Next Generation Application Enablement Tools: A Framework for Automated Performance Analysis and Tuning

David Klepacki
Advanced Computing Technology
IBM T.J. Watson Research Center
ACTC Toolkit Activities

- **HPC Toolkit**
  - Traditional ("show-me …") framework for interactive application performance analysis
  - **Analysis-based** as opposed to tuning-based
  - IBM Parallel Environment (PE) Product 4Q08
    - Blue Gene systems still supported from IBM Research

- **HPCS Toolkit (DARPA HPCS Productivity Program)**
  - Next generation framework for **automated intelligent-assist** of application performance tuning ("tell-me …")
  - Emphasis on **Tuning**
  - “Automated”…..not automatic.
System Evolution

- Device Scaling imposing **fundamental constraints on system**
  - Power dissipation and energy consumption
  - Physical size / packaging
- **Pressure to re-think system architecture**
  - Blue Gene: low power devices, embedded (small)
  - Cell: Attached (embedded) co-processing engine
- **Systems become inherently more complex**
  - Connectivity / hierarchical topology (torus, intra-cell)
  - Multi-core processors (and less memory per processor)
  - Multi-thread (SMT, hyperthreading)
- **This poses new challenge to application programming**
  - New programming paradigm? (but ~$1T in legacy codes, ISV apps, etc.)
- **Conclusion**: **New software tools essential** to mitigate evolving system complexity and improve productivity.
Enablement Productivity Gap = Hardware – Software

![Diagram showing the productivity gap between Hardware and Software over time.](image-url)
DARPA / PERCS Impact on Productivity Gap

- **State-of-Art Application Enablement circa 2002+**
  - Source code modification (e.g., timing routines)
  - Non-selective, non-source code correlated tools (e.g., PAPI)
  - Dynamic instrumentation via external agents (e.g., DynInst)
  - GUI frameworks to look at data (e.g., Vampir, Vtune, Tau)
  - No unified analysis framework (CPU, MPI, OpenMP, and I/O)
  - No management of large scale performance data

- **IBM DARPA HPCS Toolkit**
  - Next generation unified framework for automated (not automatic) intelligent-assist of application performance tuning including…
    - No source code modifications…but with source code correlation of the data
    - Selective and dynamic instrumentation without external agents
    - Large scale data management

- **In a Nutshell:**
  - Previous tools only show you the data…does not resolve the Productivity Gap.
  - The HPCS Toolkit makes sense of the data…closes the Productivity Gap.
**High Level Design Flow for HPCS Toolkit**

- **HPCS Toolkit provides Automated Framework for Performance Analysis.**
  - Intelligent automation of performance evaluation and decision system.
  - Interactive capability with graphical/visual interface always available.

**Bottleneck:** elapsed time exceeds threshold for completing work.
HPCS Toolkit Scalability

- **Self-Contained Performance Data Collection Framework**
  - Part of the instrumented application executable
    - No background processes or external agents
    - Extensible to MRNet (University of Wisconsin)

- **Use of Parallel File System (GPFS)**
  - Data managed in parallel via distributed files
    - Up to five files per process (e.g., for each MPI task):
      1. HPM data
      2. MPI data
      3. OpenMP data
      4. Memory reference data
      5. I/O data

- **Pre-runtime and Post-runtime Filtering Capability**
  - User-defined logic to reduce data to be captured and/or analyzed

- **IBM Research Blue Gene test-bed**
  - Up to 0.5 million processor systems
Closing the Enablement Productivity Gap

HPCS Toolkit = bridge to “Super”-Compiler

Complexity

Hardware

Productivity Gap

1960
(Fortran, C)

Time

HPC Programming Languages
(Fortran, C)
Automated Performance Tuning – Timetable

**2007 Deliverables:**

- **Performance Data Collection**
  - Scalable, dynamic, programmable
  - Completely binary: no source code modification to instrument application…
  - But retains ability to correlate all performance data with source code

- **Bottleneck Discovery**
  - Make sense of the performance data
  - Mines the performance data to extract bottlenecks

