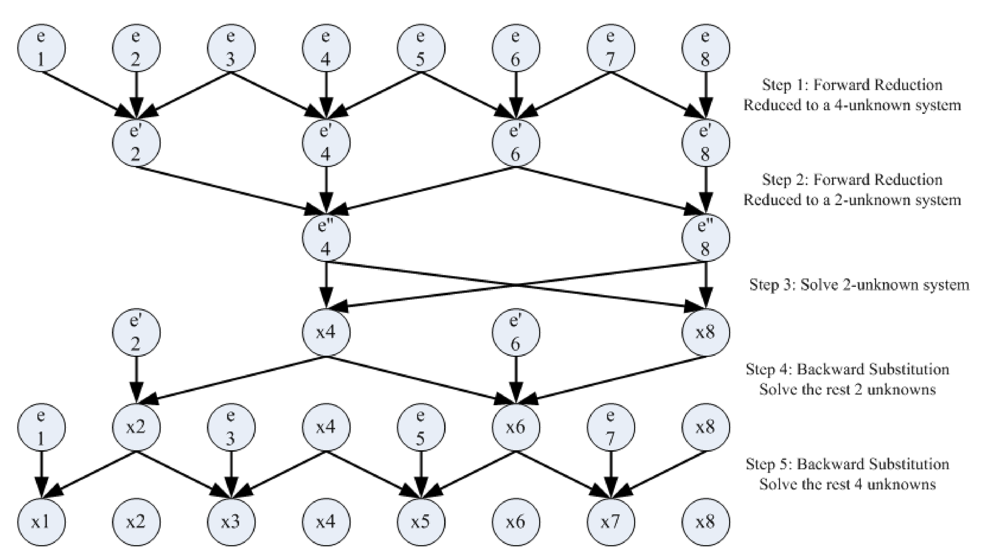
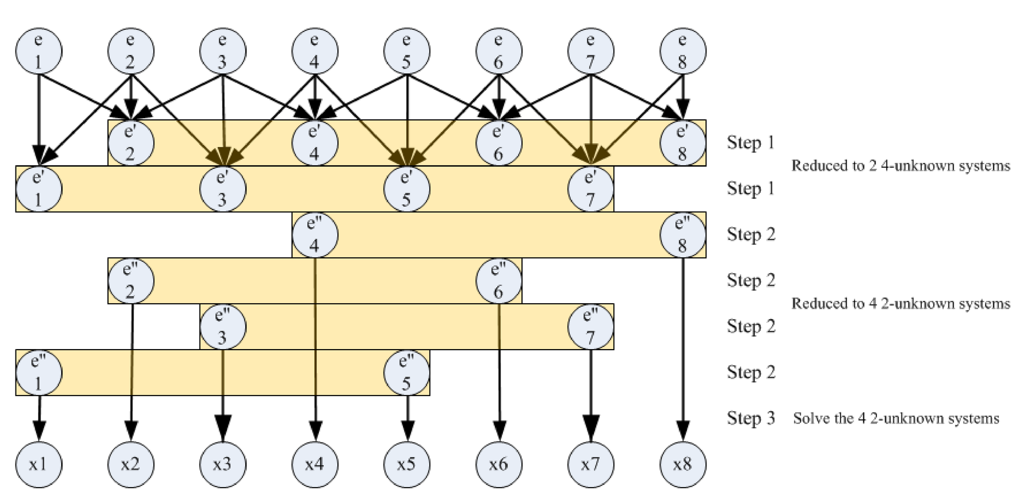
Fast Tridiagonal Solvers on GPU

