

Lecture 9:  
Data Parallel Programming Languages: A  
Renaissance for GPUs and Multi-Core

Mary Hall

[mhall@isi.edu](mailto:mhall@isi.edu), [mhall@usc.edu](mailto:mhall@usc.edu)

<http://www.isi.edu/~mhall/CS503Spring08.html>

# Administrative Issues

- Office hours: Tuesdays, 11-12 in SAL 234
- Project 2:
  - DEADLINE EXTENDED UNTIL FRIDAY!
  - ASK QUESTIONS TODAY!
- What's coming for the rest of the semester
  - CUDA (NVIDIA) assignment
    - Trying for programming assignment, but otherwise homework.
  - Partitioned Global Address Space Language (PGAS) assignment
    - Probably UPC, but looking into new language Chapel (CRAY)

## Outline

- Questions on assignment
- Data Parallel Programming Languages
  - What are they?
  - Why are they important for multi/many-core?
  - Examples: Ct, Sequoia
- Streaming Programming Languages
  - What are they?
  - Examples: StreaMIT, CUDA, PeakStream

# Sources for Today's Slides

- **Introduction to Parallel Programming**
  - [https://computing.llnl.gov/tutorials/parallel\\_comp/](https://computing.llnl.gov/tutorials/parallel_comp/)
- **Intel Ct website:**
  - <http://www.intel.com/go/Ct>
- **2007 Workshop on Data-Parallel Programming Models for Many-Core Architectures**
  - <http://www.cgo.org/cgo2007/>
  - <http://groups.google.com/group/dataparallel>
- **CUDA Workshop at SC07**
  - <http://www.gpgpu.org/sc2007/>
- **Stanford Sequoia**
  - <http://www.stanford.edu/group/sequoia/cgi-bin/node/8>
  - <http://cscads.rice.edu/workshops/july2007/autotune-slides-07/Houston.pdf>
- **StreamIT**
  - <http://cag.csail.mit.edu/streamit/talks/StreamIt-NEPLS-8-02.ppt>
- **NESL tutorial**
  - <http://www.cs.cmu.edu/~scandal/nesl/tutorial2.html>

## Assignment 2 (due 3/28)

- LU Decomposition (No pivoting)
  - Potentially numerically unstable, but we use random number generator for input
- Sequential optimization:
  - Improve locality in registers and cache
  - Demonstrate performance improvement
  - Extra credit: Also, SSE-3
- Parallel version:
  - OpenMP code
  - Try different parallelization strategies, different loop orders, different scheduling strategies, ...

## More on Project 2

- What is provided:
  - Sequential and OpenMP Fortran and C code (not equivalent)
  - Makefile
  - Way of creating input for Fortran version
  - PBS file: how to execute on HPC (next slide)
- Note: Current parallel versions
  - DO create threads
  - DO NOT actually execute anything in parallel (you must add directives for this)
- Use sequential version in same language as baseline
  - Either language is acceptable

## Executing on HPC

- Previously using the frontend machine for the HPC cluster.
  - Still use this to compile the code
- Want to run on specific nodes
  - V20z: AMD Dual Dualcore Opteron 2.0 GHz with 4GB Memory.
  - Interactively: `qsub -I -d . -l nodes=1:ppn=4:V20z`
  - For batch mode, which is preferred for timing, you'll need to create a PBS file.

## Executing on HPC, cont.

- PBS example file (on website):

```
#!/bin/bash
```

```
#PBS -l nodes=1:ppn=4:V20z
```

```
#PBS -l walltime=2:00:00
```

```
#PBS -o lu.out #PBS -j oe
```

```
#PBS -N lu-seq
```

```
WORK_HOME=/home/rcf-proj2/mwh1/mhall/proj2
```

```
cd $WORK_HOME
```

