Advancing Multicore Debug & Monitoring Technologies

DOMINIQUE TOUPIN, ERICSSON
ABOUT ME

› Tool Manager at Ericsson, helping Ericsson sites to develop better software efficiently

› Telecommunication systems
  – Open, standards-based common platform
  – High availability, 99.999 %
  – Broad range of support for both infrastructure and value-added applications
  – Multimedia, network and application processing capabilities

I do not sell processors or tools 😊
MULTICORE SYSTEMS

Adapteva 64-core Epiphany

Kalray 256-core in 16 clusters

Intel’s Xeon Phi 61-core

Tilera 100-core Tile-GX

Coherent Logix’ 100-core HyperX
Distinct pipeline stages or modes can run on different cores, can also re-affinitize dynamically to help load-balancing.
Complex interactions,
Many debug entities:
- 100s of cores
- 100s of processes
- 1000s of threads

Too much info for mere mortals!
How to view/control on that scale?
**Multicore Tool Scalability**

**Cmd line** to connect and get events from multiple processes.

**GUI tree** gives a flat list for big apps, too much detail, can’t easily see overall layout/behavior, difficult to find and interact with “important” processes/threads...

Like this one
EXAMPLE: EPIPHANY 64-CORE

- Hardware layout
- Network layout
- Core Load
- Network link load
EXAMPLE:
EPHYPHANY'S VISUALIZER

Visualizer is interactive, can select & interact with cores, processes, threads. Doesn't replace the traditional debug view, the many-core visualizer augments it.
EXAMPLE: TILERA’S GRID

- Dots = Processes
- Red = Crash
- Yellow = Stop on bkp
- Drag Selection
- Resume, Suspend, Step, etc. Selection
EXAMPLE COHERENT LOGIX’ GRID

Colored Tasks

Paths between tasks

In-view modifications
EXAMPLE:
KALRAY’S MPPA VISUALIZER
If some cores have a high load while others have low load, you should consider re-alloca
ting your cores to better address the needs of your application.
WORKGROUP FILTERING

Focus on interesting parts

Allows to display more information

Only shows a particular workgroup
How in the ^**&#@ are we doing to program AND DEBUG this??
DEBUGGING THOUSANDS OF CORES
POSSIBLE SOLUTIONS

- Zooming
- Use of scrollbars, mouse drag
- Use of colors to pinpoint issues
- Automatic issue detection

More study needed, more collaboration
Zoom directly
Pin & clone for multiple visualizer
One high-level view with one or more zoomed views
PIN & CLONE TO COMPARE DATA

Compare data
Process Thread Core (PTC) set control groups of debug elements:

```
(gdb) step .34-59
```
Step threads numbered between 34 and 59

```
(gdb) step @2
```
Step all threads running on core 2

```
(gdb) interrupt *,future@5-7
```
Stop everything running on cores 5 to 7, preventing new threads from being started
Create a named set containing cores 3 to 6 and core 8

Print value of global variable ‘globvar’ for all processes running on cores that are part of the ‘crunchers’ group

Eclipse CDT to provide user-friendly grouping for data abstraction and control
GLOBAL BREAKPOINTS

Applies to every process

Auto attach when hit

Un-started or short lived process

 Kernél Modification Required / UProbe

(gdb) gbreak return.cc:14
Global Breakpoint 6 at 0x80484aa: file return.cc, line 14.
BREAKPOINT SCALABILITY

Too slow if host debugger evaluates breakpoint conditions

Thousands of debug elements on-target

Remote Debug Protocol

Conditional breakpoint evaluated on-target / in-process
**Non-Stop Debugging**

Stopping all threads
- changes program behavior drastically
- program can fail even when the code is correct
- heartbeat, watchdog, chasing race condition

Non-stop allows to break in one or more threads while letting the others continue

```
(gdb) set target-async on
(gdb) set pagination off
(gdb) set non-stop on
(gdb) run&
(gdb) interrupt
(gdb) step&
(gdb) continue&
(gdb) info threads
  Id   Target Id       Frame
  5   Thread 0xb67e2b70 (LWP 6951) (running)
  ✯  3   Thread 0xb77e4b70 (LWP 6949) 0x0012d422 in __kernel_vsyscall ()
  2   Thread 0xb7fe5b70 (LWP 6948) (running)
  1   Thread 0xb7fe7b30 (LWP 6944) 0x0012d422 in __kernel_vsyscall ()
```
Open source MI protocol applicable to all target OS

GDB Remote Protocol

Bare Metal / JTAG

Proprietary code

Proprietary targets

Proprietary simulators

Linux GDB Stub

QEMU

Open source
Next Teleconference

Every second Tuesday of the month at 11 a.m. Ottawa time
The next one will be on Tuesday May 13th, 2014
The call info is the same as for the CDT monthly calls:
  - North America 1-866-569-4992
  - Germany 49-692-2224-6059
  - France 33-(0)1-70-07-8535
  - UK 0800-033-7806
  - Switzerland 41-44-580-2115
  - Sweden 46-85-063-8386
  - Italy 003-902-3604-8268
Attendees use this: Extension: 700 Passcode: 19455

Agenda

- Luna release after feature freeze
  - Testing
  - Any issues?

