SEEING IS UNDERSTANDING: DEBUGGING WITH THE MULTICORE VISUALIZER

EclipseCon 2012

William R. Swanson

Marc Khouzam

ABOUT US

- William R. Swanson
  - Lead IDE Developer, Tilera Corporation
  - CDT Committer, developed Visualizer framework and UI
  - 22 years in parallel hardware, software & developer UI tools

- Marc Khouzam (@marckhouzam)
  - Lead CDT developer, Ericsson
  - CDT Committer, lead of Debug component (DSF-GDB)
  - 15 years in Telecom:
    - Multicore chips
    - Multi CPU boards
    - Multi board nodes
    - Need better tools!

TILERA

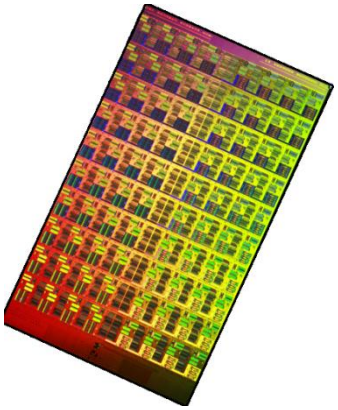
ERICSSON

AGENDA

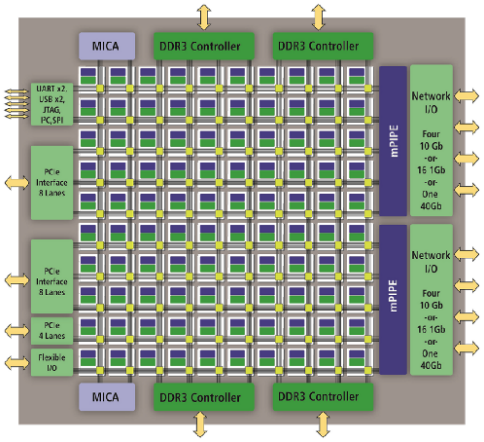
- The Challenge of Multicore
- An Answer: Visualization
- The Multicore Visualizer & Framework (demo)
- Extending the Visualizer (demo)
- The CDT Multicore Debug Workgroup
- Current Projects, Future Plans

- After the talk: demo on 24-core machine.

MULTICORE SYSTEMS



Intel's **80-core** Teraflop



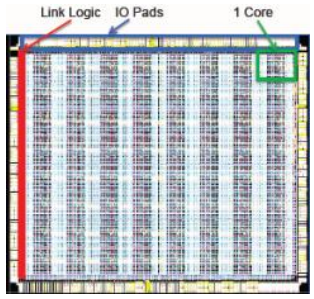
Tilera **100-core** Tile-GX

Ambric's **336-core** Am2045

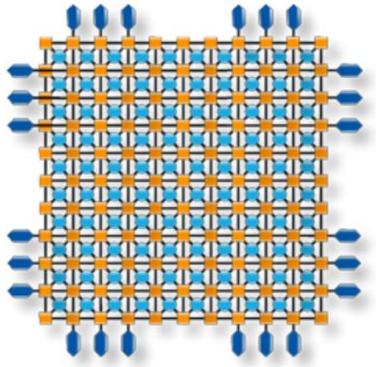
Plurality's **256-core** Hypercore

ClearSpeed **192-core** CSX700

AOCS **128-core** ModemX



Adapteva **64-core** Epiphany



Coherent Logix' **100-core** HyperX

TILERA

ERICSSON

MULTICORE SCALABILITY

Debug elements exploding
- 100s of cores
- 100s of processes
- 1000s of threads

Can be a literal embarrassment of riches...

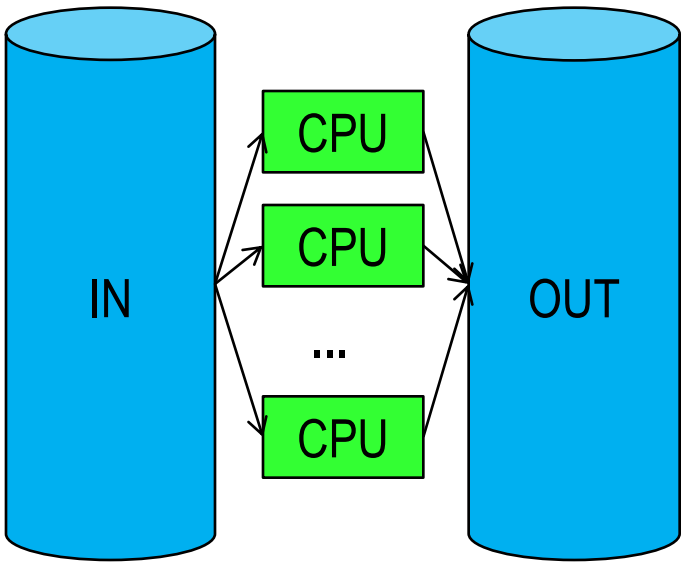
How to view and control on that scale?

