Tools and scalability at openlab

Andrzej Nowak

June 21st 2010

2nd Workshop on adapting applications and computing services to multi-core and virtualization
CERN

Contents

- Architecture outlook and performance
- Shifts in performance tuning
- Tools used for performance optimization
  - Platform usage efficiency
    - perfmon2
    - VTune and PTU
    - Piersol HE (upcoming)
  - Threading performance
    - Thread profiler
    - Piersol HE (upcoming)
  - Correctness
    - Thread Checker
    - Cantua HE (upcoming)

Architecture outlook

- **Current Intel microarchitecture: “Westmere”**
  - 32nm, Nehalem based
  - Up to 6 cores per chip, we use 2 sockets: 24 threads
- **Current Intel multiprocessor architecture: “Beckton”**
  - 45nm, Nehalem-EX
  - Up to 8 cores per chip, up to 8 sockets per platform: 128 threads
- **Forthcoming developments:**
  - 2010/2011: “Sandy Bridge” – 256 bit SSE (AVX), 8 cores
  - 2011/2012: “Ivy Bridge” – shrink to 22nm
  - 2012/2013: “Haswell” – possible uarch redesign, further integration, FMA

Architecture and platform performance (1)

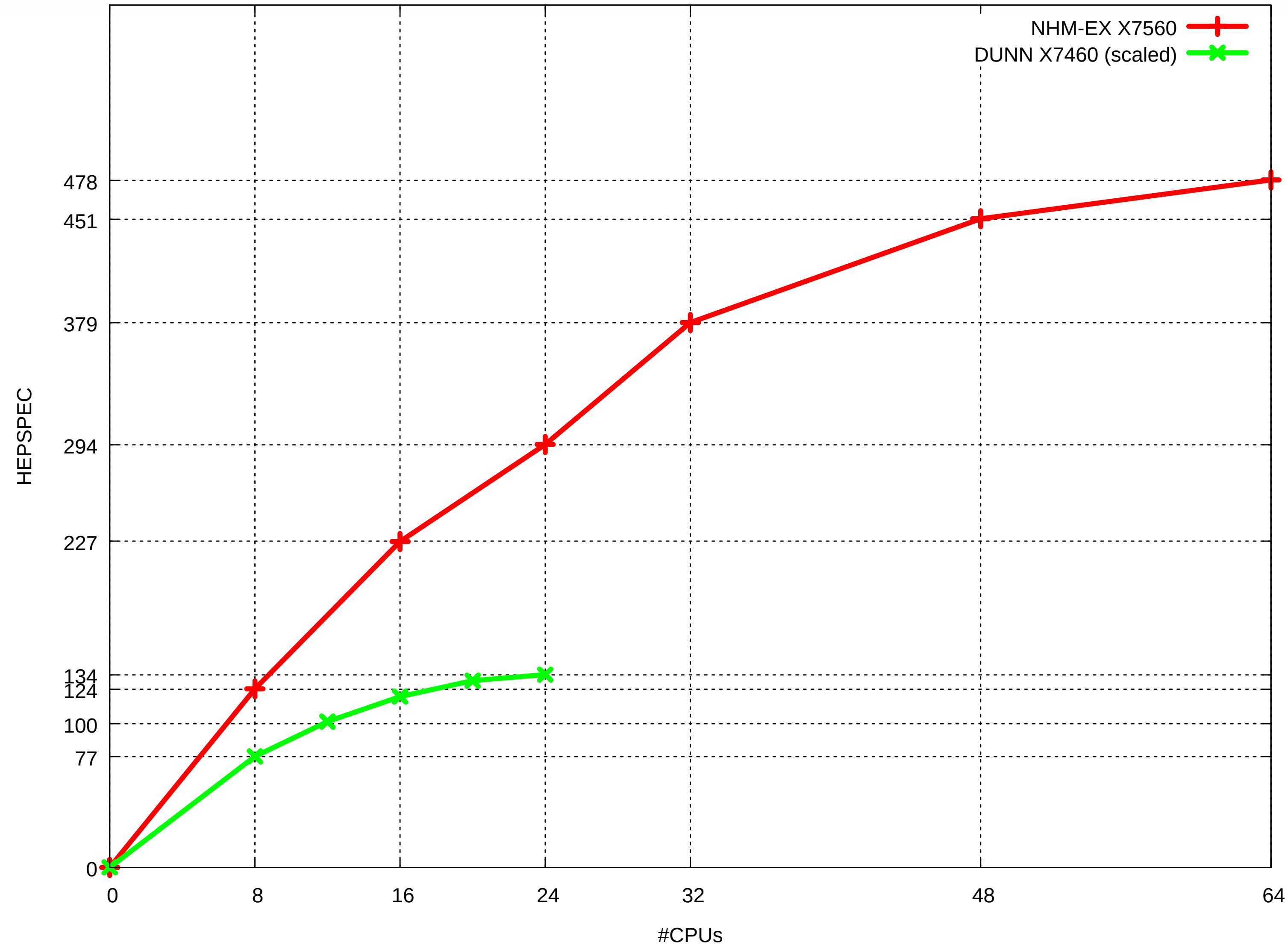
- **Openlab tested a Westmere-EP system and a Nehalem-EX system: papers available at http://cern.ch/openlab**
- **Westmere-EP (12 cores)**
  - 50% core increase, but HEPSPEC06 numbers only 32% better
  - Core Clock-per-clock throughput comparison very close to Nehalem
  - Overall improvements between 39% and 61% (mostly due to core increase) wrt Nehalem
  - SMT benefit mostly unchanged, 10-23% performance per Watt improvement
  - Some multi-core effects inhibit scalability

Architecture and platform performance (2)

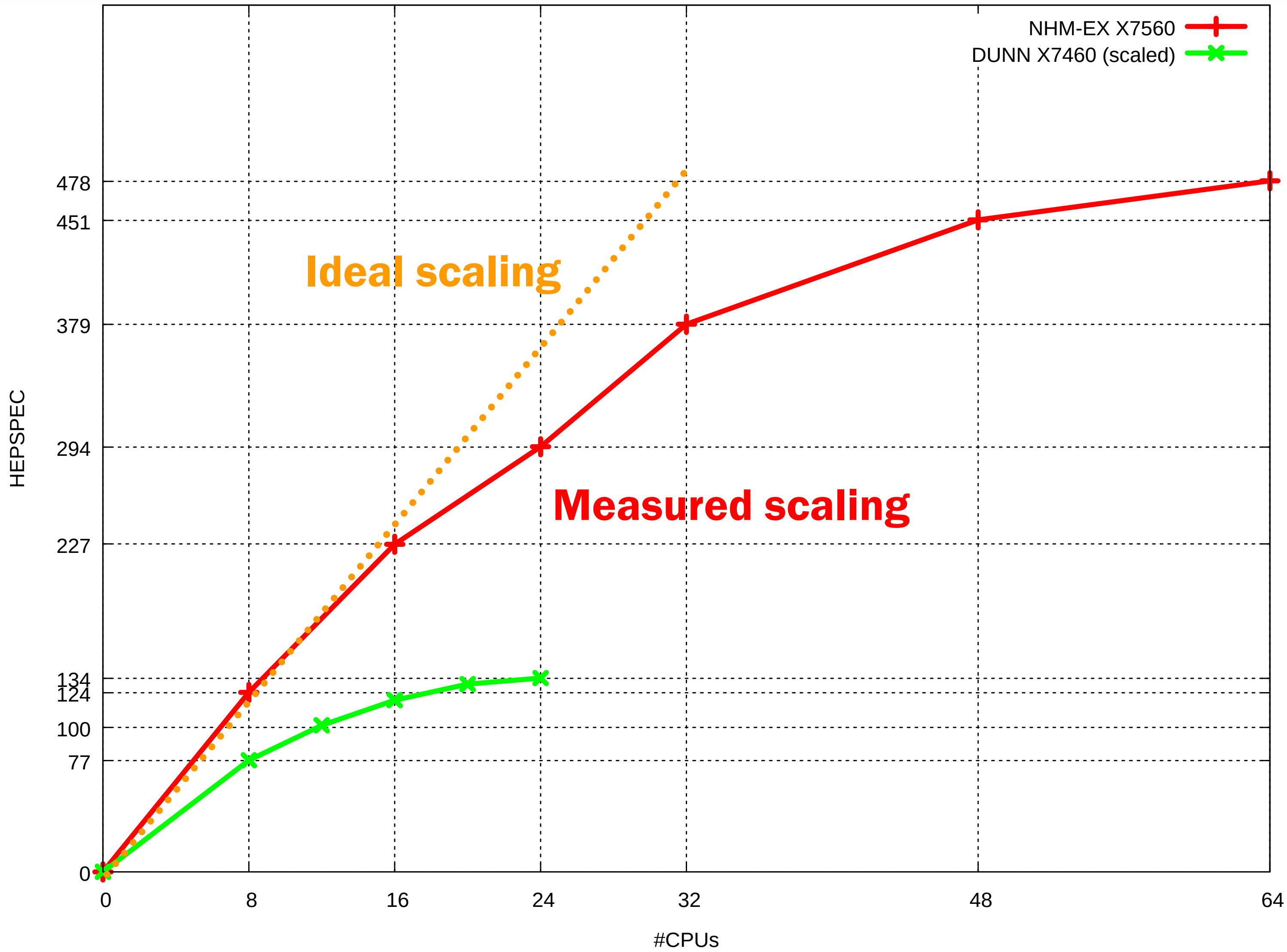
> Nehalem-EX (32 cores)

- 33% core increase reflected in performance
- 29.7x speedup obtained with a multi-threaded Geant4 prototype (FullCMS simulation) compared to 1 thread
  - Minor memory usage per thread
- "Top of the line" CPU comparison w/ Dunnington (24 cores), unscaled
  - 3x more throughput on HEPSPEC06
  - 11%-60% more throughput elsewhere
- SMT advantage: 19%-28%
  - More than expected; might be related to the fact that SW doesn't scale well to 32 cores
- Significant power consumption
- Some multi-core effects inhibit scalability – scalability lines on graphs should be straight, but are not