**FUTURE MILESTONE DELIVERABLES:**

- **Solution Determination**
  - Make sense of the bottlenecks
  - Mines bottlenecks and suggests system solutions (hardware and/or software)
  - Assist compiler optimization (including custom code transformations)

- **Performance “Visualization”**
  - Performance Data / Bottleneck / Solution Information feedback to User
    - Logging (textual information)
    - Compiler feedback
    - Output to other tools (e.g., Kojak analysis, ParaVer visualization, Tau, etc.)
BDE Architecture

Abbreviation | Meaning
---|---
BDE | Bottleneck Detection Engine
BDE DB | Bottleneck Detection Engine Database
MMOD | Metric Module
PEMOD | Performance Estimation Module
MSCHED | Performance Scheduler
HD | Hotspot Detector

= Area A: User Interface
= Area B: Bottleneck Detection
HPCST: Extensibility and Integration

- **HPCST Framework is completely extensible**
  - Open interfaces for bottleneck detection and solution analysis
  - ANY tool can be easily integrated via Module interface
    - Tau, Paraver, Kojak, etc.
  - Extends capabilities of third-party tools by allowing access to bottleneck detection and solution analysis components of HPCS Toolkit
Example Codes and Applications

- **Standard benchmark codes/kernels**
  - SPEC SWIM (Shallow Water Model)
  - NAS Parallel Bechmarks

- **DARPA Applications**
  - MHD (Magneto Hydrodynamics)
    - Model turbulence and collision in fluids
    - Integration, collision, and stream are time consuming steps
  - HYCOM (Hybrid Ocean Model)
    - 2D ocean circulation model with hybrid coordinate scheme to improve modeling accuracy as the ocean depth varies from deep stratified water to shallow coastal regions
### HD Results (Loop Level)

#### DATA WINDOW

<table>
<thead>
<tr>
<th>SELECTION</th>
<th>FILE</th>
<th>FUNCTION</th>
<th>START</th>
<th>END</th>
<th>self secs (PO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Loop</td>
<td>mhd_F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Loop</td>
<td>mhd_F</td>
<td>collision</td>
<td>512</td>
<td>592</td>
</tr>
<tr>
<td>3</td>
<td>Loop</td>
<td>mhd_F</td>
<td>collision</td>
<td>513</td>
<td>596</td>
</tr>
<tr>
<td>4</td>
<td>Loop</td>
<td>mhd_F</td>
<td>stream</td>
<td>338</td>
<td>494</td>
</tr>
<tr>
<td>5</td>
<td>Loop</td>
<td>mhd_F</td>
<td>stream</td>
<td>339</td>
<td>395</td>
</tr>
<tr>
<td>6</td>
<td>Loop</td>
<td>mhd_F</td>
<td>stream</td>
<td>337</td>
<td>414</td>
</tr>
<tr>
<td>7</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>619</td>
<td>807</td>
</tr>
<tr>
<td>8</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>750</td>
<td>804</td>
</tr>
<tr>
<td>9</td>
<td>Loop</td>
<td>mhd_F</td>
<td>equil</td>
<td>291</td>
<td>314</td>
</tr>
<tr>
<td>10</td>
<td>Loop</td>
<td>mhd_F</td>
<td>equil</td>
<td>292</td>
<td>317</td>
</tr>
<tr>
<td>11</td>
<td>Loop</td>
<td>mhd_F</td>
<td>check</td>
<td>927</td>
<td>938</td>
</tr>
<tr>
<td>12</td>
<td>Loop</td>
<td>mhd_F</td>
<td>check</td>
<td>924</td>
<td>934</td>
</tr>
<tr>
<td>13</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>696</td>
<td>748</td>
</tr>
<tr>
<td>14</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>618</td>
<td>626</td>
</tr>
<tr>
<td>15</td>
<td>Loop</td>
<td>mhd_F</td>
<td>init</td>
<td>237</td>
<td>285</td>
</tr>
<tr>
<td>16</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>644</td>
<td>651</td>
</tr>
<tr>
<td>17</td>
<td>Loop</td>
<td>mhd_F</td>
<td>init</td>
<td>273</td>
<td>281</td>
</tr>
<tr>
<td>18</td>
<td>Loop</td>
<td>mhd_F</td>
<td>neighbours</td>
<td>681</td>
<td>688</td>
</tr>
<tr>
<td>19</td>
<td>Loop</td>
<td>mhd_F</td>
<td>my_range</td>
<td>224</td>
<td>226</td>
</tr>
<tr>
<td>20</td>
<td>Loop</td>
<td>mhd_F</td>
<td>init</td>
<td>284</td>
<td>287</td>
</tr>
<tr>
<td>21</td>
<td>Loop</td>
<td>mhd_F</td>
<td>init</td>
<td>288</td>
<td>271</td>
</tr>
<tr>
<td>22</td>
<td>Loop</td>
<td>mhd_F</td>
<td>mhd</td>
<td>31</td>
<td>145</td>
</tr>
<tr>
<td>23</td>
<td>Loop</td>
<td>mhd_F</td>
<td>mhd</td>
<td>856</td>
<td>854</td>
</tr>
<tr>
<td>24</td>
<td>Loop</td>
<td>mhd_F</td>
<td>mhd</td>
<td>117</td>
<td>128</td>
</tr>
<tr>
<td>25</td>
<td>Loop</td>
<td>mhd_F</td>
<td>mhd</td>
<td>157</td>
<td>214</td>
</tr>
<tr>
<td>26</td>
<td>Loop</td>
<td>mhd_F</td>
<td>mhd</td>
<td>898</td>
<td>806</td>
</tr>
</tbody>
</table>

