IMPLEMENTATION OF THE DEFLATED PRE-CONDITIONED CONJUGATE GRADIENT METHOD APPLIED TO THE POISSON EQUATION RELATED TO BUBBLY FLOW ON THE GPU

LITERATURE STUDY

Rohit Gupta
MSc Computer Engineering
(1542702)

OUTLINE

- Discretization of Bubbly Flow Problem
- Basic Iterative Methods
- Conjugate Gradients
  - Preconditioning
  - Deflation
- Parallelization Techniques
- The Graphical Processing Unit
  - Parallelization of Iterative Methods on the GPU
- Research Questions

PROBLEM STATEMENT

- Two Phase Flow – Navier Stokes Equation
- Linear System Ax=b
- Finite Difference Disrcetization – Neumann Boundary Conditions
- Preconditioning of the Sparse Matrix
- Deflation Techniques (second Level of Preconditioning)
- Optimization of DPCG on the GPU

DISCRETIZATION TECHNIQUES

- Finite Differences
  - Centered, Backward and Forward
  - 1-D and 2-D discretization
- Finite Element
  - Weak Formulation
- Finite Volume
  - Green's Theorem

BASIC ITERATIVE METHODS

$$x_{k+1} = Gx_k + f$$

- Jacobi

  G = $I - D^{-1}A$

- Gauss Seidel

  G = $I - (D - E)^{-1}A$

- Successive Over- Relaxation

  G = $(D - \omega C_1)^{-1}((1 - \omega)D + \omega C_2)$

CONJUGATE GRADIENT

- Projection method on Krylov Subspace

- Coefficient matrix A has to be Symmetric Positive Definite

- Much better than Jacobi and Gauss Siedel in terms of speed of convergence

CONJUGATE GRADIENT ALGORITHM

- Arnoldi Orthogonalization
- Lanczos Method
- Conjugate Gradient Iteration

PRECONDITIONING

Improving the Condition Number K(A)

- Diagonal Preconditioning
- Incomplete Cholesky
- Other Forms (ILU(0), ILU(p), ILUT)

DOMAIN DECOMPOSITION

- Divide the Problem into Smaller domains.
- Solve the Linear System for each domain separately.
- Represent mesh points as a graph then build the matrix.
- Vertex, Edge or Element based Partitioning.
- Block matrix structure emerges.

DEFLATION

- Attempt to treat bad eigenvalues of Preconditioned Matrix

- A different splitting of **A.** PAx=Pb (new system to be solved)

- The deflection vectors approximate the eigenspace of **A**

PARALLELIZATION TECHNIQUES

- Preconditioner setup (internal loop within outer iteration)
- Matrix vector multiplications (storage formats)
- Preconditioning operations (multi-coloring, block-ILU)
- Using BLAS libraries

