VSCSE Summer School

Proven Algorithmic Techniques for Many-core Processors

Lecture 9: Directions for Future Studies and Closing Remarks

Course Objective (Review)

- To master the most commonly used algorithm techniques and computational thinking skills needed for many-core programming
  - Especially the simple ones!

- Specifically, to understand
  - Many-core hardware limitations and constraints
  - Desirable and undesirable computation patterns
  - Commonly used algorithm techniques to convert undesirable computation patterns into desirable ones

Agenda (Monday)

- 10:00 – 11:30 Lecture 1 – Introduction to computational thinking for many-core computing
- 12:30 – 2:00 Lecture 2 – Scatter-to-Gather transformation for scalability
- 3:00 – 4:30 Lecture 3 - Loop blocking and register tiling for locality

- Lab 1 – 2D Blocking and Register tiling

Agenda Review (Tuesday)

- Lecture 4 – Cut-off and Binning for regular data sets
- Lecture 5 – Data Layout for Grid Applications
- Keynote 1 – Michael Garland (NVIDIA)
  - Algorithm Design for GPU Computing

- Lab 1 – 2D Blocking and Register tiling
- Lab 2 - Data Layout Transformation for LBM Methods and Binning for Regular Data Sets

Agenda Review (Wednesday)

- Lecture 6 – Dealing with non-uniform data distribution
- Lecture 7 – Dealing with dynamic data sets
  - With guest lecture by Jonathan Cohen (NVIDIA) on PDE Solvers and OpenCurrent
- Keynote 2 – David Kirk (NVIDIA)
  - Fermi and Future of GPU Computing Technology

- Lab 2 - Data Layout Transformation for LBM Methods and Binning for Regular Data Sets
- Lab 3 – Binning for non-uniform data distribution

Agenda Review (Thursday)

- Keynote 3 – Lorena Barba (Boston University)
  - Multiplying speedups: fast algorithms on GPUs. A case study on biomolecular electrostatics.
- Lecture 8 – Transformations of key computation patterns
  - With guest lecture from Jeremy Meredith (Oak Ridge National Lab) on Accelerating HPC Applications with GPUs – Two Case Studies"
- Hands-on Lab 3 – Data binning for non-uniform data distribution
- Hands-on Lab wrap-up discussions

Phillip Colella's "Seven dwarfs"

High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
2. Unstructured Grids
3. Fast Fourier Transform
4. Dense Linear Algebra
5. Sparse Linear Algebra
6. Particles
7. Monte Carlo

- If add 4 for embedded, covers all 41 EEMBC benchmarks
  8. Search/Sort
  9. Filter
  10. Combinational logic
  11. Finite State Machine
- Note: Data sizes (8 bit to 32 bit) and types (integer, character) differ, but algorithms the same

Well-defined targets from algorithmic, software, and architecture standpoint

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

Seven Techniques in Manycore Programming

- Scatter to Gather transformation
- Granularity coarsening and register tiling
- Tiling for dense matrix data locality
- Data layout and traversal ordering
- Binning and cutoff
- Bin sorting and partitioning for non-uniform data
- Hierarchical queues and kernels for dynamic data

Benefit from other people's experience

- GPU Computing Gems Vol 1
  - Coming December 2010
  - 50 gems in 10 applications areas
  - Scientific simulation, life sciences, statistical modeling, emerging data-intensive applications, electronic design automation, computer vision, ray tracing and rendering, video and imaging processing, signal and audio processing, medical imaging
- GPU Computing Gems Vol 2
  - Coming in April 2011
  - 50+ gems on more application domains, tools, environments

Phillip Colella's "Seven dwarfs"

High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
2. Unstructured Grids
3. Fast Fourier Transform
4. Dense Linear Algebra
5. Sparse Linear Algebra
6. Particles
7. Monte Carlo

- If add 4 for embedded, covers all 41 EEMBC benchmarks
  8. Search/Sort
  9. Filter
  10. Combinational logic
  11. Finite State Machine
- Note: Data sizes (8 bit to 32 bit) and types (integer, character) differ, but algorithms the same

Well-defined targets from algorithmic, software, and architecture standpoint

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

Courtesy: David Patterson

How about your computation types?

- Are your applications very different from all the seven dwarfs?
- Can you think of any other algorithmic techniques that we should cover?

You can do it.

- Computational thinking is not as hard as you may think it is.
  - The purpose of the course is to make them accessible to domain scientists and engineers.
  - With practice, you can make it look easy!

ANY MORE QUESTIONS?