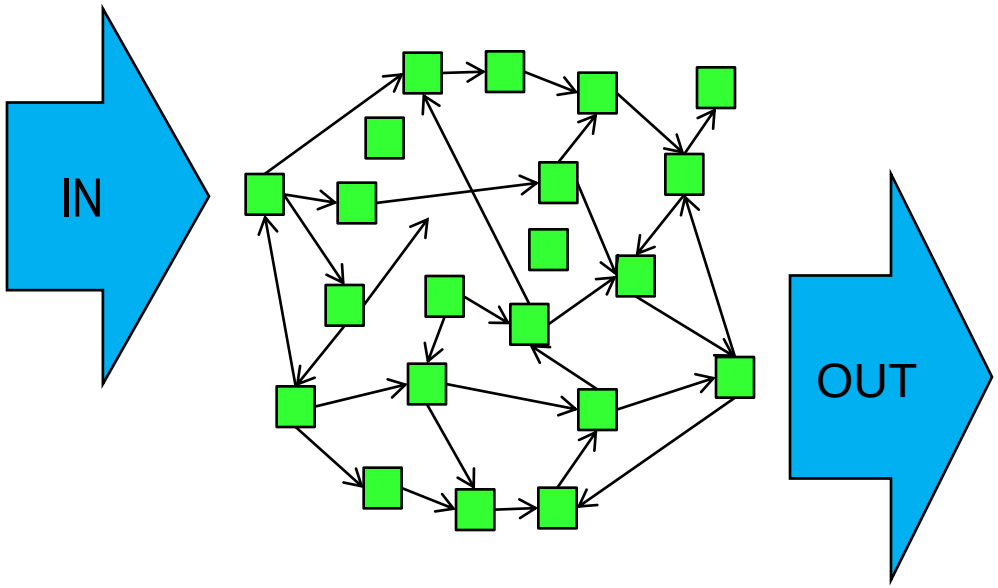
TILERA

ERICSSON

THE MULTICORE CHALLENGE

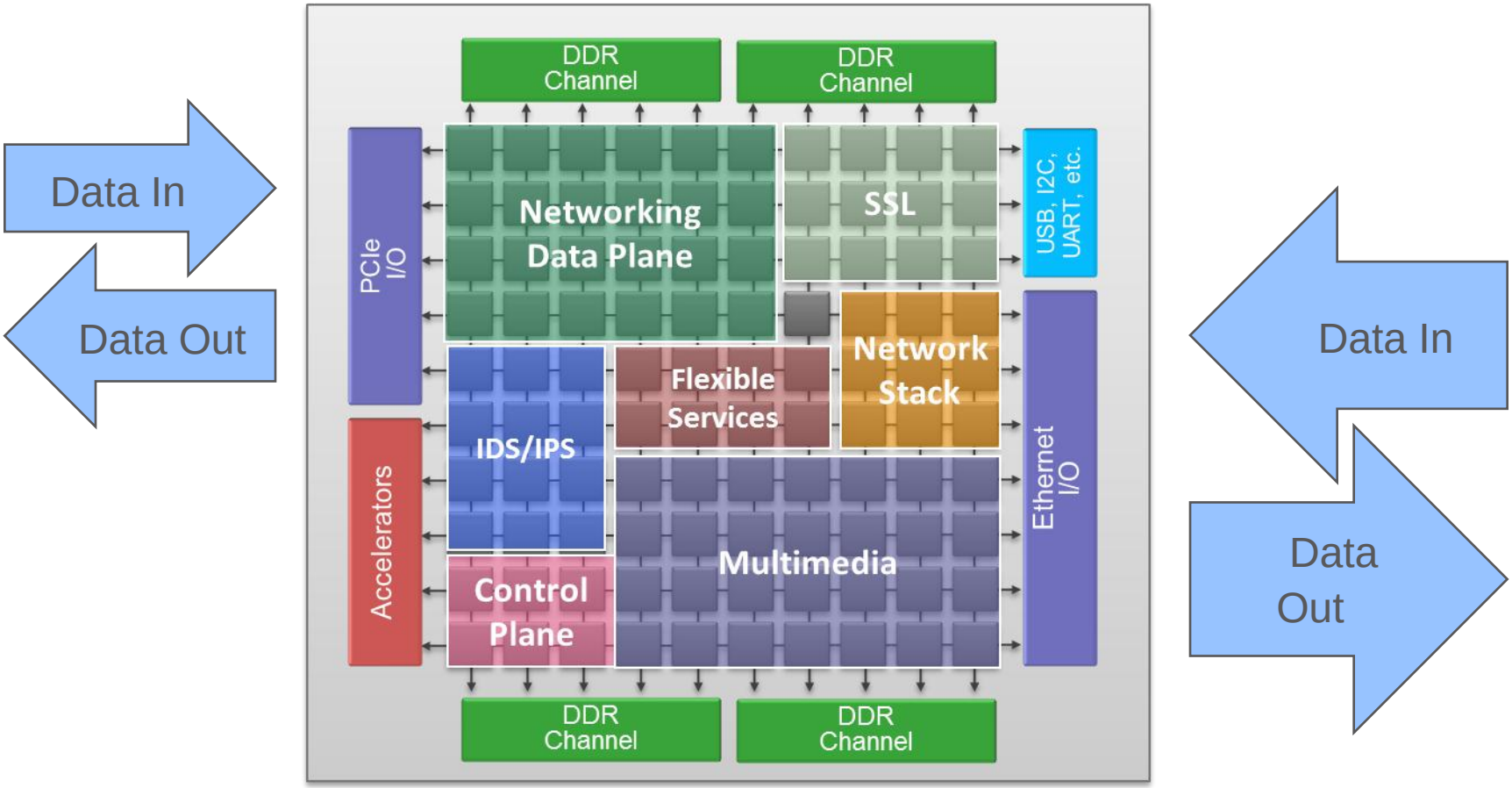
- We're in "big data" era, moving to "big algorithm" era
  - lots of processes/threads, complex interactions
  - ever-increasing need for a "big picture" overview





EXAMPLE: TILERA PROCESSOR

Distinct pipeline stages or modes can run on different tiles, can also re-affinitize dynamically to help load-balancing.



TILERA

ERICSSON

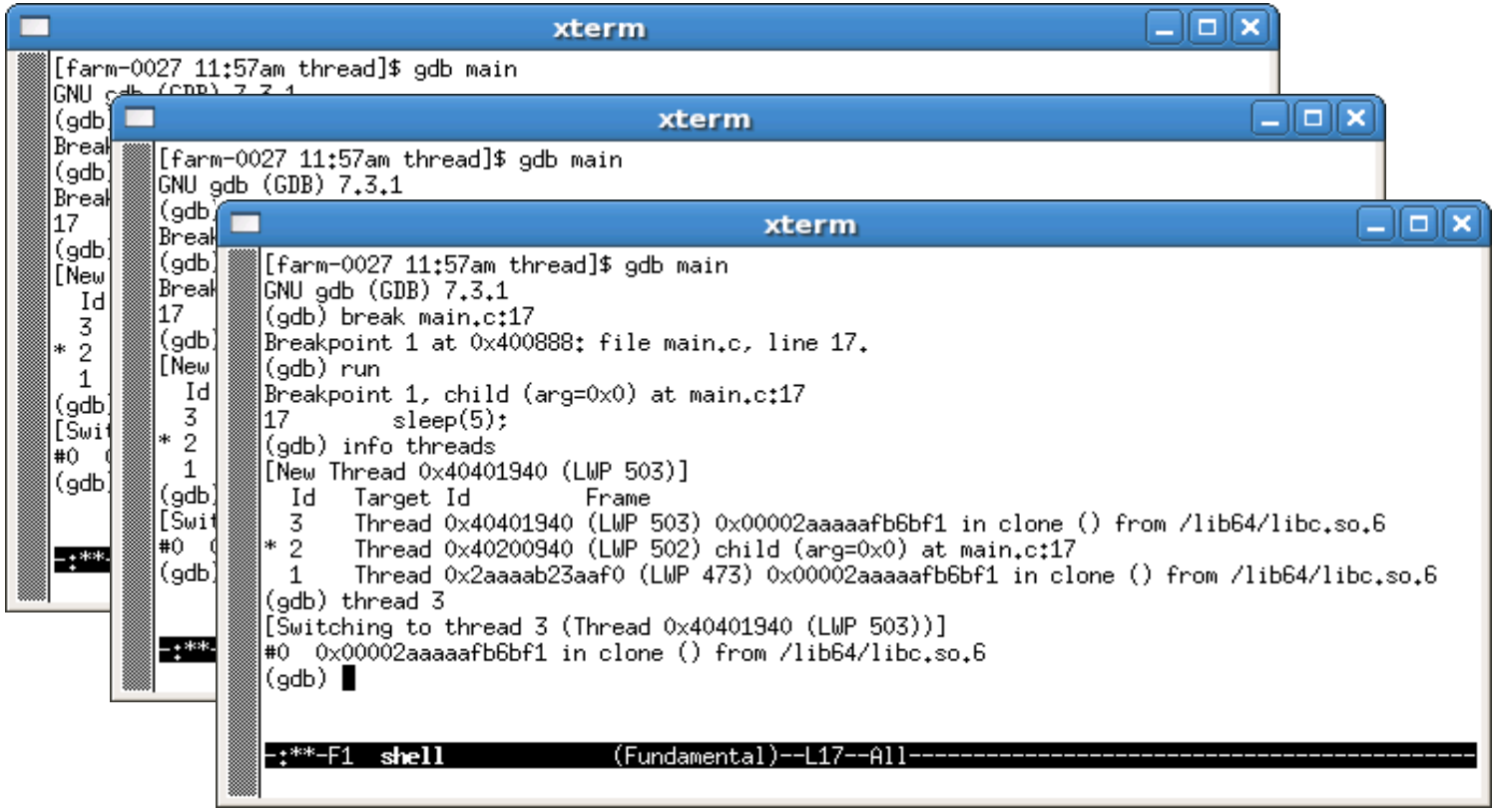
EXAMPLE: TILERA PROCESSOR

- And this is ~100 cores!
- Imagine debugging CPUs / GPUs with 100's or 1000's of cores...
- where the app is NOT data-parallel or lock-step SIMD!