GRAPHICAL PROCESSING UNIT

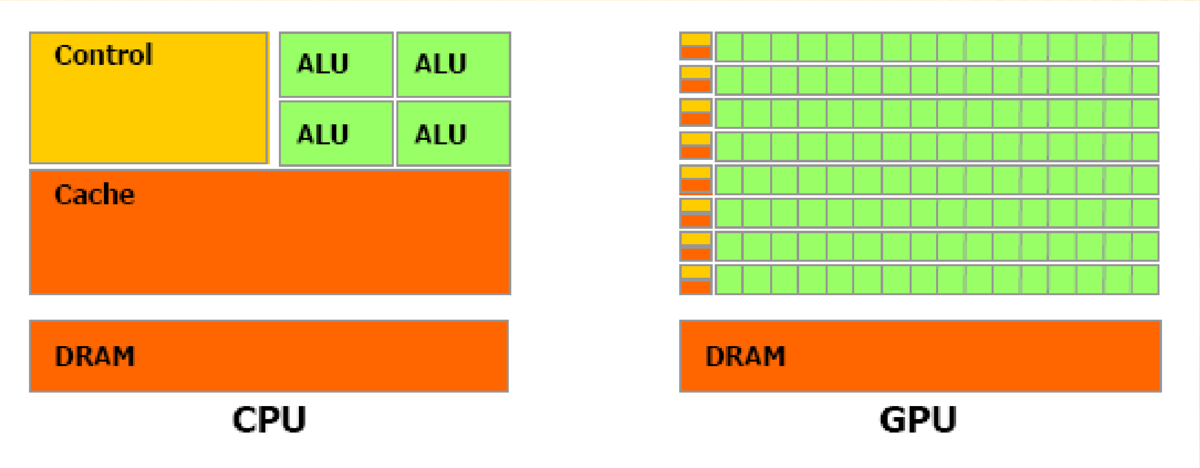
- SIMD Processor

- Hierarchy of Memories

- Capable of delivering hundreds of GFLOPS*

*Conditions Apply

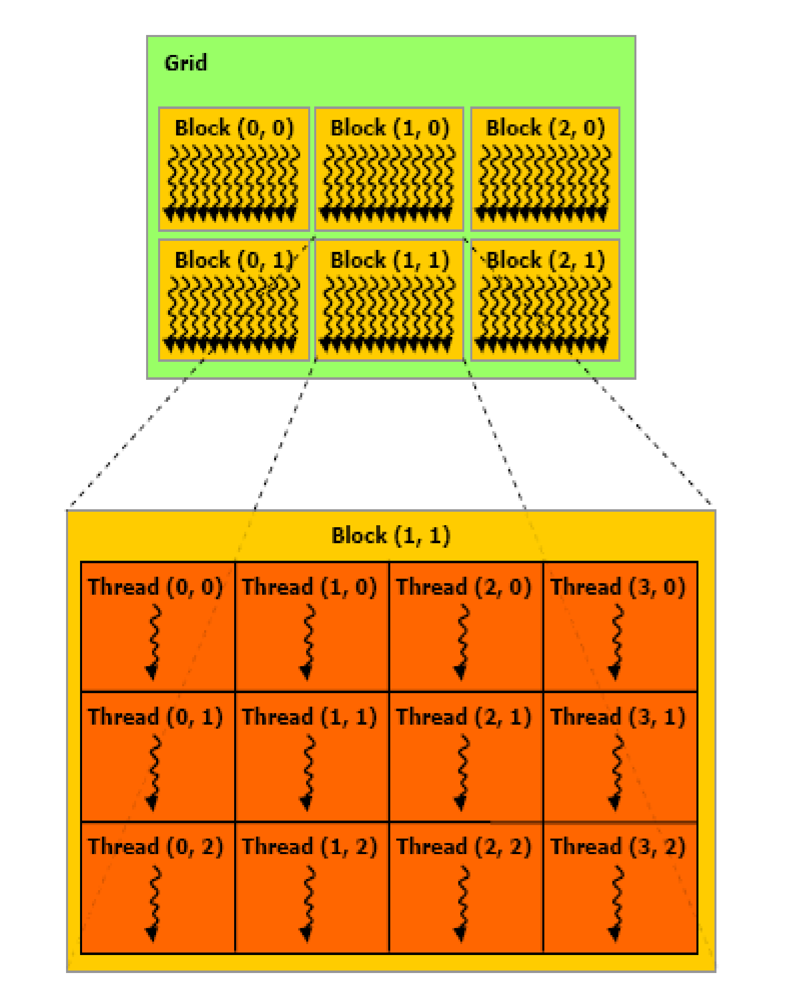
GPU ORIENTATION



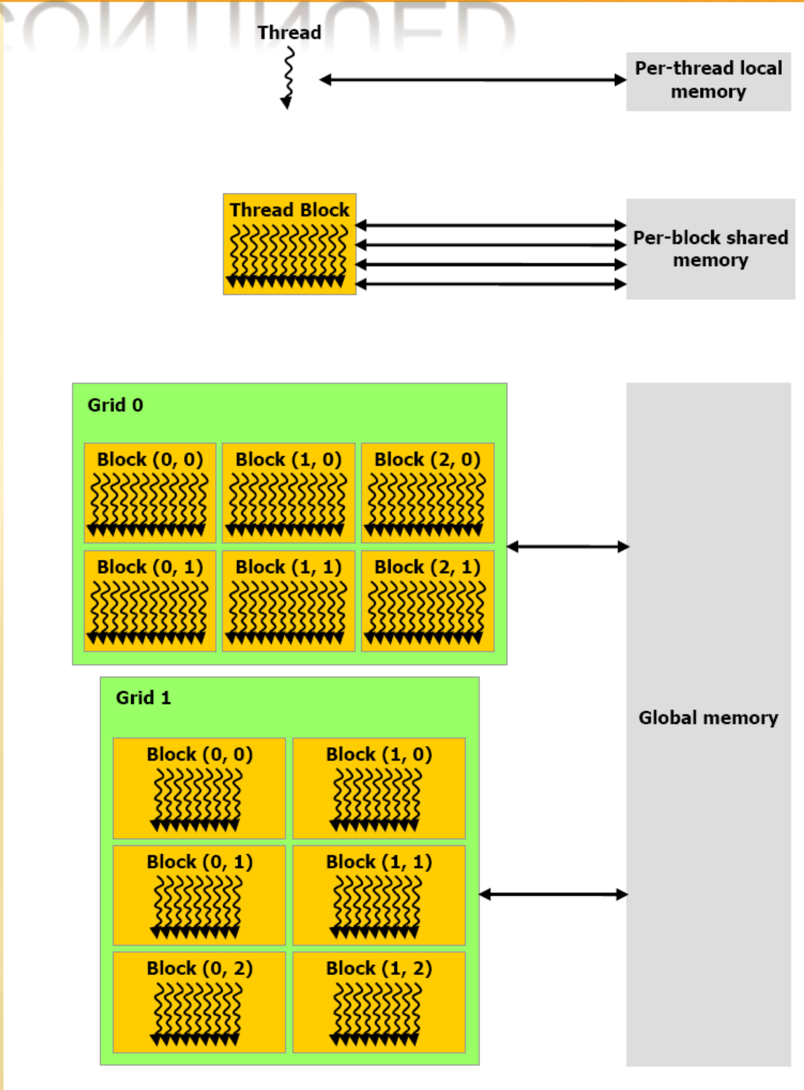
