CERN openlab II

Multi-threading and multi-core optimizations

Andrzej Nowak
andrzej.nowak at 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 imminent move to multi-core

- 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?

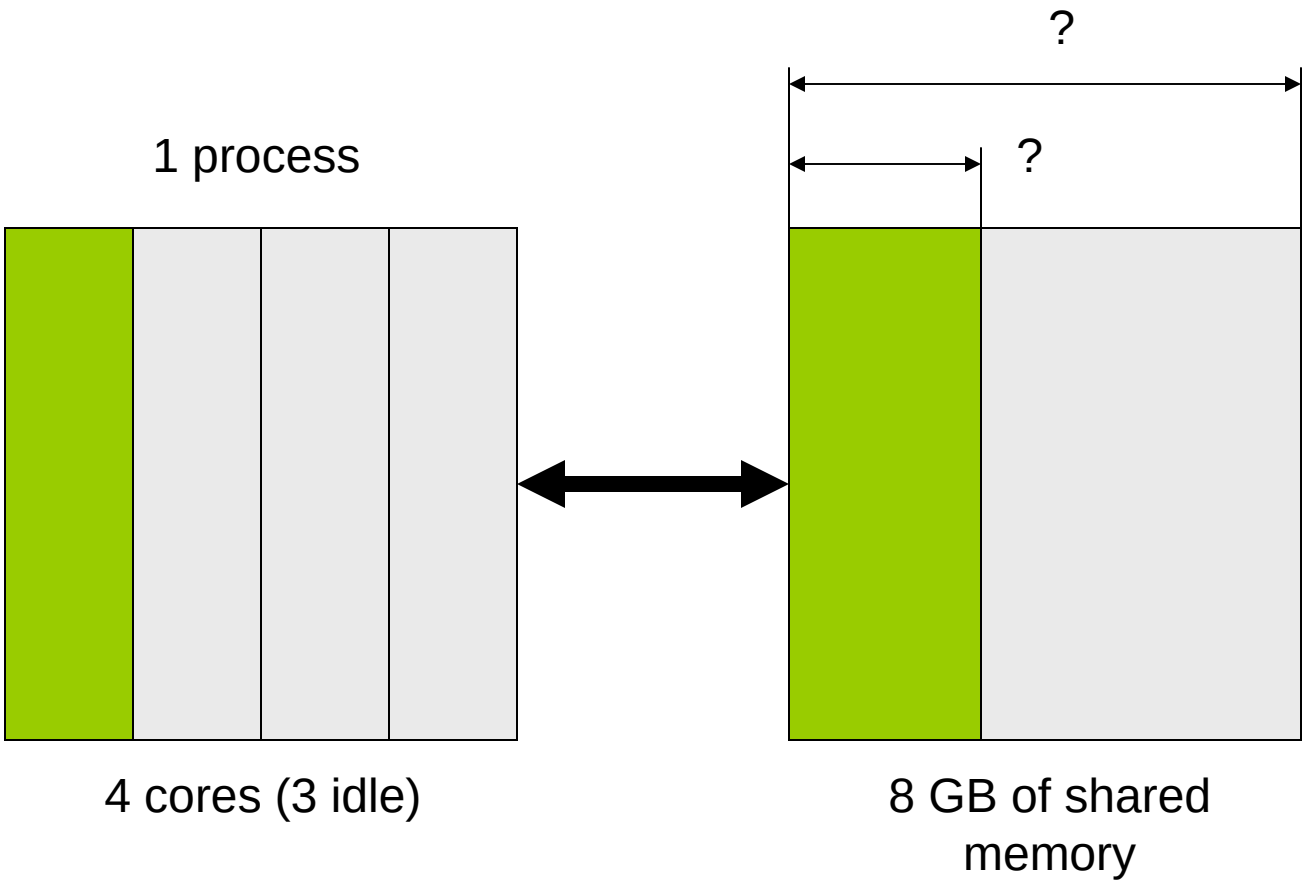
Multi-threading issues

- 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

CERN applications

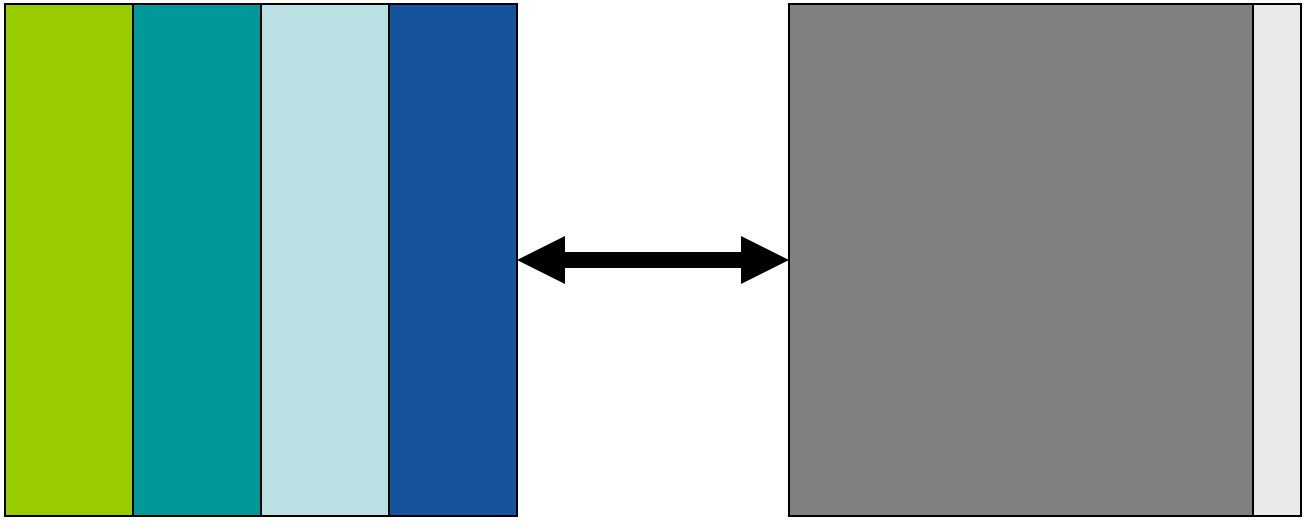
- Many applications at CERN have the following characteristics:
  - CPU-intensive
  - Relatively low amount of RAM transactions
  - Embarrassingly parallel (data parallelism)
  - The executable has a small footprint (often fitting into 1MB of cache)
  - Single-threaded
- A lot of “free” processing power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

Problem illustration example



Problem illustration example

4 processes / threads



4 cores (0 idle)

