

CERN openlab II

Multi-threading and multi-core situation overview

Andrzej Nowak
[andrzej.nowak at cern.ch](mailto:andrzej.nowak@cern.ch)



Modern supercomputing limitations

- > **The constant need for faster and more capable systems**
- > **Today's options:**
 - Frequency scaling techniques of CPUs are nearly exhausted
 - Increasing core frequency does not yield linear performance improvements
 - High power consumption
 - High heat dissipation
 - Parallel architectures introduced; additional “cores” available at a low cost

> The move to heavily multi-core architectures is imminent

■ Advantages:

- Less power used
- Less heat dissipated
- More processing power in a single package
- Fast communication between cores: nanoseconds with multi-core, 100's of nanoseconds with SMP

> Timeline:

- “Today”: 2, 4, 8 cores
- Heavily multi-core designs are already used in graphics and network processing
- 16 or 32 cores in general purpose CPUs in the near future
- As much as 80 cores might be available in the further future

> How do we prepare for this revolution?

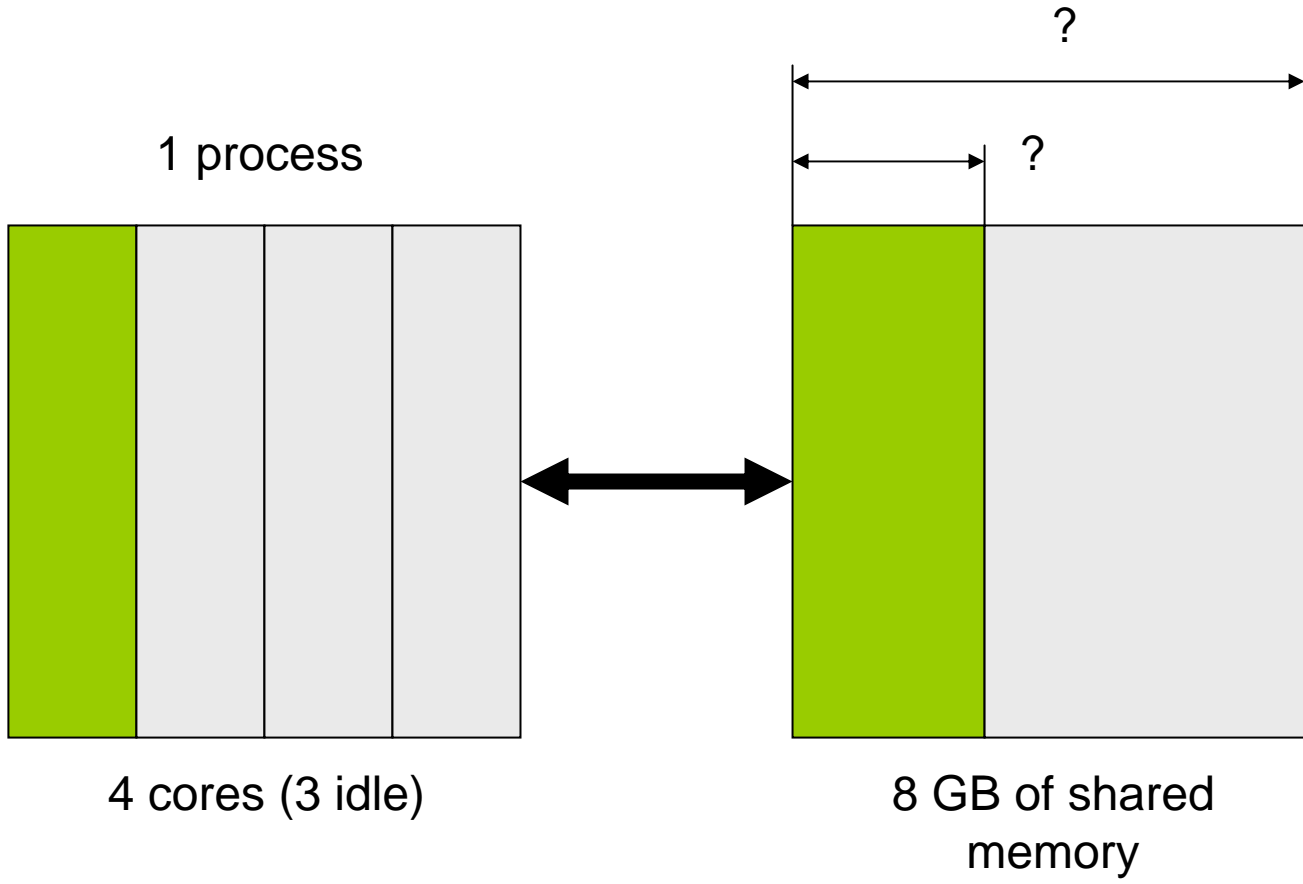
- > **Performance measurements, benchmarking**
 - Software development (pfmon, gpfmon, other tools)
 - Benchmarking our own systems...
 - ...as well as future systems – in cooperation with Intel; first results very interesting
 - Actively looking for areas in CERN physics software which could be improved
- > **Multi-core architecture and scalability study**
 - Close cooperation with Intel
- > **Multi-thread programming methodologies study**
 - openlab/Intel multi-threading workshop
- > **Studying alternative approaches to CPU performance issues**
 - General purpose computing on GPUs