UCDAVIS
UNIVERSITY OF CALIFORNIA
Yao Zhang
John D. Owens
Jonathan Cohen
NVIDIA

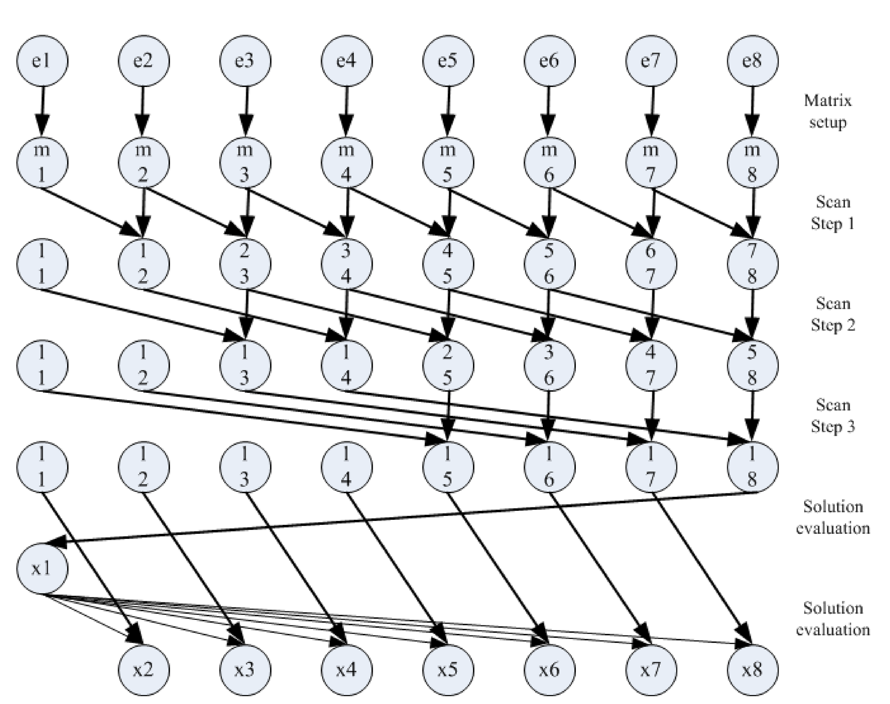
Basic Algorithms



Good: less total work (17n including 3n div)
Bad: more algorithmic steps($2\log_2 n - 1$), bank conflicts
Cyclic Reduction (CR)



Good: fewer algorithmic steps ($\log_2 n$)
Bad: more total work ($12n\log_2 n$ including $2n\log_2 n$ div)
Parallel Cyclic Reduction (PCR)



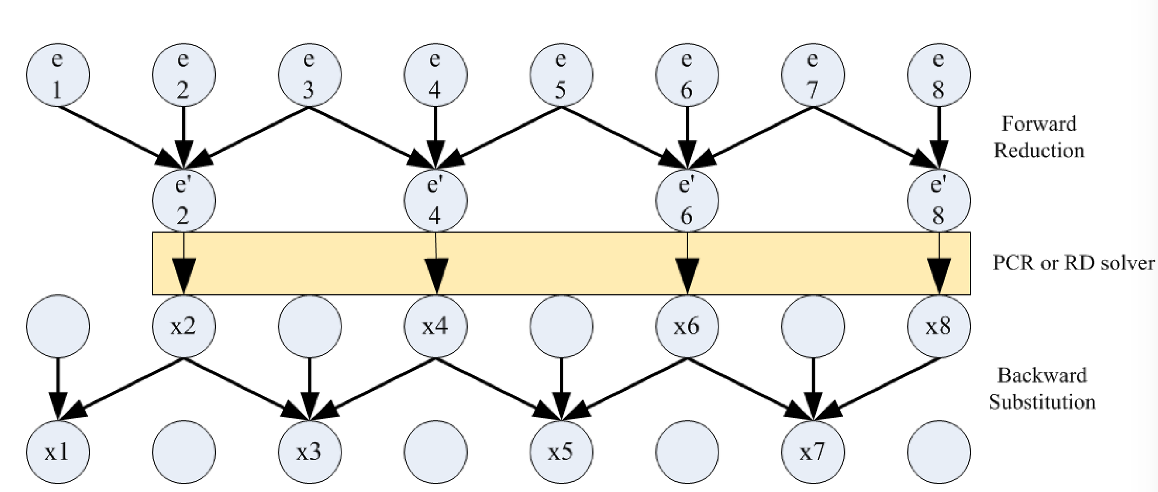
Good: fewer steps ($\log_2 n + 2$)
Bad: more total work ($20n\log_2 n$, no div in major step scan)
Recursive Doubling (RD)

Problem Statement

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ d_n \end{pmatrix}$$

Numerous Applications
Fluid Simulation
Depth-of-fields Blurs
Numerical Ocean Models
Spectral Poisson Solvers
Cubic Spline Approximation
Semi-coarsening for Multi-grid Solvers
Alternating Direction Implicit (ADI) Method
Pre-conditioners for Iterative Linear Solvers

Hybrid Algorithm



$2\log_2 n - \log_2 m - 1$ steps
$17(n - m) + 12m\log_2 m$ arithmetic operations
Fewer bank conflicts
Better parallel hardware utilization (warp size: 32)

Misc.

