

The Many Core paradigm

Session 6 - Future of LHC Computing: Virtualization/Multicores

Philip J. Clark

University of Edinburgh

GridPP25 Collaboration Week
26th August 2010



Outline

- 1 The many core paradigm
 - Many core systems
 - Multicore vs manycore
 - KSM kernel module
- 2 General Purpose GPUs
 - Advantages of GPU
 - Reality of GPUs
 - GPUs & Tracking
 - Tracking algorithms

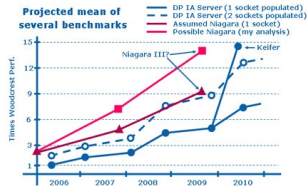
Outline

- 1 The many core paradigm
 - Many core systems
 - Multicore vs manycore
 - KSM kernel module
- 2 General Purpose GPUs
 - Advantages of GPU
 - Reality of GPUs
 - GPUs & Tracking
 - Tracking algorithms

Many core systems

Dispatch slots on a grid node: sockets*cores*hw-threads

- Total of what is in /proc/cpuinfo
 - 1 Dual-socket Intel quad core Harpertown: $2*4*1=8$
 - 2 Dual-socket Intel quad core Nehalem: $2*4*2=16$
 - 3 Quad-socket Intel six core Dunnington: $4*6*1=24$
 - 4 Quad-socket Intel six core Nehalem: $4*6*2=48$
 - 5 Quad-socket SUN Niagara (T2+): $4*8*8=256$
 - 6 Future 32 sockets with 16 processors?
- Soon to have 100s dispatch slots.



Multicore vs Manycore

Multi-core processor

Two or more independent cores on single integrated circuit die

Many-core processor

When the number of cores is large enough that that traditional multi-processor techniques are no longer efficient (often deliberately)

Amdahl's Law

Gains limited by proportion (α) of parallel vs. seq. code (s)

$$\frac{1}{(1 - \alpha) + \frac{\alpha}{s}}$$

Gustafson's Law

Gains limited only by no. of processors (P) i.e. problem size

$$P - \alpha(P - 1)$$

More than just six Grid slots?

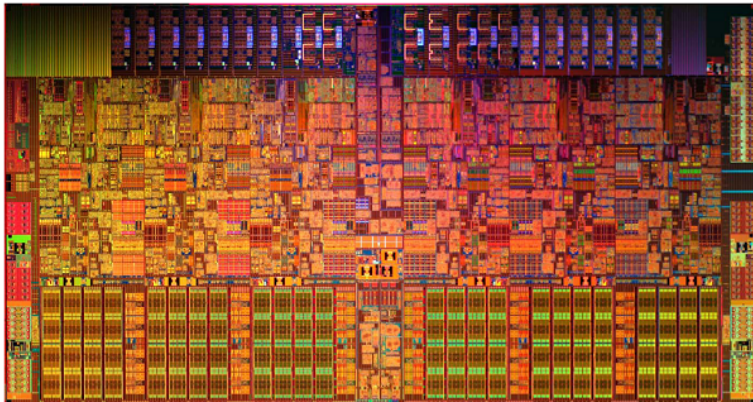
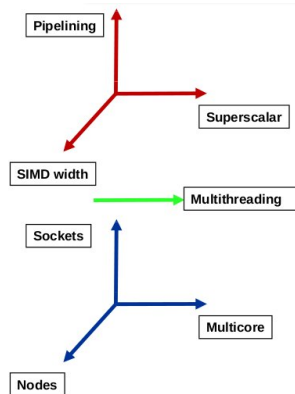


Figure: 32nm processed Westmere wafer die

Seven dimensions of CPU performance

Sverre Jarp

- “Within core” dimensions
 - Superscalar
 - Pipelining
 - Computational width (SIMD)
- “Pseudo” dimensions
 - Hardware multithreading
- Other dimensions
 - Multiple cores
 - Multiple sockets
 - Multiple compute nodes



Possible ways forward

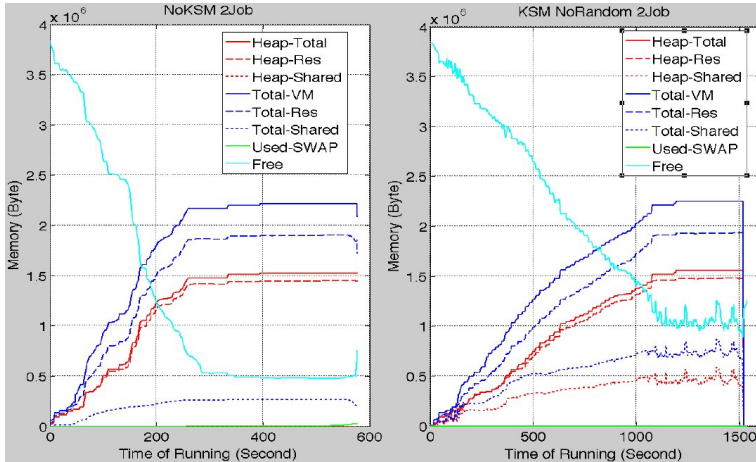
- Stay with event-level parallelisation
 - Assumes necessary memory remains affordable
 - Major I/O problem looming
- Rely on forking (just fork n processes)
 - rely on KSM
 - Or OS to do “copy-on-write” (AthenaMP idea)
- Move to a fully parallel paradigm
 - Soon to have hundreds of dispatch slots (GPUs even more)