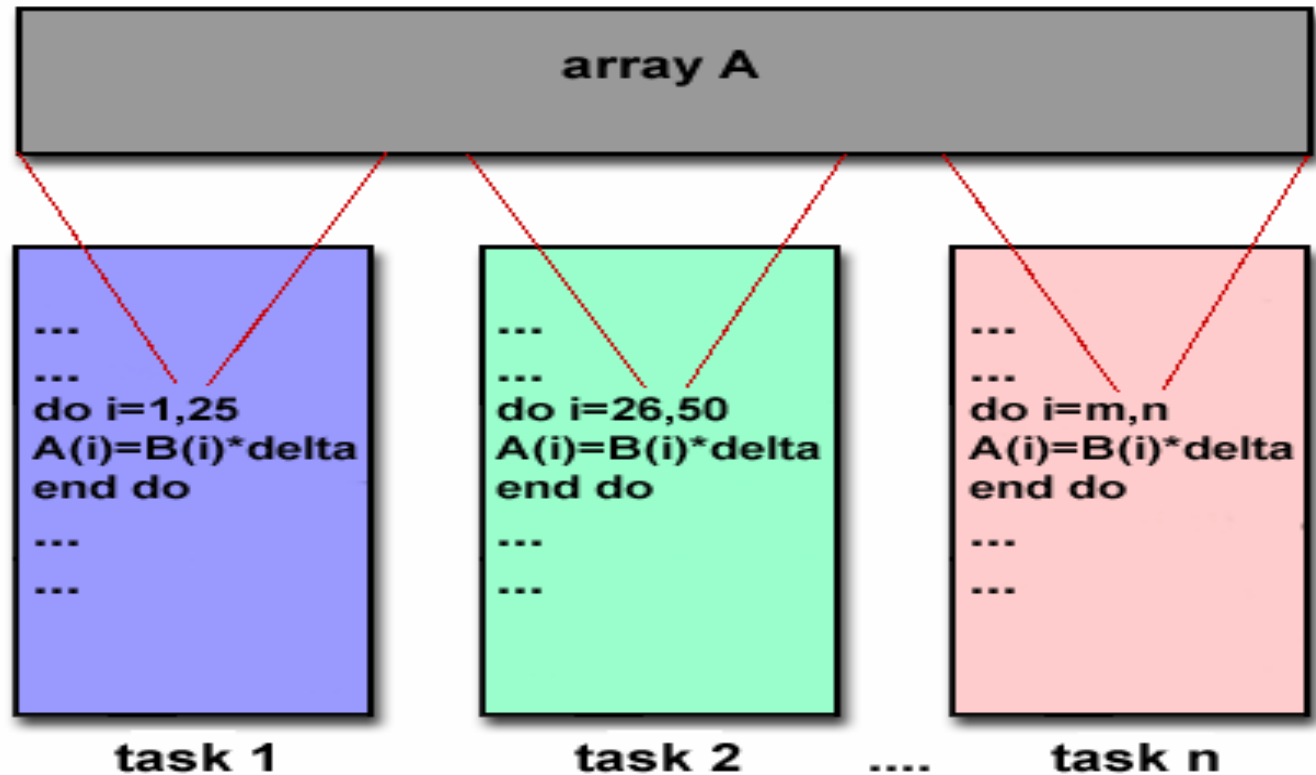
```
./lu-seq 1000
```

## Another thing to try: Prefetch

- The following can be used to insert prefetch instructions into code (not clear it works on Opteron)

```
DO K
  DO I
    call mm_prefetch(A(I+1,K),0)
    ... A(I+1,K)
```

# Data Parallel Programming



- Tasks work collectively on the same data structure. Each task works on a different partition of data.
- Tasks perform the same operation on their partition of work, for example, "add 4 to every array element".

# Data Parallel Programming Languages: History

- 60s & 70s
    - First Gen: Fortran Loops, APL
    - Flavor: Flat, embarrassingly parallel, simple reductions
    - Architectures: SIMD, Vector computers
  - 80s & 90s
    - Next Gen: Nest, Parallel Lisp, C\*, Fortran HPF, OMP
    - Flavor: Introduction of nested parallelism
    - Large-scale SIMD, MIMD-Vector computers, MIMD
  - Mid 90s: Supercomputers essentially died, focus on cluster
  - Late 90s
    - Emergence of consumer graphics acceleration
  - Early 00s
    - Mainstream Programmable pipelines: HLSL, GLSL, Cg
    - Flavor: Very constrained data parallelism
  - Mid 00s
    - GPGPU, parallel-DSP emerging: Brook, Streamit, Peakstream, RaplMind, Codeplay, Accelerator
    - Flavor: DP is back
- ← Expressive power peaked!
- ← Defense spending cut
- ← GPGPU going commercial...

