The Potential of the Cell Processor for Scientific Computing

Sam Williams, John Shalf, Leonid Oliker, Shoaib Kamil, Parry Husbands, Katherine Yelick
{SWWilliams, JShalf, LOliker, SAKamil, PJRHusbands, KAYelick}@lbl.gov
Computational Research Division – Lawrence Berkeley National Laboratory

Introduction

Cell

- 8 SIMD cores (SPEs) @ 3.2GHz
- Independent, explicitly controlled 256KB local store memory (not a cache)
- Decoupled DMAs improve bus utilization
- 1 SP SIMD instruction every cycle
- But science requires double precision
- 1 DP SIMD instruction every 7 cycles

Experimentation

- Performance modeling coupled with benchmarks on actual hardware
- Compared against actual hardware:
  - 1.13 GHz X1E MSP
  - 2.2 GHz Opteron
  - 1.3 GHz Itanium2

Programming Model

- Explicit software controlled one-sided communication (DMAs) for SPEs
- SPEs are programmed as if they were a distributed memory machine (SPMD)
- The SPEs (and PPU) must synchronize between program phases
- Programmed in C with intrinsics in the critical sections.
- Double buffered whenever possible and appropriate

Performance Modeling

Standard Cell Model

- Performance estimator written for each kernel to examine strategies before C with intrinsics Implementation
- Overlaps communication and computation
- Handles irregular problems (like SpMV)

Cell+ Model

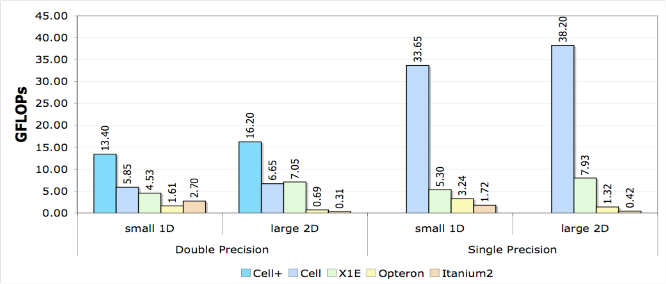
- Lengthen forwarding network to avoid stalling double precision instructions
- Modest modification; up to 3.5x benefit
- Cell+ makes DP instructions half-pumped (one SIMD every other cycle)
- How much does 6 stall cycles per double-precision instruction hurt performance?

1D & 2D Fast Fourier Transforms

Introduction

- Naïve radix 2 performance modeling
- Small explicit on-chip transpose for 1D (inter-local store)
- Large explicit transposes in DRAM for 2D
- Ensures long DMAs from global store and butterflies within local store

Performance Estimation



Future Work

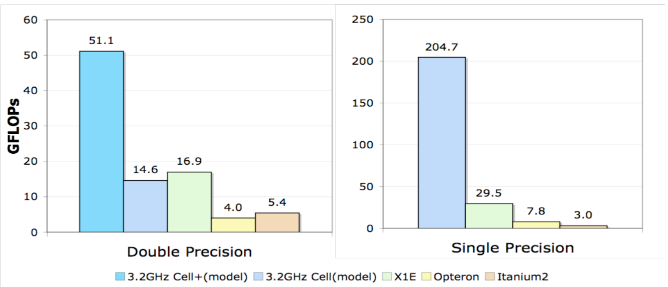
- Alternatives to explicit transposes
- Overlap of communication and computation for the 1D FFT

Dense Matrix-Matrix Multiplication

Introduction

- Straightforward performance modeling
- Matrix stored in column major order
- DMAs pack stanzas together into Local store
- 64^2 cache blocks are sufficiently large to ensure Cell is computationally bound
- Parallelize by assigning each SPE a unique cache block in the destination matrix.

