Creating Innovative Solutions

IEEE Computer Society Workshop on Multicore Computing

Trivandrum, 16 Nov 2007

Concurrency Revolution In Software Development

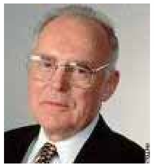
www.nestsoftware.com

Presented by

Unni Sankar

Agenda

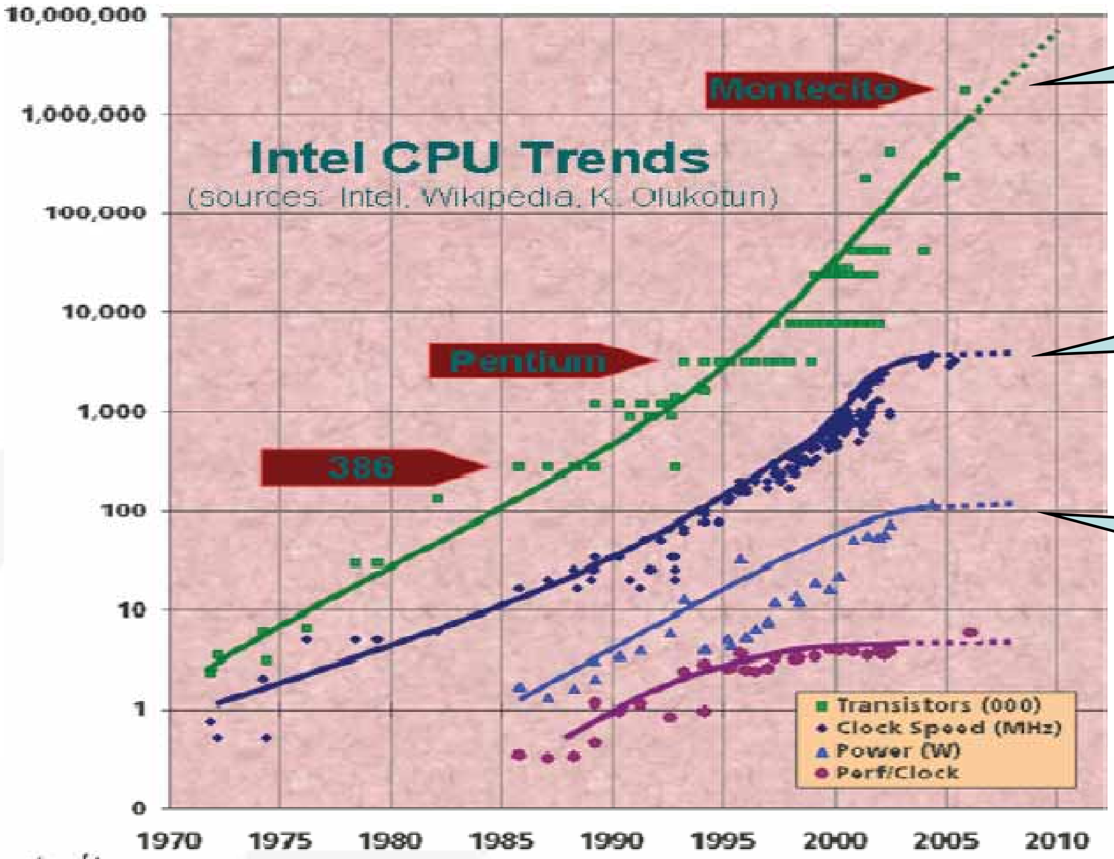
- **Trends in Processor Hardware**
  - From Single core to Multi core
  - From CPUs to Stream Processors
- **Implications to Mainstream Software Development**
  - How to exploit parallelism
    - Conventional approaches
    - Radically new approaches
  - Challenges faced by the Industry
  - Lessons to be learnt from other communities
    - Such as GPGPU
- **What lies ahead**
  - Different players, different roadmaps



Moore's Law

Number of transistors on a chip doubles about every 18 months

-predicted in 1965 by Gordon Moore



Moore's law still holds good

However, clock speed is flattening

Power per chip is peaking

What these trends mean?

