Performance and Environment Monitoring for Continuous Program Optimization

Evelyn Duesterwald

Joint work with
How to Sustain the Application Performance Growth Curve?

- Performance gains from technology advances are diminishing
- Complexity of the SW deployment stack is increasing
- Execution layers can no longer be optimized in isolation

**Need a new methodology for application performance delivery:**

- Tune entire application deployment stack on-site and on-demand
  - Vertical integration
  - Continuous and adaptive
Continuous Program Optimization (CPO) through Performance & Environment Monitoring (PEM)

- Continuous Program Optimization (CPO) increases programmer productivity by automating the laborious and challenging performance tuning effort.

- CPO aims at tuning application by optimally:
  - adapting the application to its behavior and environment
  - adapting environment resources to application behavior

- CPO is made possible through continuous whole-system Performance and Environment Monitoring (PEM).

![Diagram of the continuous optimization loop involving Tune & adapt (CPO) and Monitor & evaluate (PEM).]
A Catalogue of CPO Scenarios

Adapting applications to execution behavior and environment
- Library Tuning
- Adaptive MPI tuning
- Tuning Just-In-Time (JIT) compiler heuristics
- Interactive tuning environment to support algorithmic tuning

Adapting the execution environment to the application’s needs:
- Tuning the Virtual Machine (heap size, GC policies)
- Tuning the operating environment/OS (adaptive page size, adaptive file cache management (FCM))
CPO Architecture

2 Components to implement CPO scenarios:

- **PEM: Vertical Performance and Environment Monitoring**
  - Implements programmable monitoring functionality across execution layers
  - Provides integrated whole system view
- **CPO Agent** (Specific to CPO scenario)
  - programs PEM to provide desired monitoring information
  - Implements specific scenario optimization
- **There may be several CPO Agents active in the system, interconnected through PEM**

![Diagram of CPO Architecture]

- control performance knob
- negotiate knob
Performance & Environment Monitoring (PEM)

- Flexible monitoring infrastructure to integrate monitors across execution layers
- Platform-independent API (PEMAPI) to program CPO Agents
  - CPO Agent for offline event tracing and logging
  - CPO Agent for implementing CPO scenarios
- Designed to serve as integral part of CPO as well as stand-alone PEM tool set
1) Language-independent specification of events and their semantics

2) Generate language-specific event data structures and header files (Java, C, C++, Fortran) to support Layer instrumentation
   - PEM library
   - CPO agent

3) PEM implementation with a platform-independent API

PEM Infrastructure

- XML Event specification
- PEM Tools
- Application
- Virtual Machine
- O/S
- Hardware/Simulator
- CPO Agent
- PEM Library
- PEM trace
- PE2 Visualizer
Exploratory Monitoring with PEM

- Dual-path PEM Tool Set

1) Hardware Path

Explore application performance and identify performance problems

1) Simulator Path to explore

Performance effects of different solutions
- New hardware
- New monitoring functionality
PEM on a Commercial AIX Stack

- Integrate existing tracing support (e.g., AIX Trace) into PEM monitoring by intercepting existing tracing stream with PEM wrappers

```
Java App
WebSphere
JVM (J9) J9 Hook function
AIX
Hypervisor
H/W
```

PEMAPI event record
PEMAPI event record
PEMAPI event record
AIX Trace

PEM wrapper

PEM

Trace
PEM Demo
Exploring the Performance Impact of Large Pages

Preliminary Work for building a CPO Agent for Adaptive Page Sizing

Vertical Event Traces:
- App layer: phase markers
- O/S layer: page faults
- Hardware layer: PMU counters
Complete run of umt2
(initial configuration with 4K pages)
Zoom into innermost loop
/* Set angular fluxes for reflected angles */

snxyzref(&ncut, &npart, &nelem, &ndim, &nsets, &ndir,
    &m, &ntmp_ref(1, thnum), &x1, &x2, &cx,
    &quadw, &omega, &tmp_ref(1, thnum), &psib,
    &abdym_ref(1, thnum), &A_bdy);

/* Sweep the mesh, calculating PSI|C for each corner; the boundary */
/* flux array PSI_B is also updated here. */