> **Exploiting multi-core architectures is a necessity. What are the issues?**

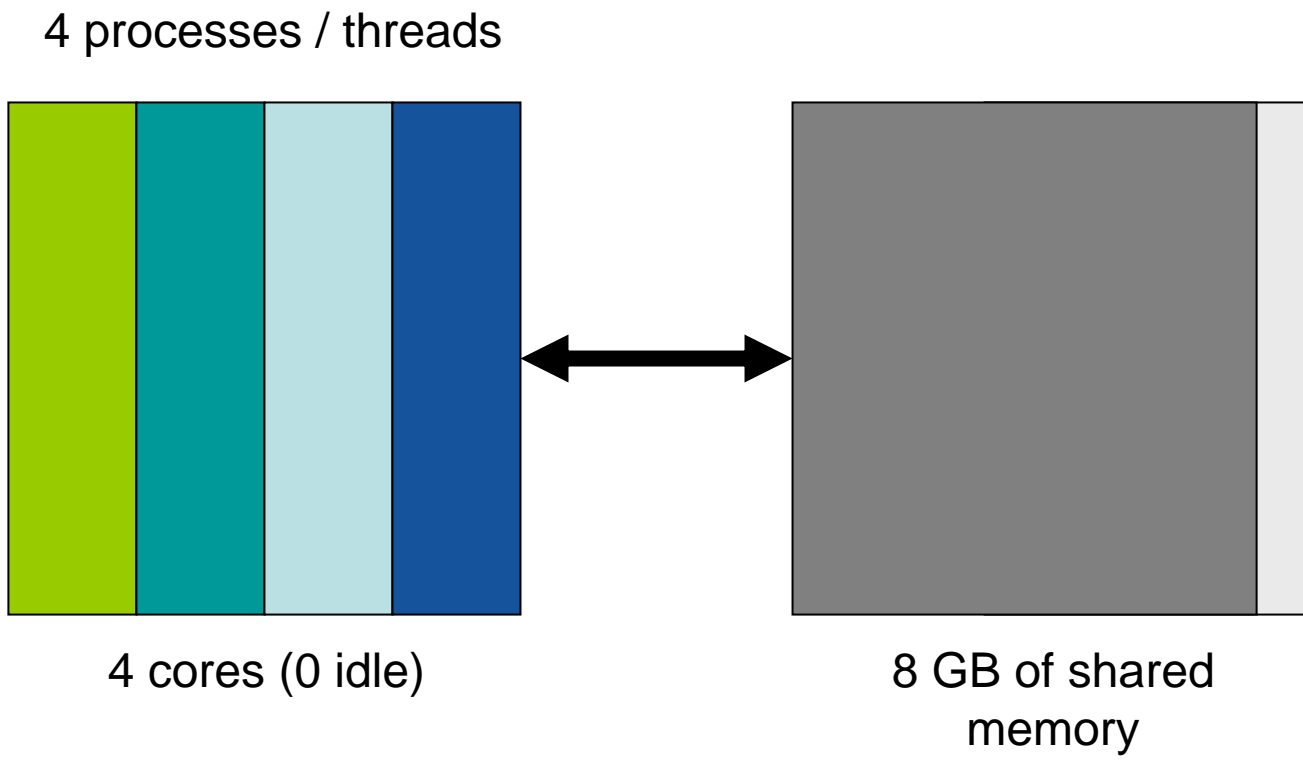
- Can the problem be solved via parallel computing?
What is the best approach?
- The implications of running multiple demanding threads in a single system: some resources might become choking points
 - Memory bandwidth/size
 - System bus
 - Inter-CPU communication
 - Network
 - Hard drive performance
 - Hard drive space

- > **Many applications at CERN have the following characteristics:**
 - CPU-intensive
 - Relatively low amount of RAM transactions
 - Embarrassingly parallel (data parallelism)
 - The executable has a small footprint (often fitting into 1MB of cache)
 - Single-threaded
- > **A lot of “free” processing power is wasted “between the lines”**