- **In the past, performance gains came from raising the CPU clock speed**
  - – By shorter circuit paths on chip made possible by higher integration density
  - ➔ This allowed serial programming to thrive
- **But now, serial CPU performance isn't improving as before**
  - – For single threaded CPUs, no much benefit in having more transistors
  - – Memory speed hasn't kept pace
  - – Physics is catching up, finally
    - Power consumption
    - Heat dissipation
    - Current leakage
  - ➔ CPU designers looking at alternatives

A turning point in computing history

More Semiconductor Trends

- ***Processor*** **Capability**
  - increases by ***71%*** every year
- **Semiconductor *Memory* technology also advances, but not as much**
  - ***Capacity doubles*** every year
  - ***25%*** increase in ***bandwidth*** every year
  - ***15%*** decrease in ***latency*** every year

Source: ITRS (International Technology Roadmap for Semiconductors)

- **Consequences, when this continues for a long time:**
  - *One metric changes at a different rate than other → requires rethinking of the assumptions behind processor and system design*
    - Compute Vs Communicate
    - Latency Vs Bandwidth
    - Power

Time to rethink computing?

Towards High Performance Computing

- **Transistors in the Processor hardware**
  - Control: to direct computation
  - Data path: to perform computation
  - Storage: to store data
- **For Efficient Computation**
  - Transistors in data path to be maximized
    - By Parallelism
      - Task Parallelism: several tasks on different data
      - Data Parallelism: same task on several data elements
      - Instruction Parallelism: several simple operations at the same time
    - By specialization
      - Special purpose hardware
- **For Efficient Communication**
  - Prefer on chip communication to off-chip
  - Caching

The key lies in using more transistors for computation

Evolution of Processor Hardware

- **How to use the extra transistors?**
  - By having more cores per chip
    - Multi-core architectures
  - By having specialized accelerators
    - Eg: Java Virtual Machines
  - By having SIMD units
    - Single Instruction Multiple Data (SIMD)
    - Also termed Stream Processing Units
- **Why SIMD?**
  - Exploit data parallelism
  - Hide memory latencies
  - Memory optimized for throughput
  - Extremely power efficient

Future belongs to heterogeneous multi-core processors

Mainstream Computing at Crossroads

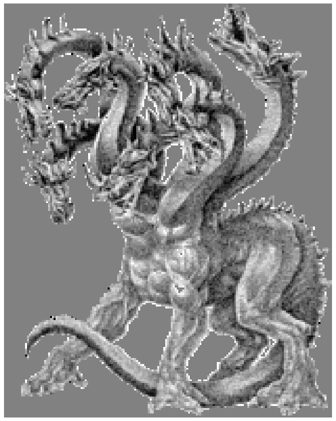
- **Legacy SIMD Architecture**
  - All active processors execute same instructions at same time
- **Modern SIMD Implementations**
  - Groups of SIMD units, with each group following different paths of the program
- **Different Approaches**
  - Extending the Micro Architecture
    - SSE (Streaming SIMD Extensions) for x86
  - Leveraging commodity graphics cards
    - GPUs (Graphical Processing Units) from nVIDIA, ATI
  - Entirely new designs
    - Cell BE (IBM, Sony, Toshiba) – *launched 2006*
    - Fusion (AMD) – *expected 2008*
    - Larrabee (Intel) – *expected 2009*

NeST

Moving beyond von Neumann Architecture

Implications to Software Development

- **Software holds the key**
  - In extracting parallelism from modern processors
- **The Free Lunch is Over***
  - With clock speed gains ➔ applications just run faster
  - With new processor architectures ➔ applications need to be (re)-written to exploit parallelism
    - Task level parallelism is only part of the solution
    - Data-level parallelism is the way to go
  - Parallelism is hard for programmers
    - Data parallelism is especially so



*Herb Sutter, Microsoft, ***The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software***, Dr. Dobb's Journal, March 2005*