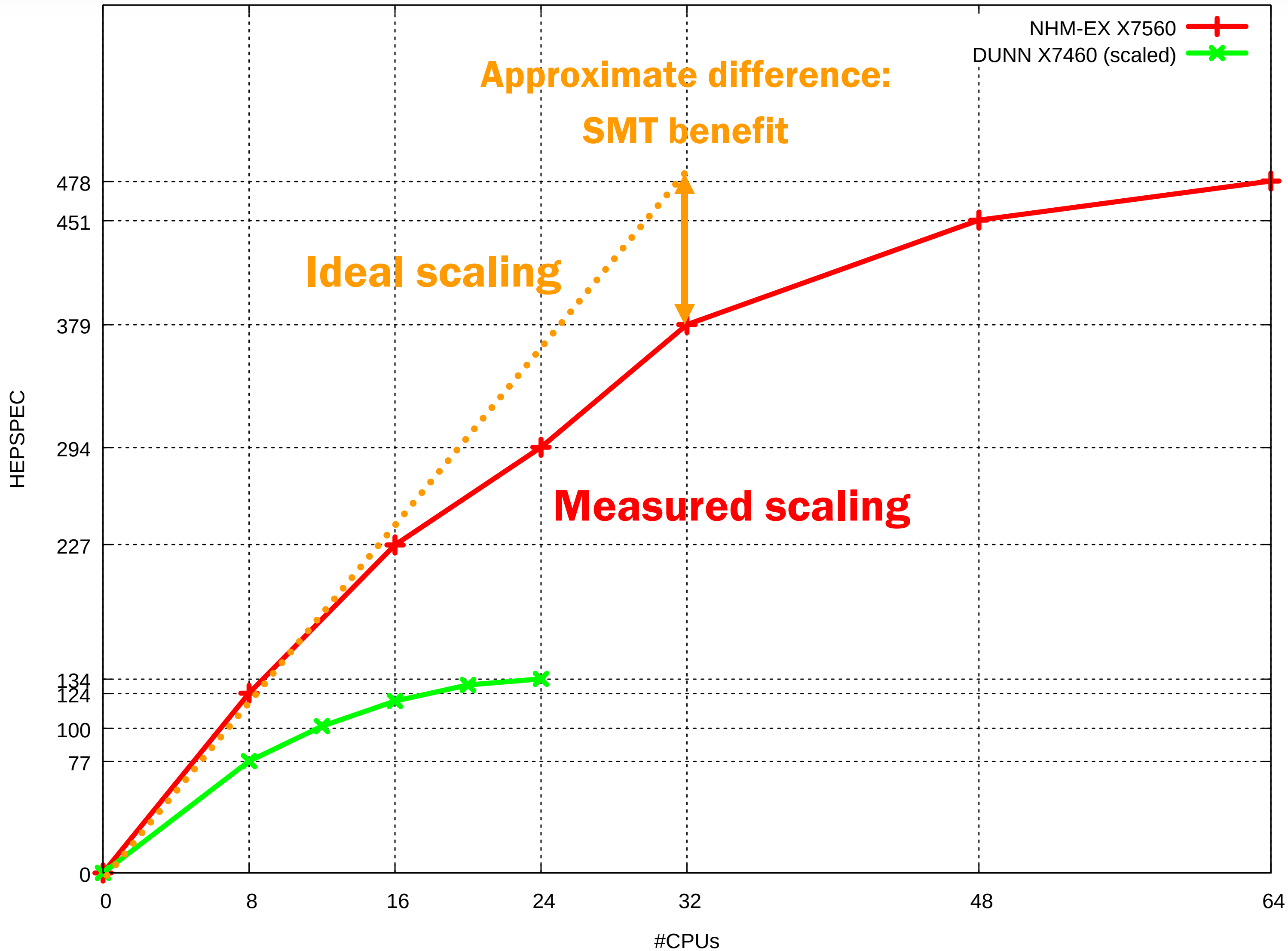
Nehalem-EX – HEPSPEC06 curve



Nehalem-EX – HEPSPEC06 curve



Nehalem-EX – HEPSPEC06 curve



Mainstream architecture and platform trends

> **Core development:**
- X86-64 remains the architecture of choice
- Continued <u>incremental</u> core increase
  - Decreasing relative cache advantage: smaller size, larger distances, increased penalty risk, lower predictability
- Added functional units
  - i.e. Crypto, AVX, FMA, Graphics
- 4-way hardware threading unlikely on OOO

> **Platform development:**
- Many-core scaling effects already noticeable
- NUMA effects already noticeable – large impact on multi-threading and SHM-based software, impact on multi-processing
- Increasing memory demand and pressure
  - More memory capacity expected (increasing part of the cost and power!)
  - More memory levels
  - More system memory does not guarantee more bandwidth, throughput or performance
- DP likely to remain the platform of choice and develop well for the next several years – market "sweet spot"
  - Pricing on more developed systems is prohibitive

Performance tuning outlook

- **Good news:**
  - Our window onto the hardware (performance counters) is being improved and systematized
    - The importance of hardware counters has been understood by the industry
    - openlab has direct links to HW/SW architects and other key people
  - Counter-level software becoming more widespread, growing adoption and understanding in the HEP community
  - Software is becoming more robust and accessible (i.e. GUIs)
- **Bad news:**
  - The complexity of performance analysis is growing
    - Architecture complexity is increasing (i.e. NUMA)
    - Language complexity is increasing (C++, Java proliferation)
    - It's harder to implement the tools because of the above and because of the increased complexity of the environment (kernel, etc)
  - And it was already hard in the first place

- **Where Linux "PCL" will miss, private companies (like Intel) will hit**
- **Hardware counters can be used to measure I/O, but it's hard**

perfmon2

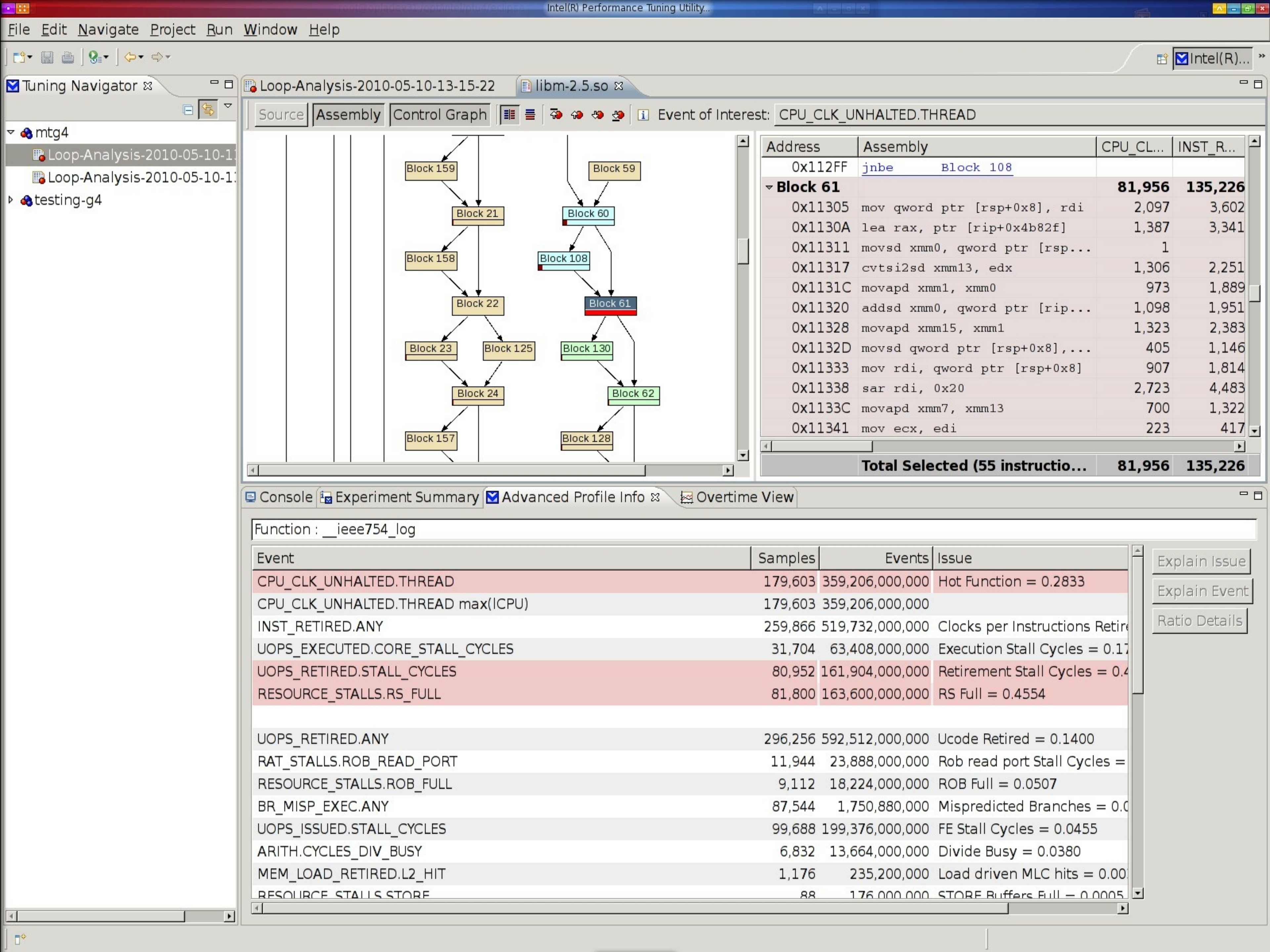
- **A Linux interface to the performance monitoring counters**
  - Supports counting, flat profiles and sampling in intervals
- **Pros:**
  - Lightweight, robust and open source
  - Long running project with extensive experience
  - Relatively stable and reliable, even with HEP software (in part thanks to efforts at openlab)
  - Easy to modify, easy to use to instrument HEP code (see CMS)
  - Direct link with the main architect
- **Cons:**
  - SLC-based kernel is provided in binary form, but is not standard SLC
  - Limited analytical capabilities in the package:
    - counting, flat profiles, sampling in intervals
  - Little hopes of further extensive development – “PCL” hijacked the functionality (far from being ready)