#### SOURCE WINDOW

```plaintext
421    ifetch one line strides = 18 elements at a time
422    ptr += geq(ptr+1, j, 1, k)
423    CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
424    ptr += geq(ptr+1, j, 1, k)
425    CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
426    ptr += geq(ptr+1, j, 5, k)
427    CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
428    ptr += geq(ptr+1, j, 7, k)
429    CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
430    do i = imp, MIN(imp+18*NMP_PREF_LINES-1, lend)
431    #else
432    do i = lsta, lend
433    #endif
434    g(i, j, k) = geq(i-1, j, 1, k)
435    g(i, j, k) = geq(i-1, j, 3, k)
436    g(i, j, k) = geq(i-1, j, 5, k)
437    g(i, j, k) = geq(i-1, j, 7, k)
438    enddo
439    ifndef NMP
440    endif
441    ifndef
442    endif
443    do j = lsta, lend
444    #endif
```
Source Code with Clock Ticks

```c
ptr => feq[inmp, j=1, 3]
CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
ptr => feq[inmp+1, j=5]
CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
ptr => feq[inmp+1, j=7]
CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
do i = inmp, MIN(inmp+(16*NMP_PREF_LINES)-1, lend)
  #else
  do i = ista, lend
  #endif
  ! write(*,*) i, ' ', j, ' ', inmp
  f[i,j] = feq[i+1,j,1]
  f[i,j,3] = feq[i,j-1,3]
  f[i,j,5] = feq[i+1,j,5]
  f[i,j,7] = feq[i,j+1,7]
enddo
#ifdef NMP
enddo
#endif
#endif
#endif
#endif
#endif
```

! fetch one line strides = 18 elements at a time
ptr => feq[inmp, j=2]
CALL Mambo_Prefetch(ptr, 128, 1, NMP_PREF_LINES)
BDE Results