8 GB of shared memory

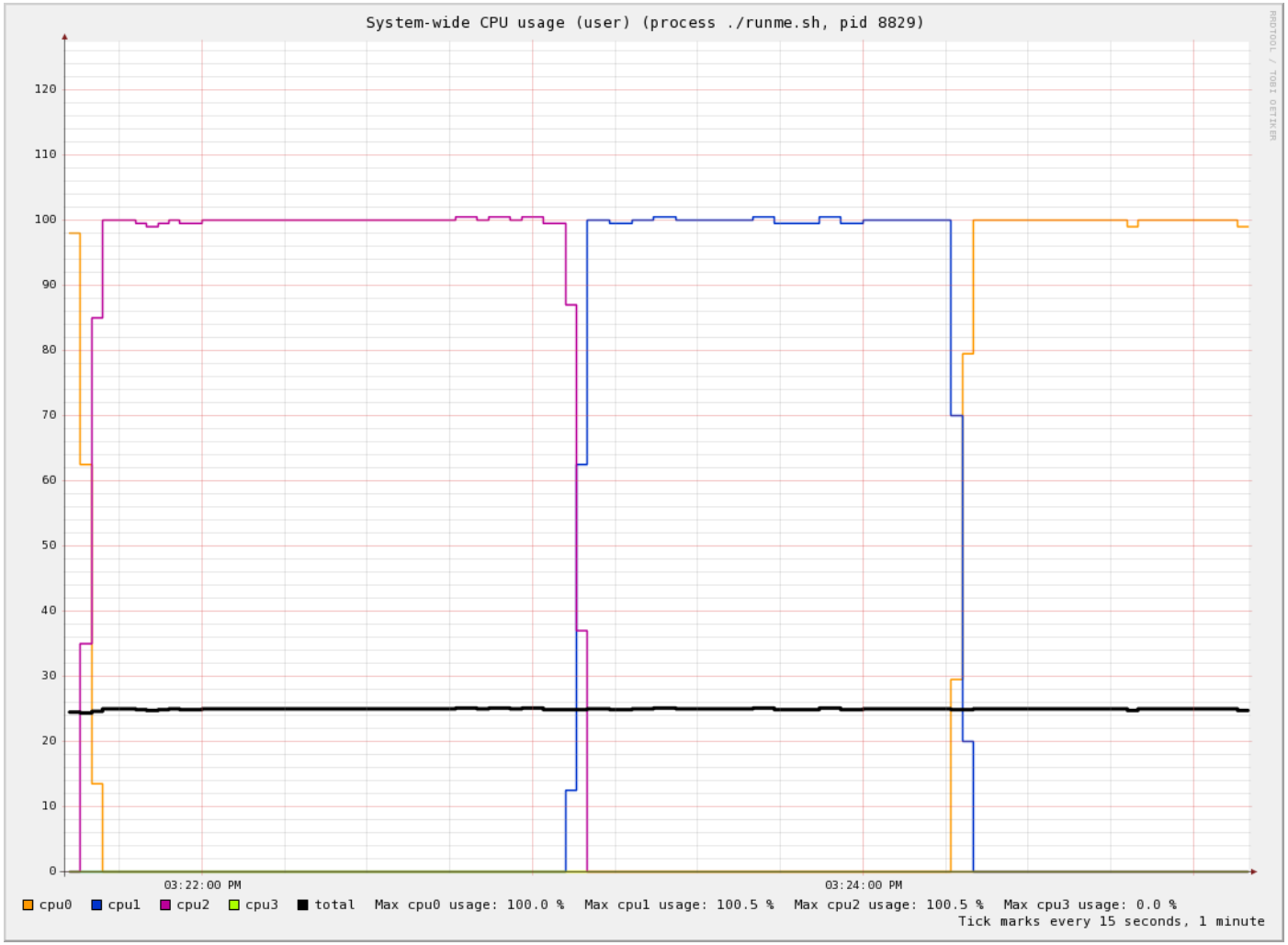
Performance monitoring

- System benchmark, based on Geant 4 scientific software
  - Simulates particles passing through matter
  - Real detector geometry from a LHC experiment