perfmon2 – obtaining and using

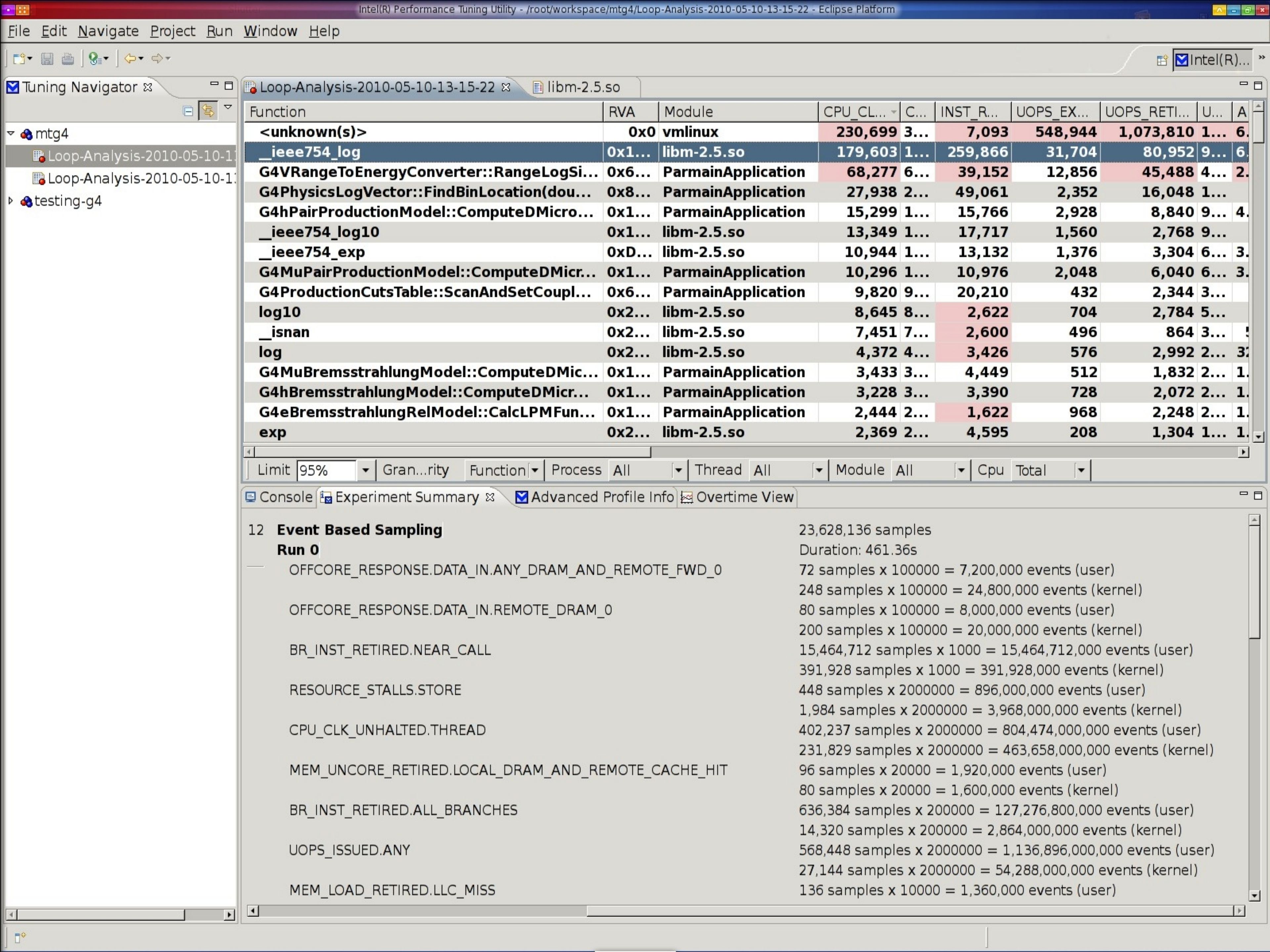
- **Restrictions:**
  - Kernel is modified – SLC based config but different version
  - Superuser access needed for installation
- **Profiles available for managed SLC5**
- **RPMs available for manual installation on SLC5**
- **Manual installation: complicated but possible**
- **Successfully used at CERN in many contexts**
  - Extensive monitoring in the computing center
  - Framework analysis and debugging
  - Snippet analysis
  - CMS instrumentation
- **All resources:**
  - twiki.cern.ch/twiki/bin/view/Openlab/Perfmon2

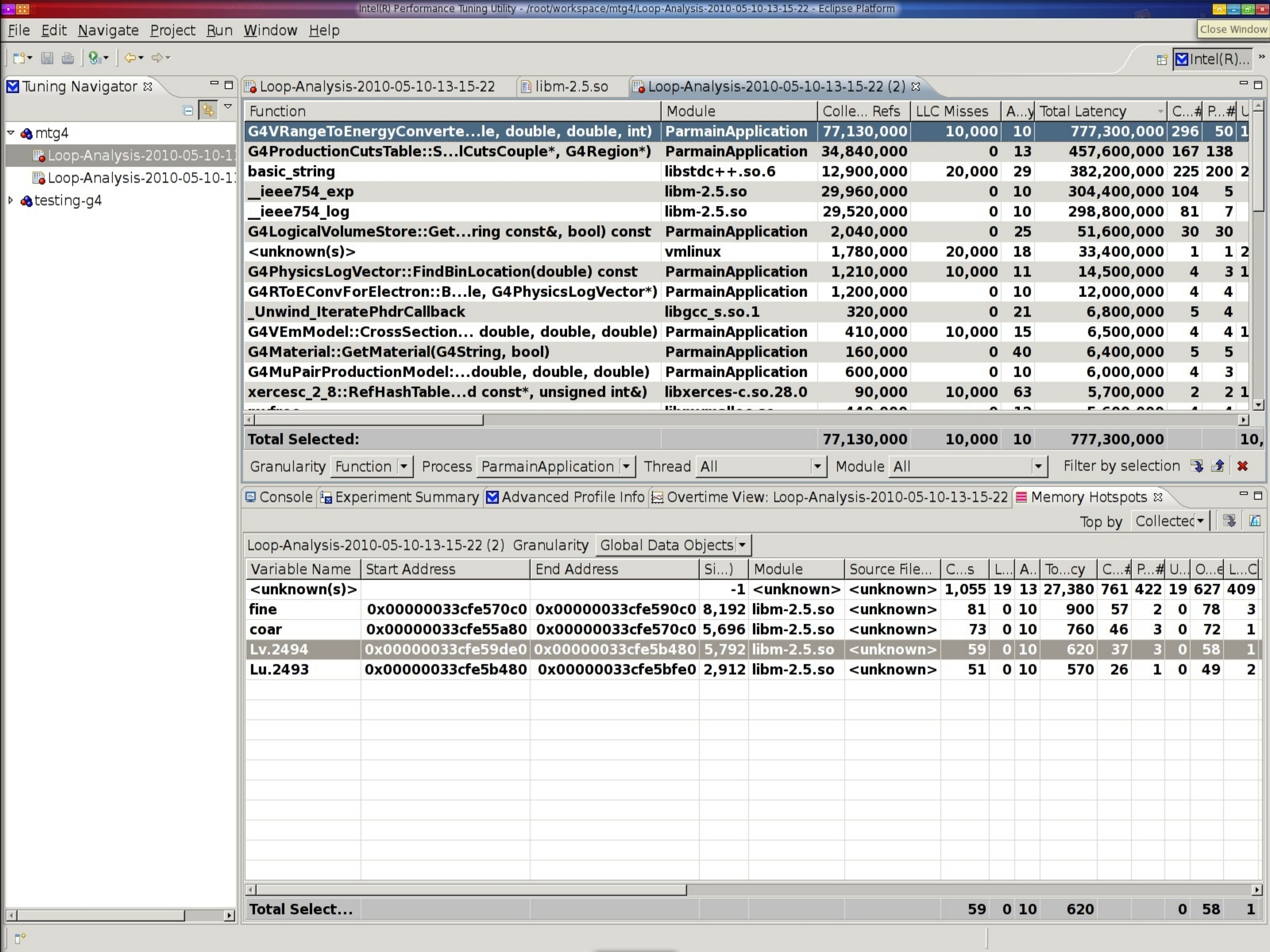
VTune and PTU

- **PTU – "VTune on steroids"**
  - The tool of choice of a performance monitoring expert
  - Supports counting, profiles, comparisons, memory profiling, call graphs (statistical and instrumentation based), basic block analysis, call count
  - Supports graphing: Events over time, events over IP, Memory traffic (heap profiler, data access profiling)
- **Pros:**
  - Very robust tool with a variety of analytical capabilities
  - Code instrumentation possible
  - Does not need a special kernel, works fine with SLC5
  - Direct link with the main architect (coming to CERN in July)
- **Cons:**
  - Might be too sophisticated for occasional users
  - Proprietary – license needed (although we have plenty at CERN)
    - Free support and discounted/free licenses for academic institutions
  - No source, modifications impossible
  - Official support very limited
  - Stability with HEP software sometimes leaves room for improvement
    - To be fixed in an upcoming future version 4.0

