TimeDoctor
Use the Strength of Eclipse to Visualize (Multi)Processor Execution Behavior

Ruud Derwig (NXP Semiconductors)
Stuart Fullmer & Klaas van Gend (Monta Vista)
TimeDoctor – display execution traces

Debugger, target connection

Read trace data from target (jtag, memory buffer, SW agent)

Linux PC, (multi-processor) chip, or virtual prototype

Instrumented software to capture execution events

TimeDoctor trace file
Contents

- What is it: demo
- Why
  - history
  - goals, requirements
- How – part 1
  - TimeDoctor Eclipse based architecture
  - TimeDoctor input format
  - Instrumentation & logging
- How – part 2
  - TimeDoctor & Linux
  - LTTng: getting & converting trace data
- Conclusions
  - status
  - plans
Why TimeDoctor?

- A pictures says more than a thousand words
- TimeDoctor started as a side-result of a student assignment in 1994
  - visualize schedules and additional information
    - release times & deadlines
    - wake-up calls, send/receive actions

Tools for real-time constraints

an off-line scheduler and a schedule viewer

By:
R.J.C. Derwig

CONCLUSIONS - Time Doctor (1)

Example of test-run data
## History of TimeDoctor

<table>
<thead>
<tr>
<th>Generation &amp; date</th>
<th>host OS &amp; toolkit</th>
<th>language</th>
<th>kLoCs</th>
<th>new features</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1994</td>
<td>Unix, Motif</td>
<td>C++</td>
<td>2.5</td>
<td>schedules, events</td>
</tr>
<tr>
<td>1, 1999</td>
<td>Windows, GDI</td>
<td>Visual Basic C++</td>
<td>0.75 (VB) 3 (C++)</td>
<td>queues, semaphores, counters</td>
</tr>
<tr>
<td>2, 2003</td>
<td>Windows, .NET</td>
<td>C#</td>
<td>10</td>
<td>colors, statistics, agents</td>
</tr>
<tr>
<td>3, 2007</td>
<td>Linux + Windows, Eclipse SWT</td>
<td>Java</td>
<td>10</td>
<td>Eclipse integration</td>
</tr>
</tbody>
</table>
TimeDoctor – Eclipse plug-in objective

- **Objective**: make TimeDoctor available under Linux and Windows
  - Existing version is Windows only (written in C#)

- **Solution**: port the existing TimeDoctor version to Eclipse/Java
  - Eclipse supports both Linux and Windows
  - Align with NXP Eclipse policy

- **Objective**: attract a community of TimeDoctor users and developers
  - Customers can add specific features
  - NXP internal users can easily contribute

- **Solution**: donate TimeDoctor to open source
  - First cleanup the code to allow further extensions
  - Actively contribute with new features
TimeDoctor – related work

- Trace generation implemented in Linux
- Binary file format with extension options
- Visualization tool: LTTV
  → TimeDoctor may be used to visualize LTT trace files

  - Eclipse plug-in from [Montavista]

- Based on LTTng
  - Implemented as Eclipse plug-in
Main TimeDoctor requirements & features

- Visualization
  - Display the usage of CPU (or CPU’s) in a graph over time: threads, ISRs, agents
  - Show events and OS objects that influence scheduling: semaphores, queues, etc.
    - tailored for ‘streaming’ applications like e.g. media processing: queue sizes
  - Support performance analysis: counters for e.g. cache misses, bus cycles, etc.
  - Support multi-core systems: indicate core for tasks and ISRs
  - Visualize user defined events/notes and additional descriptions/information
  - Provide statistical information: nr. of context switches, CPU load, etc.
  - Show sample properties and additional information when mouse hovers over

- Other
  - Linux and Windows development host environments
  - Stand-alone and Eclipse IDE plug-in
  - Good performance: quick layout and response time to the user
  - Efficient memory usage: deal with trace files of 10’s of MBytes
  - Support measuring time intervals by setting baseline and distance to current mouse
  - Zooming and easy navigation through the trace set
Contents

- What is it: demo
- Why
  - history
  - goals, requirements
- How – part 1
  - TimeDoctor Eclipse based architecture
  - TimeDoctor input format
  - Instrumentation & logging
- How – part 2
  - TimeDoctor & Linux
  - LTTng: getting & converting trace data
- Conclusions
  - status
  - plans
Eclipse development environment
Eclipse architecture
TimeDoctor Eclipse plug-in