Parallel Algorithm Overview

Coarse-grained algorithms (multi-core CPU)
-Two-way Gaussian elimination
-Sub-structuring method
Fine-grained algorithms (many-core GPU)
-Cyclic Reduction (CR)
-Parallel Cyclic Reduction (PCR)
-Recursive Doubling (RD)
-Hybrid CR-PCR, CR-RD algorithms

Performance Measure

A manual differential method:
Step 1: comment out the whole code
Step 2: uncomment it incrementally in program order and measure the execution time
Step 3: calculate time difference between neighboring timing results
Tricks:
-Stop loop early at each iteration
-Allocate shared memory to maintain same number of concurrent blocks

Performance Pitfalls

-The higher computation rate and sustained bandwidth, the better. (They may have different algorithm complexity)
-The lower algorithm complexity, the better. (What if there is a considerable amount of control overhead, or bank conflicts, or low hardware utilization)

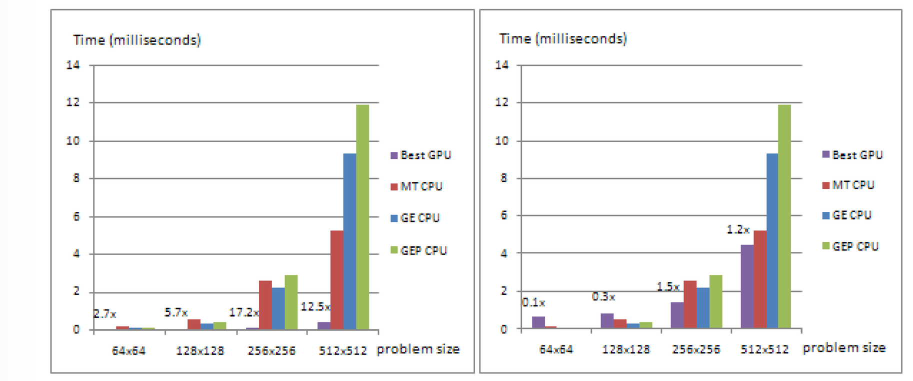
Major Lesson Learned

Performance is determined by a composition of several factors including computation, memory access and control overhead. We show that sometimes these factors can be equally important, and making the right tradeoff between them can lead to the best performance, as in the hybrid solvers. This component-based GPU performance view should replace the traditional bottleneck-based model, in which performance is considered either bandwidth-bound or computation-bound.

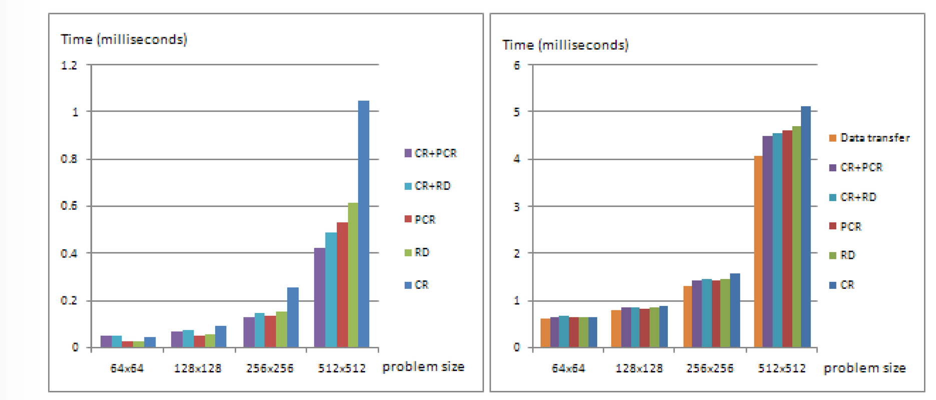
Know Issues and Future Research

-The PCI-E data transfer bottleneck
-Double precision
-Pivoting
-Block tridiagonal system
-Handle large systems that cannot fit into shared memory
-Automatic performance profiling

