GSRC Gigascale Systems Research Center

Programming Frameworks and Tools for Many-Core Processors

Wen-mei Hwu
GSRC Concurrent Themeand UIUC
University of Illinois, Urbana-Champaign

MARCO

VIA 2008 at UIUC

- QP Cluster: 16 nodes, each with 2 AMD duo-core CPUs and 4 NVIDIA GPUs and 1 Nalletch/Xilinx HPC FPGA card
  - 32.5 TFLOSP SP peak + FPGA
- Apps include 3D model extractions, video event extraction, human emotion detection, physics modeling.
- MPI/CUDA programming with research frameworks and tools
- To be upgraded to a 32-node, 128 TFLOPS single precision, 16 TFLOPS double precision in August 2008

Why does programming need to be low cost on VIA?

- The apps folks have not figured out the right models and approaches!
  - Most are still Ph.D. thesis efforts.
  - Many development efforts must be done for them to figure it out!
  - High software development cost will kill app innovation and productivity
  - A large amount of new functionalities will have to be created and maintained.

Why is programming many-core processors costly today?

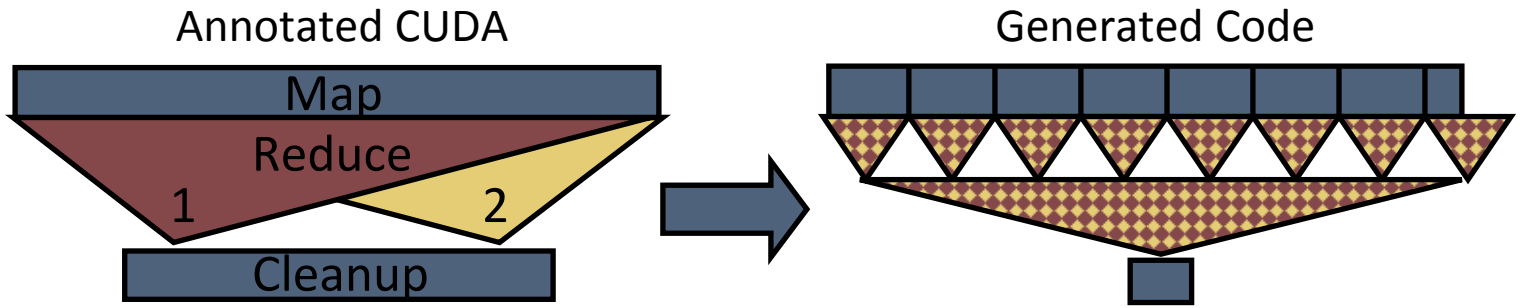
- Separate structure from CPU
  - Data isolation and marshalling with pressure to optimize away overhead
- Lack of standardized programming interface
  - Each has its own app development models and tools
- Management of specialized execution and memory resources
- Multi-dimensional optimizations required for achieving performance goals