- We're going to need better tools to handle this!

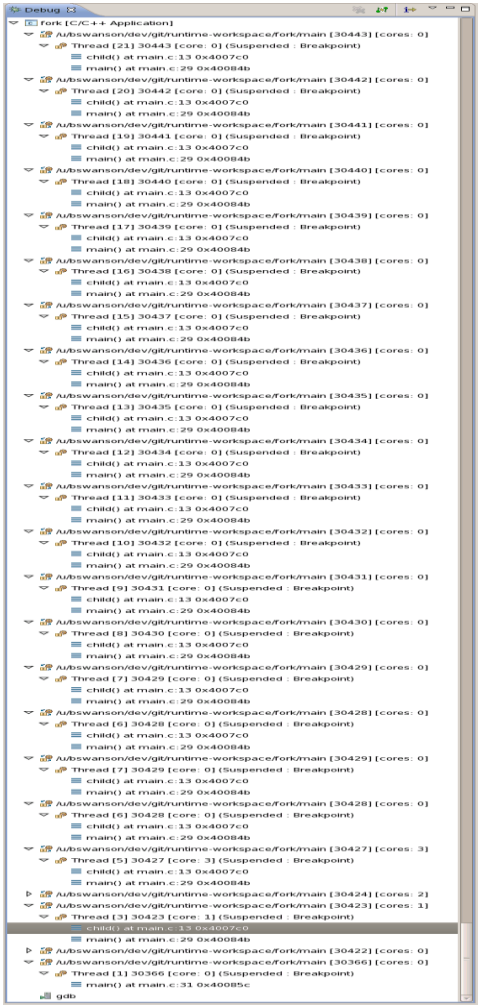
CURRENT TOOLS DON'T SCALE

- Command-line GDB
  - multiple shells, even for only a few processes, or
  - for multithreaded GDB, having to remember which thread you're on



CURRENT TOOLS DON'T SCALE

- Eclipse's Debug View
  - *way* better for managing large numbers of processes/threads
  - but...
  - for big apps, essentially a flat list
  - too much repetitive detail
  - can't easily see overall layout/behavior
  - difficult to find and interact with "important" processes/threads...
  - (like this one) ---------->



THE BIG PICTURE: VISUALIZATION

- Need for new ways of looking at applications...
- One answer: visualization tools
- Visualization is the “big picture”: the important stuff made easily visible.

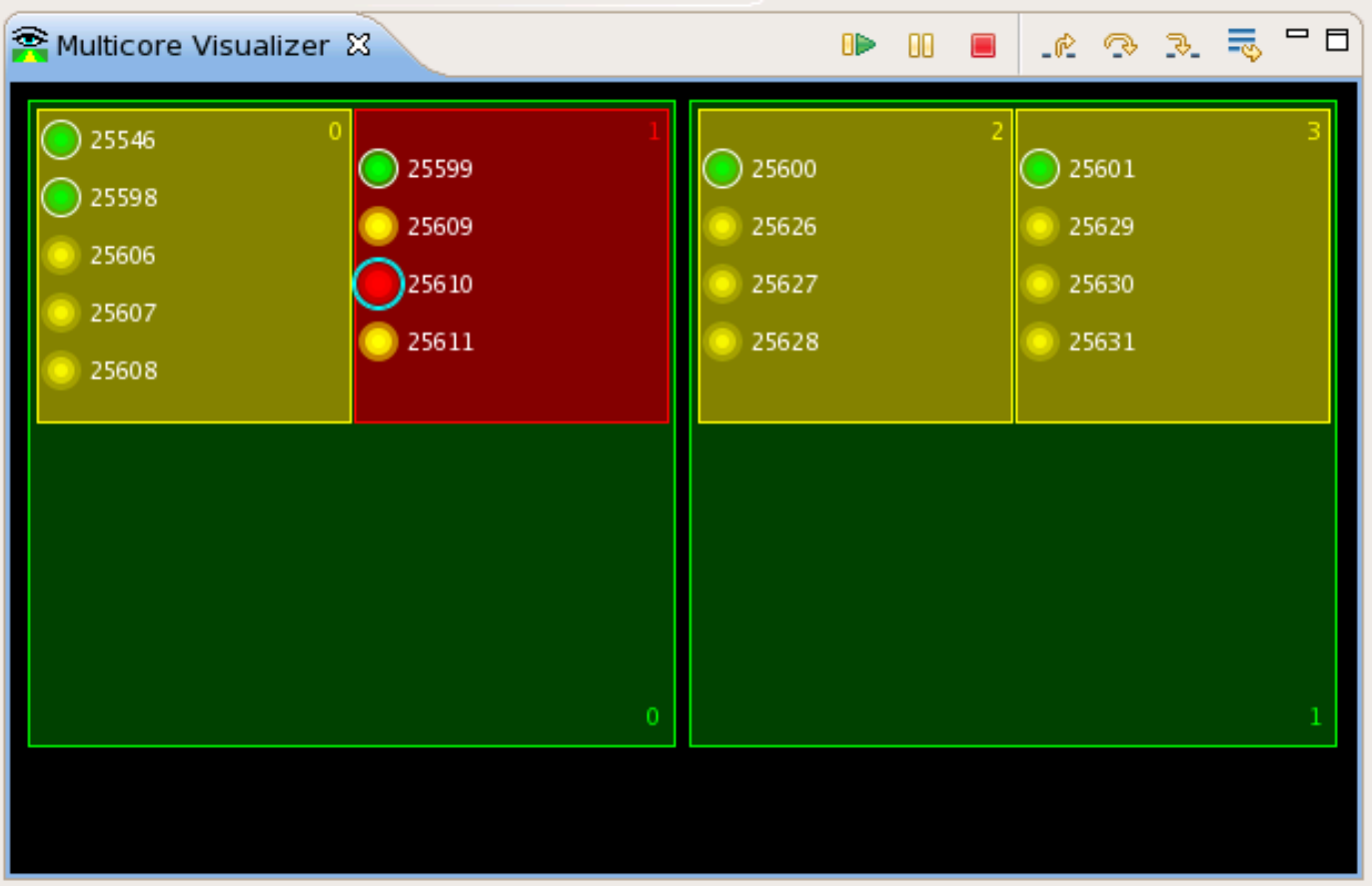
- *What this means is that we shouldn't abbreviate the truth but rather get a new method of presentation.*
  Edward Tufte

A VISUALIZATION FRAMEWORK

- There's also a need for a framework
- Rather than having everyone recreate the wheel, we need a platform to support visualization
- In the Eclipse spirit it should be flexible and extensible
- Should handle boilerplate code:
  - view/workbench interaction
  - selection
  - toolbars and context menus
- Visualizations should be able to focus on presentation.

