Clock Synchronization in Cell BE Traces

☑ Marina Biberstein (biberstein@il.ibm.com)
☑ Yuval Harel
☑ Andre Heilper
Cell BE background

- Heterogeneous multicore architecture
- 1 PPU core (Power)
- 8 SPU cores
  - Novel ISA
  - No cache
  - 256 K local store
  - Explicit asynchronous load/store using DMA
  - Other means of communication: mailboxes, mutexes, ...

64-bit Power Architecture with VMX

http://fabiocpn.files.wordpress.com/2006/10/cell.jpg
Performance Debugging Tool (PDT)

- Trace collection for Cell
- Application-level events
- Requires rebuild but no code changes

Traced events:
- SPE thread creation and completion
- DMA transfer events
  - Requests (put, get)
  - Waiting for request completion (get_tag_status)
- Lock-based synchronization
- Mailboxes and other channel communication
- User-defined events

Part of Cell BE SDK

http://www.ibm.com/developerworks/power/cell/pkgdownloads.html
Trace Analyzer

- Read the PDT trace
- Fill in implicit data
  - Wall-clock time
  - Core ids
  - Context switches
- Show the trace
  - Graphical view
  - Text-based browser
  - Full record details on selection

Part of Visual Performance Analyzer
http://www.alphaworks.ibm.com/tech/vpa
Tracing on Cell

Problem: each log event comes with a *core-local* timestamp
Existing solutions: Lamport’s virtual time

Leslie Lamport, Time, clocks, and the ordering of events in a distributed system, Comm. ACM, 1978
Existing solutions: Lamport’s virtual time (cont.)

Leslie Lamport, Time, clocks, and the ordering of events in a distributed system, Comm. ACM, 1978
Existing solutions: Network Time Protocol

- **References:**

- **Problems with implementing NTP on our system:**
  - Precision/overhead tradeoff
  - Context switch on SPU resets clock relative offset
  - Use of scarce resources (communication channels, decrementer register)
Target solution

- Agrees with event order
- Agrees with durations within thread
Definitions

**Trace**
- $E$ set of events
- $\text{tid}: E \rightarrow N$ function that matches each event to its thread id
- $\leq$ happened-before partial order relation over $E$
- $\text{ttime}: E \rightarrow \mathbb{R}$ function that matches each event to its thread-local timestamp

**Global time**
- $\text{gtime}: E \rightarrow \mathbb{R}$
  - Agrees with thread-local time: if $e, e'$ belong to the same thread, then
    $\text{gtime}(e) - \text{gtime}(e') = \text{ttime}(e) - \text{ttime}(e')$
  - Agrees with event order: if $e \leq e'$, then
    $\text{gtime}(e) \leq \text{gtime}(e')$
Time computation

\[ T_1 \quad e_{11} \quad T_2 \quad e_{21} \quad T_3 \quad e_{31} \]

\[ 0 = g_1 \quad g_2 \quad g_3 \quad 8 \]

\[ 0 = g_1 \quad 10 = e_{12} \quad 16 = e_{13} \quad 10 = e_{22} \]

\[ 0 = e_{21} \quad 0 = e_{31} \]
Time computation

\begin{align*}
&\text{T1} & e_{11} & 0 & g_1 < g_2 \\
&\text{T2} & e_{21} & 0 & \\
&\text{T3} & e_{31} & 0 & \\
& & e_{12} & 10 & \\
& & e_{13} & 16 & \\
& & e_{22} & 10 & \\
& & e_{22} & 8 & \\

g_1 & & g_2 & & g_3
\end{align*}
Time computation

\[ g_1 - g_2 < 0 \]
Time computation

\begin{align*}
T1 & \quad e_{11} \\
T2 & \quad e_{21} \\
T3 & \quad e_{31}
\end{align*}

\begin{align*}
\text{if} & \quad g_1 - g_2 < 0 \\
& \quad g_1 - g_3 < 0
\end{align*}
Time computation

\[ g_1 - g_2 < 0 \quad g_1 - g_3 < 0 \]
Time computation

\[ g_1 - g_2 < -6 \]
\[ g_1 - g_3 < 0 \]