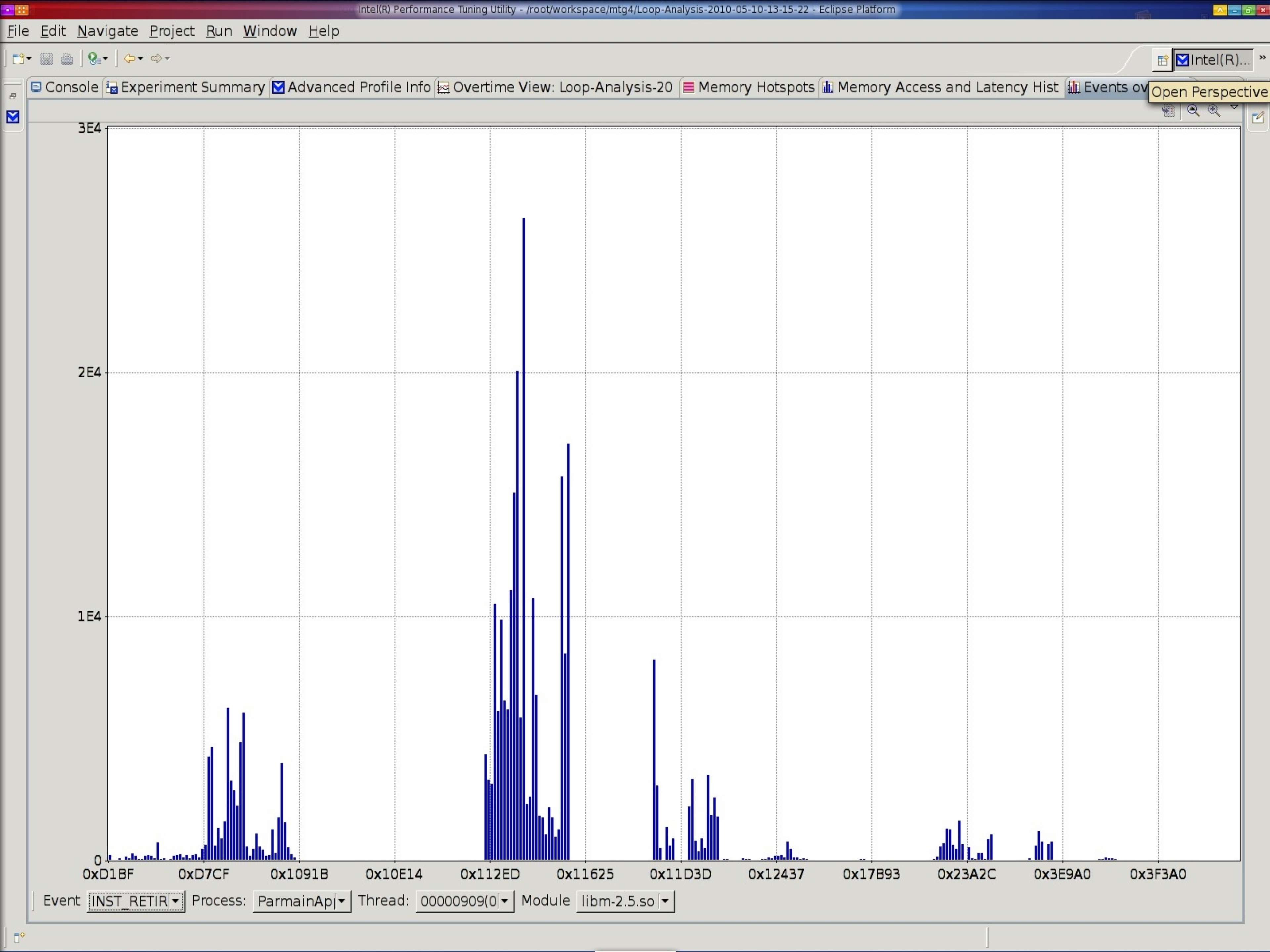


Intel(R) Performance Tuning Utility
File Edit Navigate Project Run Window Help
Tuning Navigator
mtg4
Loop-Analysis-2010-05-10-1
Loop-Analysis-2010-05-10-1
testing-g4
Loop-Analysis-2010-05-10-13-15-22
libm-2.5.so
Source
Assembly
Control Graph
Event of Interest: CPU_CLK_UNHALTED.THREAD
Block 159
Block 59
Block 21
Block 60
Block 158
Block 108
Block 22
Block 61
Block 23
Block 125
Block 130
Block 24
Block 62
Block 157
Block 128
Address | Assembly | CPU_CL... | INST_R...
0x112FF | jnbe Block 108 | |
Block 61 | | 81,956 | 135,226
0x11305 | mov qword ptr [rsp+0x8], rdi | 2,097 | 3,602
0x1130A | lea rax, ptr [rip+0x4b82f] | 1,387 | 3,341
0x11311 | movsd xmm0, qword ptr [rsp... | 1 |
0x11317 | cvtsi2sd xmm13, edx | 1,306 | 2,251
0x1131C | movapd xmm1, xmm0 | 973 | 1,889
0x11320 | addsd xmm0, qword ptr [rip... | 1,098 | 1,951
0x11328 | movapd xmm15, xmm1 | 1,323 | 2,383
0x1132D | movsd qword ptr [rsp+0x8],... | 405 | 1,146
0x11333 | mov rdi, qword ptr [rsp+0x8] | 907 | 1,814
0x11338 | sar rdi, 0x20 | 2,723 | 4,483
0x1133C | movapd xmm7, xmm13 | 700 | 1,322
0x11341 | mov ecx, edi | 223 | 417
Total Selected (55 instructio... | 81,956 | 135,226
Console
Experiment Summary
Advanced Profile Info
Overtime View
Function : __ieee754_log
Event | Samples | Events | Issue
CPU_CLK_UNHALTED.THREAD | 179,603 | 359,206,000,000 | Hot Function = 0.2833
CPU_CLK_UNHALTED.THREAD max(ICPU) | 179,603 | 359,206,000,000 |
INST_RETIRED.ANY | 259,866 | 519,732,000,000 | Clocks per Instructions Retir
UOPS_EXECUTED.CORE_STALL_CYCLES | 31,704 | 63,408,000,000 | Execution Stall Cycles = 0.17
UOPS_RETIRED.STALL_CYCLES | 80,952 | 161,904,000,000 | Retirement Stall Cycles = 0.4
RESOURCE_STALLS.RS_FULL | 81,800 | 163,600,000,000 | RS Full = 0.4554
UOPS_RETIRED.ANY | 296,256 | 592,512,000,000 | Ucode Retired = 0.1400
RAT_STALLS.ROB_READ_PORT | 11,944 | 23,888,000,000 | Rob read port Stall Cycles =
RESOURCE_STALLS.ROB_FULL | 9,112 | 18,224,000,000 | ROB Full = 0.0507
BR_MISP_EXEC.ANY | 87,544 | 1,750,880,000 | Mispredicted Branches = 0.0
UOPS_ISSUED.STALL_CYCLES | 99,688 | 199,376,000,000 | FE Stall Cycles = 0.0455
ARITH.CYCLES_DIV_BUSY | 6,832 | 13,664,000,000 | Divide Busy = 0.0380
MEM_LOAD_RETIRED.L2_HIT | 1,176 | 235,200,000 | Load driven MLC hits = 0.00
RESOURCE_STALLS.STORE | 88 | 176,000,000 | STORE Buffers Full = 0.0005
Explain Issue
Explain Event
Ratio Details


Intel(R) Performance Tuning Utility - /root/workspace/mtg4/Loop-Analysis-2010-05-10-13-15-22 - Eclipse Platform
File Edit Navigate Project Run Window Help
Intel(R)...
Tuning Navigator
mtg4
Loop-Analysis-2010-05-10-1
Loop-Analysis-2010-05-10-1
testing-g4
Loop-Analysis-2010-05-10-13-15-22
libm-2.5.so
Function | RVA | Module | CPU_CL... | C... | INST_R... | UOPS_EX... | UOPS_RETI... | U... | A
<unknown(s)> | 0x0 | vmlinux | 230,699 | 3... | 7,093 | 548,944 | 1,073,810 | 1... | 6.
__ieee754_log | 0x1... | libm-2.5.so | 179,603 | 1... | 259,866 | 31,704 | 80,952 | 9... | 6.
G4VRangeToEnergyConverter::RangeLogSi... | 0x6... | ParmainApplication | 68,277 | 6... | 39,152 | 12,856 | 45,488 | 4... | 2.
G4PhysicsLogVector::FindBinLocation(dou... | 0x8... | ParmainApplication | 27,938 | 2... | 49,061 | 2,352 | 16,048 | 1...
G4hPairProductionModel::ComputeDMicro... | 0x1... | ParmainApplication | 15,299 | 1... | 15,766 | 2,928 | 8,840 | 9... | 4.
__ieee754_log10 | 0x1... | libm-2.5.so | 13,349 | 1... | 17,717 | 1,560 | 2,768 | 9...
__ieee754_exp | 0xD... | libm-2.5.so | 10,944 | 1... | 13,132 | 1,376 | 3,304 | 6... | 3.
G4MuPairProductionModel::ComputeDMicr... | 0x1... | ParmainApplication | 10,296 | 1... | 10,976 | 2,048 | 6,040 | 6... | 3.
G4ProductionCutsTable::ScanAndSetCoupl... | 0x6... | ParmainApplication | 9,820 | 9... | 20,210 | 432 | 2,344 | 3...
log10 | 0x2... | libm-2.5.so | 8,645 | 8... | 2,622 | 704 | 2,784 | 5...
__isnan | 0x2... | libm-2.5.so | 7,451 | 7... | 2,600 | 496 | 864 | 3...
log | 0x2... | libm-2.5.so | 4,372 | 4... | 3,426 | 576 | 2,992 | 2... | 3.
G4MuBremsstrahlungModel::ComputeDMic... | 0x1... | ParmainApplication | 3,433 | 3... | 4,449 | 512 | 1,832 | 2... | 1.
G4hBremsstrahlungModel::ComputeDMicr... | 0x1... | ParmainApplication | 3,228 | 3... | 3,390 | 728 | 2,072 | 2... | 1.
G4eBremsstrahlungRelModel::CalcLPMFun... | 0x1... | ParmainApplication | 2,444 | 2... | 1,622 | 968 | 2,248 | 2... | 1.
exp | 0x2... | libm-2.5.so | 2,369 | 2... | 4,595 | 208 | 1,304 | 1... | 1.
Limit 95% Gran...rity Function Process All Thread All Module All Cpu Total
Console Experiment Summary Advanced Profile Info Overtime View
12 Event Based Sampling 23,628,136 samples
Run 0 Duration: 461.36s
OFFCORE_RESPONSE.DATA_IN.ANY_DRAM_AND_REMOTE_FWD_0 72 samples x 100000 = 7,200,000 events (user)
248 samples x 100000 = 24,800,000 events (kernel)
OFFCORE_RESPONSE.DATA_IN.REMOTE_DRAM_0 80 samples x 100000 = 8,000,000 events (user)
200 samples x 100000 = 20,000,000 events (kernel)
BR_INST_RETIRED.NEAR_CALL 15,464,712 samples x 1000 = 15,464,712,000 events (user)
391,928 samples x 1000 = 391,928,000 events (kernel)
RESOURCE_STALLS.STORE 448 samples x 2000000 = 896,000,000 events (user)
1,984 samples x 2000000 = 3,968,000,000 events (kernel)
CPU_CLK_UNHALTED.THREAD 402,237 samples x 2000000 = 804,474,000,000 events (user)
231,829 samples x 2000000 = 463,658,000,000 events (kernel)
MEM_UNCORE_RETIRED.LOCAL_DRAM_AND_REMOTE_CACHE_HIT 96 samples x 20000 = 1,920,000 events (user)
80 samples x 20000 = 1,600,000 events (kernel)
BR_INST_RETIRED.ALL_BRANCHES 636,384 samples x 200000 = 127,276,800,000 events (user)
14,320 samples x 200000 = 2,864,000,000 events (kernel)
UOPS_ISSUED.ANY 568,448 samples x 2000000 = 1,136,896,000,000 events (user)
27,144 samples x 2000000 = 54,288,000,000 events (kernel)
MEM_LOAD_RETIRED.LLC_MISS 136 samples x 10000 = 1,360,000 events (user)