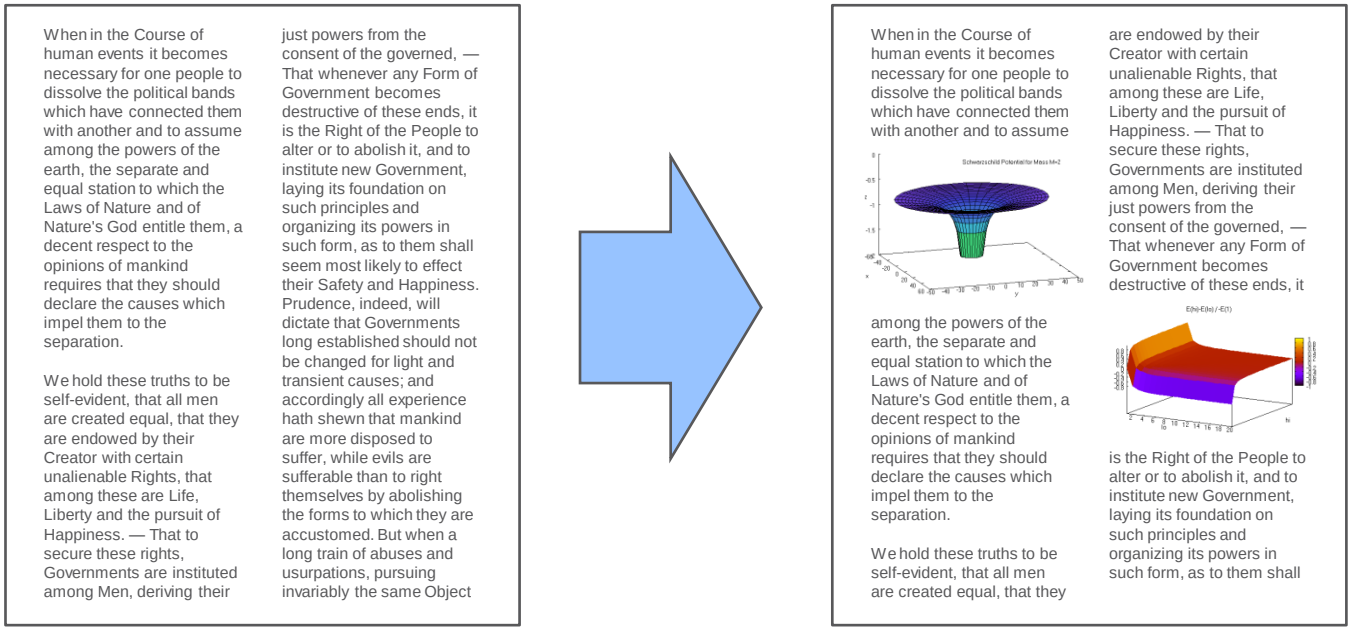
THE MULTICORE VISUALIZER

- New feature, currently optional
- Being added to CDT in Juno.



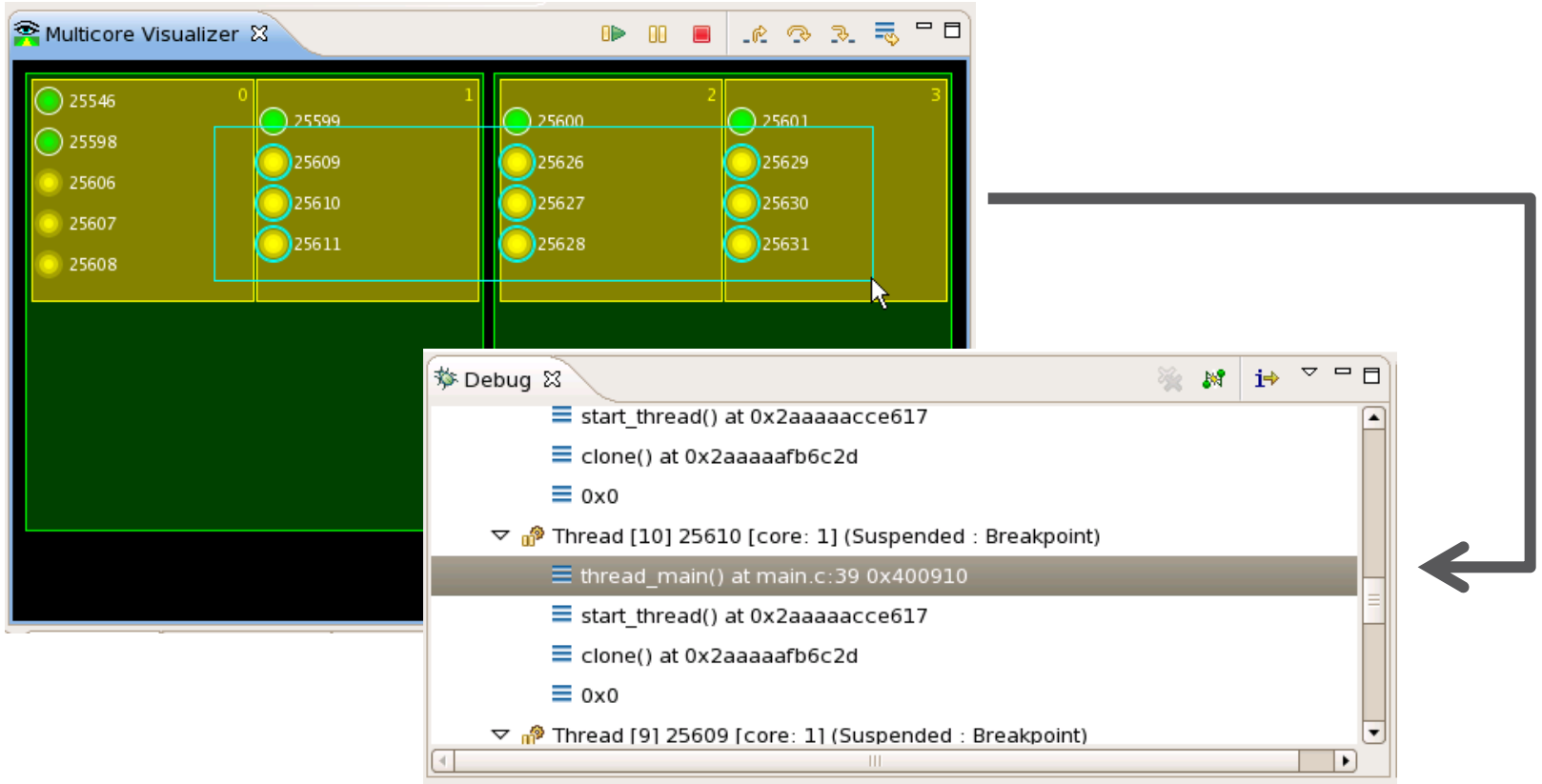
THE MULTICORE VISUALIZER

- Visualizer presents an overview of application debug state
- Doesn't *replace* the Debug View, it *augments* it
- Analogy: adding pictures to a journal article.
  - Pictures don't replace the text, instead they provide high-level context that makes it easier to parse the details.



