A Balanced Programming Model for Emerging Heterogeneous Multicore Systems

Wei Liu, Brian Lewis, Xiaocheng Zhou, Hu Chen, Ying Gao, Shoumeng Yan, Sai Luo, Bratin Saha

Intel Labs, Intel Corporation
Emerging trends in computer systems

• Moving towards heterogeneous architectures
  – Combinations of CPUs & accelerators (e.g., GPUs)
    – Integrated & discrete accelerators, more and more cores
  – GPUs becoming more capable
    – Large virtual address spaces, unified memory, atomics
  – Manycore CPUs with wide vector instruction sets
    – Substantial data- and task-parallel computational abilities

How best to program these new systems?
Current programming models

- Heterogeneous programming languages today
  - OpenCL, Cuda, DirectCompute, ...
  - Low-level, but effective for many applications
    - Emphasize data parallelism
  - Support coarse-grain offloading
    - Underutilize CPU, don’t support fine-grain computation well
  - Share flat data structures like arrays
    - No pointer sharing, require data conversion & marshalling

- Want to
  - Improve programmer productivity
  - Harness computational power of both CPUs & accelerators
  - Extend range of applications easily programmed
**Goal: a balanced programming model**

- Balanced computation between CPU & accelerators
  - Enable fine-grained computation using all cores
  - Better support for task and data parallelism
  - Load balancing, dynamic reconfiguration
  - Leverage substantial computational ability of CPUs

- Virtual memory (VM) sharing
  - Directly share pointer-containing data structures like trees

- Lightweight atomics, locks, ...
  - Low-cost task dispatch & coordination

*Want Improved Programmer Productivity*
Our implementations

- **Existing shared VM system for discrete Larrabee**
  - Targets throughput computing on Larrabee platforms
  - Described in 2009 PLDI paper by B. Saha et al.
  - Includes memory sharing, synchronization, task placement
  - Example uses: Bullet physics and Offset game engines

- **Shared memory prototype for CPU-integrated GPU**
  - Processors with on-die Intel Integrated Graphics GPU
  - Uses OpenCL with shared VM extensions
  - Buffer-level coherency & fine-grained concurrent access
Shared memory with discrete Larrabee

- **Supports release consistency**
  - Natural model for many programs, reduces coherence cost
  - Mine/Yours ownership rights enable coherency optimizations

- **OS on both sides**
  - Shared virtual address range allocated at startup
  - VM page protection used for coherency
    - E.g., updates detected using page faults
  - Implementation resembles SW distributed shared memory
    - Complicated by different page tables, virtual-to-physical mappings

- **Daemons on each side communicate over PCI aperture**
  - Pinned pages hold message queues & copy buffers
  - Enables user-level communication between runtimes
Shared memory on CPU-integrated graphics

- Device driver, no OS services like page fault handling
  - E.g., can’t detect updates using page faults

- Exploits shared physical memory
  - No data copying, supports fine-grain concurrent data sharing

- To share the virtual address space
  - Reserve a common virtual memory region
  - Allocate pinned physical pages & map to same address

- To manage memory coherence
  - Flush caches as needed
    - Discard stale data, make updates visible
A Balanced Programming Model for Emerging Heterogeneous Multicore Systems

Cost of sharing virtual memory?

Larrabee memory sharing performance comparable to traditional marshal-copy
Summary

• Current situation
  – Languages like OpenCL, Cuda, and DirectCompute
    – Good match for today’s hardware, effective for many applications
    – Emphasize data parallelism
  – Focus on coarse-grain offloading, underutilize CPU
    – Limited ability for work sharing & fine-grain computation
  – No pointer sharing, require data conversion & marshalling

• Goals
  – Better programmer productivity & performance
    – Pointer sharing, memory consistency, low-cost task dispatch
  – Harness computational power of all cores
  – Extend range of applications easily programmed