Intel(R) Performance Tuning Utility - /root/workspace/mtg4/Loop-Analysis-2010-05-10-13-15-22 - Eclipse Platform
File Edit Navigate Project Run Window Help
Close Window
Intel(R)...
Tuning Navigator
mtg4
Loop-Analysis-2010-05-10-1
Loop-Analysis-2010-05-10-1
testing-g4
Loop-Analysis-2010-05-10-13-15-22
libm-2.5.so
Loop-Analysis-2010-05-10-13-15-22 (2)
Function | Module | Colle... Refs | LLC Misses | A...y | Total Latency | C...# | P...# | U
G4VRangeToEnergyConverte...le, double, double, int) | ParmainApplication | 77,130,000 | 10,000 | 10 | 777,300,000 | 296 | 50 | 1
G4ProductionCutsTable::S...lCutsCouple*, G4Region*) | ParmainApplication | 34,840,000 | 0 | 13 | 457,600,000 | 167 | 138
basic_string | libstdc++.so.6 | 12,900,000 | 20,000 | 29 | 382,200,000 | 225 | 200 | 2
__ieee754_exp | libm-2.5.so | 29,960,000 | 0 | 10 | 304,400,000 | 104 | 5
__ieee754_log | libm-2.5.so | 29,520,000 | 0 | 10 | 298,800,000 | 81 | 7
G4LogicalVolumeStore::Get...ring const&, bool) const | ParmainApplication | 2,040,000 | 0 | 25 | 51,600,000 | 30 | 30
<unknown(s)> | vmlinux | 1,780,000 | 20,000 | 18 | 33,400,000 | 1 | 1 | 2
G4PhysicsLogVector::FindBinLocation(double) const | ParmainApplication | 1,210,000 | 10,000 | 11 | 14,500,000 | 4 | 3 | 1
G4RToEConvForElectron::B...le, G4PhysicsLogVector*) | ParmainApplication | 1,200,000 | 0 | 10 | 12,000,000 | 4 | 4
_Unwind_IteratePhdrCallback | libgcc_s.so.1 | 320,000 | 0 | 21 | 6,800,000 | 5 | 4
G4VEmModel::CrossSection... double, double, double) | ParmainApplication | 410,000 | 10,000 | 15 | 6,500,000 | 4 | 4 | 1
G4Material::GetMaterial(G4String, bool) | ParmainApplication | 160,000 | 0 | 40 | 6,400,000 | 5 | 5
G4MuPairProductionModel:...double, double, double) | ParmainApplication | 600,000 | 0 | 10 | 6,000,000 | 4 | 3
xercesc_2_8::RefHashTable...d const*, unsigned int&) | libxerces-c.so.28.0 | 90,000 | 10,000 | 63 | 5,700,000 | 2 | 2 | 1
Total Selected: | 77,130,000 | 10,000 | 10 | 777,300,000 | 10,
Granularity Function Process ParmainApplication Thread All Module All Filter by selection
Console
Experiment Summary
Advanced Profile Info
Overtime View: Loop-Analysis-2010-05-10-13-15-22
Memory Hotspots
Top by Collected
Loop-Analysis-2010-05-10-13-15-22 (2) Granularity Global Data Objects
Variable Name | Start Address | End Address | Si...) | Module | Source File... | C...s | L... | A... | To...cy | C...# | P...# | U... | O...e | L...C
<unknown(s)> | | | -1 | <unknown> | <unknown> | 1,055 | 19 | 13 | 27,380 | 761 | 422 | 19 | 627 | 409
fine | 0x00000033cfe570c0 | 0x00000033cfe590c0 | 8,192 | libm-2.5.so | <unknown> | 81 | 0 | 10 | 900 | 57 | 2 | 0 | 78 | 3
coar | 0x00000033cfe55a80 | 0x00000033cfe570c0 | 5,696 | libm-2.5.so | <unknown> | 73 | 0 | 10 | 760 | 46 | 3 | 0 | 72 | 1
Lv.2494 | 0x00000033cfe59de0 | 0x00000033cfe5b480 | 5,792 | libm-2.5.so | <unknown> | 59 | 0 | 10 | 620 | 37 | 3 | 0 | 58 | 1
Lu.2493 | 0x00000033cfe5b480 | 0x00000033cfe5bfe0 | 2,912 | libm-2.5.so | <unknown> | 51 | 0 | 10 | 570 | 26 | 1 | 0 | 49 | 2
Total Select... | 59 | 0 | 10 | 620 | 0 | 58 | 1


Intel(R) Performance Tuning Utility - /root/workspace/mtg4/Loop-Analysis-2010-05-10-13-15-22 - Eclipse Platform
File Edit Navigate Project Run Window Help
Intel(R)...
Console
Experiment Summary
Advanced Profile Info
Overtime View: Loop-Analysis-20
Memory Hotspots
Memory Access and Latency Hist
Events ov
Open Perspective
3E4
2E4
1E4
0
0xD1BF
0xD7CF
0x1091B
0x10E14
0x112ED
0x11625
0x11D3D
0x12437
0x17B93
0x23A2C
0x3E9A0
0x3F3A0
Event
INST_RETIR
Process:
ParmainAp
Thread:
00000909(0
Module
libm-2.5.so

PTU – obtaining and using

> **Practical restrictions:**
- Superuser access needed for installation
- Works best with the GUI, command line obscure
- Rather sluggish – there are some overheads which are still being investigated

> **Obtaining PTU:**
- Download from the Intel website
- Source the license file in your shell as described here:
  - twiki.cern.ch/twiki/bin/view/Openlab/IntelTools
- Install PTU by running the installer
- Run script to compile and install the kernel driver

> **Running:**
- Use GUI or refer to user guide for numerous command line options

> **Successfully used at CERN for several projects**
- Framework and snippet analysis and debugging
- CMS monitoring and source annotation
- ATLAS, Geant4 monitoring and analysis

Upcoming tool from Intel – VTune replacement

> **A promising next-generation performance monitoring tool from Intel**

- A fusion of PTU and Thread Profiler, will support threaded profiling
- Simplified and modernized GUI, without sacrificing functionality
- Several levels of analysis depth available
  - Many PIN-based engines (http://pintool.org)
- Linux native
- Kernel driver will be needed (but no kernel change required)
- More in H2 2010 – possible workshops

Threading performance – Thread Profiler

- **Thread Profiler from Intel**
  - Bottleneck analysis
  - Synchronization issues
  - Locates inefficiencies and sub-optimal resource usage
- **Visual representation of threading performance issues**
  - Concurrency and thread state graph ("profile view")
    - Different breakdowns available
  - Transitions graph and timeline ("timeline view")
- **Supports OpenMP and pthreads (and Windows API threads)**
- **Source instrumentation and binary instrumentation**
- **Remote data collection possible**
- **Used successfully for debugging prototype multi-threaded applications at CERN**

Thread profiler – view

- Dark green = good work, light green = no work, waiting, idle
- Orange/red – critical path
- Yellow - transitions

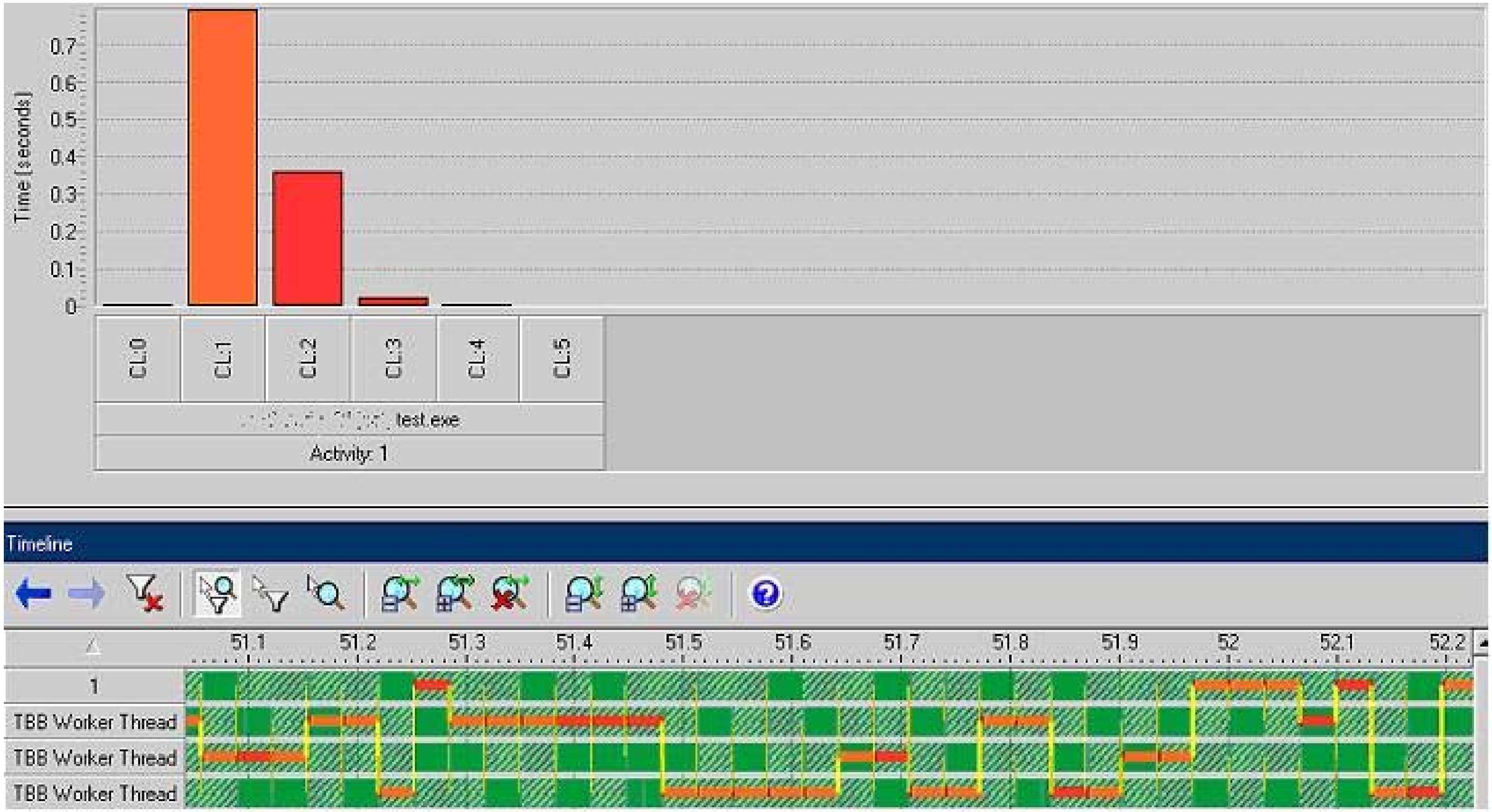
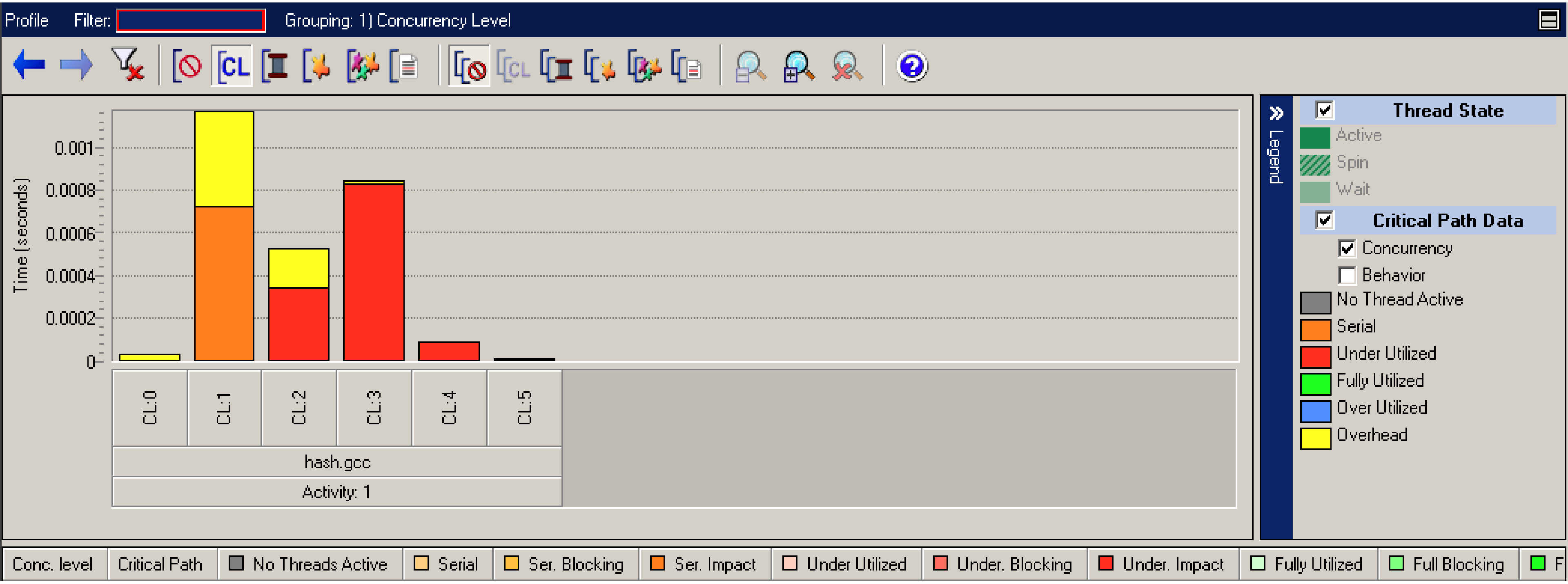