Problem illustration example



Problem illustration example



> System benchmark, based on Geant 4 scientific software

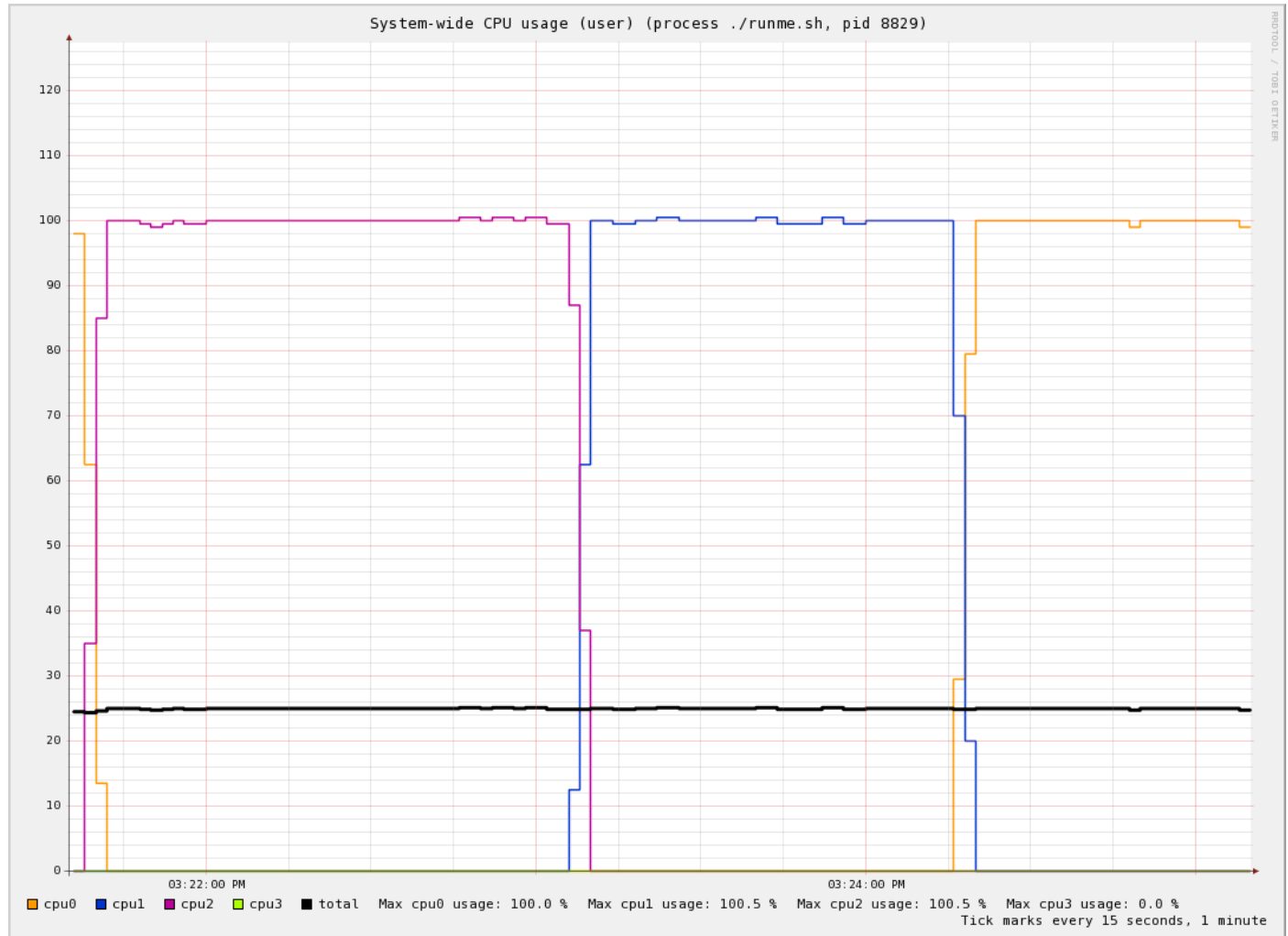
- Simulates particles passing through matter
- Real detector geometry from a LHC experiment (CMS)
- Real physics processes
- Loads similar to those expected during LHC operation