TimeDoctor menu and toolbar

TimeDoctor help, preference pages, update site

Trace editor

TimeDoctor perspective

Visible trace lines outline view

Statistics view

Property view: values of the selected sample
Stand-alone application & Eclipse plug-in

Eclipse plug-in

Stand-alone (RCP) application
TimeDoctor architecture – plug-ins & features
TimeDoctor trace file format

- Setup
  - CPU <id> <name>
  - SPEED <CPU clocks per sec>
  - MEMSPEED <memory clocks per sec>
  - TIME <time ticks per sec>

- Creation/delete
  - CRE <type> <id> <time> [<…>]
  - DEL <type> <id> <time>

- Events
  - STA <type> <id> <time> [<size>]
  - STO <type> <id> <time> [<size>]
  - OCC <type> <id> <time>
  - TIM <time>
  - VAL <type> <id> <value>

- Descriptions
  - DSC <type> <id> <value>
  - DNM <type> <id> <name>
  - NAM <type> <id> <name>

- Event types
  - Task
  - Interrupt service routine
  - Agent
  - Semaphore
  - Queue
  - Message event
  - Value count
  - Cycle count
  - Memory cycle count
  - Notes

- Description types
  - String
  - Number
  - Time
  - Color
### TDI example – 1/2

| STA 0 788 9493 | start task 788 at 9403 |
| STA 0 788 9676 | stop task 788 at 9676 |
| STA 0 788 10493 | etc. |
| STA 1 1 10530 | start ISR 1 at 10530 |
| NAM 1 | name of ISR 1 is “testing” |
| STO 1 1 10640 | etc. |
| STO 0 788 10676 |
| STA 0 788 11493 |

| NAM 3 1 Queue_ABC | name of queue 1 is “Queue_ABC” |
| STA 3 1 11000 | send a message to queue 1 at 11100 |
| STA 3 1 11100 |
| STA 3 1 11200 |
| STA 3 1 11300 | etc. |
| STO 3 1 11400 | receive from queue 1 at 11400 |
| STA 3 1 11500 |
| STA 3 1 11600 |

![Diagram showing TDI example](image-url)
### TDI example – 2/2

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAM 2 1</td>
<td>Sema_ABC</td>
<td>name of semaphore 1 is “Sema_ABC”</td>
</tr>
<tr>
<td>STA 2 1</td>
<td>10000</td>
<td>release sem. 1 at 10100</td>
</tr>
<tr>
<td>STO 2 1</td>
<td>10100</td>
<td>acquire sem. 1 at 10300</td>
</tr>
<tr>
<td>STA 2 1</td>
<td>10200</td>
<td>...</td>
</tr>
<tr>
<td>STA 2 1</td>
<td>10300</td>
<td>...</td>
</tr>
<tr>
<td>STO 2 1</td>
<td>10400</td>
<td>...</td>
</tr>
<tr>
<td>STA 2 1</td>
<td>10500</td>
<td>...</td>
</tr>
<tr>
<td>STA 2 1</td>
<td>10600</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAM 6 234</td>
<td>MemCycleValue</td>
<td>define name</td>
</tr>
<tr>
<td>TIM 11200</td>
<td>set time for next sample</td>
<td>next value, delta is shown</td>
</tr>
<tr>
<td>VAL 6 234</td>
<td>100</td>
<td>set time for counter 234</td>
</tr>
<tr>
<td>STA 3 1</td>
<td>11300</td>
<td>counter value uses time from previous event</td>
</tr>
<tr>
<td>VAL 6 234</td>
<td>200</td>
<td>...</td>
</tr>
<tr>
<td>TIM 11400</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>VAL 6 234</td>
<td>400</td>
<td>...</td>
</tr>
<tr>
<td>STO 3 1</td>
<td>11500</td>
<td>...</td>
</tr>
</tbody>
</table>

![Diagram](image-url)
Instrumentation & logging

Debugger, target connection

Read trace data from target (JTAG, memory buffer, SW agent)

TimeDoctor trace file

Linux PC, (multi-processor) chip, or virtual prototype

Instrumented software to capture execution events
Instrumentation & logging

- For NXP’s TriMedia processor family (not running Linux) the infrastructure software has been fully instrumented
  - context switches
  - interrupts
  - performance counters
  - queues
  - etc.