T1  \quad g_1 + 16 < g_2 + 10

T2  \quad e_{21}

T3  \quad e_{31}

\[ e_{11} \]
\[ e_{12} \]
\[ e_{13} \]
\[ e_{22} \]
\[ e_{31} \]
\[ e_{21} \]
\[ e_{22} \]
Time computation

\[
\begin{align*}
T_1 & \quad e_{11} & 0 \\
T_2 & \quad e_{21} & 0 \\
T_3 & \quad e_{31} & 0 \\
g_1 & \quad e_{13} & 10 \\
g_2 & \quad e_{23} & 10 \\
g_3 & \quad e_{33} & 8
\end{align*}
\]

\begin{align*}
g_1 - g_2 & < -6 & g_1 - g_3 & < -8 \\
g_2 - g_1 & < 10 & g_2 - g_3 & < 0 \\
g_3 - g_1 & < 10 & g_3 - g_2 & < 2
\end{align*}
Problem: complexity

- Relatively few threads (low tens)
- Many events (100 K +)
- Can’t really afford algorithm that is non-linear in number of events
- How to collect all the constraints?
Solution:

- Collect constraints from **consecutive** pairs of events
- Improve the constraints by computing transitive closures
Constraint propagation

- $G(V, E, w)$
- **Nodes:**
  - Threads
- **Edges:**
  - All (clique)
- **Weights:**
  - constraints collected from consecutive events

$w(g_i, g_j)$ is the upper bound on $g_i - g_j$

<table>
<thead>
<tr>
<th></th>
<th>$g_1 - g_2 &lt; 0$</th>
<th>$g_1 - g_3 &lt; -8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$g_2 - g_3 &lt; 0$</td>
<td>$g_3 - g_1 &lt; 10$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$g_3 - g_2 &lt; 2$</td>
</tr>
</tbody>
</table>
Constraint propagation

\[
\begin{align*}
g_1 &- g_2 < 0 & g_1 &- g_3 < -8 \\
g_2 &- g_3 < 0 & g_2 &- g_3 < 0 \\
g_3 &- g_1 < 10 & g_3 &- g_2 < 2 \\
\end{align*}
\]
Constraint propagation

- $G'(V, E, w')$
- **Nodes:**
  - Threads
- **Edges:**
  - All (clique)
- **Weights:**
  - improved constraint

$w'(g_i, g_j)$ is the weight of the shortest path in $G$ from $g_i$ to $g_j$

<table>
<thead>
<tr>
<th></th>
<th>$g_1 - g_2 &lt; -6$</th>
<th>$g_1 - g_3 &lt; -8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_2 - g_1 &lt; 10$</td>
<td></td>
<td>$g_2 - g_3 &lt; 0$</td>
</tr>
<tr>
<td>$g_3 - g_1 &lt; 10$</td>
<td>$g_3 - g_2 &lt; 2$</td>
<td></td>
</tr>
</tbody>
</table>
Constraint propagation (cont.)

- **Theorem**: constraint propagation process above generates the same constraints both...
  - when we start with constraints from all event pairs, and
  - when we start with constraints from consecutive events
Constraint propagation (cont.2)

- **Proof**: let \( e \leq e' \) be any two events.
  
  - Implied constraint
    
    \[ g_{\text{tid}(e)} + \text{ttime}(e) \leq g_{\text{tid}(e')} + \text{ttime}(e') \]
    
    \[ g_{\text{tid}(e)} - g_{\text{tid}(e')} \leq \text{ttime}(e') - \text{ttime}(e) \]
  
  - There exists sequence
    
    \( e = e_0 \leq e_1 \ldots \leq e_n = e' \)
    
    such that every \( e_i \) is the immediate predecessor of \( e_{i+1} \)
  
  - Then
    
    \[
    \text{ttime}(e') - \text{ttime}(e) = \\
    = \Sigma_{i=0..n-1} (\text{ttime}(e_{i+1}) - \text{ttime}(e_i)) \\
    \geq \Sigma_{i=0..n-1} w(g_i, g_{i+1}) \\
    \geq w'(g_{\text{tid}(e)}, g_{\text{tid}(e')})
    \]
  