Performance Estimation



Future Work

- Cannon's Algorithm

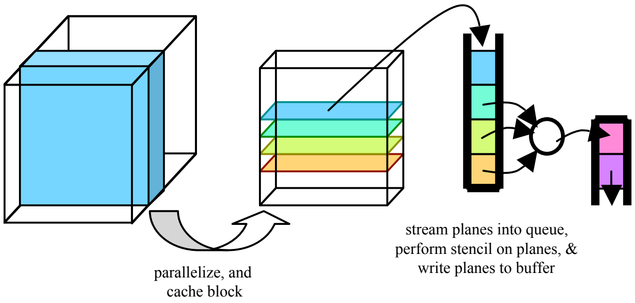
Stencils on Structured Grids

Introduction

- 7 point stencil (Heat Equation, with unweighted neighbors) on a regular grid
- Double
- and Single precision implementations on actual hardware

Implementation

- Cache blocked
- Double buffered planes
- SIMDized with C intrinsics



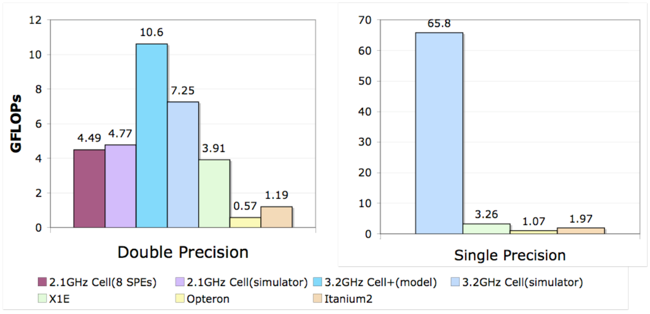
parallelize, and
cache block

stream planes into queue,
perform stencil on planes, &
write planes to buffer

Temporal Blocking using Time Skewing

- Time skewing performs multiple time steps per memory access, and allows the machine to attain near peak algorithmic performance
- One queue exists for each time step ~ 3(TimeSteps + 1) planes
- Planes are read from DRAM once, shuffled from one queue to the next, and written to DRAM once.
- Only used on single precision version but applicable to Cell+

Performance



Future Work

- Complex PDE solvers
- Cell+ double precision time skewed
- Non cubical domains

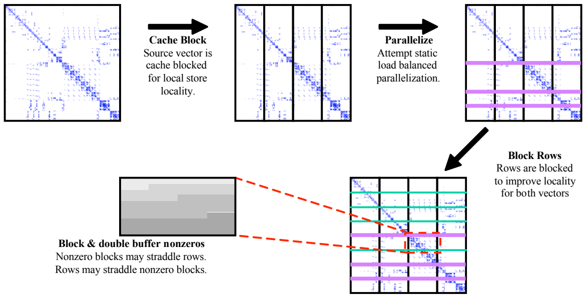
Sparse Matrix-Vector Multiplication

Introduction

- Evaluate Y=**A**X, where **A** is a generic sparse matrix
- Potential load balancing problems based on nonzero structure
- Double precision implementation on actual hardware

Implementation

- Uniprocessor CSR modified so that all row lengths are even (for SIMDization)
- Cache blocked source vector (for temporal locality within each SPE)
- Parallelization based on a cost function of rows and nonzeros per block
- Nonzeros are streamed and double buffered into local store
- Performance model guided actual implementation



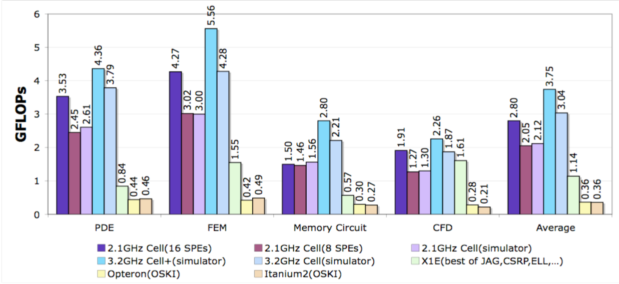
Cache Block
Source vector is
cache blocked
for local store
locality.

Parallelize
Attempt static
load balanced
parallelization.

Block Rows
Rows are blocked
to improve locality
for both vectors

Block & double buffer nonzeros
Nonzero blocks may straddle rows.
Rows may straddle nonzero blocks.

Double Precision Performance



Future Work

- Better load balancing
- Explicitly parallel storage formats
- Segmented scan to amortize branch misprediction and loop overhead
- BCSR to improve SIMDization

http://www.cs.berkeley.edu/~samw/
http://crd.lbl.gov/~oliker/