Image from software.intel.com

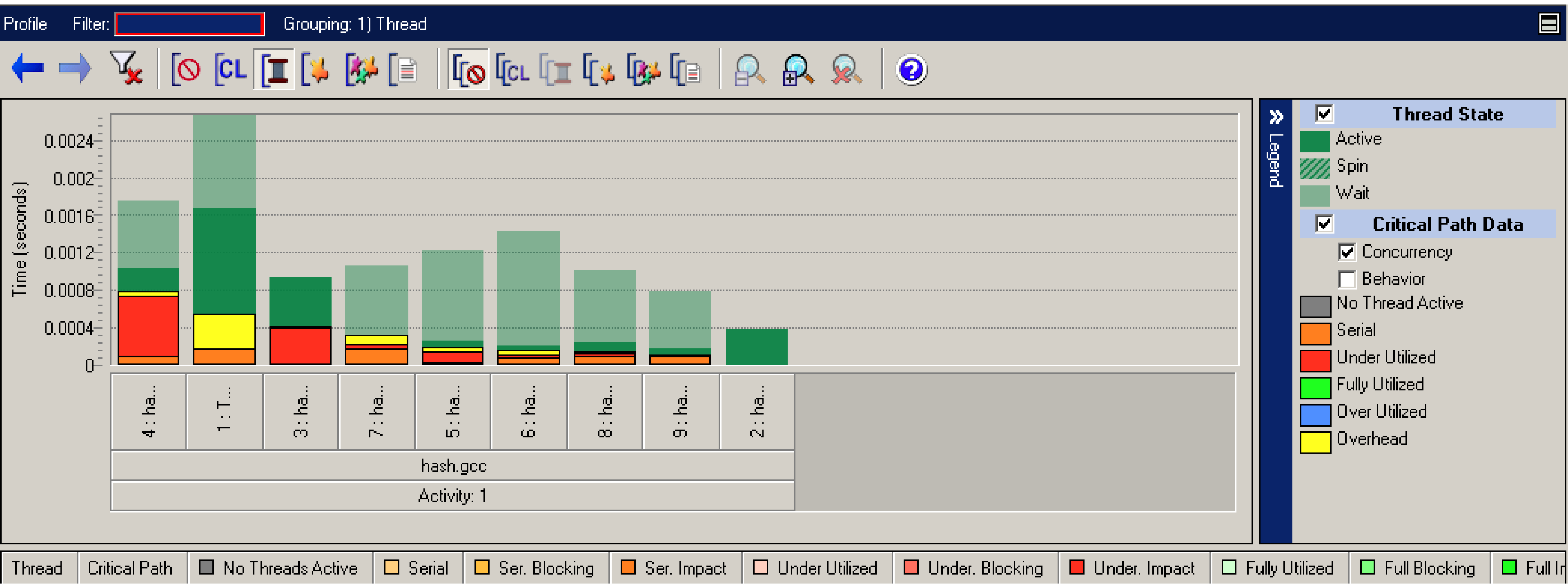
Thread profiler – profiler view

> Concurrency level grouping



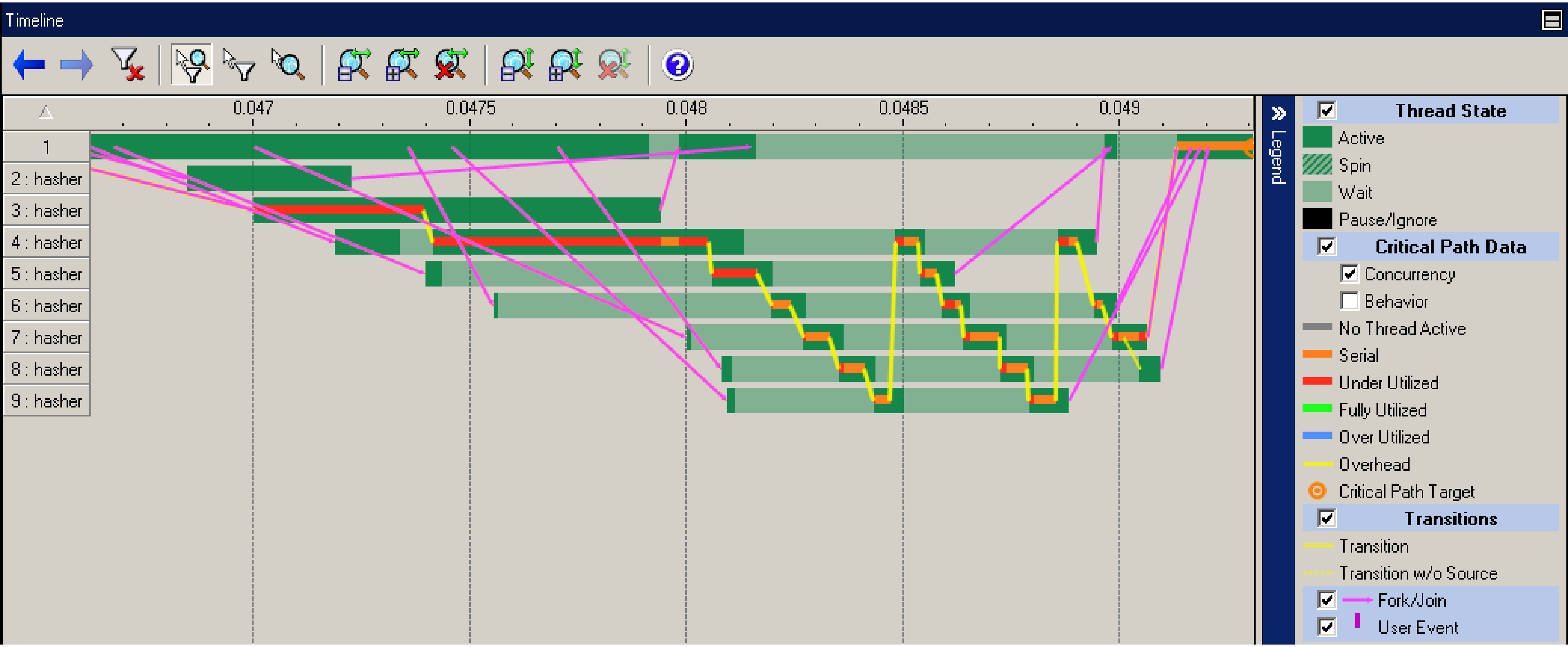
Thread profiler – profiler view

> Thread grouping



Thread profiler – timeline view

- Critical path display
- Light green portions show wait time, dark green show work



Thread Profiler - typical workflow

1. **Collect on Linux**
   - Either compile with **`–tprofile or –openmp_profile`** (source instrumentation - better option)
   - Or just run **`tprofile_cl yourprogram`** (binary instrumentation)
2. **Copy bistro/.tp output files over to Windows and analyze in the GUI**

Checking for correctness

> Intel Thread Checker

- Rather old tool with interesting capabilities
- Anomaly detection: Deadlocks, Stalls, API usage violations, Race conditions, Memory overwrites, Abandoned locks

> Data analysis:

- Data collection and text analysis in Linux
- The trace file can later be opened in the Windows version of the tool for detailed analysis

Thread Checker screenshot

```
|ID|Short De|Sever|C|Contex|Description                        |1st Acc|2nd Acc|
|  |scriptio|ity  |o|t[Best|                                   |ess[Bes|ess[Bes|
|  |n       |Name |u|]     |                                   |t]     |t]     |
|  |        |     |n|      |                                   |       |       |
|  |        |     |t|      |                                   |       |       |

|1 |Read -> |Error|3|"datar|Memory write at "dataraces.c":26   |"datara|"datara|
|  |Write da|     | |aces.c|conflicts with a prior memory read|ces.c":|ces.c":|
|  |ta-race |     | |":20  |at "dataraces.c":26 (anti          |26     |26     |
|  |        |     | |      |dependence)                        |       |       |

|2 |Write ->|Error|3|"datar|Memory read at "dataraces.c":26    |"datara|"datara|
|  |Read dat|     | |aces.c|conflicts with a prior memory      |ces.c":|ces.c":|
|  |a-race  |     | |":20  |write at "dataraces.c":26 (flow    |26     |26     |
|  |        |     | |      |dependence)                        |       |       |

|3 |Write ->|Error|3|"datar|Memory write at "dataraces.c":26   |"datara|"datara|
|  |Write da|     | |aces.c|conflicts with a prior memory      |ces.c":|ces.c":|
|  |ta-race |     | |":20  |write at "dataraces.c":26 (output  |26     |26     |
|  |        |     | |      |dependence)                        |       |       |
```

Thread Checker – typical workflow

1. **Collect on Linux**
   - Either compile with **`-tcheck`** (source instrumentation - better option)
   - Or just run **`tcheck_cl yourprogram`** (binary instrumentation)
2. **Analyze**
   1. Either directly use the text output produced by the program
   2. Or copy files over to Windows and analyze in the GUI (more information available, “clickable” source)

Upcoming tool from Intel: Inspector

> **Next-generation correctness checking tool from Intel**
  - Threading support: pthread, OpenMP, TBB
  - Detects race conditions, memory conflicts, locking conflicts and other issues
  - Robust memory correctness checker