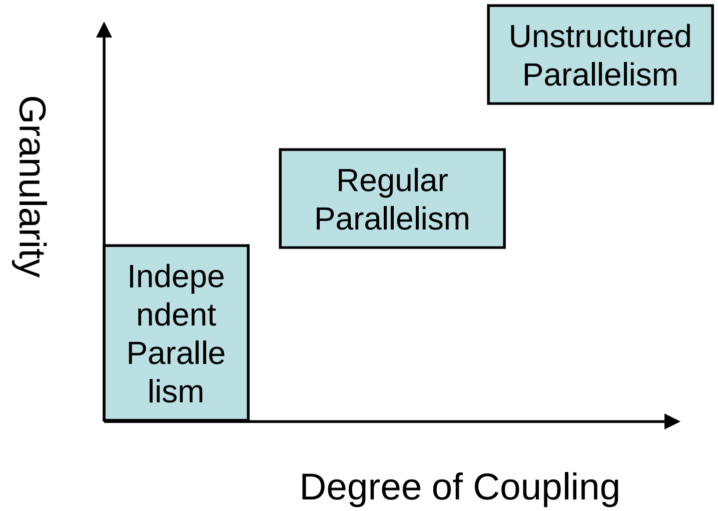
The Concurrency Revolution

- **Concurrency: Next major revolution in writing software***
  - Similar to Object-Oriented in the 90s.
    - Scale / Applicability
    - Complexity / Learning curve
  - Concurrency in mainstream software till now
    - To logically separate control flows
    - To take advantage of latency in other parts
      - IO, Database, Network
  - Concurrency from now
    - Explicit, implicit and automatic parallelization
    - Better abstractions
    - New tools

Herb Sutter, James Larus, Microsoft, **Software and the Concurrency Revolution, ACM Queue, Sep 2005*

Programming Models

- **Different ways to express parallelism**
  - Models differ in two dimensions
    - Granularity of parallel operations
    - Degree of coupling between tasks
  - Types of Parallelism
    - Independent parallelism
    - Regular parallelism
    - Unstructured parallelism
- **Problem of shared state**
  - Locks are not the answer
    - Difficult to program
    - Cause debugging nightmares
    - Doesn't scale well
  - Beyond locks
    - Lock-free programming
    - Transactional memory



Think Parallel or Perish!

How the Industry is gearing up

- **Conventional Approaches**
  - Processes and Threads as supported by OS
- **Industry Standards**
  - Pthreads
  - OpenMP
  - MCAPI
- **Popular Programming Languages**
  - Imperative languages such as C++, Java, C#
  - Expect more constructs to express parallelism
- **Functional Languages**
  - Such as Scheme, ML, Haskell
  - Natural ways of expressing parallelism
- **Tools**
  - To detect defects, debug, find performance bottlenecks, and test

Programming Heterogeneous Cores

- **Heterogeneous Multi-core Processors**
  - Cores vary in
    - Functionality
    - Instruction sets (ISA)
    - Performance
  - Programming them will be a key challenge
    - Not attempted in traditional mainstream computing
    - Different way of thinking is required
  - Lessons from
    - Traditional Super Computing
    - DSP Programming
    - **GPGPU (General Purpose Computation on GPU)**

GPGPU Overview

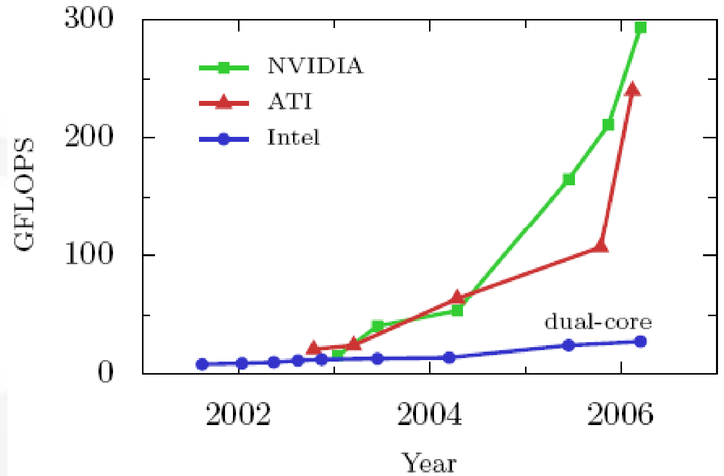
- **GPGPU**
  - General Purpose Computation using GPU
  - To harness the power of GPU for general purpose computing
- **GPU: Graphics Processing Units**
  - Primarily designed for interactive graphics
    - Graphics pipeline with programmable stages (groups of SIMD units)
  - Driven by Entertainment Market
    - Gaming, Media
  - High processing throughput at affordable cost
    - Unbeatable FLOPS per $!
  - Stream Programming Model
    - High efficiency through task and data parallelism