KSM—Dynamically sharing identical memory

A. Arcangeli — RedHat

- Sorts pages by raw page content with memcmp()
- If memory is identical then share it
 - **stable** shared write protected pages
 - **unstable** tracked anonymous pages (index can change)
- Developed to aid KVM hypervisor
- 500 MB real memory gain demonstrated.
- **Pros:** if code running within KVM then all done automatically.
- **Cons:** VSZ unchanged though (Linux memory accounting again)

KSM results



500MB less memory consumed

Processor level multi-threading

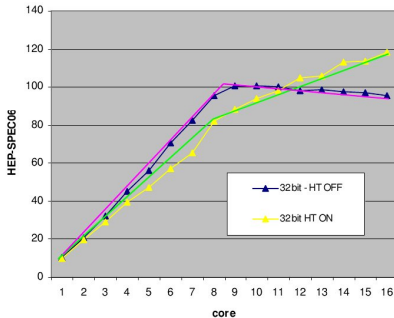
Simultaneous multi-threading (SMT)

- SMT not present in Intel consumer processors since 2006.
- Latest Intel Nehalem architecture brought it back (hyperthreading)... [used also in some Intel Atom CPUs]
- Initial reports indicate performance boost (next slide)
- Advantage, transparent to SMP enabled OSes.
- Disadvantage, OS unaware of logical vs physical cores.
- Can treat this by scheduling (subset/part of NUMA (later))
- Major ramification: more memory required per physical core.
- Hyper-threading performance for ATLAS jobs. vs. it turned off?

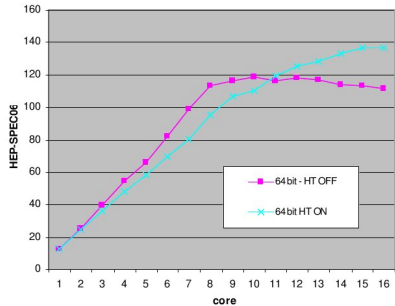
Hyperthreading results (HEPiX)

Michele Michelotto

32 bit HT OFF vs ON



64 bit HT OFF vs ON



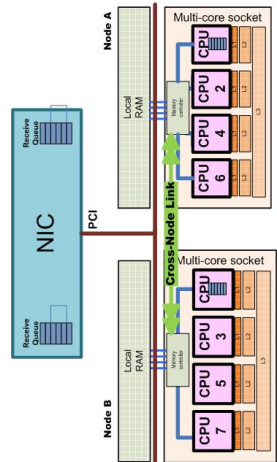
HT better for more than 11 threads with HEP_SPEC 2006 benchmark.

Non-

Non Uniform Memory Access (NUMA)

A. Washbrook

- Memory organised into number of distinct “nodes” to which CPUs can be attached (depending on locality).
- Tasks are then given node affinity (faster access to cpu cache)
- Memory bottlenecks may be alleviated by NUMA architecture.

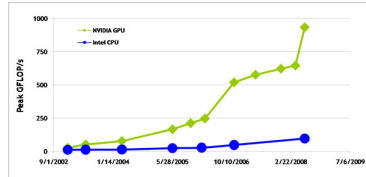


Outline

- 1 The many core paradigm
 - Many core systems
 - Multicore vs manycore
 - KSM kernel module
- 2 General Purpose GPUs
 - Advantages of GPU
 - Reality of GPUs
 - GPUs & Tracking
 - Tracking algorithms

Advantages of a GPU

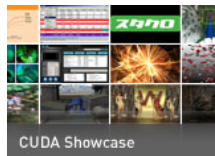
- Staggering *potential* performance
- The main strengths are:
 - Many more floating point units than a CPU
 - Can provide many more threads in flight than a CPU
 - The memory interface is much faster (GDDR3)
- The total memory for the Tesla S1070 is 4x4 GBs
- Theoretical bandwidth of 408 GB/s.
- Total proc. power 4.147 TFLOPS.