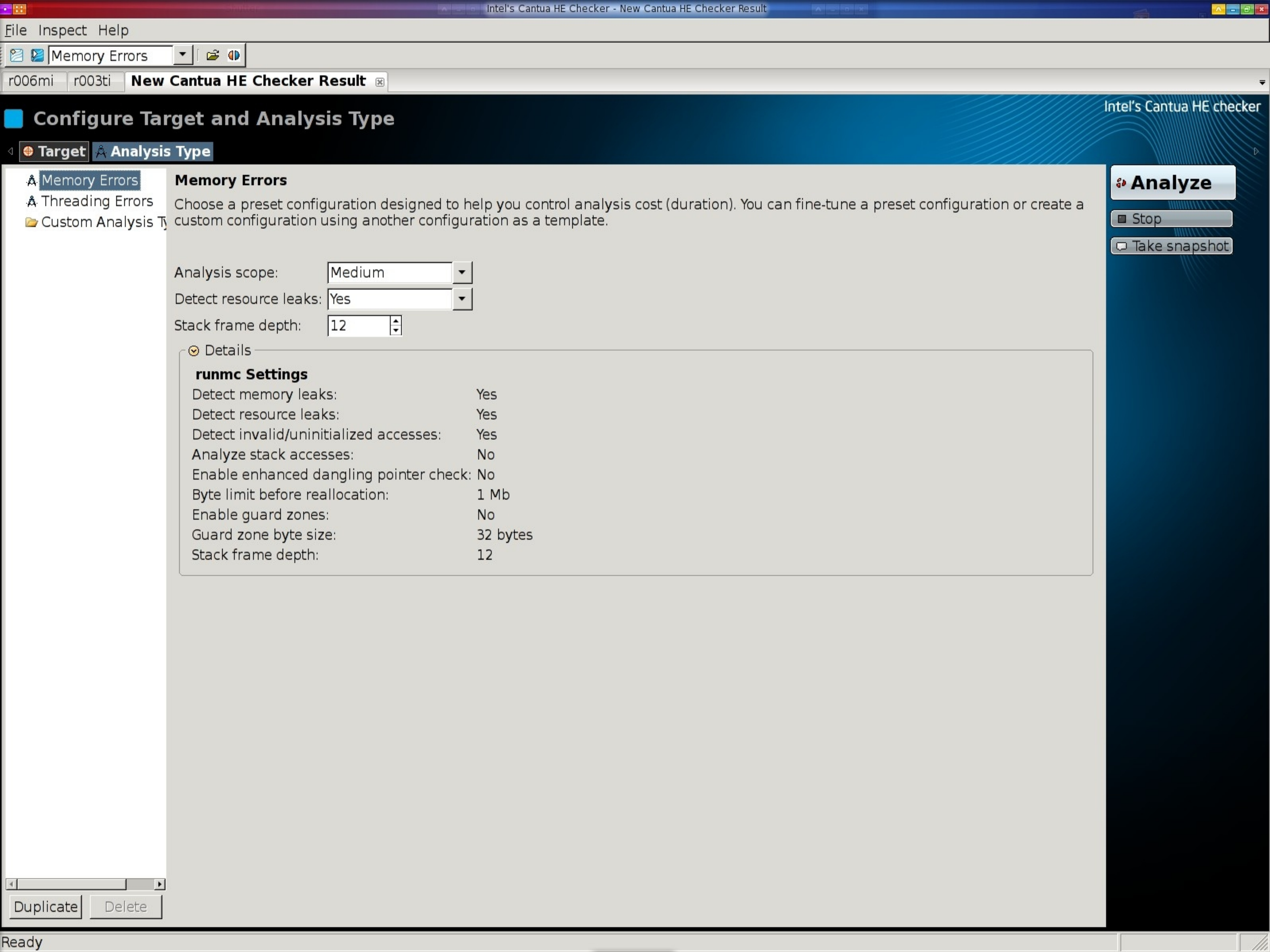
> **Several levels of analysis depth available**
  - Many PIN-based engines (http://pintool.org)

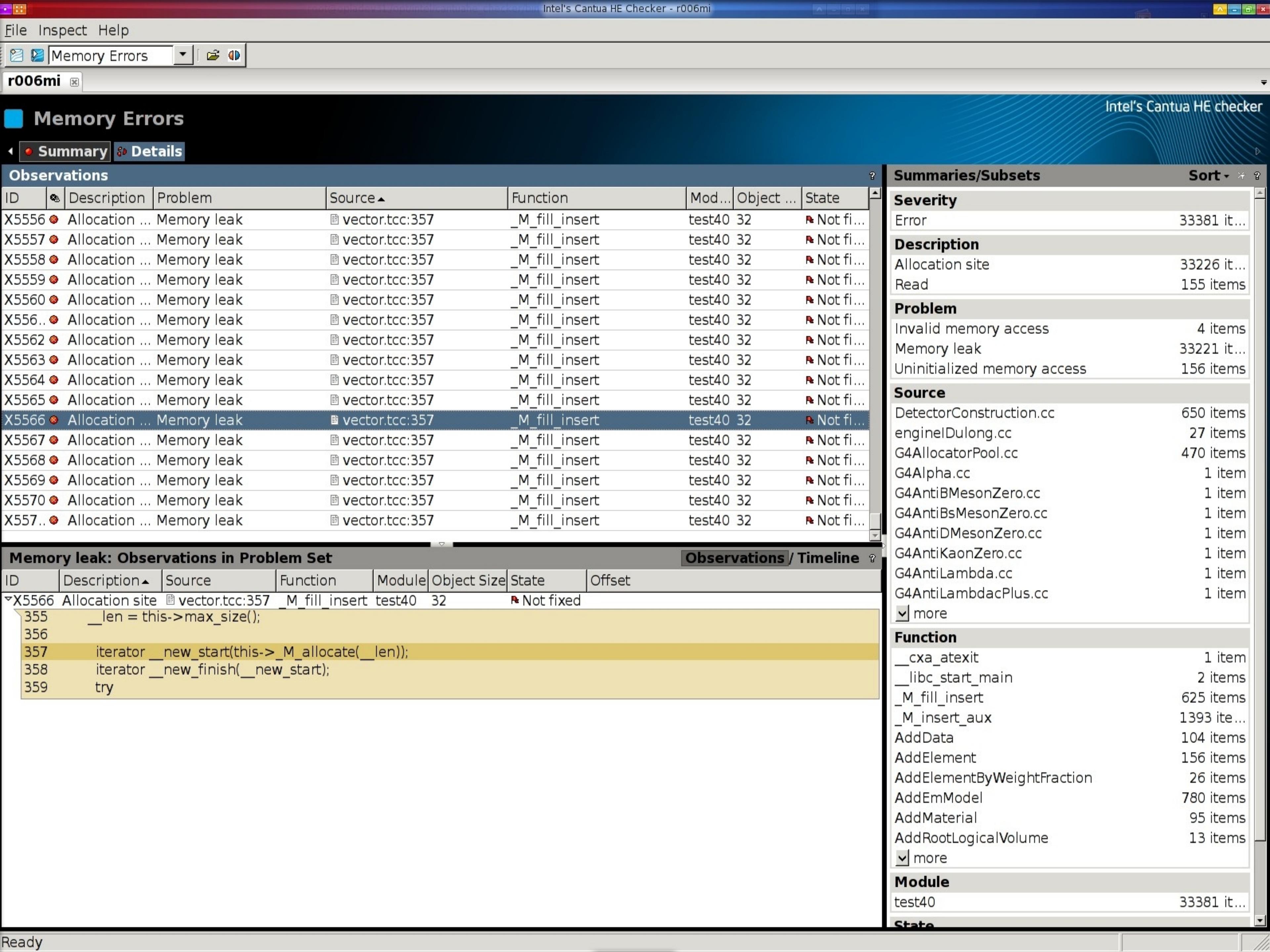
> **Linux native**

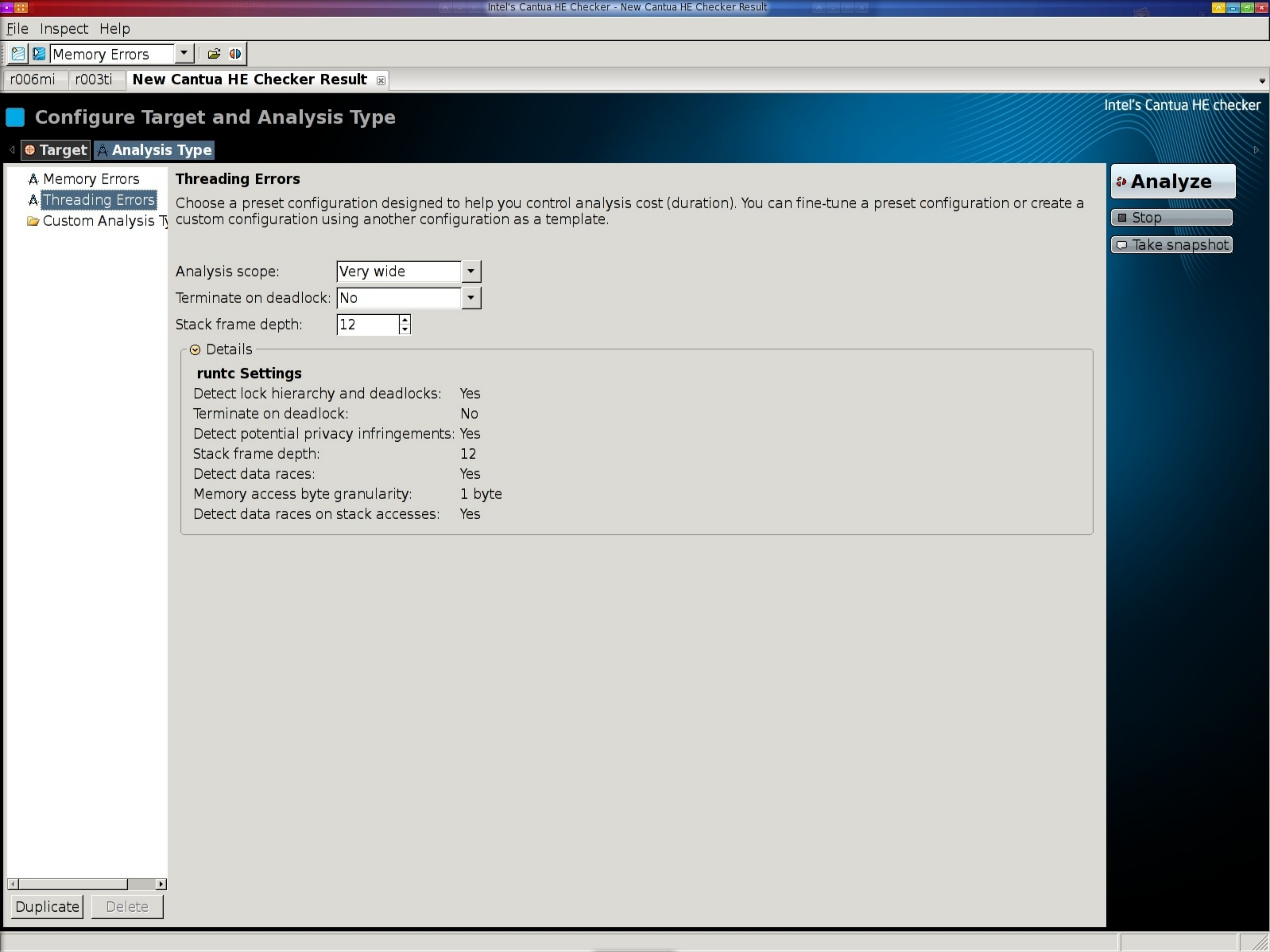
> **No kernel driver needed, no privileges needed**