A vibrant community at www.gpgpu.org

Why GPU?

- **GPU scores over CPU in**
  - Computational Horsepower
    - 240 GFLOPS in ATI Radeon X1900 XTX
    - 25.6 GFLOPS in a comparable CPU*
  - Memory Bandwidth
    - 51.2 GB/sec in NVIDIA GeForce 7900 GTX
    - 8.5 GB/sec in a comparable CPU*
  - Rate at which they get faster

* Intel Pentium Extreme Edition 965 dual core 3.7GHz



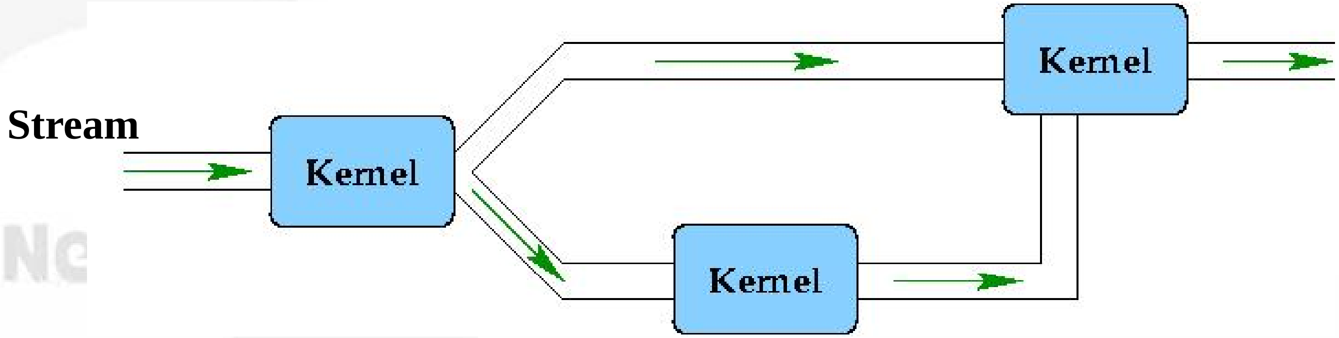
nVIDIA GeForce 6600GT

ATI Radeon 9800

Computational Power (FLOPs) per $ of GPU is unparalleled

Stream Programming

- **Stream Programming Model**
  - Organize an application into *streams* and *kernels*
  - Data is represented as ***stream***
    - Ordered set of data of the same data type
  - Computation performed by ***kernels***
    - Operates on entire streams of elements
  - Allows high efficiency in computation and communication
    - By exposing the inherent concurrency and locality



Stream Programming Options from GPGPU

- **Domain specific APIs**
  - OpenGL, DirectX
- **Domain specific Programming Languages**
  - GLSL, HLSL, Cg
- **Stream Programming Models**
  - Abstracts GPU as a stream processor
  - Brook, CUDA, SPUR, Sh
- **Stream Programming Virtual Machines**
  - GPU optimized math libraries as extensions to standard languages
  - Allows GPU acceleration without explicit GPU programming
  - RapidMind, PeakStream

