.0 MPI Type vector() n,c,t 2,0,0 86.63% n.c.t 3.0.0 MPI Waitall() n.c.t 3.0.0 81.59% n.c.t 4.0.0 Recv Dependency [MPI n,c,t 4,0,0 89.58% n.c.t 5.0.0 Reductions [MPISchedu n,c,t 5,0,0 84.00% n,c,t 6,0,0 Send Dependency [MPI n,c,t 6,0,0 89.11% n.c.t 7.0.0 Task Graph Output [MP n,c,t 7,0,0 Task execution [MPISch Test Some [MPISchedu Topological Sort [MPISc mean profile main() void (int, char \*\*) • Help File Value Order Mode close mean 84.26% Task execution [MPIScheduler::execute()] 6.77% MPI Waitall() close 3.51% MPIScheduler::gatherParticles 1.21% MPI Probe() MPIScheduler::gatherParticles profile • 0.95% MPI Finalize() File Value Mode Units Help 0.68% MPI Allreduce() 0.53% MPI Type indexed() MPIScheduler::gatherParticles 0.44% main() void (int, char \*\*) 3.51% mean 0.33% MPIScheduler::scatterParticles 0.28% Initial Send Recv [MPIScheduler::execute()] 3.97% n.c.t 0.0.0 0.18% MPIScheduler::execute() 3.26% n.c.t 1.0.0 0.14% MPI Isend() 3.51% n,c,t 2,0,0 0.14% Topological Sort [MPIScheduler::execute()] 3.32% n.c.t 3.0.0 0.13% MPI Recv() 3.39% n.c.t 4.0.0 0.10% Recv Dependency [MPIScheduler::execute()] 3.95% n.c.t 5.0.0 0.10% MPI Testsome() 3.34% n,c,t 6,0,0 0.08% MPI Irecv() 3.33% n.c.t 7.0.0 0.06% Send Dependency [MPIScheduler::execute()] 0.06% MPI Init()  $\triangleleft$ close close

### Using External Associations

- When task is created, a timer is created with the same name
- Two level mappings:
   Level 1: <task name, timer>
   Level 2: <task name, patch, timer>

## Using Task Mappings



# **Tracing Uintah Execution**

- VAMPIR - Global Timeline																				
					S	us32_	bigb	ar.p	ov: G	lobal	Time	line (1	0.0 u	s - 40	.29	3 s =	40.29	3 s)		
	er	E.E.	10.0 s	E.C.	E.E.	40	2	:0.0	S	40	E.C.		30.	JS		EE.	40	ce.	40.0	IDIF
Process U 26 00	60	00	49	00	00	49	00		00	49	00	00	49	90		99	49	00	20	TAU_DEFAULT
Process 1 28 65		33 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 2 28 65		33 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	SerialMPM::actuallyInitialize
Process 3 28 65		3 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Serial MPM::interpolateParticlesToGrid
Process 4 28 65		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Contact::exMomInterpolated
Process 5 28 65		1 <mark>3</mark> 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Serial MPM::compute Stress Tensor
Process 6 28 65		35	43 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Serial MPM::compute Internal Heat Rate
Process 7 28 65		15	43 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Serial MPM::solveEquations Motion
Process 8 28 65		80 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Serial MPM::integrateAcceleration
Process 9 28 65		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	Contact::exMom Integrated
Brooses 10 22 65		242 EE	40	56	55	40	50		EE.		50	55	40	56		EE	40	CC.	26	SerialMPM::interpolateToParticlesAndUpdate
Plucess IU 20 00		20 20 00 FF	49	00	00	43	00		00 55	49	00	00	43	00		00 65	49	00	20	
Process 11 28 65		53 55	49	56	55	49	55		55	49	56	55	49	56		55	49	56	26	
Process 12 28 65		43 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 13 28 65		43 55	49	56	55	49	56	ł	55	49	56	55	49	56		55	49	56	26	
Process 14 <mark>28 65</mark>		55	<b>43</b> 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 15 <mark>2865</mark>		55	43 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 16 <mark>28 65</mark>		<mark>43</mark> 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 17 28 65		<b>43</b> 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 18 28 65		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 19 28 65		43 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 20 28 65		43 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 21 28 65		43 55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Brocoss 22 28 65		55	40	56	55	40	56		55	10	56	55	/0	56		55	10	56	26	
Process 22 20 00		55	42 40	00	55	40	50		55	40	50		40	50		55	40	ou re	20	
Process 23 26 00		00	40 40	00	00	49	00		00 55	49	00	00	49	00		00 55	49	00	20	
Process 24 28 65		55	43	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 25 28 65		55	43 9	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 26 28 65		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 27 <mark>28 65</mark>		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 28 <mark>28 65</mark>		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 29 <mark>28 65</mark>		55	49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 30 28 65		55	<b>43</b> 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	
Process 31 28 65		55	<b>43</b> 49	56	55	49	56		55	49	56	55	49	56		55	49	56	26	

### Two-Level Mappings: Tasks+Patch



### Conclusions

- □ New performance mapping model (SEAA)
- □ Application of SEAA to:
  - o asynchronously executed work packets in POOMA
    o packet-task-patch mapping in Uintah
- Mapping performance data helps bridge the gap in understanding performance data
- □ Complex mapping problems

O cross-context mapping

### Information

- □ TAU (http://www.acl.lanl.gov/tau)
- PDT (http://www.acl.lanl.gov/pdtoolkit)
- Tutorial at SC'01: M11
   B. Mohr, A. Malony, S. Shende, "Performance Technology for Complex Parallel Systems" Nov. 7, 2001, Denver, CO.
- □ LANL, NIC Booth, SC'01.

### Support Acknowledgement

TAU and PDT support:
 Department of Engergy (DOE)
 DOE 2000 ACTS contract
 DOE MICS contract
 DOE ASCI Level 3 (LANL, LLNL)

O DARPA

 $\odot$  NSF National Young Investigator (NYI) award