> Monitoring using own tool + pfmon

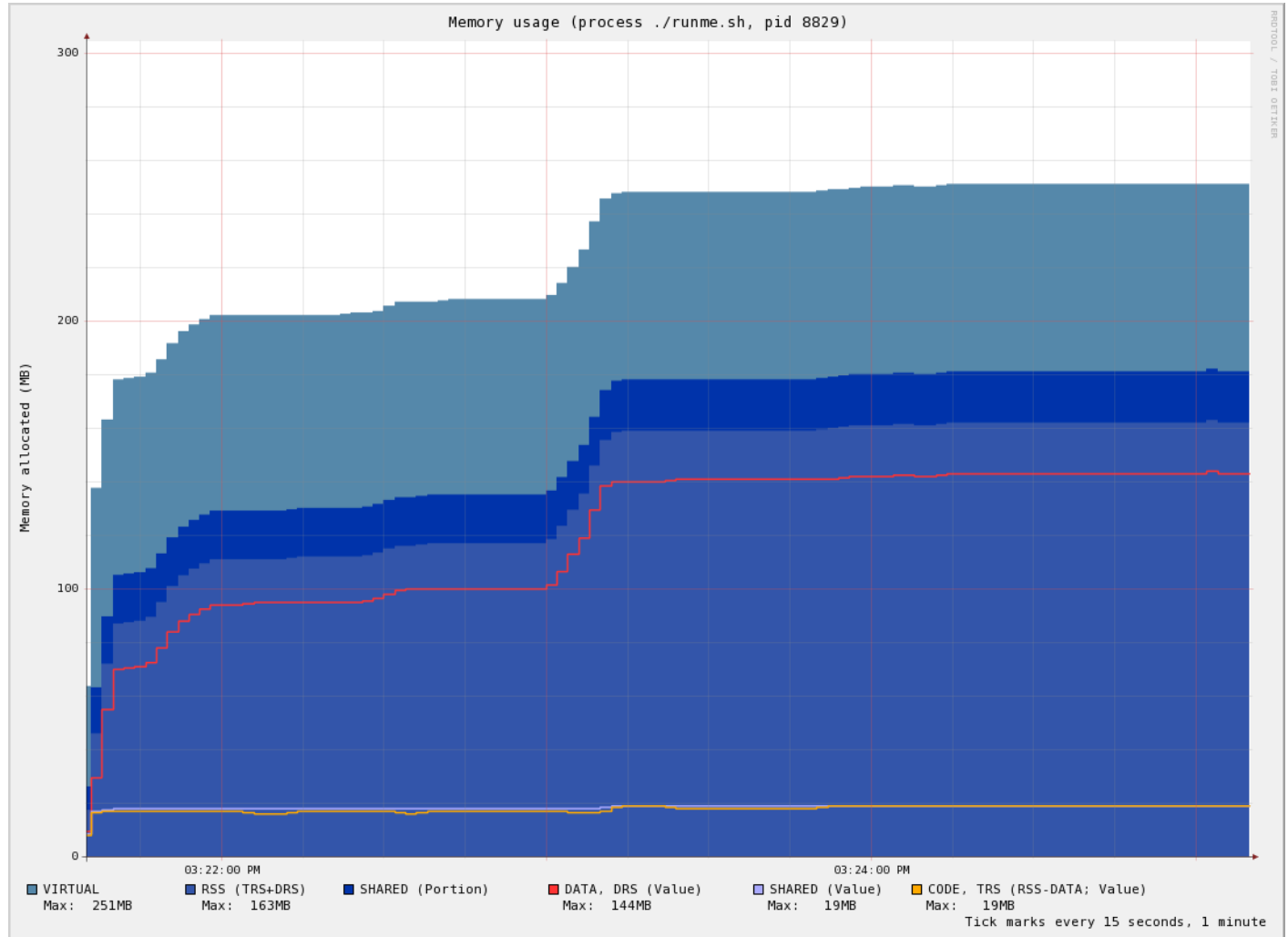
Processing time for 100 events (real time)

| | | | | |
|--------------------|-------------|-------------|-------------|-------------|
| 1 process | 118s | - | - | - |
| 4 processes | 120s | 121s | 121s | 121s |