Slide source: "Data Parallel Programming Models for Many-Core Architectures," Anwar Ghuloum and Matthew Papakipos.

## Some Data-Parallel Models we have Studied

- SIMD
  - Typically, not nested
  - Recall, particularly challenging in the presence of control flow
- SPMD
  - More general than what we are describing today, but convenient to express data parallelism

# Motivation

- Graphics processors have enormous compute capacity at modest price
    - Commodity high-performance small-scale systems!
  - Programming is difficult due to specialized hardware
    - What is a frame buffer? Texture sampler?
    - Sometimes fairly obscure relationship between program and performance.
    - Separate device: How to debug?
- Idea: Develop programming models that make GPUs more approachable to average programmers
- Side benefit: Same or similar programming models can be used for more conventional multi-core architectures

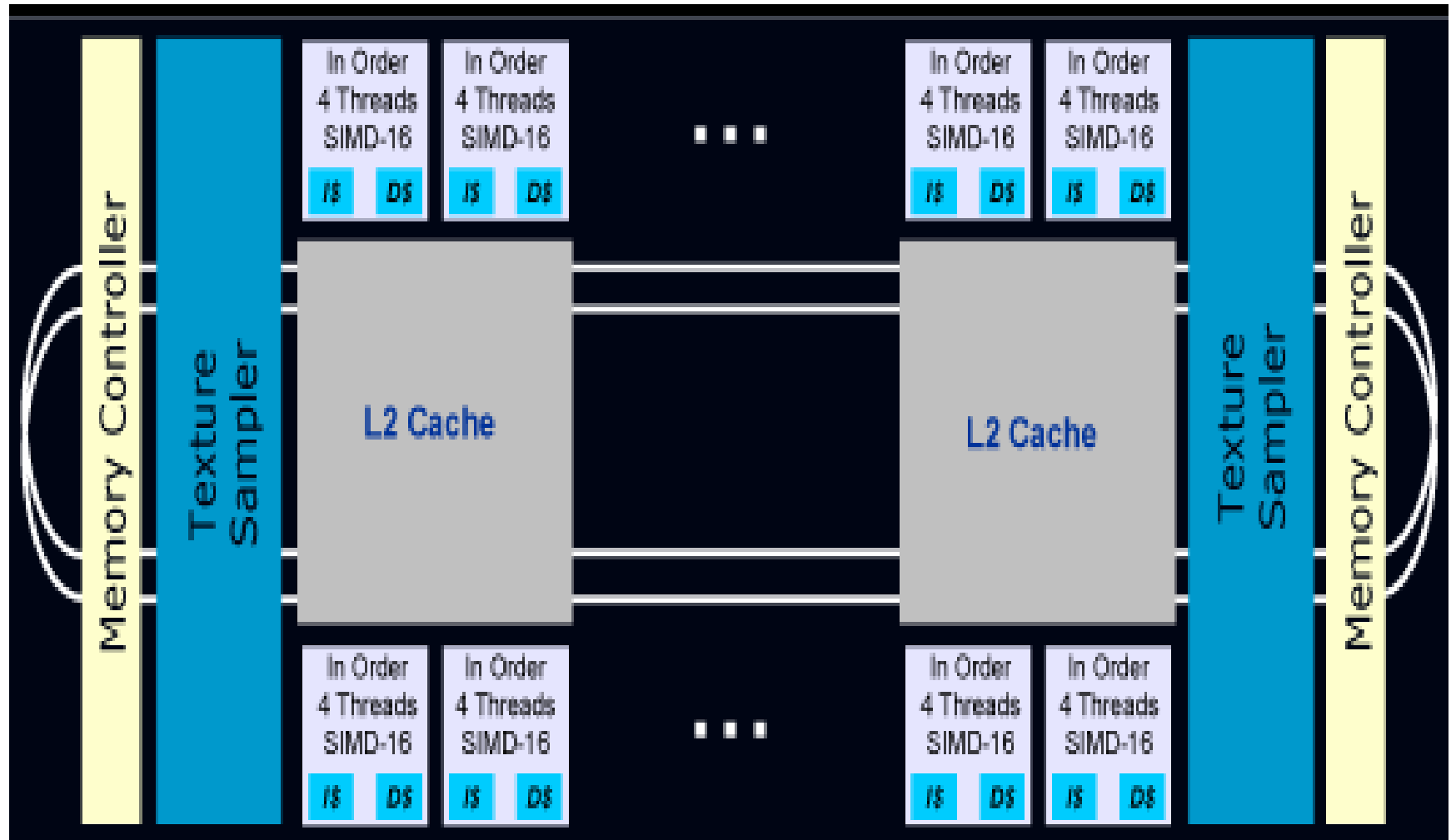
# Data Parallel for Multi-Core and GPUs

- Data Parallel criticized for not being widely applicable.
  - Not effective on “task parallel” computations
- But **MANY** advantages
  - Hardware can do this very efficiently
  - Scalable solutions if data set is sufficiently large
- Programmability
  - Well-suited to certain domains
  - Andrew Brownsword, EA: “If data parallel coding were easy, we would do a lot more of it. What works well informs game design as well as technical design.”
  - Deterministic behavior

## A Brief Look at a Few Examples

- Ct (Intel)
- Sequoia (Stanford)

# Intel Larrabee (announced last summer)



<http://media.arstechnica.com/news.media/larrabee-gpu.gif>

# Ct Features

- How to guarantee determinism?
  - All data-parallel operations on TVECs
  - Only operators allowed on TVECs are Ct operators
  - Ct operators are "functional", no side effects

# Streams:

## A Specific Data-Parallel Programming Model

- A programming model oriented around data flow