Still evolving and rapidly changing

EXOCHI: a promising alternative

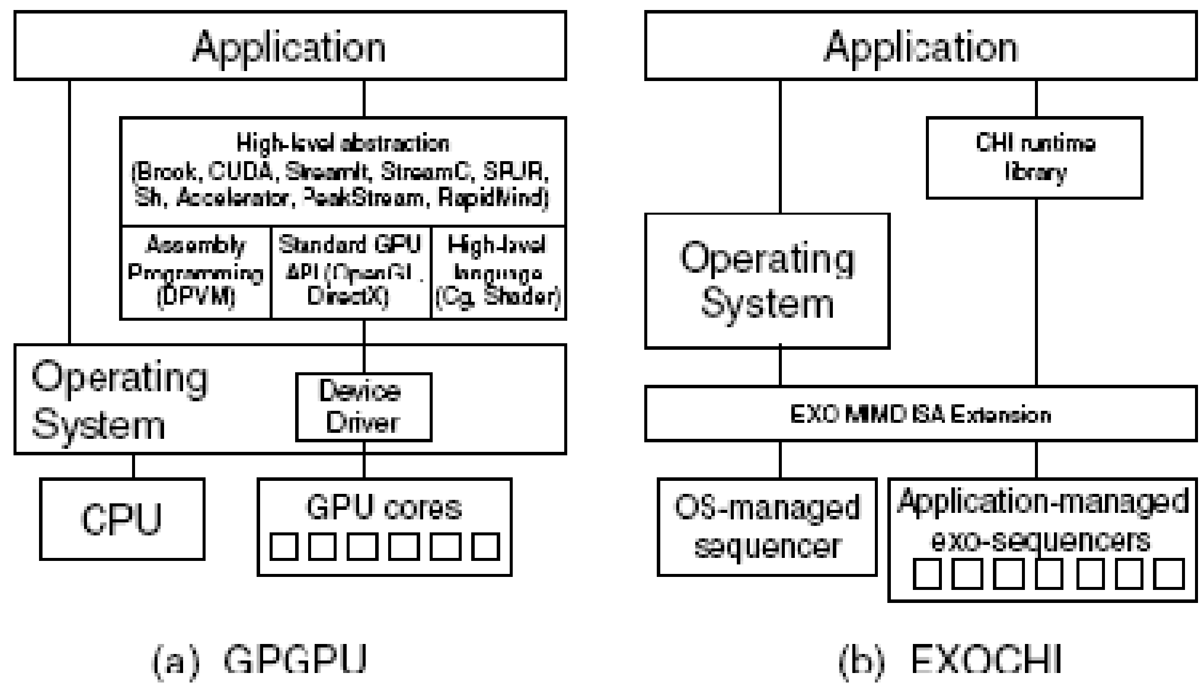
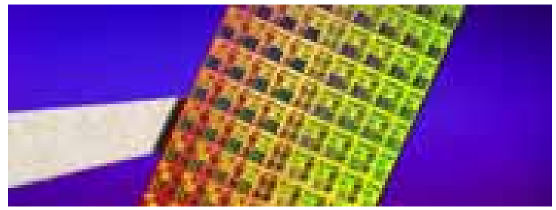


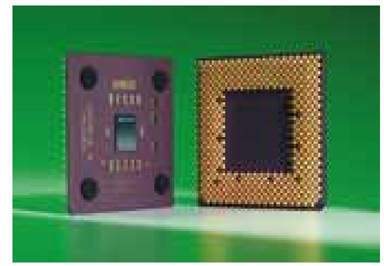
Figure 1. Alternate Programming Environments

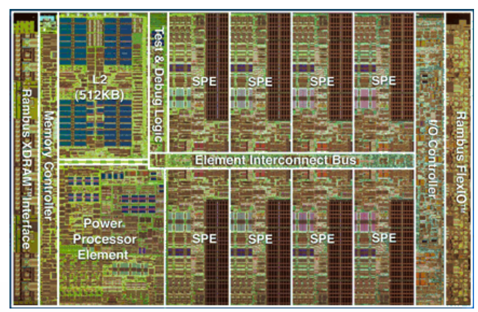
Perry H. Wang et al, **Intel,** *EXOCHI: Architecture and Programming Environment for a Heterogeneous Multi-core Multi-threaded System*, **PLDI 2007**