  - The constraint implied by \( e, e' \) is not stronger than the corresponding \( w' \)
Building the solution

- **Given:** constraints on \((g_i - g_j)\)
- **Goal:** set of values for the \(g_i\)‘s satisfying the constraints
- **Iterative process**
  - \(g_0 = 0\)
  - Assume \(g_i\) is already defined for \(i = 0, \ldots, k\)
  - **For** \(i = k + 1\):  
    - Substitute \(g_0, \ldots, g_k\), into constraints involving them and \(g_{k+1}\):  
      \begin{align*}
      g_0 - g_{k+1} & \leq w'(0, k+1), \ldots, g_k - g_{k+1} \leq w'(k, k+1) \\
      g_{k+1} - g_0 & \leq w'(k+1, 0), \ldots, g_{k+1} - g_k \leq w'(k+1, k)
      \end{align*}
    - Obtain a range of possible values for \(g_{k+1}\):  
      \[
      m_{k+1} \leq g_{k+1} \leq M_{k+1}
      \]
    - Choose a value for \(g_{k+1}\) from that range

- **Theorem:** this process generates all the possible solutions
Building the solution: example

\[ g_1 = 0 \]
\[ g_2 = 8 \]
\[ g_3 = 9 \]

\begin{align*}
g_1 - g_2 & < -6 \\
g_1 - g_3 & < -8 \\
g_2 - g_1 & < 10 \\
g_2 - g_3 & < 0 \\
g_3 - g_1 & < 10 \\
g_3 - g_2 & < 2 \end{align*}
Algorithm summary

- **Collect constraints from consecutive events**
  - Complexity depends on $\leq$
  - For linear order: $O(|E|)$

- **Compute transitive closure of the constraints**
  - Complexity $O(|\text{tid}(E)|^3)$ using Floyd-Warshall

- **Build a global time function $g_{time}$**
  - Showed construction taking $O(|\text{tid}(E)|^2)$
  - Alternatively, there is an $O(|\text{tid}(E)|)$ construction that doesn’t cover solution space

- **Complexity: $O(\text{NEvents}) + (\text{NThreads}^3)$**
Precision estimate

- Estimate on solution’s precision
  \[-w'(g_t, g_t') \leq g_t - g_t' \leq w'(g_t, g_t')\]

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>SPU</td>
</tr>
<tr>
<td>BlackScholes</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>FFT16M</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>JuliaSet</td>
<td>34</td>
<td>19</td>
</tr>
<tr>
<td>Matrix_Mul</td>
<td>35</td>
<td>11</td>
</tr>
</tbody>
</table>

Errors shown in ticks, 10 ticks = 0.7 µs
Coping with clock imprecision

\[ g_1 - g_2 < 0 \]
\[ g_2 - g_1 < -2 \]
Coping with clock imprecision (cont.)

\[
\begin{align*}
T1 & : e_{11} & 0 \pm \delta \\
T2 & : e_{21} & 0 \pm \delta \\
& : e_{12} & 8 \pm \delta \\
& : e_{22} & 10 \pm \delta
\end{align*}
\]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1 - g_2 &lt; 2\delta$</td>
<td></td>
</tr>
<tr>
<td>$g_2 - g_1 &lt; -2 + 2\delta$</td>
<td></td>
</tr>
</tbody>
</table>

- Negative loop disappears at $\delta = \frac{1}{2}$
Coping with clock imprecision: summary

1. Collect constraints from consecutive events
2. Compute transitive closure of the constraints
3. Success?
   - Y: Generate a solution
   - N: Find simple loop with the smallest weight $\delta$
4. Bump all constraints by $\delta/2$
Conclusions and future work

- **Offline algorithm for computing global time using thread-local time and event order data**
- **Linear in # events, cubical in # threads**
- **Features:**
  - Doesn’t contradict trace timing/order data
    - Unless trace errors are detected
  - Provides estimate for its own precision
  - Can recover from minor errors in trace collection
    - And provide estimate on error magnitude
- **Future work**
  - Balancing precision vs. tracing overhead (order data)
  - Handling other common tracing errors
    - Single event with a bad timestamp