  - Think of code as a series of filters
  - Each filter takes input "stream(s)" and produces an output "stream"
- Features - not general data-parallel model:
  - Sometimes limited to linear filters
  - Typically, not nested
- Advantages:
  - Significant parallelism across independent streams
  - Stream provides a good boundary for copying data from one memory structure to another
  - Can be restricted to produce deterministic behavior
  - Size of stream can be adjusted and filters can be merged or split to map to architecture
- Examples:
  - StreamIT
  - PeakStream
  - CUDA (NVIDIA)
    - A full lecture devoted to CUDA next week! (Guest lecturers)

## Other Data-Parallel Approaches

- AMD CTM (Close to the Metal)
- API-based solutions from
  - Microsoft
  - RapidMind
  - ...

# Ways to evaluate alternative data-parallel programming models

## General:

- Type of data-parallel language
  - SIMD, Streaming, or something more general
- Does it support nested parallelism?
- New language or just extensions/library/API to existing language?

## Some modern twists:

- Does it have constructs for data movement and software-controlled storage?
- Does it support heterogeneous compute units?
- Recursion, managed run-time environment, garbage collection...

# Taxonomy

	Ct	Sequoia	CUDA	PeakStream	StreamIT
Nested Parallelism	✓	✓	?	?	X
Language	C/C++ "API-like interface	C++-based language	C with Extensions	Language-neutral API	Java classes
Data movement	X	✓	✓	✓	Implicit
Heterogeneity	X	Implicit	✓	✓	X
Recursion, managed runtime, etc.	✓	?	?	✓	X

# Summary

- Assignment details
- A discussion of data-parallel programming languages
- Is convergence possible?
  - Like OpenMP and MPI?

## What's ahead

- The next two weeks you will have 2 guest lectures:
  - 4/1 -- CUDA (NVIDIA): John Tran and Gene Wagenbreath
  - 4/8 -- HPCView and other performance tuning tools:  
Jacqueline Chame
- Next week a CUDA assignment
  - Stay tuned to your email and check the website