Previous minutes of meetings

Previous minutes of meetings

Features of current interest

The following is a list of features have had some effort invested in them but are not

1. Multi Level Hierarchy in the Debug View
2. Grouping of debug view elements
3. Hiding of debug view elements
4. User-selectable Debug View layouts
5. Visualizer View
   - A first version of the Multicore Visualizer as well as the Visualizer Framework are part of th
   - Many enhancements are still of interest.
6. Synchronized/Grouped Run Control Operations
   - User defines a group (cores or processes or threads) which makes up the application, and
7. Global breakpoints
8. Enhanced Breakpoint Support

Features already completed

List of features that have already been implemented.

Potential future features
Debuggers are invaluable for algorithm issues but they fall short for the above category of problems, not to mention that some problems are not reproducible in the developer’s environment!

Problems can be hard/impossible to reproduce in the lab

Single core to multicore makes it harder:
  Race conditions, Deadlocks, Non-deterministic behavior

Many layers makes it worst:
  Middleware, VM, OS, hypervisor, sometimes across nodes or clusters
TOOLS INFRASTRUCTURE WORKING GROUP (TIWG™)

Objective

Promote interaction of existing and new tools to support migration of legacy sequential applications to multicore platforms and development of new parallel software.

Overview

When migrating existing sequential software to multicore platforms or developing new parallel software, programmers must use a range of tools that include tracers, profilers, and analysis tools. Although mechanisms exist to support tool interoperability for embedded applications that utilize a single, integrated development environment (IDE), today no standard exists to support the embedded multicore environment. To mitigate the issue of interoperability, the TIWG’s first year goal is to establish a trace data format standard. A subsequent goal will be to define interfaces between profilers and analysis/visualization tools. Meeting these goals will aid programmers searching for performance bottlenecks that limit the speed and capabilities of their multicore platforms.

The TIWG is also collaborating with the CE Linux Forum with Ericsson and the Linux Foundation CE Workgroup on an implementation for a de-facto trace data format standard that TIWG will define. The Linux Trace Toolkit Next Generation (LTTng) embeds a reference implementation of this format. It will subsequently be used to develop the Multicore Association’s industry-standard specification.

Status

The TIWG holds regularly scheduled meetings, typically, on a bi-weekly basis. Although the end product of this working group will be publicly available, we encourage you to join the Multicore Association to participate in the discussion and development phases of this project.

The specification for the initial common trace format (CTF) is available now at http://www.efficios.com/ctf. Even as a non-member, we’d like you to review this and provide feedback.
COMMON TRACE FORMAT

› Linux Tracing Toolkit (LTTng)
› GDB Tracepoint
› Linux Distros
› Babeltrace: convert to common trace format
› Eclipse Tracing Monitoring Framework
› Application, Kernel, HW, bare metal SW, etc.
FUTURE WORK

› Tracing / Monitoring info into the many core visualizer

› Analysis: power consumption, memory, etc.

› cmd line trace operations, similar to GDB

› Correlate with system end results

› Show multi-core programing model concepts e.g. OpenMP, OpenCL, etc. in the CDT

› Etc.
OPEN INNOVATION
Open Source & Industry

Industrial User Driven
Feature Completeness, Speed
Long Term Availability, Quality (now)

Tool Vendor Driven
Framework (2000)

Individual Driven
Hackers (1980)

Open Source
PolarSys: a user driven ecosystem

- Open Innovation and advanced features
- No Lock-in: you or third parties can add features
- Open source with commercial support
- No license fees
- Industrial user community driven
- World class intellectual property management for open source

- Very long term support
- Systematic maturity assessment
- Interoperability
- Accelerates product development
- Technology platform
- Designed for extensibility and adaptation to your context
FULL SPEED INNOVATION

- Learning curve, accessibility
- Innovation, advanced features
- More features, empowerment
- Re-use, faster development
- Cost ↓ Improvement budget
- Controlling own destiny