  - Real physics processes
  - Loads similar to those expected during LHC operation
- Monitoring using own tool + pfmon

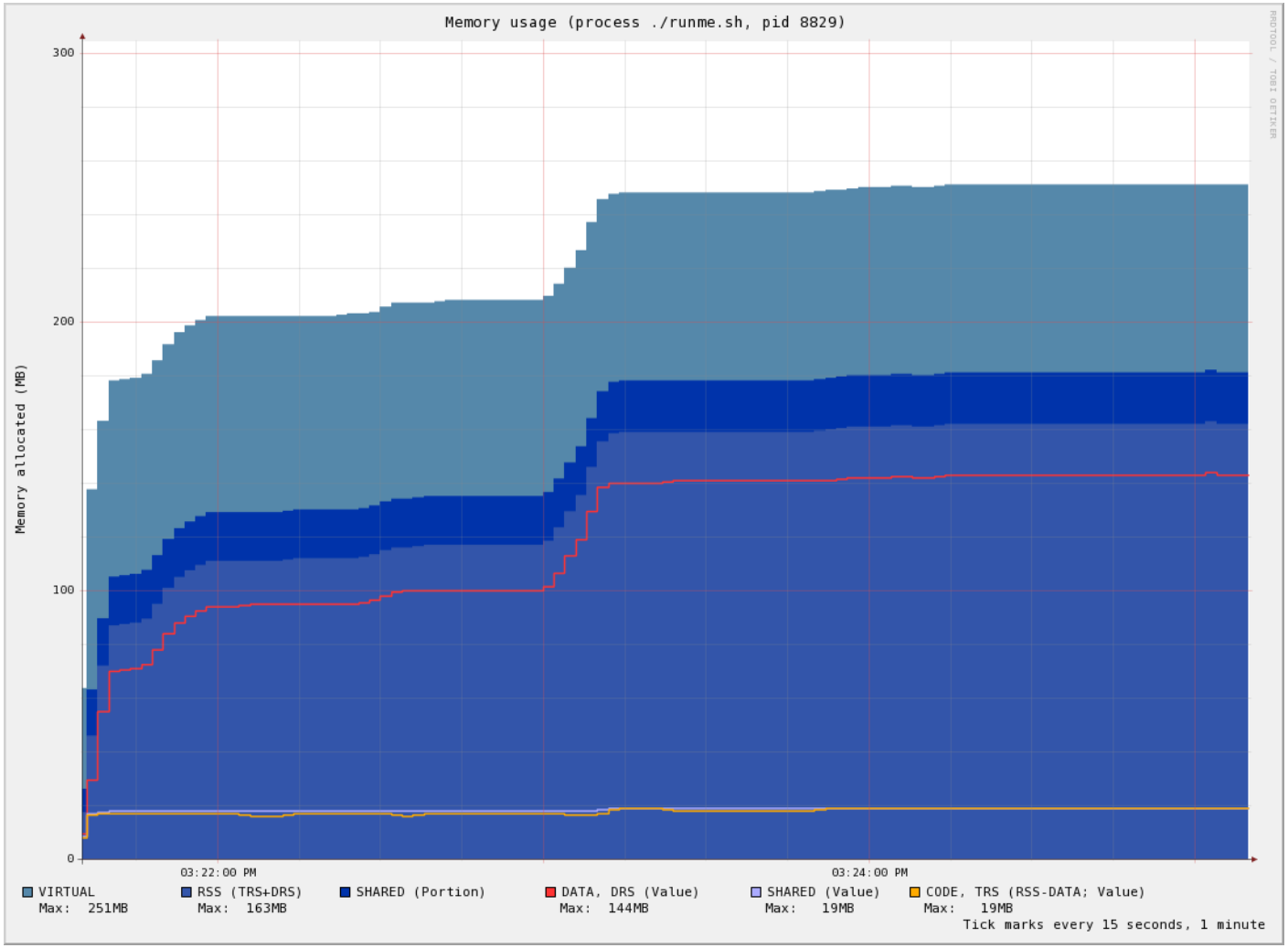
Processing time for 100 events (real time)