Single benchmark process – CPU usage



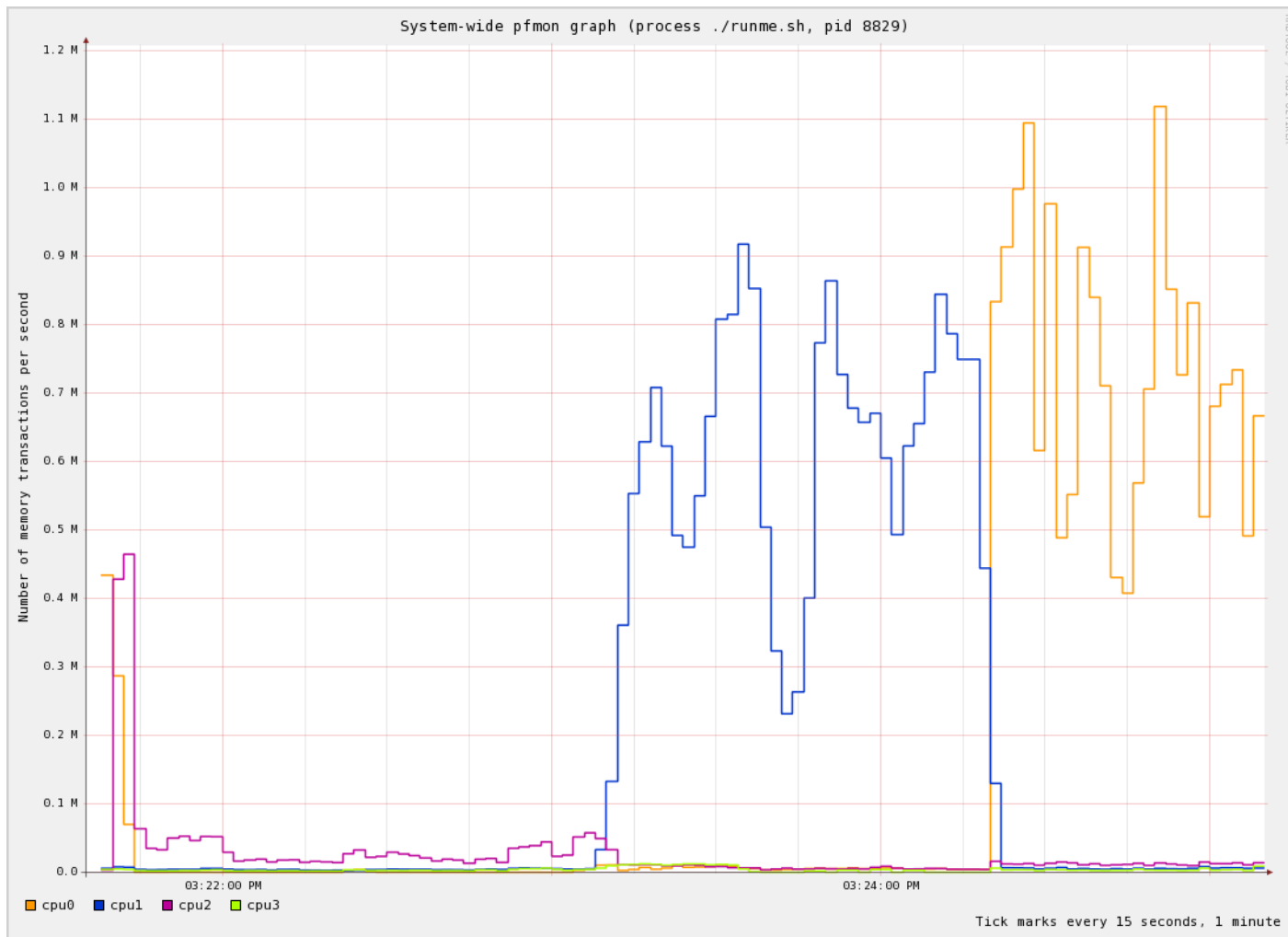
Single benchmark process – memory usage





Single benchmark process – memory transactions

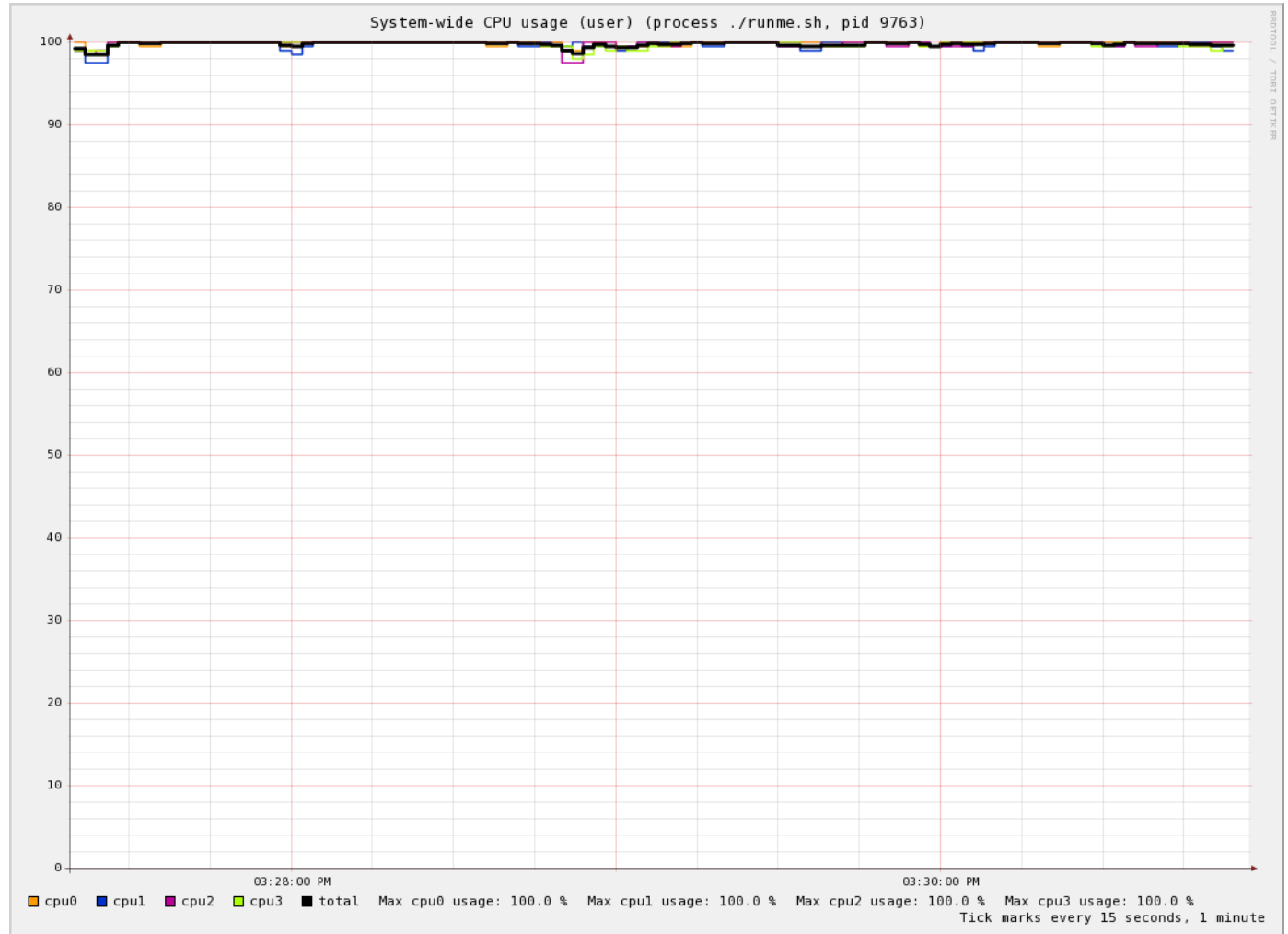
CERN
openlab



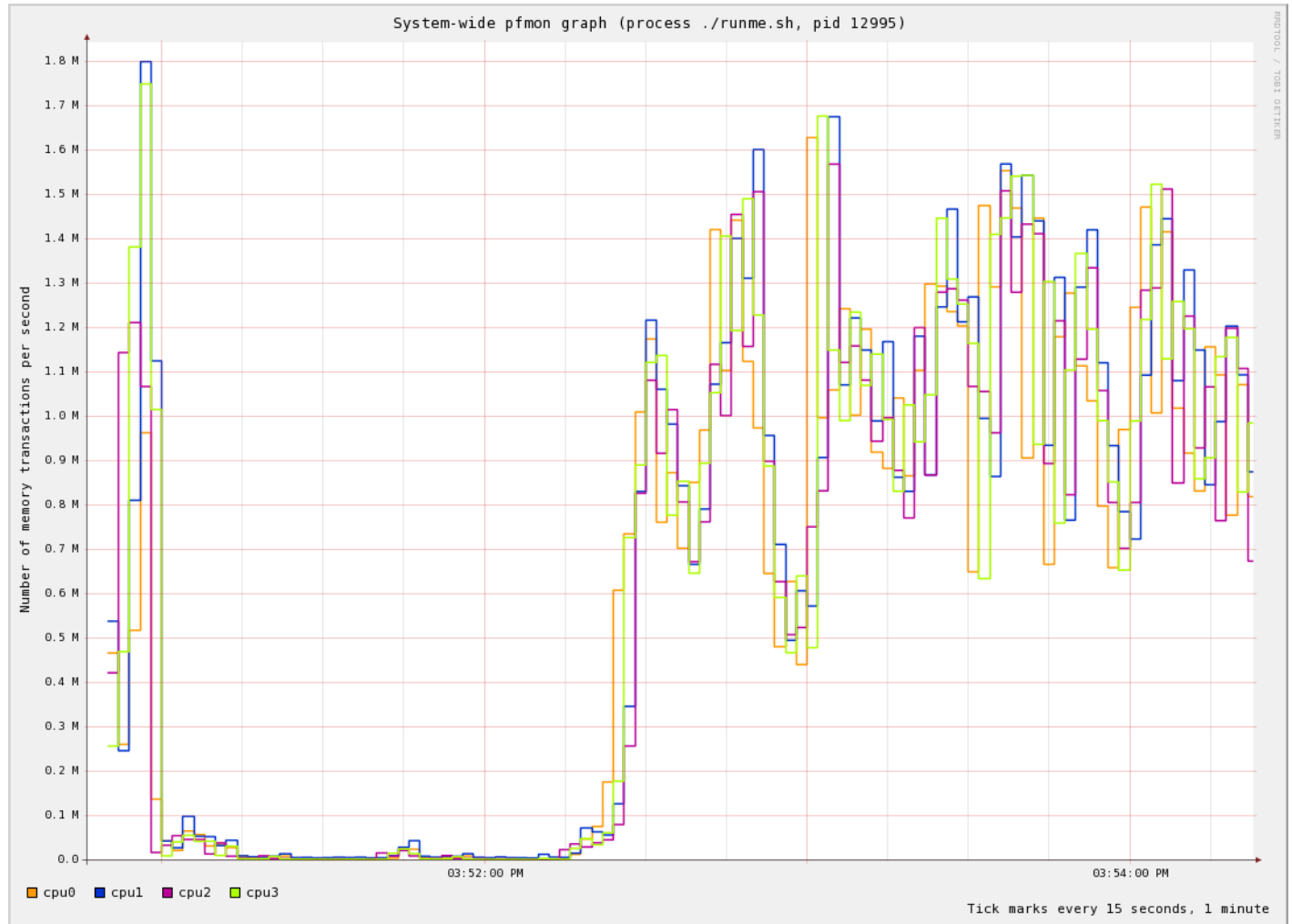


**CERN
openlab**

Multiple processes – CPU usage



Multiple processes – memory transactions



> Graphical pfmon frontend being developed

- With users from CERN in mind

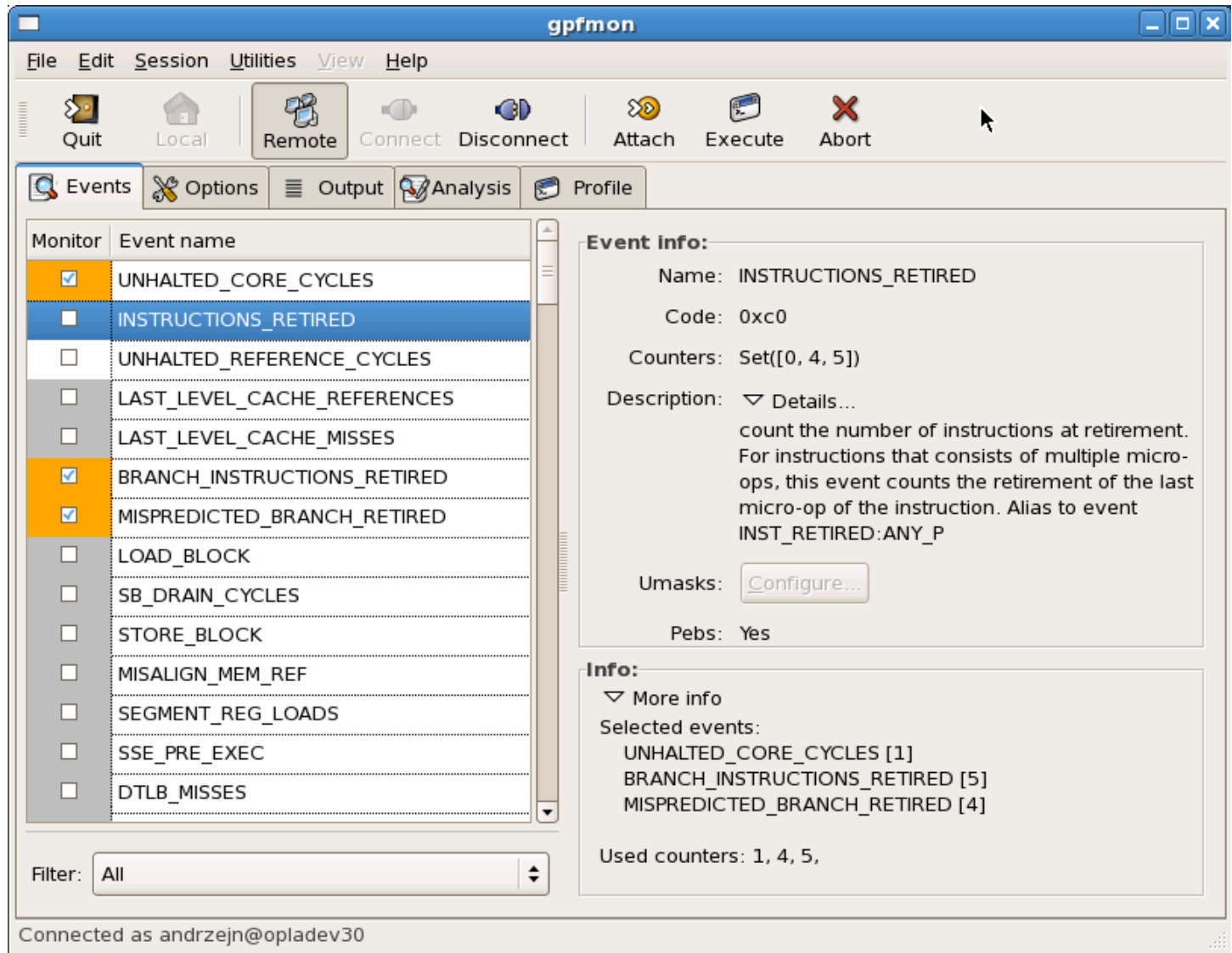
> Features

- Simple, user-friendly GUI
- Many pfmon options supported
- Supports all architectures supported by pfmon
- Remote monitoring sessions