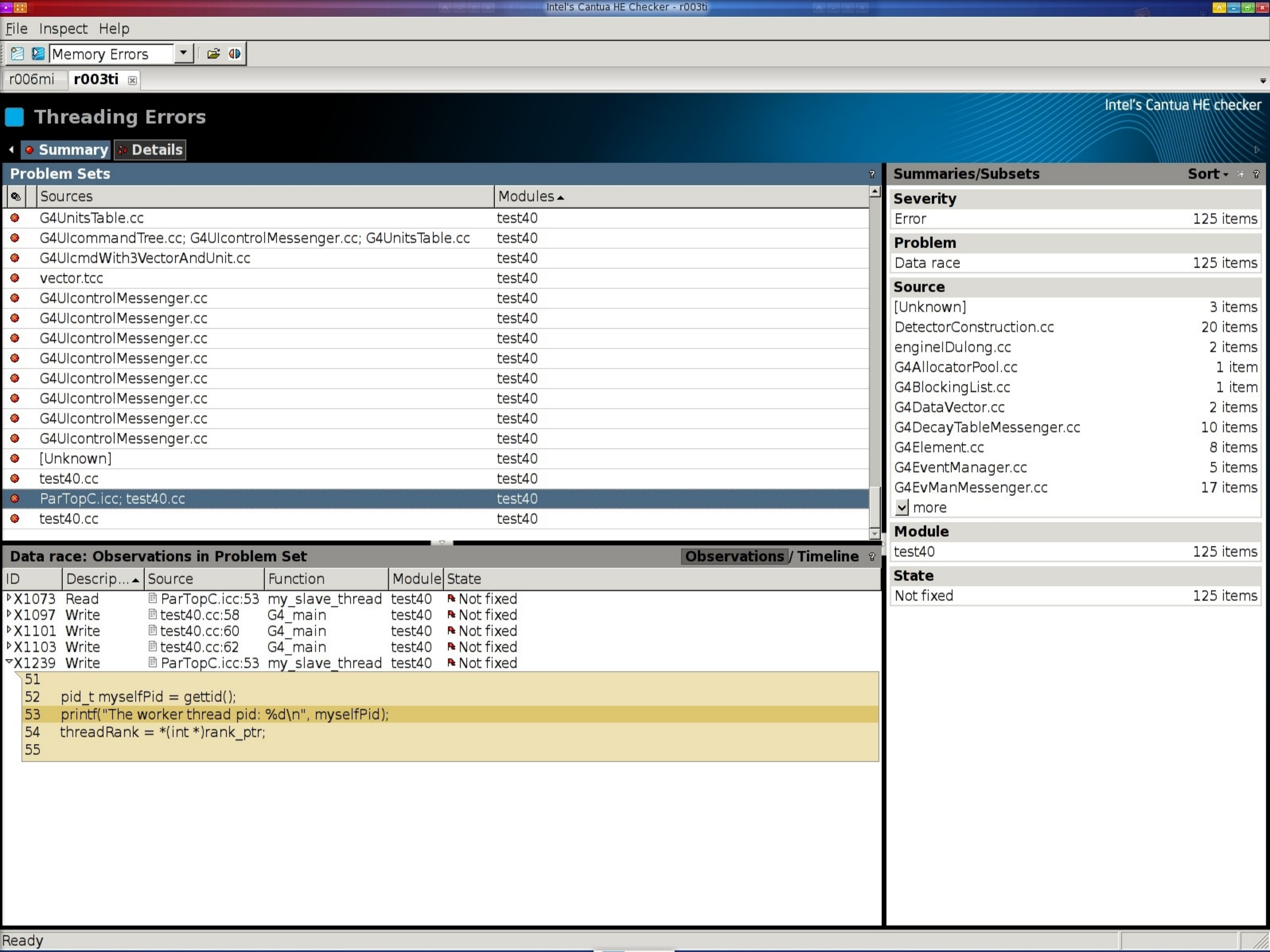
> **Successfully tested at openlab, feedback provided**
  - ROOT
  - Geant4

> **Public beta in H2 2010**


Intel's Cantua HE Checker - New Cantua HE Checker Result
File Inspect Help
Memory Errors
r006mi r003ti New Cantua HE Checker Result
Configure Target and Analysis Type
Intel's Cantua HE checker
Target Analysis Type
Memory Errors
Threading Errors
Custom Analysis Ty
Memory Errors
Choose a preset configuration designed to help you control analysis cost (duration). You can fine-tune a preset configuration or create a custom configuration using another configuration as a template.
Analysis scope: Medium
Detect resource leaks: Yes
Stack frame depth: 12
Details
runmc Settings
Detect memory leaks: Yes
Detect resource leaks: Yes
Detect invalid/uninitialized accesses: Yes
Analyze stack accesses: No
Enable enhanced dangling pointer check: No
Byte limit before reallocation: 1 Mb
Enable guard zones: No
Guard zone byte size: 32 bytes
Stack frame depth: 12
Analyze
Stop
Take snapshot
Duplicate
Delete
Ready


Intel's Cantua HE Checker - r006mi
File Inspect Help
Memory Errors
r006mi
Memory Errors
Intel's Cantua HE checker
Summary
Details
Observations
ID | Description | Problem | Source | Function | Mod... | Object ... | State
X5556 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5557 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5558 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5559 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5560 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X556.. Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5562 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5563 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5564 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5565 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5566 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5567 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5568 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5569 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X5570 Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
X557.. Allocation ... Memory leak vector.tcc:357 _M_fill_insert test40 32 Not fi...
Memory leak: Observations in Problem Set
Observations / Timeline
ID | Description | Source | Function | Module | Object Size | State | Offset
X5566 Allocation site vector.tcc:357 _M_fill_insert test40 32 Not fixed
355 __len = this->max_size();
356
357 iterator __new_start(this->_M_allocate(__len));
358 iterator __new_finish(__new_start);
359 try
Summaries/Subsets
Sort
Severity
Error 33381 it...
Description
Allocation site 33226 it...
Read 155 items
Problem
Invalid memory access 4 items
Memory leak 33221 it...
Uninitialized memory access 156 items
Source
DetectorConstruction.cc 650 items
engineIDulong.cc 27 items
G4AllocatorPool.cc 470 items
G4Alpha.cc 1 item
G4AntiBMesonZero.cc 1 item
G4AntiBsMesonZero.cc 1 item
G4AntiDMesonZero.cc 1 item
G4AntiKaonZero.cc 1 item
G4AntiLambda.cc 1 item
G4AntiLambdacPlus.cc 1 item
more
Function
__cxa_atexit 1 item
__libc_start_main 2 items
_M_fill_insert 625 items
_M_insert_aux 1393 ite...
AddData 104 items
AddElement 156 items
AddElementByWeightFraction 26 items
AddEmModel 780 items
AddMaterial 95 items
AddRootLogicalVolume 13 items
more
Module
test40 33381 it...
State
Ready


Intel's Cantua HE Checker - New Cantua HE Checker Result
File Inspect Help
Memory Errors
r006mi
r003ti
New Cantua HE Checker Result
Configure Target and Analysis Type
Intel's Cantua HE checker
Target
Analysis Type
Memory Errors
Threading Errors
Custom Analysis Ty
Threading Errors
Choose a preset configuration designed to help you control analysis cost (duration). You can fine-tune a preset configuration or create a custom configuration using another configuration as a template.
Analysis scope: Very wide
Terminate on deadlock: No
Stack frame depth: 12
Details
runtc Settings
Detect lock hierarchy and deadlocks: Yes
Terminate on deadlock: No
Detect potential privacy infringements: Yes
Stack frame depth: 12
Detect data races: Yes
Memory access byte granularity: 1 byte
Detect data races on stack accesses: Yes
Analyze
Stop
Take snapshot
Duplicate
Delete



Intel's Cantua HE Checker - r003ti
File Inspect Help
Memory Errors
r006mi
r003ti
Threading Errors
Intel's Cantua HE checker
Summary
Details
Problem Sets
Sources
Modules
G4UnitsTable.cc test40
G4UIcommandTree.cc; G4UIcontrolMessenger.cc; G4UnitsTable.cc test40
G4UIcmdWith3VectorAndUnit.cc test40
vector.tcc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
G4UIcontrolMessenger.cc test40
[Unknown] test40
test40.cc test40
ParTopC.icc; test40.cc test40
test40.cc test40
Data race: Observations in Problem Set
Observations / Timeline
ID Descrip... Source Function Module State
X1073 Read ParTopC.icc:53 my_slave_thread test40 Not fixed
X1097 Write test40.cc:58 G4_main test40 Not fixed
X1101 Write test40.cc:60 G4_main test40 Not fixed
X1103 Write test40.cc:62 G4_main test40 Not fixed
X1239 Write ParTopC.icc:53 my_slave_thread test40 Not fixed
51
52 pid_t myselfPid = gettid();
53 printf("The worker thread pid: %d\n", myselfPid);
54 threadRank = *(int *)rank_ptr;
55
Summaries/Subsets
Sort
Severity
Error 125 items
Problem
Data race 125 items
Source
[Unknown] 3 items
DetectorConstruction.cc 20 items
engineIDulong.cc 2 items
G4AllocatorPool.cc 1 item
G4BlockingList.cc 1 item
G4DataVector.cc 2 items
G4DecayTableMessenger.cc 10 items
G4Element.cc 8 items
G4EventManager.cc 5 items
G4EvManMessenger.cc 17 items
more
Module
test40 125 items
State
Not fixed 125 items
Ready

Food for thought– loose propositions for possible future activities

> **Expand scope of automated performance collection**
  - Every CPU has a performance monitoring unit that for the better part is unused
  - Adopting a wide, common strategy for performance feedback
  - Nightly build runs for instant performance feedback and regression monitoring
  - Annotated source a la CMS (Peter Elmer)
  - Background monitoring of batch servers for live and historical data

> **Extending the availability of readily available performance monitoring C++ classes (Daniele Kruse)**

> **Increasing the common availability of low level performance monitoring**
  - Simplification of performance monitoring processes and outputs – a long and hard task

> **Most of the discussed scenarios could be deployed via our management system (possibly with Intel's help – if needed)**
  - Some ideas already are on the table at openlab

Q & A

Questions?

Andrzej.Nowak@cern.ch