- For MIPS & ARM running Linux, a simple logging module was written, and the kernel was instrumented
  - context switches & interrupts only
  - not suited for community, just a quick hack

- Logging module
  - Sysfs interface
    - start/stop/reset
    - read sample data
  - kernel logging functions
    - phStbLog_task_create()
    - phStbLog_task_switch()
    - phStbLog_interrupt_enter/exit()
    - phStbLog_Info()

- Instrumented Linux files
  - fork.c
    - create task; CRE
  - sched.c
    - context switch; STA/STO
  - irq.c
    - enter/exit: STA/STO

- Perl script for conversion to TDI format
  - get process names etc.
Switching speakers
Klaas van Gend - MontaVista
Contents

- What is it: demo
- Why
  - history
  - goals, requirements
- How – part 1
  - TimeDoctor Eclipse based architecture
  - TimeDoctor input format
  - Instrumentation & logging
- How – part 2
  - TimeDoctor & Linux
  - LTTng: getting & converting trace data
- Conclusions
  - status
  - plans
A single Eclipse-based IDE for development, debugging and system profiling
Using TimeDoctor as LTTng viewer

- 4.0 Editions: MontaVista authored proprietary Eclipse plug-in to view LTT tracefiles.

- 5.0 Editions no longer use LTT but switch to LTTng
  - Yet, LTTng visualiser doesn’t run on Windows nor Solaris
  - Improving the existing plugin would be a lot of work
  - Adding another backend to TimeDoctor appeared to be a good alternative

- LTTng usage doesn’t change
- Just a different viewer
Parsing LTTng files

- Problem: LTTng is a binary format that is very flexible and very complicated.

- First attempt: a script to convert LTTng to TimeDoctor
- Second attempt: a parser – required new hooks in TimeDoctor
Issues with the current TimeDoctor (and wish list)

- No raw event view
- TimeDoctor wants all trace data in memory
  - Imposes size limit on tracefiles
  - Streaming not possible
- Events, IRQs and syscalls are in separate items – contrary to the LTT viewer.
  - Determining what causes a context switch is more difficult
- Synchronizing various data streams
  - Add data from syslogd
Switching speakers
Ruud Derwig – NXP Semiconductors
Contents

- What is it: demo
- Why
  - history
  - goals, requirements
- How – part 1
  - TimeDoctor Eclipse based architecture
  - TimeDoctor input format
  - Instrumentation & logging
- How – part 2
  - TimeDoctor & Linux
  - LTTng: getting & converting trace data
- Conclusions
  - status
  - plans
Status – TimeDoctor

- Current version 1.3.0-Alpha
  - Basic visualization of trace data
  - Both IDE and RCP support
  - Acceptable performance for large trace files
  - Help documentation

- Next release: 1.3.0
  - April/May
  - Outline view to select trace lines
  - Preference pages

- First fully functional version planned for June 2007
  - at least same feature level as TimeDoctorII

- To do list
  - Task/ISR/Queue statistics export to CVS, Excel
  - Graphical display of statistics
  - Property view
  - Many small UI features
Status – sourceforge

- NXP donated TimeDoctor to open source
  - Released under Eclipse public license (EPL)
  - Active contribution and project management by NXP
  - Project “TimeDoctor” hosted on sourceforge

http://sourceforge.net/projects/timedoctor

- MontaVista plans to release LTTng parser as open source
  - first version shipped to customers proprietary

- To do list
  - Advertise on Eclipse plugin central and others
Plans

- Complete TimeDoctorII features

- Many new ideas
  - real-time sample data feed
  - statistics graphs instead of text
  - new representations
    - events in same traceline as task instead of own traceline
      - same representation as LTT
      - configurable
    - highlight samples above/below a certain watermark
  - support for very large tracesets (GBytes)
  - synchronized scrolling of multiple trace windows
    - and other support for comparing traces
  - user annotations
  - saving/restoring configuration data (freeze view)
    - for sharing/analysing with others
  - specific Linux extensions?
Conclusions

- A picture says more than a thousand words!
- Eclipse is powerful, but has a steep learning curve
  - lot’s of functionality for free
  - but many standard widgets not adequate for specific TimeDoctor purposes
- Changing from a proprietary model to Open Source works
  - what if you could: Linux was started by a student, and v1.0 was also released in 1994 …

Help us make TimeDoctor in Eclipse a success
⇒ Use & Contribute to TimeDoctor ⇐