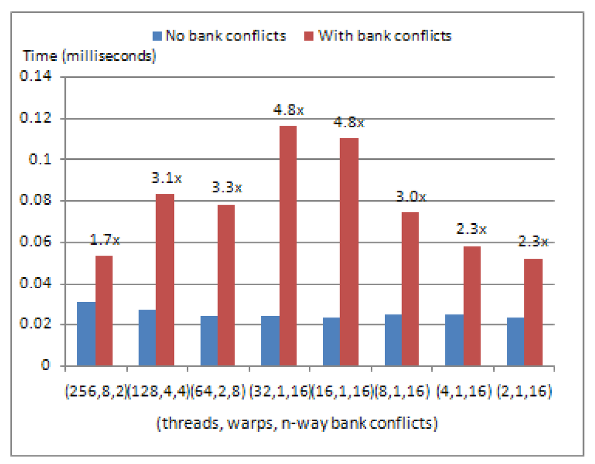
Performance



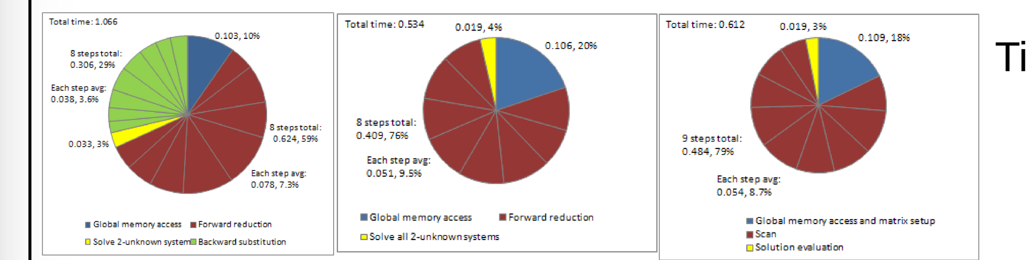
GPU vs. CPU
GTX 280
2.5 GHz Intel Core 2 Q9300 quad-core CPU
CUDA 2.0
CentOS 5
12.5x speedup over multi-threaded CPU solver
28x speedup over LAPACK solver



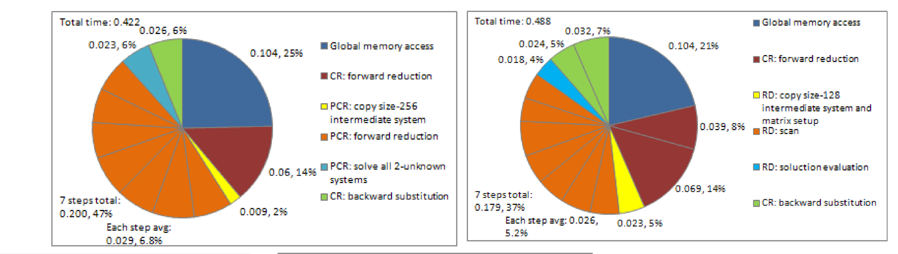
Basic vs. Hybrid
Hybrid solver improves the performance of PCR, RD and CR by 21%, 31% and 61% respectively



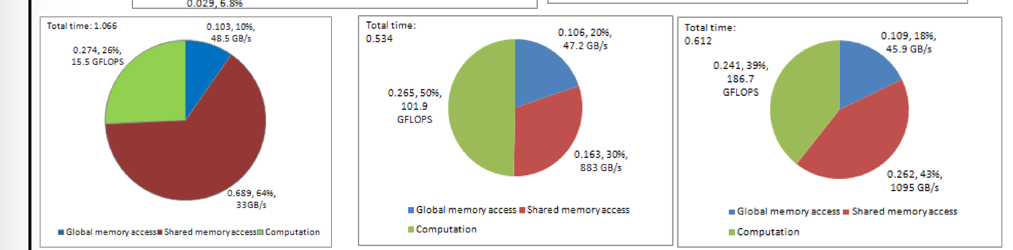
Bank Conflicts of CR
Enforce a shared memory access stride of one



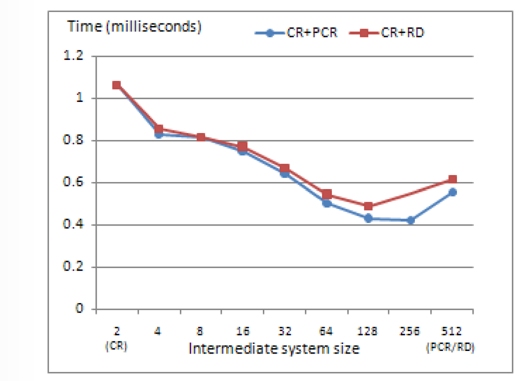
Time Breakdown
CR, PCR, RD



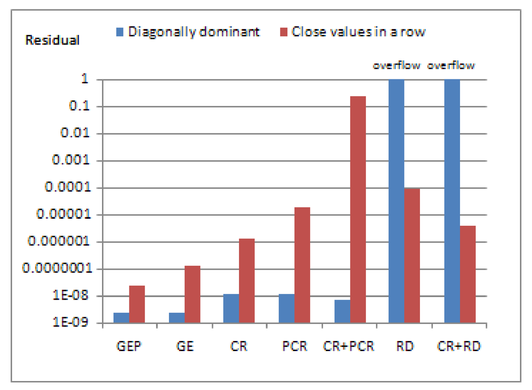
CR-PCR, CR-RD



CR, PCR, RD



Optimal Performance of hybrid solver



Accuracy