<table>
<thead>
<tr>
<th>BOTTLENECK</th>
<th>DIMEN</th>
<th>DESCRIPTION</th>
<th>RULE</th>
<th>FUNCTION</th>
<th>START LINE</th>
<th>END LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DcacheMiss</td>
<td>CPU</td>
<td>cycles wasted due to stalls</td>
<td>(PM_RUN_CYC - (PM_GCT_EMPTY_CYC + PM_GRP_CMPL))</td>
<td>stream</td>
<td>416</td>
<td>493</td>
</tr>
<tr>
<td>DcacheMiss</td>
<td>CPU</td>
<td>cycles wasted due to stalls</td>
<td>(PM_RUN_CYC - (PM_GCT_EMPTY_CYC + PM_GRP_CMPL))</td>
<td>equill</td>
<td>291</td>
<td>314</td>
</tr>
<tr>
<td>DcacheMiss</td>
<td>CPU</td>
<td>cycles wasted due to stalls</td>
<td>(PM_RUN_CYC - (PM_GCT_EMPTY_CYC + PM_GRP_CMPL))</td>
<td>neighbours</td>
<td>750</td>
<td>804</td>
</tr>
<tr>
<td>DcacheMiss</td>
<td>CPU</td>
<td>cycles wasted due to stalls</td>
<td>(PM_RUN_CYC - (PM_GCT_EMPTY_CYC + PM_GRP_CMPL))</td>
<td>collision</td>
<td>512</td>
<td>592</td>
</tr>
<tr>
<td>DcacheMiss</td>
<td>CPU</td>
<td>cycles wasted due to stalls</td>
<td>(PM_RUN_CYC - (PM_GCT_EMPTY_CYC + PM_GRP_CMPL))</td>
<td>neighbours</td>
<td>619</td>
<td>807</td>
</tr>
<tr>
<td>MPI:Barrier1</td>
<td>COMM:LL</td>
<td>imbalanced wait time</td>
<td>MAX(mpl_wlt_hot_sum_time) - MIN(mpl_wlt_hot_sum_time) &gt; 1, neighbours</td>
<td></td>
<td>619</td>
<td>807</td>
</tr>
<tr>
<td>Stalls</td>
<td>CPU</td>
<td>cycles spent in stalls</td>
<td>(PM_CMPLU_STALL_LSU + PM_CMPLU_STALL_FXU + PM_CN)</td>
<td></td>
<td>619</td>
<td>807</td>
</tr>
<tr>
<td>Stalls</td>
<td>CPU</td>
<td>cycles spent in stalls</td>
<td>(PM_CMPLU_STALL_LSU + PM_CMPLU_STALL_FXU + PM_CN)</td>
<td></td>
<td>416</td>
<td>493</td>
</tr>
<tr>
<td>Stalls</td>
<td>CPU</td>
<td>cycles spent in stalls</td>
<td>(PM_CMPLU_STALL_LSU + PM_CMPLU_STALL_FXU + PM_CN)</td>
<td></td>
<td>512</td>
<td>592</td>
</tr>
<tr>
<td>Stalls</td>
<td>CPU</td>
<td>cycles spent in stalls</td>
<td>(PM_CMPLU_STALL_LSU + PM_CMPLU_STALL_FXU + PM_CN)</td>
<td></td>
<td>750</td>
<td>804</td>
</tr>
<tr>
<td>Stalls</td>
<td>CPU</td>
<td>cycles spent in stalls</td>
<td>(PM_CMPLU_STALL_LSU + PM_CMPLU_STALL_FXU + PM_CN)</td>
<td></td>
<td>291</td>
<td>314</td>
</tr>
</tbody>
</table>
HPCS Toolkit Platform Availability

- **Application programming languages supported**
  - Fortran, C/C++
  - On Horizon: UPC, CoA Fortran, X10 ?

- **GUI environments supported**
  - Qt (Nokia/Trolltech) - C/C++ based, open source
  - Eclipse (IBM) - Java based, open source

- **Hardware platforms supported**
  - PERCS and IBM POWER servers
  - IBM Blue Gene systems
  - On Horizon: AMD and Intel-based machines, Cell, ?
Both Qt and Eclipse are Open Source and have similar functionalities.

- Eclipse is funded by the DARPA HPCS Project and is popular in Commercial environment (Rational Development tools)
- Qt is much more popular in HPC environments (e.g., Linux standard desktop GUI)
HPCT-Eclipse: Instrumentation

The instrumented binary is successfully generated!
The instrumented application can be found at
/home/hfweyh/hycon/mpi4_inst.
HPCST is Ongoing Research

- We welcome **scientific collaborators** and early adopters to help grow BDE DB and provide more feedback on real applications.

- We welcome **tools developers** to extend and integrate their capabilities via Modules.

- We also have **post-doc**, co-op and student internships available at the IBM Watson lab!
Thank you!