University of Central Florida

A Comparison of Various Multi-core/Many-core Architectures

Huiyang Zhou

School of Electrical Engineering and Computer Science
University of Central Florida

Challenges facing microprocessors

+ Moore's law still holds
  - Smaller transistor size
  - More transistors

- Power wall
  - Voltage does not scale
  - Power density problem

- Memory wall
  - Higher latency to hide

- Frequency wall
  - Diminishing returns in performance
  - Worsen the power problem

Multi-core/Many-core Processors

- General purpose multi-core CPUs
- Graphics processors
  - AMD/ATI RV670, RV770, etc.
  - Nvidia G80, GTX280, etc.
- Cell Broadband Engine
- Larrabee (later)

Power wall

- Key idea: each core runs at lower voltage (potentially lower frequency)
- Dynamic power consumption is proportional to FV^2
- Same idea for mutli-core CPU, GPUs, Cell BE and Larrabee
  - Homogeneous cores vs. heterogeneous cores

Memory wall

- Multi-core CPUs
  - Large on-chip caches
  - On-chip memory controllers
- GPUs
  - Fine grain multithreading + coarse grain multithreading
  - Small caches
  - Software managed local shared memory (or cache)
  - On-chip memory controllers: Memory coalescing
  - Ratio of number of threads over the number of cores in each SIMD engine (AMD GPUs) or MP (Nvidia GPUs)
    - E.g., 400 cycles memory access latency
    - Warp size 32. Independent ops 2 after a memory access.
    - How many warps necessary to hide the latency
- Cell BE
  - Software managed shared memory (or cache)
  - DMA (enforced/explicit coalescing)
  - Overlapping DMA and computation
  - Coarse grain multithreading

Frequency wall

- Multi-core CPUs
  - Balancing number of pipeline stages vs. complexity
    - Complicated branch predictor for control hazard
    - OOO + Renaming + Bypass for data hazard
  - Most advanced semiconductor technologies
- GPUs
  - Simplest pipeline: small number of pipeline stages, in-order, scalar (Nvidia GPUs) / VLIW (AMD/ATI GPUs)
- CellBE
  - Vector pipeline (a relatively high number of pipeline stages vs. GPU)
  - Rely on software to handle branches

Programming model

- Multi-core CPUs
  - Shared memory model (cache coherent)
  - Most flexible, compatible with legacy code
  - Explicit multithreading for high performance (task-level parallelism)
    - The cost of communication among threads is low
  - SSE (SIMD instructions) for data-level parallelism
- GPUs
  - SPMD
    - High cost of communication among threads if not in the same thread block
  - Data level parallelism
- CellBE
  - Explicit multithreading
  - Vector/SIMD programming
  - Explicit management of communication

Key to Performance

- Multi-core CPUs
  - Instruction-level parallelism exploited by hardware
  - Thread-level parallelism by user
    - Workload balance
    - Inter-thread communication & synchronization
- GPUs
  - Data-level parallelism by user
  - Local data share (or shared memory) explicitly managed by user
  - User also needs to overcome the limited HW support for inter-thread communication and synchronization
- CellBE
  - Data-level parallelism by user
  - Local storage explicitly managed by user
  - Vectorization, either by user or compiler
  - User needs to manage DMAs and hide their latency