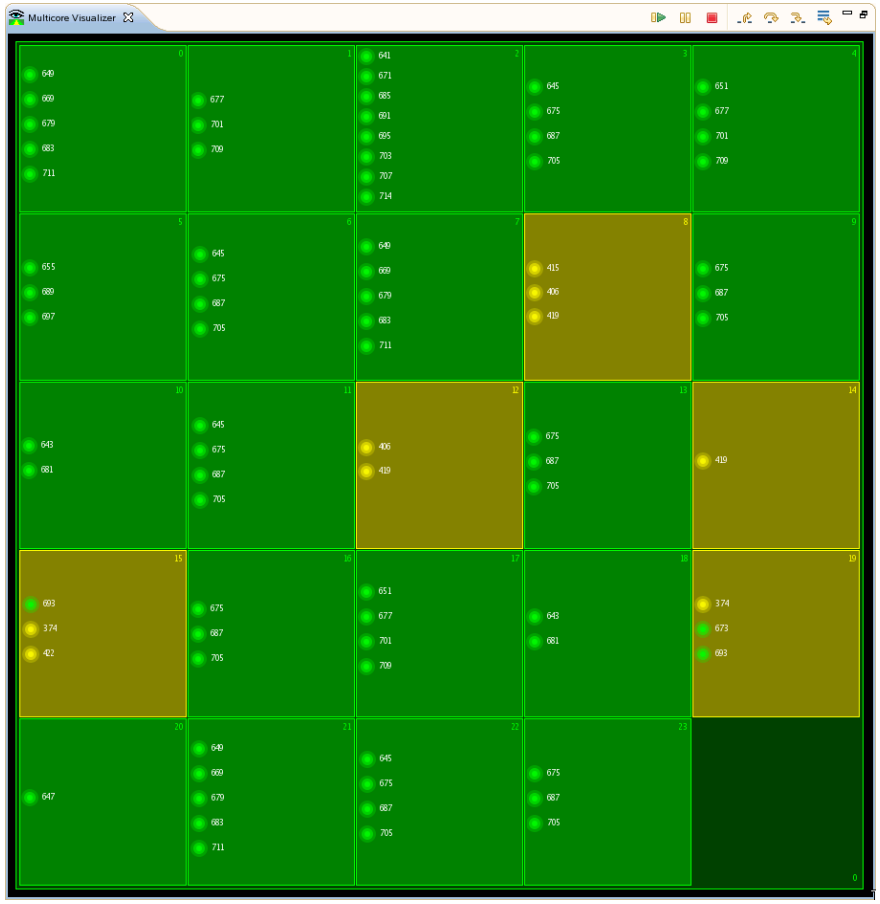
THE MULTICORE VISUALIZER

- Visualizer is interactive – can select & interact with program elements (cores, processes, threads)
- Can still use Debug View to get more detail



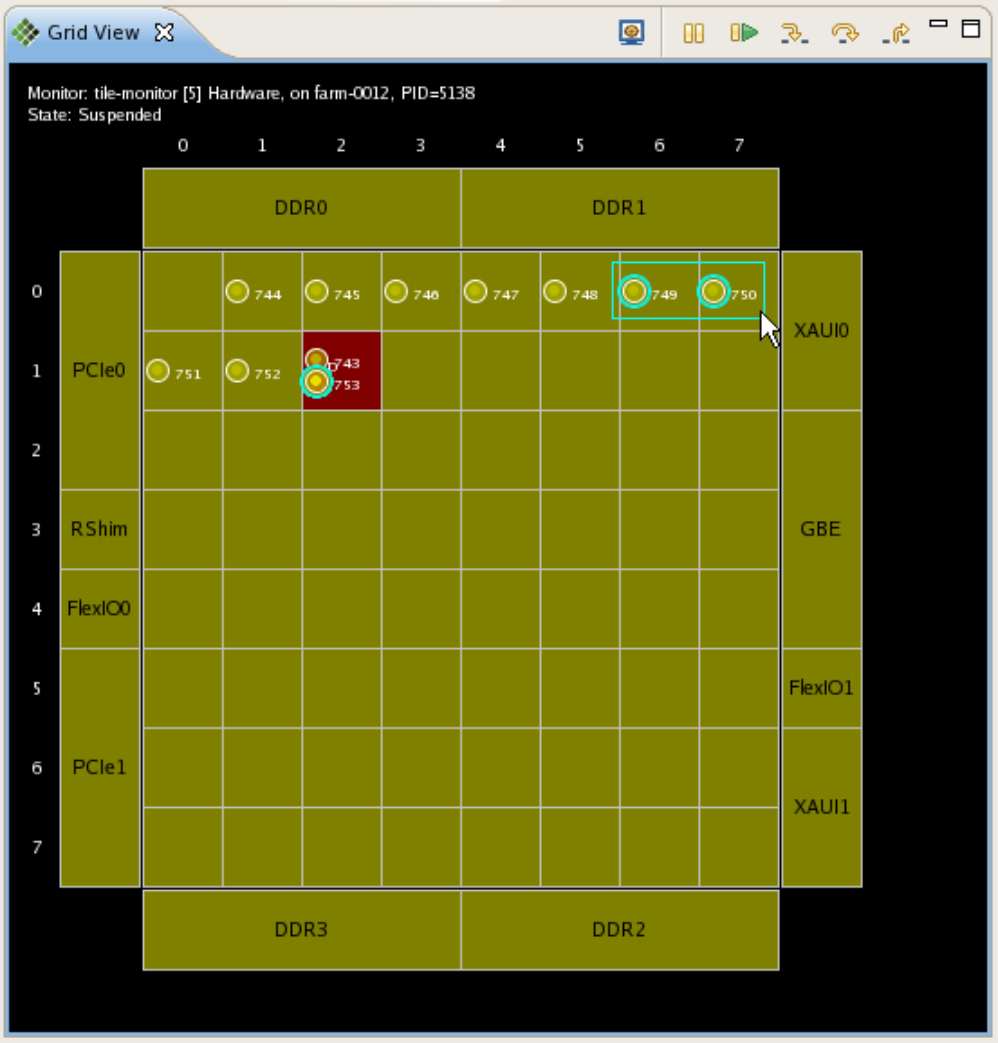
THE MULTICORE VISUALIZER

› Visualizer display scales to different cpu/core layouts:



