How to Exploit MultiCore: Summary & Outlook

Kickoff workshop April '08

High Performance Computing
for High Energy Physics

Technology Trend (Intel & OpenLab)

- Moore's law still apply in foreseeable future
  - » Challenge is not surface, is power!
- Multi-core micro-architecture
  - » Evolving, do not expect major changes
- New memory layout and connections to cpu
  - » New levels in hierarchy
- Hyper-threads
  - » They are back!
- SIMD: Intel AVX
  - » 1024 bit registers
- Mini-core
  - » Not just GPU, full IA
  - » 1024 threads in one box

The Challenge

Exploit all 7 dimensions of modern computing architecture for HPC

- Inside a core
  - » Superscalar : Fill the ports (maximize instruction per cycle)
  - » Pipelined : Fill the stages (avoid stalls)
  - » SIMD : Fill the register width (exploit SSE)
- Inside a Box
  - » HW threads: Fill up a core (share core & caches)
  - » Processor cores: Fill up a processor (share of low level resources)
  - » Sockets: Fill up a box (share high level resources)
- LAN & WAN
  - » Optimize scheduling and resource sharing on the Grid

- HEP has been traditionally good (only) in the latter

Experience and requirements

- HEP code does not exploit all the power of current processors
  - » One instruction per cycle at best
  - » Little or no use of SIMD
  - » Poor code locality
  - » Abuse of the heap
- Running N jobs on N=8 cores still efficient but:
  - » Memory (and to less extent cpu cycles) wasted in non sharing
    - "static" condition and geometry data
    - I/O buffers
    - Network and disk resources

Experience and requirements

- Complex and dispersed software
  - » Difficult to manage/share/tune resources (memory, I/O) w/o support from OS and compiler
  - » Coding and maintaining thread-safe software at user-level is difficult
  - » Need automatic tools to identify code to be made thread-aware
    - Geant4: 10K lines modified!
    - Not enough, many hidden (optimization) details such as state-caches
- Multi process seems more promising
  - » ATLAS: fork() (exploit copy-on-write), shmem (needs library support)
  - » LHCb: python
  - » Proof-lite
- Other limitations are at the door (I/O, communication, memory)
  - » Proof: client-server communication overhead in a single box
  - » Proof-lite: I/O bound >2 processes per disk
  - » Online (Atlas, CMS) limit in in/out-bound connections to one box

WP8: The Project

- 4 years
- Today 2x0.5 FTE + visitors
  - » Possible evolution to 2 FTE including contribution from EU
- Hardware resources to perform tests
  - » in a controlled environment
  - » using leading edge HW, OS, OS-tuning and tools not available on lxplus
- Collaboration
  - » LHC Experiments
  - » OpenLab
  - » Other HPC projects in HEP (and elsewhere?)

WP1: Technology Tracking & Tools

- Objective
  - » Investigate current and future multi-core architectures.
  - » Evaluate tools to measure performance.
  - » Develop a measurement and analysis methodology.
- Deliverables
  - » Assessment of industry trend in multi-core architectures.
  - » Recommendations on tools, metrics and methodology to assess the performance of LHC physics application software on such architectures

WP1: Technology Tracking & Tools

- Short term issues
  - » Deploy a stable environment to measure LHC production code
    - Kernel patches
    - Quattor support
    - Validation
  - » Provide a set of platforms where to perform such measurements
    - Patch lxbuild
    - Make more lxbench available
  - » Make available tools and documentation
    - Work with OpenLab to match experiment development environment
    - Develop a tool to monitor memory sharing

WP2: System and core-lib optimization

- Objective
  - » Measure and analyze performance of current LHC physics application software on multi-core architectures
  - » Identify bottlenecks
  - » Prototype solutions at the level of **system** and **core** libraries
- Deliverables
  - » Reports on performance of current LHC physics application software
  - » Recommendations on best practices to avoid bottlenecks and best exploit multi-core architectures
  - » Eventual materialization in software library components to implement them

WP2: System and core-lib optimization

- Short term issues
  - » Make sure we use an optimized OS and Compiler tuning
    - Check level of OS support for core2, opteron & NUMA (test new kernels)?
    - -m32 and SIMD: need to recompile libc, libm etc?
    - Investigate Large (d/I)tlb
    - provide library support to allocate memory in large pages
    - Investigate alternative “malloc”
    - Understand issues related to fork, shmen, threads in linux/gcc

WP3: Framework Parallelization

- Objective
  - » Investigate solutions to parallelize current LHC physics software at **application framework** level
  - » Identify reusable design patterns and implementation technologies to achieve parallelization
  - » produce prototypes
- Deliverables
  - » Recommendations on reusable design patterns and implementation technologies to use to achieve parallelization
  - » Eventual materialization in software library components to implement them

WP3: Framework Parallelization

- Short term issues
  - » Evaluate pro&cons of various alternatives
    - fork()
    - allocate C++ objects in shmem??
    - threads
  - » Identify, investigate and solve key showstoppers
    - Sharing I/O "channels"
    - Use of shared memory
    - Framework thread library
    - Parser to identify "thread sensitive" code
      - Define heuristic identification rules first!

WP4: Algorithm Parallelization

- Objectives
  - » Investigate solutions to parallelize **algorithms** used in current LHC physics application software
  - » Identify reusable design patterns and implementation technologies to achieve effective **high granularity** parallelization
  - » produce prototypes
- Deliverables
  - » Recommendations on reusable design patterns and implementation technologies to use to achieve effective high granularity parallelization
  - » Eventual materialization in software library components to implement them

WP4: Algorithm Parallelization

- Short term issues
  - » Evaluate gcc4.3 and port foundation and experiment code to it
  - » Experiment with posix-thread, OMP and parallel gcclib
    - Collaboration with OpenLab to setup training program
  - » Provide basic thread-safe/multi-thread library components
    - Random number generators
    - Parallelize minimization/fitting algorithms
    - Parallelize/Vectorize linear algebra

Organizational Matters

- Wiki
  - » Savannah?
- HyperNews, mailing-list?
- Meeting: Biweekly
  - » Monday 3pm, Thursday 2pm (alternate with AF)
  - » EVO, phone?
    - Do we need a physical room at cern???