TraceAPPAppPhaseStart(6, __LINE__, __FILE__);

snswp3d(&npart, &nelem, &nelempad, &incorr, &nfaces, &nzones,
    &nside, &ndim, &nelem, &m, &mpi, &nps, &ndir, &ipath,
    &neRef(1, m), &connec, &kktc, &kktf,
    &kkcz, &kkby, &kkztcf,
    &faces_corner, &listcf, &corner_zone,
    &omega_ref(1, m), &A_fep, &A_pcz, &A_fpz,
    &abdym_ref(1, thnum), &sigvol, &scsi_ref(1, 1, thnum),
    &scsz_ref(1, 1, thnum), &qf_ref(1, 1, thnum),
    &facewt, &zonewt, &etwt, &xactin_ref(1, 1, thnum),
    &tpsic_ref(1, 1, thnum), &psic, &psib,
    &psic_inc_ref(1, 1, thnum), 0L);

TraceAPPAppPhaseEnd(6, __LINE__, __FILE__);

/* Add this angle's contribution to the flux moments */

snmamt(&npart, &incorr, &m, &ndir, &nmomt, &quadw_ref(m),
    &ynm, &tsic_ref(1, 1, thnum), &tphic_ref(1, 1, thnum));

Application instrumentation to trace phase marker events
Complete run across OS and H/W metrics (4k pages)
<table>
<thead>
<tr>
<th>Event</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue page faults small</td>
<td>666</td>
</tr>
<tr>
<td>Blue page faults large</td>
<td>679</td>
</tr>
<tr>
<td>Blue TLB misses small</td>
<td>666</td>
</tr>
<tr>
<td>Blue TLB misses large</td>
<td>679</td>
</tr>
<tr>
<td>Blue data ERAT misses</td>
<td>19528</td>
</tr>
<tr>
<td>Blue L1 load misses small</td>
<td>666</td>
</tr>
<tr>
<td>Blue L1 load misses large</td>
<td>679</td>
</tr>
<tr>
<td>Brown page faults small</td>
<td>666</td>
</tr>
<tr>
<td>Brown page faults large</td>
<td>679</td>
</tr>
<tr>
<td>Brown TLB misses small</td>
<td>666</td>
</tr>
<tr>
<td>Brown TLB misses large</td>
<td>679</td>
</tr>
<tr>
<td>Brown data ERAT misses</td>
<td>19523</td>
</tr>
<tr>
<td>Brown L1 load misses small</td>
<td>666</td>
</tr>
<tr>
<td>Brown L1 load misses large</td>
<td>679</td>
</tr>
</tbody>
</table>

**Notes:**

- **Blue – 4k pages**
- **Brown – Initial large 16M page mapping:** map data structures of inner loop to large pages (each aligned at large page boundary)
<table>
<thead>
<tr>
<th>Structure</th>
<th>Blue 4k pages</th>
<th>Brown Initial Large Page Mapping</th>
<th>Red Optimized Large Page Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>5282.0 page faults</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>673 record pgflts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5282.0 page faults</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record pgflts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5282.0 page faults</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>387 record pgflts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4062.0 TLB misses</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record TlbMiss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4062.0 TLB misses</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record TlbMiss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19526.0 data ERAT misses</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record Datamiss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19526.0 data ERAT misses</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record Datamiss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30368.0 L1 load misses</td>
<td>0.0(Min)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record LdmMissL1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30368.0 L1 load misses</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>673 record LdmMissL1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Blue – 4k pages**

**Brown – Initial Large Page Mapping:** each structure aligned at large page boundary

**Red – Optimized large page mapping:** Offset each data structure to avoid conflicts
Demo Summary

![Chart showing comparison between small page, large page, and large page opt for time, page faults, TLB, data, ERAT, and L1 data.]
Conclusions

- **CPO** aims at automatically tuning performance by cross-layer adaptation of code and environment

- **PEM** global vertical monitoring infrastructure
  - Integral part of CPO
    - CPO scenarios are implemented as CPO Agents across the execution stack
  - Stand-alone PEM Tool Set
    - A CPO Agent for vertical performance exploration and visualization
      - Explore application behavior
    - Hardware-Software Co-design
      - Explore effect of new hardware on application performance
      - Explore new monitoring functionality