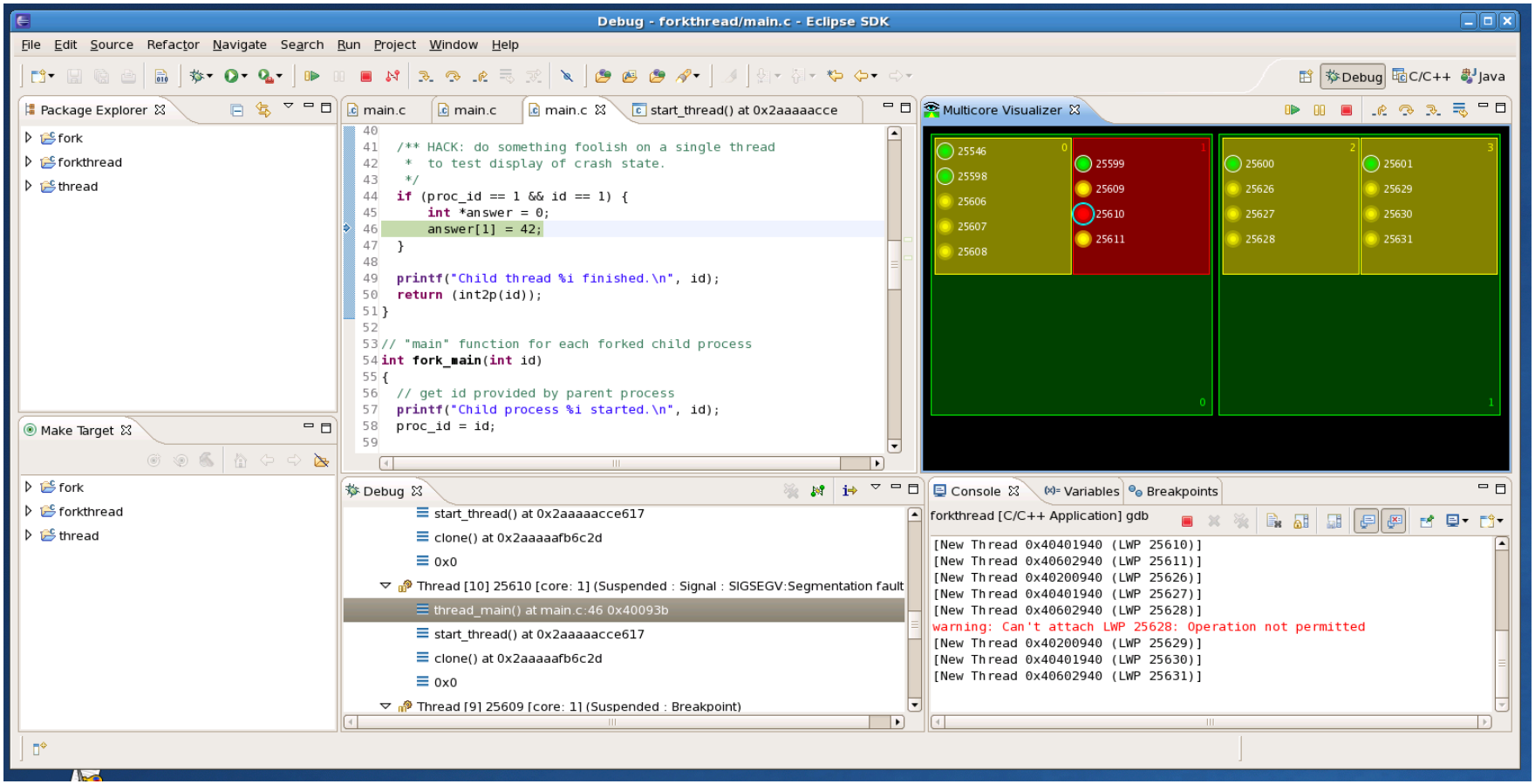
THE MULTICORE VISUALIZER

› Based on “Grid View” visualization in the Tilera IDE:



MULTICORE VISUALIZER

› Demo



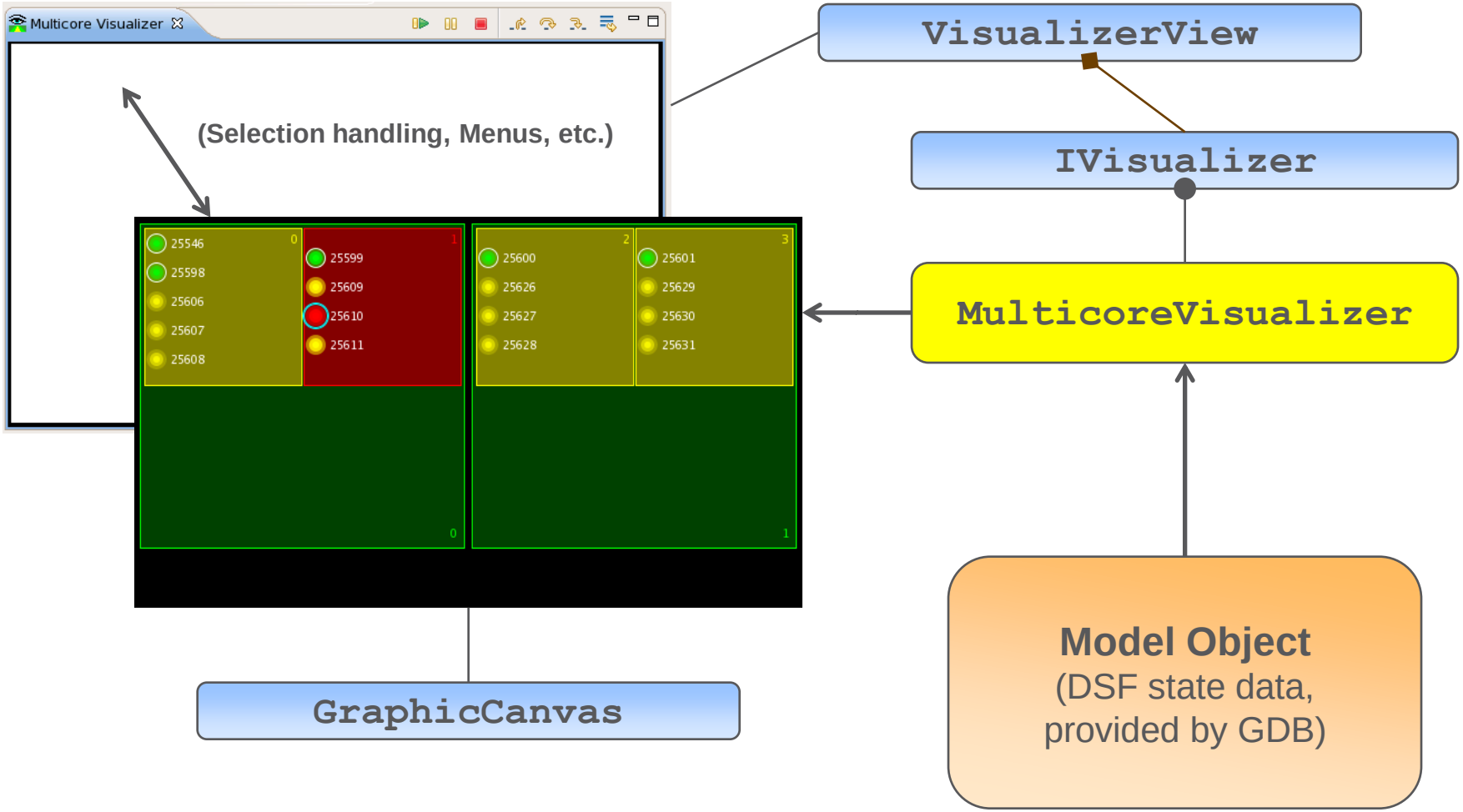
TILERA

ERICSSON

MULTICORE VISUALIZER

- Demo:
  - New Visualizer view in workbench
  - Displays cpus, cores, processes & threads
  - Sample program to debug: multi-process, multi-threaded
  - View updates automatically as program layout changes
  - Shows execution state of cpus, cores, processes, threads
  - Click/drag selection, updating of Debug View
  - Debug View state reflected in Visualizer
  - Breakpoint handling (resume, step, etc.)
  - Commands accessible from toolbars, context menu, and shortcuts
  - Variables view (as usual) reflects current selection
  - Displays crashed processes/threads in red
  - Can click on problem thread(s) to select and view source
  - Especially handy where there are multiple failures!

VISUALIZER FRAMEWORK



ON BEYOND DEBUGGING

- There's a framework because one view isn't sufficient
- Multicore Visualizer is currently aimed at debugging on homogenous multicore platforms; there are others
- Can extend multicore visualizer to suit your platform

- Also potential visualization uses beyond debugging
  - Visualizations of selected data
  - Use of "overlays" to select different kinds/views of data
  - Static code analysis displays, etc.
- Framework includes examples you can build upon