> Currently entering the alpha stage

> Plans for the near future

- Profiling and sampling support
- Graphical and tabular representation of data



The screenshot shows the gpfmon application window with the following components:

- Menu Bar:** File, Edit, Session, Utilities, View, Help
- Toolbar:** Quit, Local, Remote, Connect, Disconnect, Attach, Execute, Abort
- Tabbed Interface:** Events (selected), Options, Output, Analysis, Profile
- Monitor Panel:**

| Monitor | Event name |
|-------------------------------------|-----------------------------|
| <input checked="" type="checkbox"/> | UNHALTED_CORE_CYCLES |
| <input type="checkbox"/> | INSTRUCTIONS_RETIRED |
| <input type="checkbox"/> | UNHALTED_REFERENCE_CYCLES |
| <input type="checkbox"/> | LAST_LEVEL_CACHE_REFERENCES |
| <input type="checkbox"/> | LAST_LEVEL_CACHE_MISSES |
| <input checked="" type="checkbox"/> | BRANCH_INSTRUCTIONS_RETIRED |
| <input checked="" type="checkbox"/> | MISPREDICTED_BRANCH_RETIRED |
| <input type="checkbox"/> | LOAD_BLOCK |
| <input type="checkbox"/> | SB_DRAIN_CYCLES |
| <input type="checkbox"/> | STORE_BLOCK |
| <input type="checkbox"/> | MISALIGN_MEM_REF |
| <input type="checkbox"/> | SEGMENT_REG_LOADS |
| <input type="checkbox"/> | SSE_PRE_EXEC |
| <input type="checkbox"/> | DTLB_MISSES |
- Event info Panel:**