Hot Trends to Watch

- **From Intel**
  - Larrabee
  - Terrascale
  - Compiler Initiatives
- **From AMD / ATI**
  - Fusion
  - CTM
- **From nVIDIA**
  - Tesla
  - CUDA
- **From Microsoft**
  - DirectX 10
  - Accelerator
- **From IBM**
  - Cell/B.E



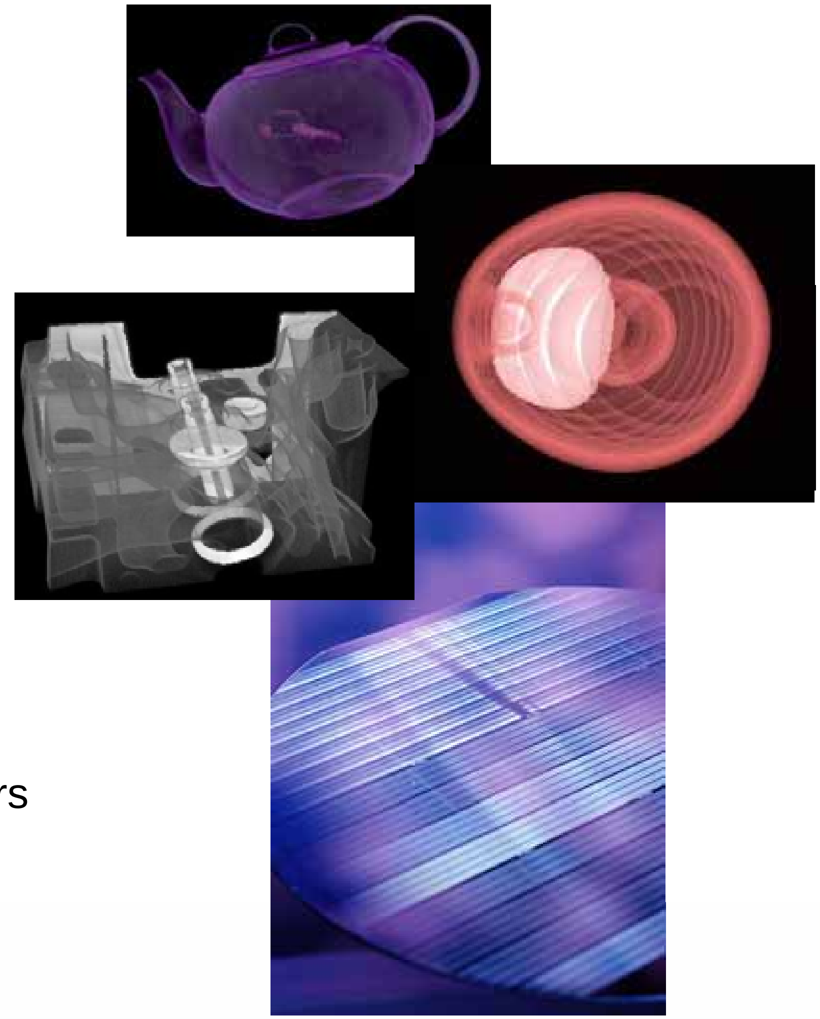




While the overall direction is same, roadmaps are different

Potential Applications

- **Graphics Applications**
  - Medical Imaging
  - Simulation
  - Gaming
  - GUI with 3D effects
- **General Purpose Applications**
  - Image Processing
  - Digital Signal Processing, esp. HD Video
  - Scientific Computing
  - Computational Finance
- **Traditional HPC applications**
  - That till recently needed Super Computers



Possibilities are mind-boggling

Conclusion

- **Parallel Computing is getting mainstream, and evolving**
  - Hardware platforms & technologies
  - Languages, Compilation techniques & Tools
  - Feature rich & computation-hungry Applications
- **Multi-core throws up new challenges in software development**
  - From single threaded to concurrent designs
- **Lessons to be learnt from others**
  - GPGPU community
  - DSP community
  - Traditional HPC community
- **Software holds the key**
  - No matter whatever advances are made in processor hardware

Future of Multicore lies in software

Questions?

www.nestsoftware.com

Thank You

www.nestsoftware.com