Performance Visualization of Hybrid Cell Applications

ScicomP 15, May 19th, Barcelona

holger.brunst@daniel.hackenberg@tu-dresden.de
Outline

Introduction
Software Tracing on Cell Systems
Implementation and Functionality
Examples and Overhead
Summary
Cell Broadband Engine

Element Interconnect Bus

SPE: Synergistic Processor Element
LS: Local Store

PowerPC Processor Element (PPE)

L2 → L1 → PowerPC Core

Memory Interface Controller (MIC)

Bus Interface Controller (BIC)

Dual XDR

FlexIO
Vast Resources

- SPEs: SIMD-Cores for fast calculations, 256 KB local store (LS, software controlled), dedicated DMA engine (MFC)
- PPE: very simple PowerPC Core for OS (Linux) and control tasks

Sophisticated Architecture

- Complex software development process
- Different compilers and programs for PPE and SPEs
- SPEs use DMA commands to access main memory or LS of other SPEs, asynchronous execution by MFC
- Mailbox communication between PPE and SPEs

Tool Support
Why do we still need to analyze?

- HPC: System complexity increases constantly
- Parallelism enters main stream market and not many people know how to deal with it

Approaches

- Profilers do not give detailed insight into timing behavior of an application
- Detailed online analysis pretty much impossible because of intrusion and data amount

Tracing

- Records application behavior step-wise
- Tracing is an option to capture the dynamic behavior of parallel applications
- Performance analysis done on a post-mortem basis
Background

What is Vampir?
- Performance monitoring and analysis tool
- Targets the visualization of dynamic processes on massively parallel (compute-) systems

History
- Development started more than 15 years ago at Research Centre Jülich, ZAM

Availability
- Unix, Windows, and Mac OS
- Visualization components (Vampir) are commercial
- Monitor components (VampirTrace) are Open Source
Components

- Application
- VampirTrace
- Trace Data (OTF)

1. Application
   - VampirTrace
   - OTF Trace Part 1

2. Application
   - VampirTrace
   - OTF Trace Part 2

3. Application
   - VampirTrace
   - OTF Trace Part 3

4. Application
   - VampirTrace
   - OTF Trace Part 4

... 

10,000 Application
   - VampirTrace
   - Trace Data Part m
Flavors

Vampir
- Sequential event analysis
- Rich set of graphical performance views
- For desktops and small parallel production environments
- Less scalable

VampirServer
- Distributed client/server approach
- Parallel analysis
- New features

Vampir for Windows
- Modern QT-based GUI
- Released at ISC 2009, Hamburg
- Currently: Beta-Release
Outline

Introduction
Software Tracing on Cell Systems
Implementation and Functionality
Examples and Overhead
Summary
**PPE**
- Conventional tools with PowerPC support run unmodified
- Modifications necessary to support SPE threads

**SPE**
- New concept needs to be designed, suitable for this architecture
- New monitor necessary to generate events
- Local store too small, only temporary storage of events
- Synchronization of PPE and SPE timers necessary
Trace Monitor Concept

SPE 0

SPU 0

Instrumented SPE program

Buf 1

Buf 2

Local Store

I/O System

trace file PPE

trace file SPE 0

trace file SPE n

* Buffers will switch each time the current trace buffer is full

PPE

processes SPE trace buffers (post mortem) and writes trace files to disk

Conventional monitoring tool with enhancements to cover e.g. mailbox communication with SPEs

PPE

DMA transfer of full trace buffer to main memory in background, SPE program keeps running

Buf 1/n

Buf 2/n

Buf 3/n

... Buf m/n

Main Memory

Buf 1/0

Buf 2/0

Buf 3/0

...

Buf m/0

Element Interconnect Bus

SPE n

SPU n

Instrumented SPE program

Buf 1

Buf 2

Local Store

Holger Brunst, Daniel Hackenberg
Illustration of parallel processes in a typical timeline display
Illustration of SPE threads as children of the PPE process
Illustration of mailbox messages

- Classic two-sided communication (send/receive)
- Illustrated by lines similar to MPI messages
Illustration of DMA transfers between SPEs and main memory

- PPE is not involved
- Main memory is represented as independent bar
- Allows graphical representation of memory states (read/write)
DMA transfers between SPEs
- Challenge: Communication is one-sided
- Peer-to-peer send/receive representation unsuitable

Distinction of active and passive partner?
- Additional lines
- Additional bullets (active partner)
- Even more bullets? (passive partner)
Trace Visualization for Cell (6)

- DMA wait operation creates two events (at $t_1$ and $t_2$)
- Allows illustration of DMA wait time
- Similar for mailbox messages

```c
    t_0 = get_timestamp();
    mfc_get();
    [...]
    t_1 = get_timestamp();
    wait_for_dma_tag();
    t_2 = get_timestamp();
```
Outline