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

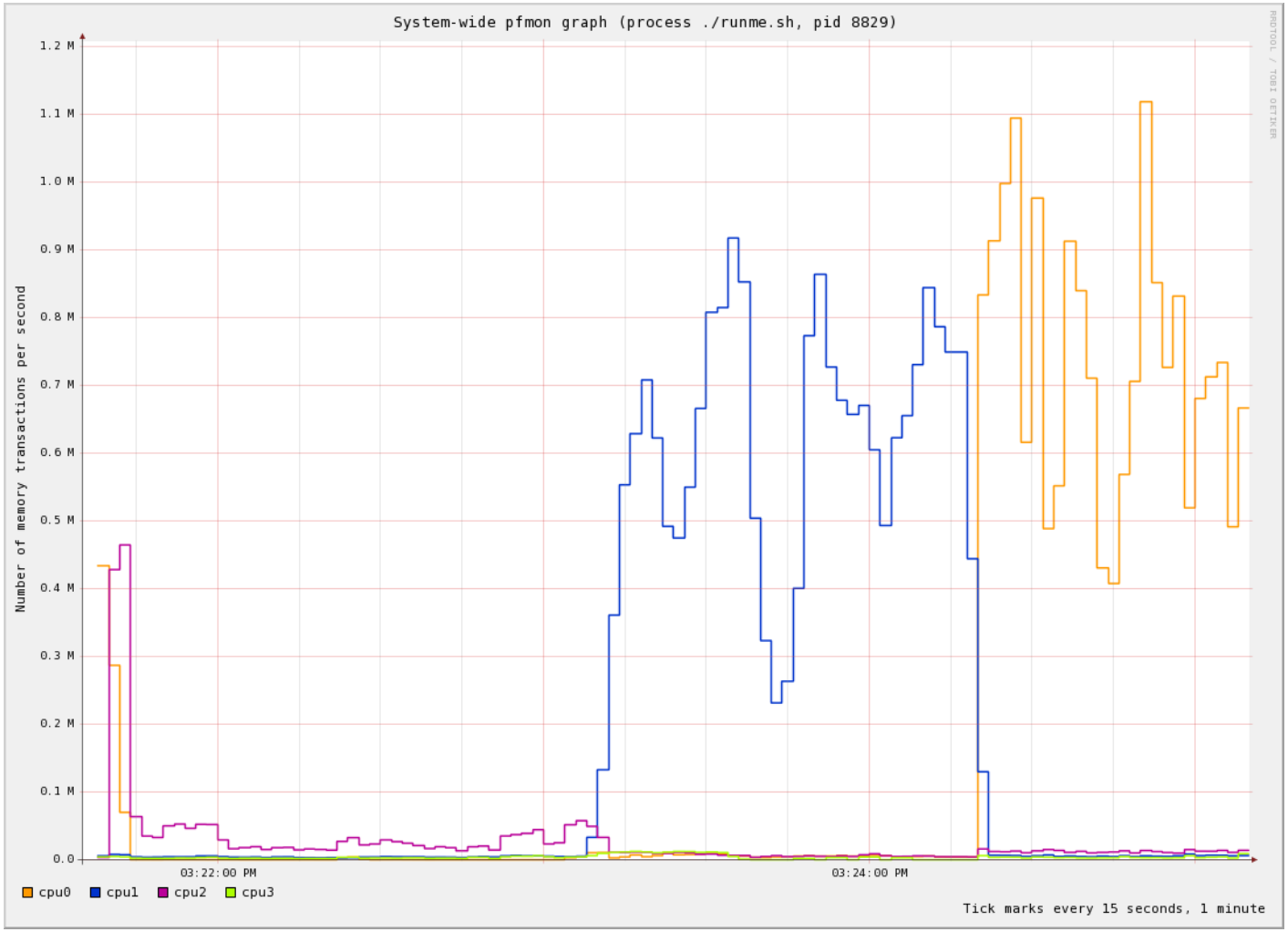
Single process – CPU usage