Event info:

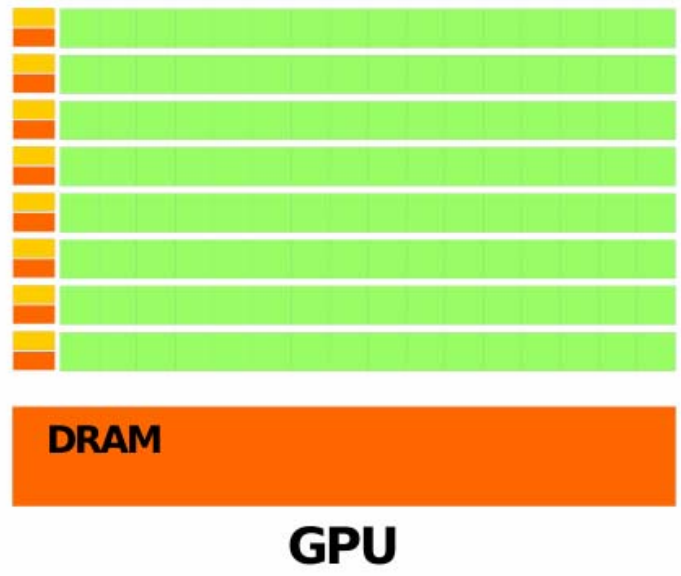
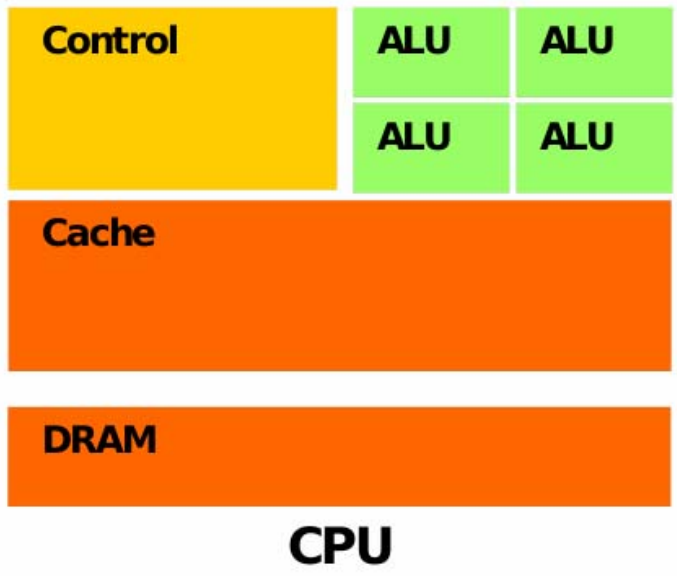
 - Name: INSTRUCTIONS_RETIRED
 - Code: 0xc0
 - Counters: Set([0, 4, 5])
 - Description: Details...
count the number of instructions at retirement. For instructions that consists of multiple micro-ops, this event counts the retirement of the last micro-op of the instruction. Alias to event INST_RETIRED:ANY_P
 - Umask:
 - Pebs: Yes
- Info Panel:**

Info:

 - More info
 - Selected events:
 - UNHALTED_CORE_CYCLES [1]
 - BRANCH_INSTRUCTIONS_RETIRED [5]
 - MISPREDICTED_BRANCH_RETIRED [4]
 - Used counters: 1, 4, 5,
- Filter:** All
- Status Bar:** Connected as andrzejn@opladev30

- > **Brief study of GPGPU performed in Q1**
- > **Main focus on the NVidia G80 chip**
 - Up to 330 GFlops
 - 128 stream processors (16x batch of 8)
 - Not that expensive
- > **Pros:**
 - GPGPU programming encouraged by GPU vendors (NVidia, ATI)
 - Several immediate scientific applications available
 - Huge processing power

GPGPU – Architecture comparison



> Cons:

- Non-IEEE FP representation
- FP precision a very big problem (128bit is in fact 4x32 bit)
- Low precision math ops
- Great hunger for data (intermediate storage problems, PCIe width, RAM bus utilization)
- Large power consumption (140W for the G80)
- Hardware – 1U servers, blades, GPUs don't fit
- Internal GPU processing unit memory might be too small for our applications

> **We will be monitoring what this market has to offer**

- > No magic bullet**
 - No easy way to “parallelize” existing software, although efforts are being made
- > Numerous tools for programmers simplify common parallelism concepts**
 - OpenMP
 - MPI/PVM
- > The solution for now: multiple independent processes and threads per physical processor**
- > Programmer awareness and education is key to good results with multi-core systems**

Q&A

This research project has been supported by a Marie Curie Early Stage Research Training Fellowship of the European Community's Sixth Framework Programme under contract number (MEST-CT-2004-504054)