Introduction
Software Tracing on Cell Systems
Implementation and Functionality
Examples and Overhead
Summary
Prototype implementation based on VampirTrace (VT)
- Open Source
- http://www.tu-dresden.de/zh/vampirtrace

Additional tool: CellTrace
- Header files for PPE and SPE programs: Instrumentation of inline functions provided by the Cell SDK
- Library for PPE programs + library for SPE programs
Visualization of a Cell trace using Vampir

- Simple demo program
- 4 SPEs only
Trace Visualization with Vampir (2)
Trace Visualization with Vampir (3)

Complex DMA transfers of SPE 3
Outline

Introduction
Software Tracing on Cell Systems
Implementation and Functionality
Examples and Overhead
Summary
Example Cell Applications: FFT (1)

FFT at a synchronization point
8 SPEs, 64 KByte page size, 11.9 GFLOPS
Example Cell Applications: FFT (2)

FFT at a synchronization point
8 SPEs, 16 MByte page size, 42.9 GFLOPS
Example Applications: Cholesky (1)

Cholesky transformation with 8 SPEs overview with DMA communication of SPE 3
Example Applications: Cholesky (2)

Cholesky transformation with 8 SPEs
enlargement with DMA communication of SPE 3
Example Cell Applications: RAXML (1)

RAXML (Randomized Accelerated Maximum Likelihood) with 8 SPEs, ramp-up phase
Example Cell Applications: RAXML (2)

RAxML with 8 SPEs, 4000 ns window enlargement of a small loop shifted start of loop, constant runtime
RAxML with 8 SPEs, 4000 ns window enlargement of a small loop (modified) synchronous start, memory contention
Example Cell Applications: RAXML (4)

RAXML with 16 SPEs, load imbalance
PBPI (Parallel Bayesian Phylogenetic Inference) on 3 QS21 blades (6 Cell processors)
Combined Charts in New GUI
Overhead sources

- Creating events
- Transferring trace data from the SPEs to main memory
- Trace buffer and trace library use space in local store (< 12 KByte)

Additional overhead

- Initialization and processing of SPE event data
- Outside of SPE runtime → Analysis unaffected

Experimental overhead measurements (QS21, 8 SPEs):

<table>
<thead>
<tr>
<th></th>
<th>Original (GFLOPS)</th>
<th>Tracing (GFLOPS)</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGEMM</td>
<td>203,25</td>
<td>200,73</td>
<td>1,3 %</td>
</tr>
<tr>
<td>FFT</td>
<td>11,93</td>
<td>11,85</td>
<td>0,7 %</td>
</tr>
<tr>
<td>Cholesky, SPOTRF</td>
<td>143,17</td>
<td>139,32</td>
<td>2,8 %</td>
</tr>
<tr>
<td>Cholesky, DGEMM</td>
<td>4,48</td>
<td>4,10</td>
<td>9,2 % (*)</td>
</tr>
<tr>
<td>Cholesky, STRSM</td>
<td>5,73</td>
<td>5,64</td>
<td>1,7 %</td>
</tr>
</tbody>
</table>

(*) Increased overhead due to intense usage of DMA lists.
Trace overhead without DMAs: 1,4 %
Summary & Future Work

Concept and Prototype for Performance Tracing on Cell
- CellTrace
- Typical overhead: less than 5 percent

Visualization of Traces with Vampir
- Creates valuable insight into the runtime behavior of Cell applications
- Intuitive performance visualization and verification
- Support for large, hybrid Cell/MPI applications

Future work
- Improved tracing, e.g. full integration in VampirTrace, providing additional analysis features such as alignment checks
- Improved visualization, e.g. by colorizing DMA messages (tag, size or bandwidth), displaying intensity of main memory accesses
<table>
<thead>
<tr>
<th></th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>BLAGOJEVIC et. al.: Scheduling Asymmetric Parallelism on a PlayStation3 Cluster. To appear in ISCCG 2008</td>
</tr>
<tr>
<td>5.</td>
<td>KURZAK, Jakub; BUTTARI, Alfredo; DONGARRA, Jack: Solving Systems of Linear Equations on the CELL Processor Using Cholesky Factorization. IEEE Transactions on Parallel and Distributed Systems 2007</td>
</tr>
<tr>
<td>7.</td>
<td>KNÜPFER et. al.: Introducing the Open Trace Format (OTF). ICCS 2006</td>
</tr>
<tr>
<td>8.</td>
<td>IBM, SCEI, Toshiba: Cell Broadband Engine Architecture. 2005</td>
</tr>
<tr>
<td>10.</td>
<td>EICHENBERGER et. al.: Using advanced compiler technology to exploit the performance of the Cell Broadband Engine architecture. IBM Systems Journal 45, 2006</td>
</tr>
</tbody>
</table>