The TAU Performance System: Advances in Performance Mapping

Sameer Shende
University of Oregon

Tuning and Analysis Utilities
Outline

- Introduction
- Motivation for performance mapping
- SEAA model
- Examples:
  - POOMA II
  - 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
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++) {
        /* particles on this face */
        int particles_on_this_face = num(face);
        for (int i=last; i < particles_on_this_face; i++) {
            /* particle properties are a function of face */
            P[i] = ... f(face);
            ...
        }
        last+= particles_on_this_face;
    }
}
Hypothetical Mapping Example (continued)

```c
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

- Performance tools with SEAA mapping can observe performance with respect to scientist’s programming and problem abstractions
Semantic Entities/Attributes/Associations

- New dynamic mapping scheme - SEAA
  - Entities defined at any level of abstraction
  - Attribute entity with semantic information
  - 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
• **Performance system framework** for scalable parallel and distributed high-performance computing

• **General complex system computation model**
  - nodes / contexts / threads
  - Multi-level: system / software / parallelism
  - Measurement and analysis abstraction

• **Integrated toolkit** for performance instrumentation, measurement, analysis, and visualization
  - Portable performance profiling/tracing facility
Multi-Level Instrumentation in TAU

- Uses multiple instrumentation interfaces
- Shares information: cooperation between interfaces
- Targets a common performance model
- Taps information at multiple levels
  - source (manual annotation)
  - preprocessor (PDT, OPARI/OpenMP)
  - compiler (instrumentation-aware compilation)
  - library (MPI wrapper library)
  - runtime (DyninstAPI[U.Wisc, U.Maryland])
  - virtual machine (JVMPi [Sun])
Program Database Toolkit (PDT)
Performance Mapping in TAU

- Supports both **embedded** and **external** associations:

**Embedded association**

**External association**

![Diagram showing 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);
POOMA [LANL] is a C++ framework for Computational Physics

- Provides high-level abstractions:
  - Fields (Arrays), Particles, FFT, etc.
- Encapsulates details of parallelism, data-distribution
- Uses custom-computation kernels for efficient expression evaluation [PETE]
- Uses vertical-execution of array statements to re-use cache [SMARTS]
POOMA II Array Example

```cpp
#include "Pooma/Arrays.h"
#include <iostream.h>

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

int main(
    int argc, // argument count
    char * argv[]) // argument list
)
{
    // Initialize Pooma.
    Pooma::initialize(argc, argv);

    // The array we'll be solving for
    Array<2> A(N, N), B(N,N), C(N,N), D(N,N), E(N,N);

    // 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;
}
```

- Multi-dimensional array statements
- \( A = B + C + D; \)
POOMA, PETE and SMARTS

\[ A = B + C + D; \]
\[ C = E - A + 2 \times B; \]
Using Synchronous Timers

```cpp
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
A = 1.0;
bool Pooma::finalize()
void Pooma::debugLevel(int)
Inform &Inform::Inform(const char *, std
Inform::id_t Inform::open(Inform::Conte
void Inform::setup(const char *) Inform
void Inform::setOutputLevel(Inform::Le
bool Pooma::initialize(int &, char **, bo
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;
```
Form of Expression Templates in POOMA

```cpp
10.3 2.064 2.064 1 0 2064221 run ExpressionKernel<Array<2, View0<Array<2, double, Brick>::This_t>::NewT_t, View0<Array<2, double, Brick>::This_t>::NewEngineTag_t>, OpAssign, ConstArray<2, View0<ConstArray<2, MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<double>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf_t>>>>::T_t, ExpressionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<double>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf_t>>>>::Tree_t>::::This_t>::::NewT_t, View0<ConstArray<2, MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<double>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf_t>>>>::T_t, ExpressionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<double>>, Reference<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<OpMultiply, Scalar<double>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf_t>>>>::T_t, ExpressionTag<MakeReturn<BinaryNode<OpSubtract, BinaryNode<OpMultiply, Scalar<double>>, Reference<ArrayCreateLeaf<2, double, Brick>::ArrayLeaf_t>>>>::Tree_t>::::This_t>::::Type_t::Kernel_t
```
Mapping Problem

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

```cpp
template <class LHS, class RHS, class Op, class EvalTag>
void ExpressionKernel<LHS,RHS,Op,EvalTag>::run()
{
    /* iterate execution */
}
```

```plaintext
A = 1.0;
B = 2.0;
...
A = B + C + D;
C = E - A + 2.0*D;
...```
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!
POOMA Traces

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
  - e.g., “interpolate particles to grid”
Without Mapping
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

<table>
<thead>
<tr>
<th>Process</th>
<th>10.0 s</th>
<th>20.0 s</th>
<th>30.0 s</th>
<th>40.0 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23 65</td>
<td>23 65</td>
<td>23 65</td>
<td>23 65</td>
</tr>
<tr>
<td>1</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>2</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>3</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>4</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>5</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>6</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>7</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>8</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>9</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>10</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>11</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>12</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>13</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>14</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>15</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>16</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>17</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>18</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>19</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>20</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>21</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>22</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>23</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>24</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>25</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>26</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>27</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>28</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>29</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>30</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
<tr>
<td>31</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
<td>23 55</td>
</tr>
</tbody>
</table>
Two-Level Mappings: Tasks+Patch
Conclusions

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

- TAU (http://www.acl.lanl.gov/tau)
- PDT (http://www.acl.lanl.gov/pdtoolkit)
- Tutorial at SC’01: M11
- LANL, NIC Booth, SC’01.
Support Acknowledgement

- TAU and PDT support:
  - Department of Energy (DOE)
    - DOE 2000 ACTS contract
    - DOE MICS contract
    - DOE ASCI Level 3 (LANL, LLNL)
  - DARPA
  - NSF National Young Investigator (NYI) award