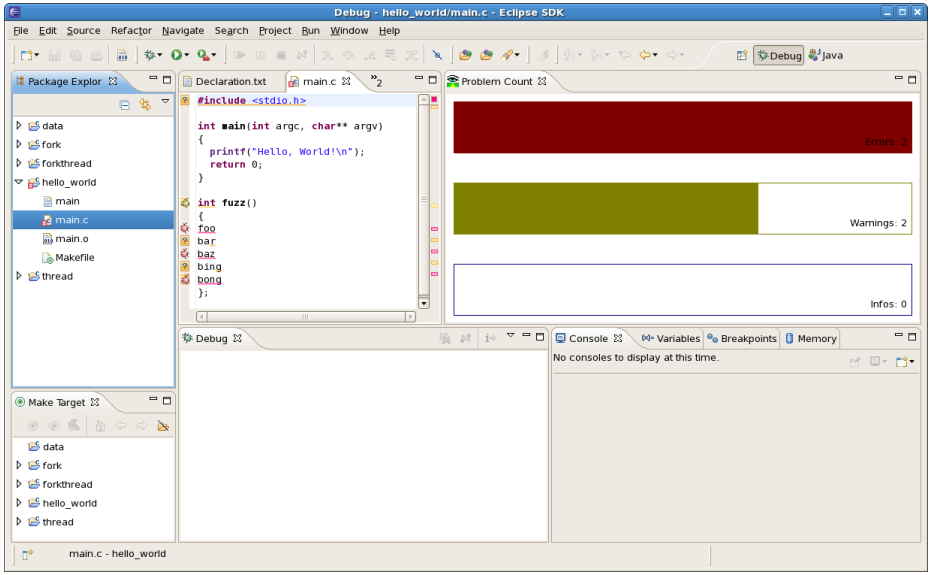
MULTIPLE VISUALIZERS

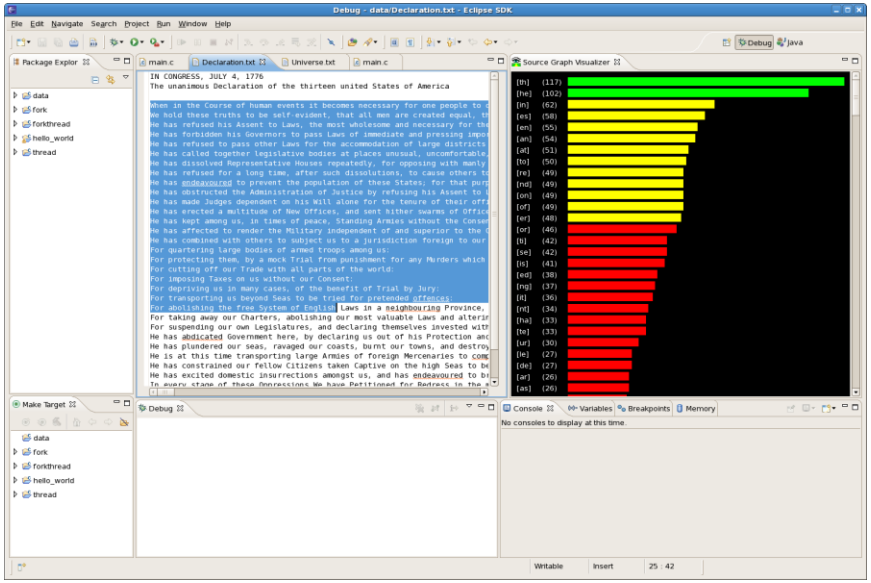
- Displayed visualizer is based on current selection.
- For multicore visualizer, the selection is DSF-GDB data
- Can also base visualizer on other information:
  - Selected projects (example: display of warnings/errors)
  - Selected text (program text, etc.)
- Anything that's selectable can potentially have a visualization associated with it.

- Visualizers report a weight indicating what they can handle
- The visualizer view selects the right visualizer for the current selection, based on visualizers' reported weights

MULTIPLE VISUALIZERS

- Demo:
  - problem count visualizer
  - source text analysis graph





TILERA

ERICSSON

SEEING IS UNDERSTANDING

- Visualization is necessary as a “big picture” approach to large applications on multicore hardware
- You can’t debug what you can’t see
- What you *can* see, you can understand.
- What you understand, you can reason and feel sure about

TILERA

ERICSSON

WE NEED NEW TOOLS

- **The Multicore Visualizer is not an end, but a beginning.**
- Visualizer exemplifies new kinds of tools and approaches needed for multicore and the “big application” era.
- These new tools are going to come from YOU the Eclipse developer community.
- The Visualizer framework provides a platform for development of more (and better!) visualizations.

- Let’s make sure our favorite platform, Eclipse, is ready for the tasks ahead!

IT TAKES MORE THAN ONE TOOL

› Visualizer is just one of the projects currently being worked on by the CDT Multicore Debug Workgroup

MULTICORE DEBUG WORKGROUP

- Joint effort to bring multicore debugging to the CDT
  - Visualizer, Pin&Clone, Multiprocess debug, etc
- Support for those that want to add new features
- Monthly conference calls (open to all interested and free ☺)
  - http://wiki.eclipse.org/CDT/MultiCoreDebugWorkingGroup

WORKGROUP PLANS

- Process/Thread/Core sets, as supported by GDB
- Global breakpoints
- OS Awareness
- Dynamic Printf

- Fully-integrated GDB console
  - complete GDB console that can be used jointly with Eclipse
- Scalability and performance
  - Ability for GDB to handle 100s or 1000s of threads/processes
- and more…

PTC SETS

Process Thread Core (PTC) sets control groups of debug elements:

- Step threads numbered between 34 and 59

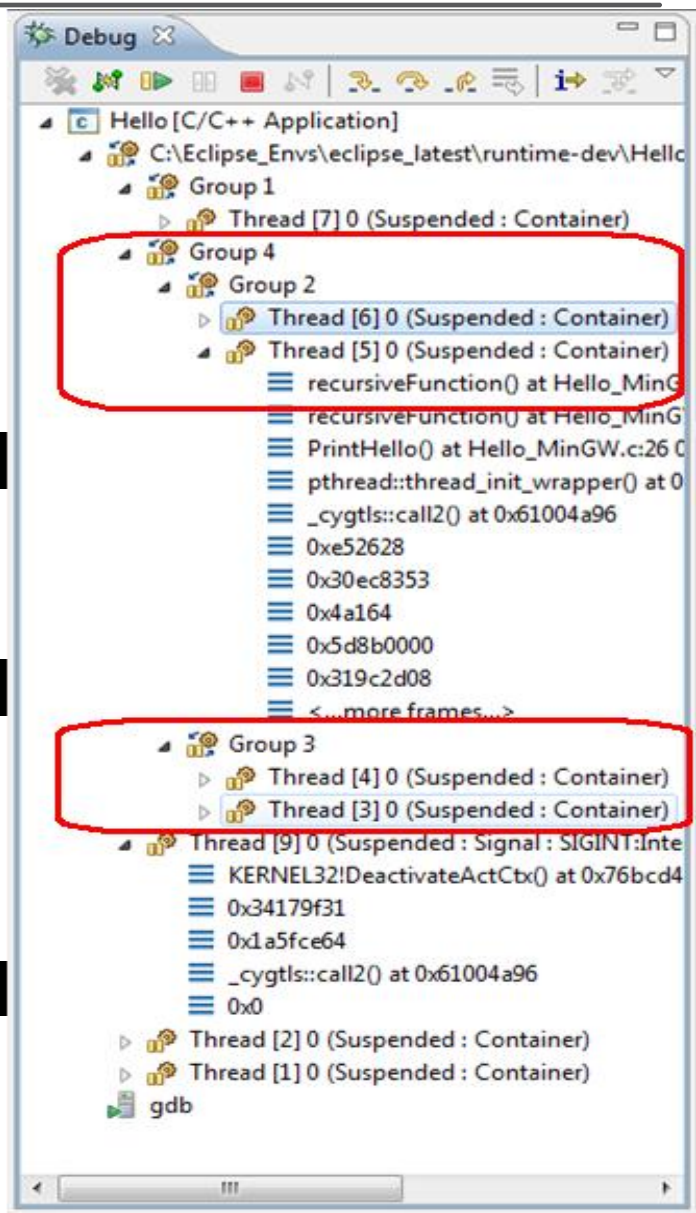
```
(gdb) step .34-59
```

- Step all threads running on core 2

```
(gdb) step @2
```

- Stop everything running on cores 5 to 7, preventing new threads from being started

```
(gdb) interrupt *.future@5-7
```