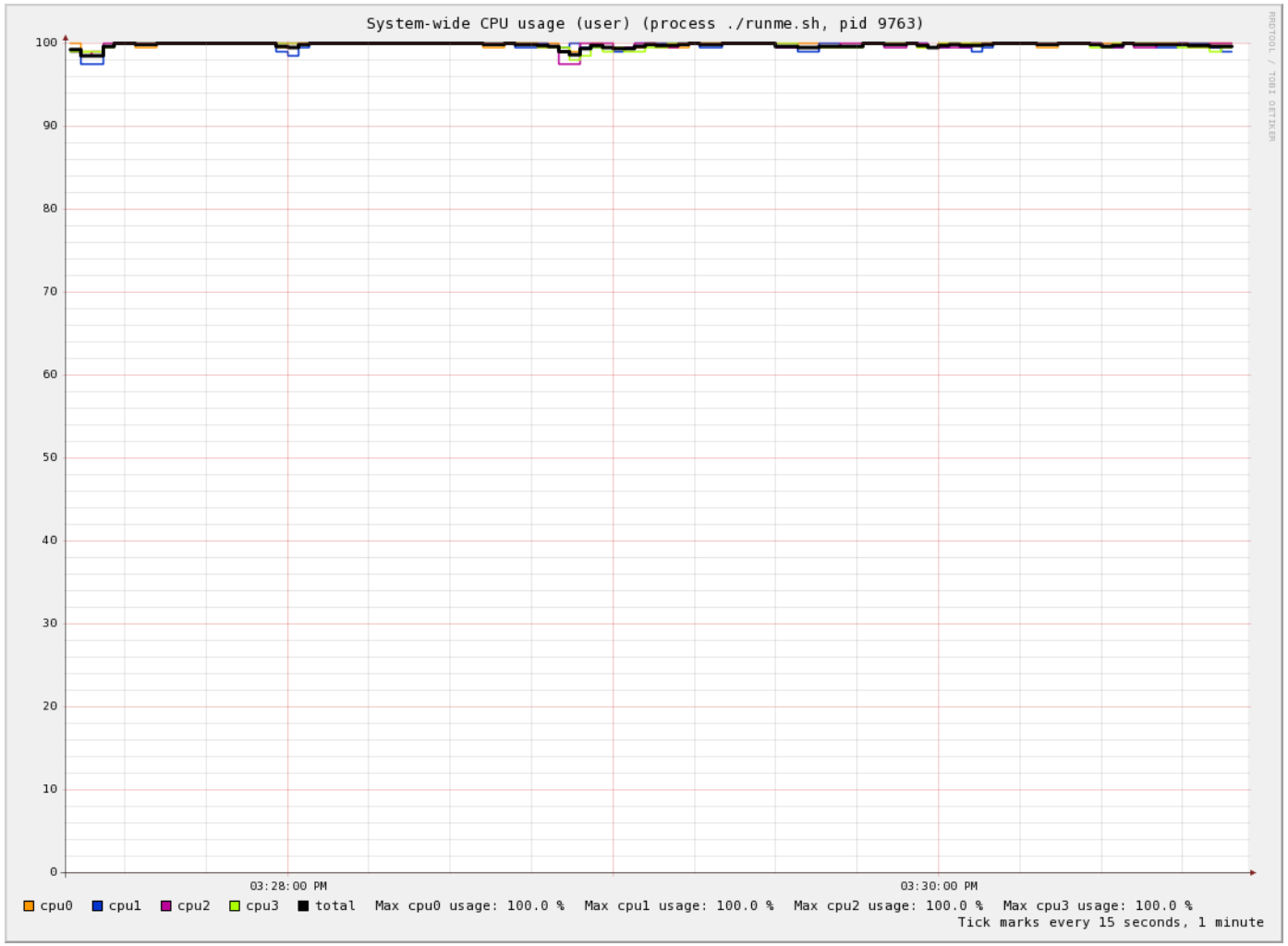
Single process – memory usage



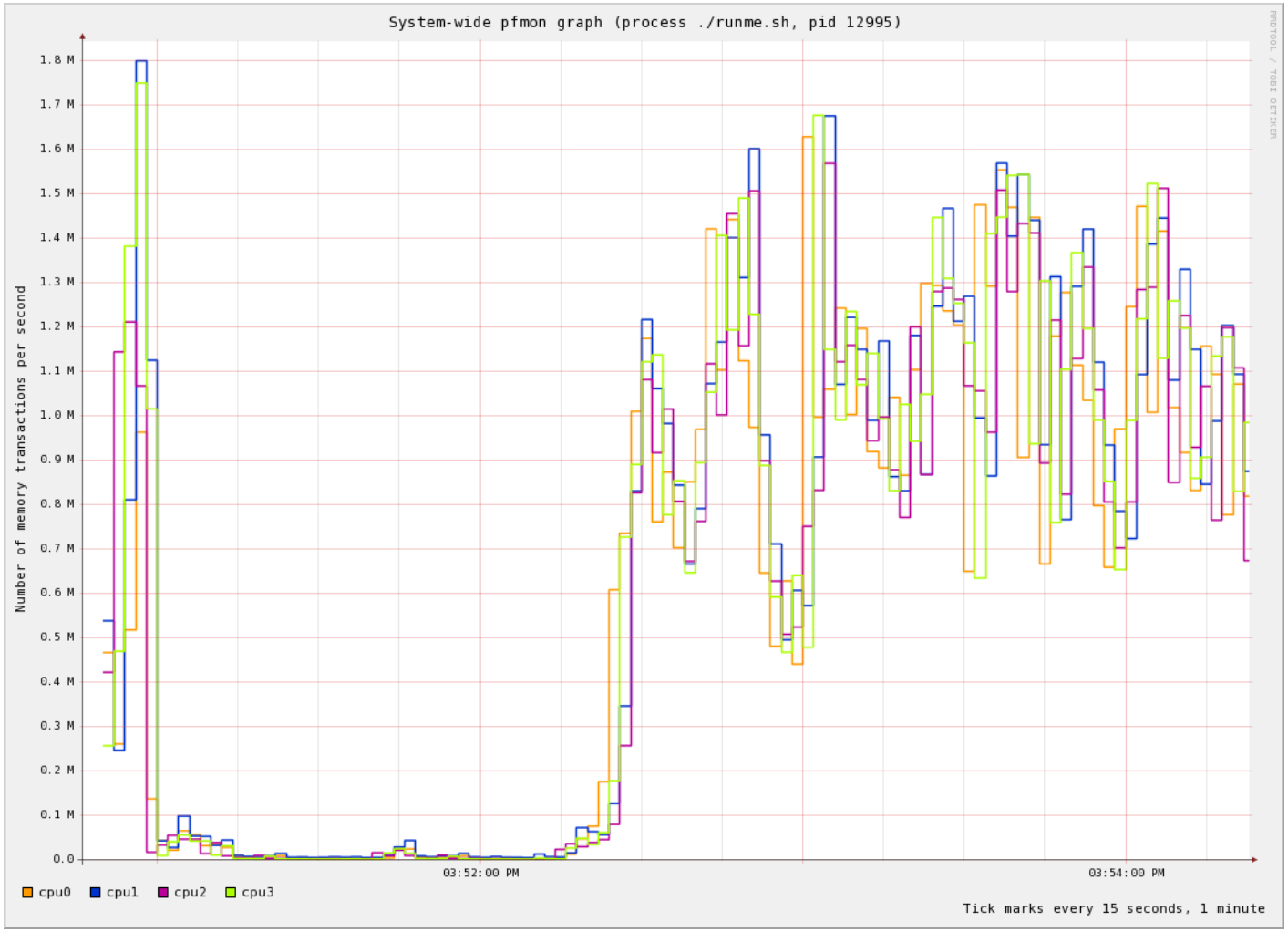
Single process – memory transactions



Multiple processes – CPU usage



Multiple processes – memory transactions



Conclusions

- 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