- No. flops increasing exponentially
- Doubling time is roughly half that of CPUs
- Shared memory has equivalent speed to CPU cache

Reality of GPUs

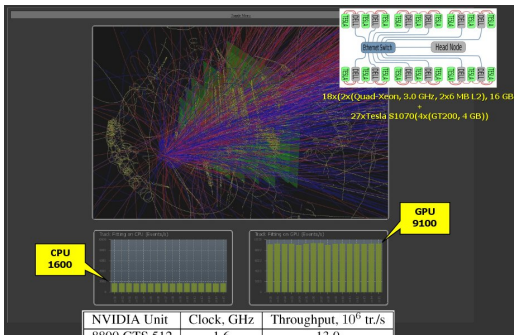
- Programming model very different from normally that used
- Code can branch (not fully vectorised), but then you lose performance gain (currently requires low level design)
- GPU memory area has to be specially handled and the architecture is evolving rapidly
- Various SDKs CUDA (nvidia), Stream (AMD) and OpenCL (Apple & DirectCompute)
- For examples see the CUDA Community Showcase (~1200 apps)



http://www.nvidia.com/object/cuda_home_new.html

GPUs & Tracking

- Kalman Filter port to CUDA (GSI Scientific Report 2008, FAIR-EXPERIMENTS-38)
- ALICE TPC HLT code GPU based / Future PANDA TPC code
- GPUs to be used for STS (Silicon Tracking System) within CBM (Compressed Baryonic Matter) experiment at FAIR/GSI.



ATLAS plans

- Some test cases to gain experience (next slides)
- Full requirement capture
- Design of ATLAS tracking prototype

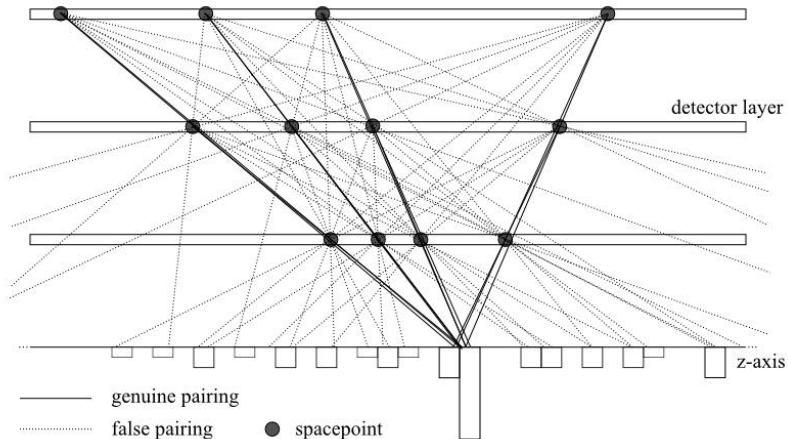
Tracking algorithms

Need relatively standalone application to gain experience

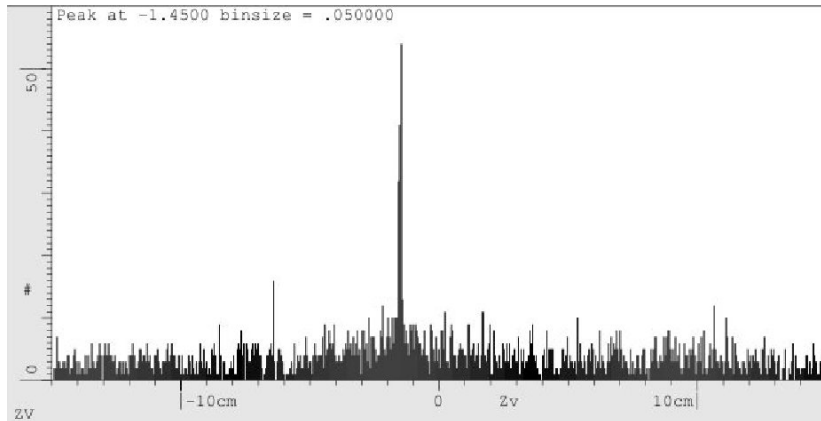
⇒ Used the ATLAS L2 trigger algorithm TrigIDScan

- Performs pattern recognition in the silicon trackers using space points.
- Start with zFinder which reconstructs z-position of primary pp collision
⇒ Port to CUDA and use to study performance optimisation
- The Extended Kalman filter algorithm used for track fitting.
⇒ Port to CUDA (more difficult).
- Could use performance boost to study alternative more time consuming tracking algorithms, e.g., fast Hough Transforms etc.

Z-finder algorithm



Z histogram

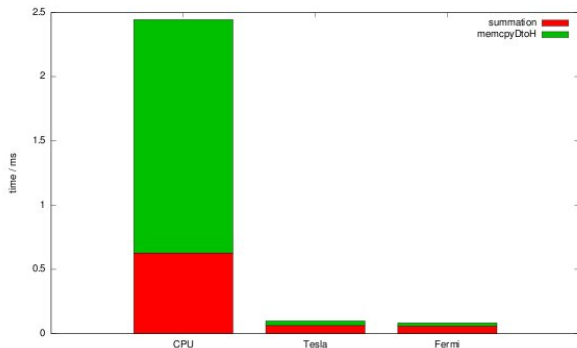


Example histogram for a low luminosity event.

z-finder port to GPU

Chris Jones

- The z-finder algorithm has been ported from C++ to CUDA
- Targeted both Tesla and Fermi GPU architectures.



Summary

- Many core paradigm is upon us
- Easy years of CPU frequency doubling are long gone
- Must work to use architectures more intelligently

GPU technology is evolving rapidly, future HPC machines will be GPU based. Useful for massively parallel tasks, but memory architecture evolving rapidly...

Conclusion: Think parallel, but avoid lock-in