GLOBAL BREAKPOINTS

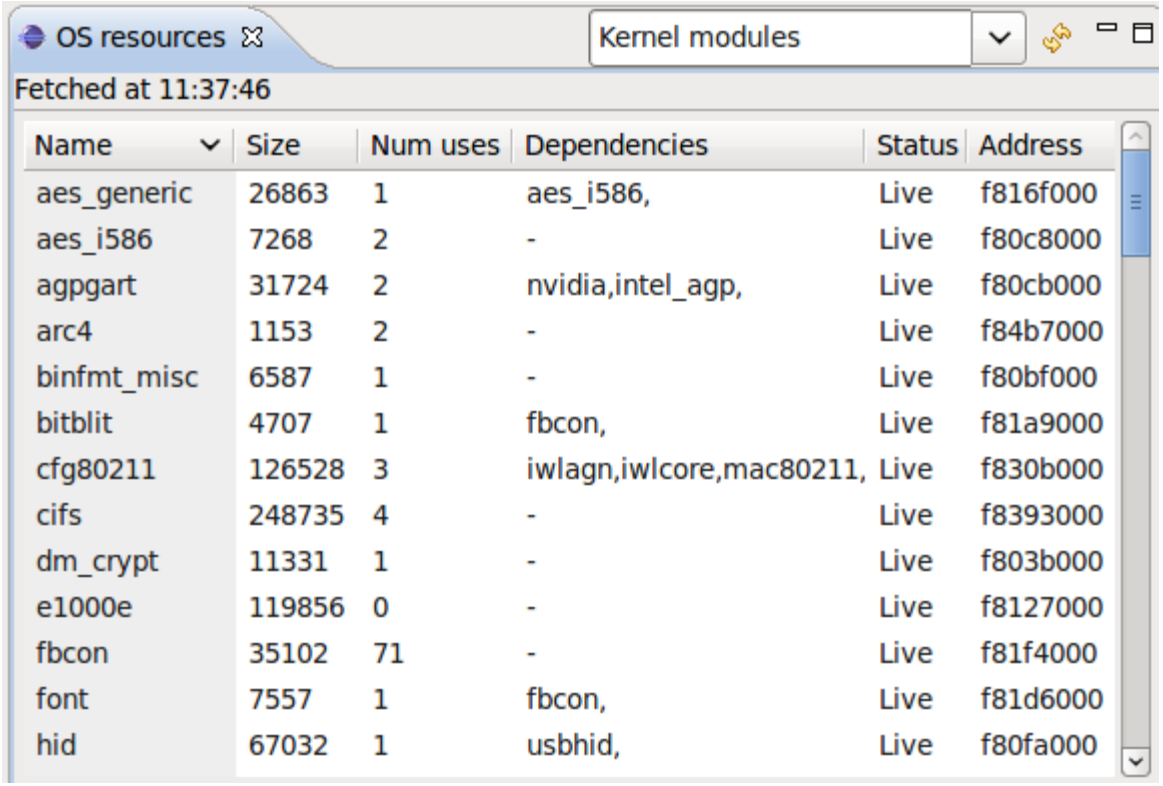
- Applies to every process
- Auto attach when hit
- Un-started or short lived process

[KERNEL MODULE REQUIRED]

OS AWARENESS

Sometimes examining OS Resources can help find bug

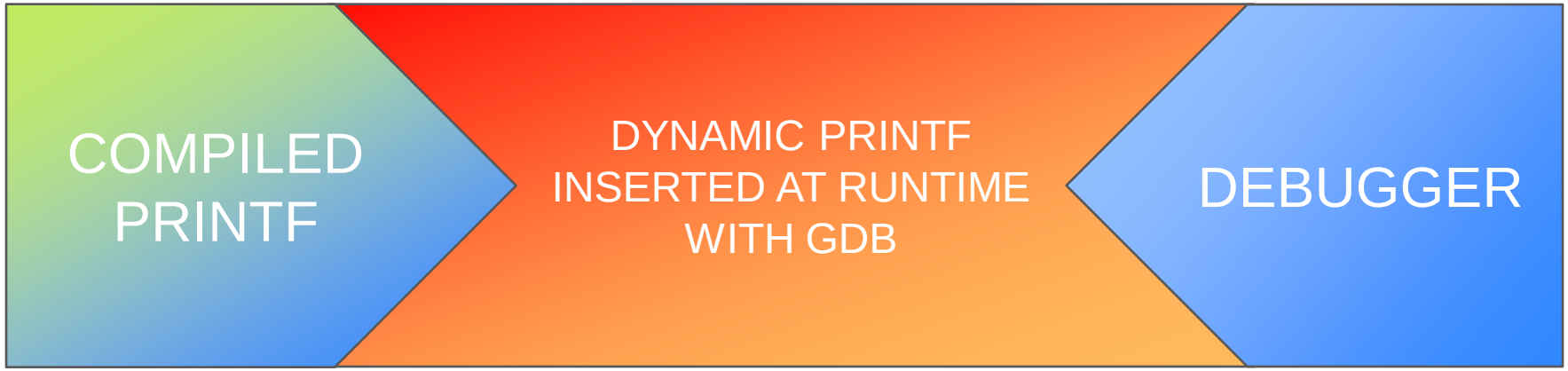
- Message Queues
- Loaded Kernel Modules
- Sockets
- Process Groups
- File Descriptors
- All Processes
- Semaphores
- All Threads
- Shared Memory Segments



Name	Size	Num uses	Dependencies	Status	Address
aes_generic	26863	1	aes_i586,	Live	f816f000
aes_i586	7268	2	-	Live	f80c8000
agpgart	31724	2	nvidia,intel_agp,	Live	f80cb000
arc4	1153	2	-	Live	f84b7000
binfmt_misc	6587	1	-	Live	f80bf000
bitblit	4707	1	fbcon,	Live	f81a9000
cfg80211	126528	3	iwlagn,iwlcore,mac80211,	Live	f830b000
cifs	248735	4	-	Live	f8393000
dm_crypt	11331	1	-	Live	f803b000
e1000e	119856	0	-	Live	f8127000
fbcon	35102	71	-	Live	f81f4000
font	7557	1	fbcon,	Live	f81d6000
hid	67032	1	usbhid,	Live	f80fa000

DYNAMIC-PRINTF

Sometimes tracing is necessary



TILERA

ERICSSON

SOME REFERENCES

› Multicore Debug workgroup, http://wiki.eclipse.org/CDT/MultiCoreDebugWorkingGroup

› CDT project, http://www.eclipse.org/cdt

› CDT Wiki, http://wiki.eclipse.org/CDT

› GDB, http://sourceware.org/gdb/

› Multicore Visualizer, http://bugs.eclipse.org/335027

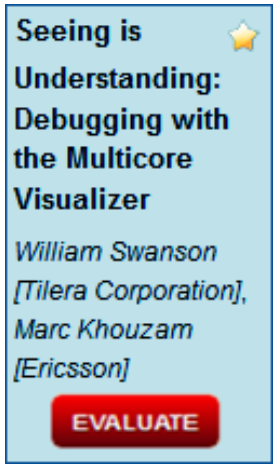
Contact:

cdt-dev@eclipse.org

Q&A

› Thanks for your time.

› Reminder: you can provide feedback:
› 1) go to www.eclipsecon.org
› 2) on Visualizer session, click “EVALUATE”
› 3) Vote (+1 0 -1) and add comments!



› Enjoy the rest of EclipseCon!

Multicore Visualizer Demo Slides, Version 1.0, 3/22/2012