A different approach from the past

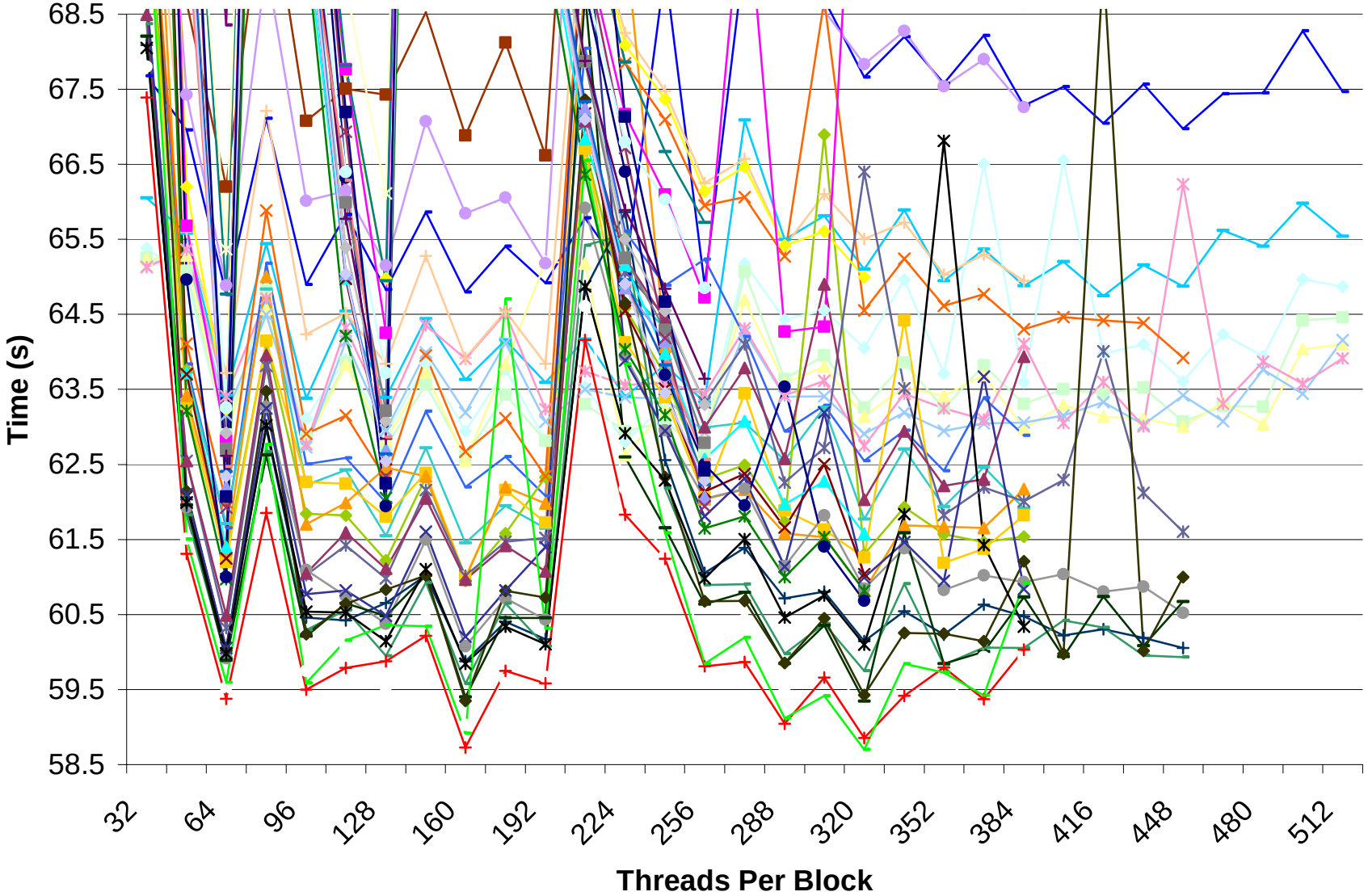
- Simple parallelism
  - Focus on simple forms of parallelism for programmers
  - Trade some generality and performance for productivity
- Power tools
  - Leverage and strengthen app development frameworks
  - Empower tools with specification, analysis and domain information

High-level Frameworks for GPU

- Programming many-core GPUs requires restructuring computations around its coordination capabilities
- Global communication is very complicated
- Approach: put this complication in a code generation framework
  - Coordination is made explicit by expressing computation as MapReduce
- User specifies set of reduction functions, map & cleanup functions
- Framework generates efficient multistage reductions implemented in CUDA kernels



Reduced Tuning Efforts



Implicitly parallel programming with data structure and algorithm property annotations to enable auto parallelization

CUDA-auto

Locality annotation programming to eliminate need for explicit management of memory types and data transfers, potential ATI entry point

CUDA-lite

Parameterized CUDA programming using auto-tuning and optimization space pruning

CUDA-tune

1st generation CUDA programming with explicit, hardwired thread organizations and explicit management of memory types and data transfers

MCUDA/ OpenMP

NVIDIA SDK 1.1

IA multi-core & Larrabe

NVIDIA GPU

Summary – A multi-level attack on the parallel programming beast

- Simple programmer-level parallelism with power tools
- New Algorithm Frameworks
  - MapReduce (UCB), Convolution (UIUC), etc.
- New Application Frameworks
  - Video (UIUC), Game Physics (NVIDIA), etc.
- More consistent programming across HW platforms
  - MCUDA (IA Multi-core/Many-core), CUDA-lite (ATI GPU, FPGA)
- Better heavy-lifting tools
  - CUDA-tune, CUDA-auto