More transistors for Data Processing

Pictures used *without permission* from NVIDIA CUDA Programming Guide v 2.2.1 . Copyrights NVIDIA Corporation

GPU ORIENTATION CONTINUED

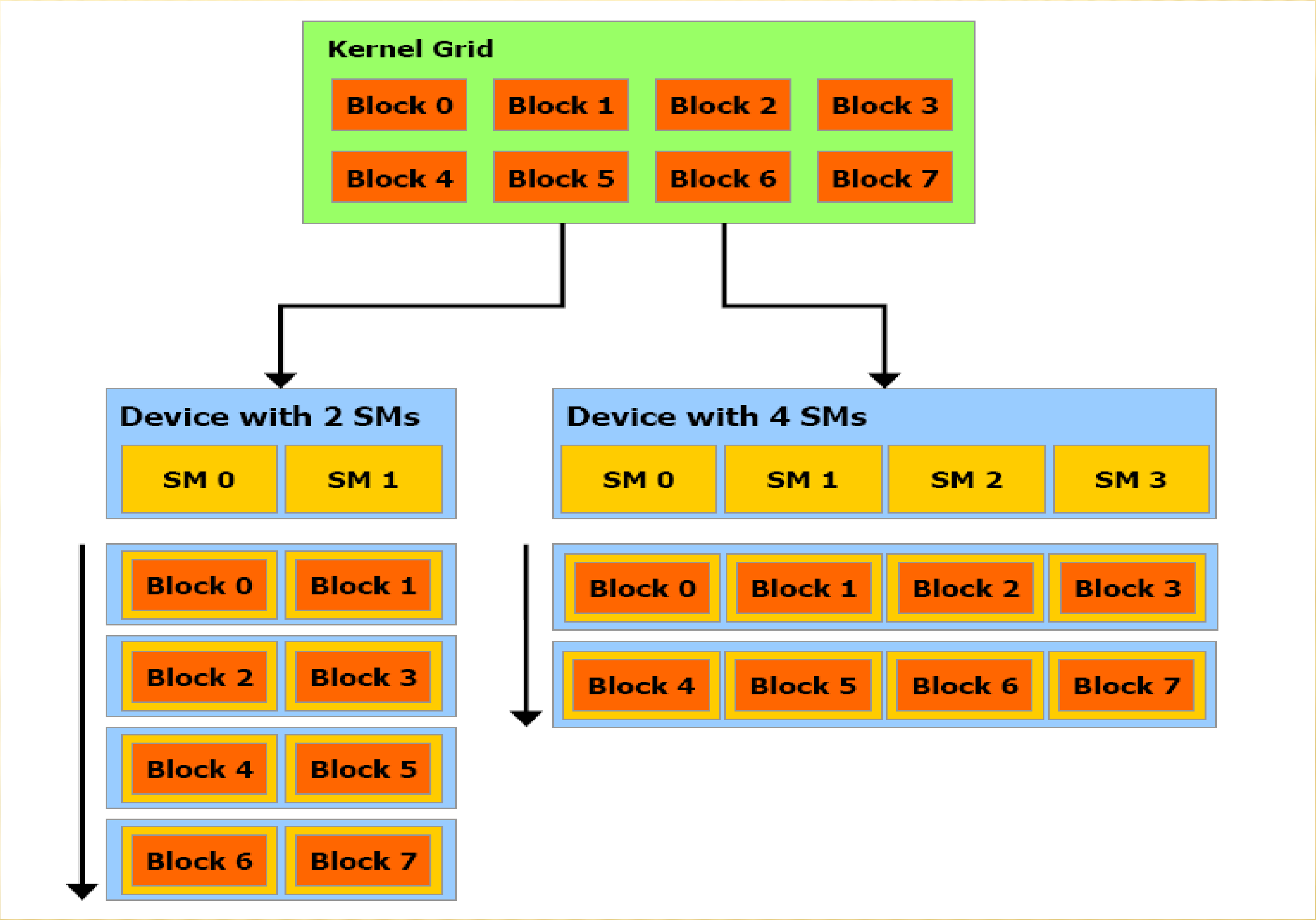


Execution Model

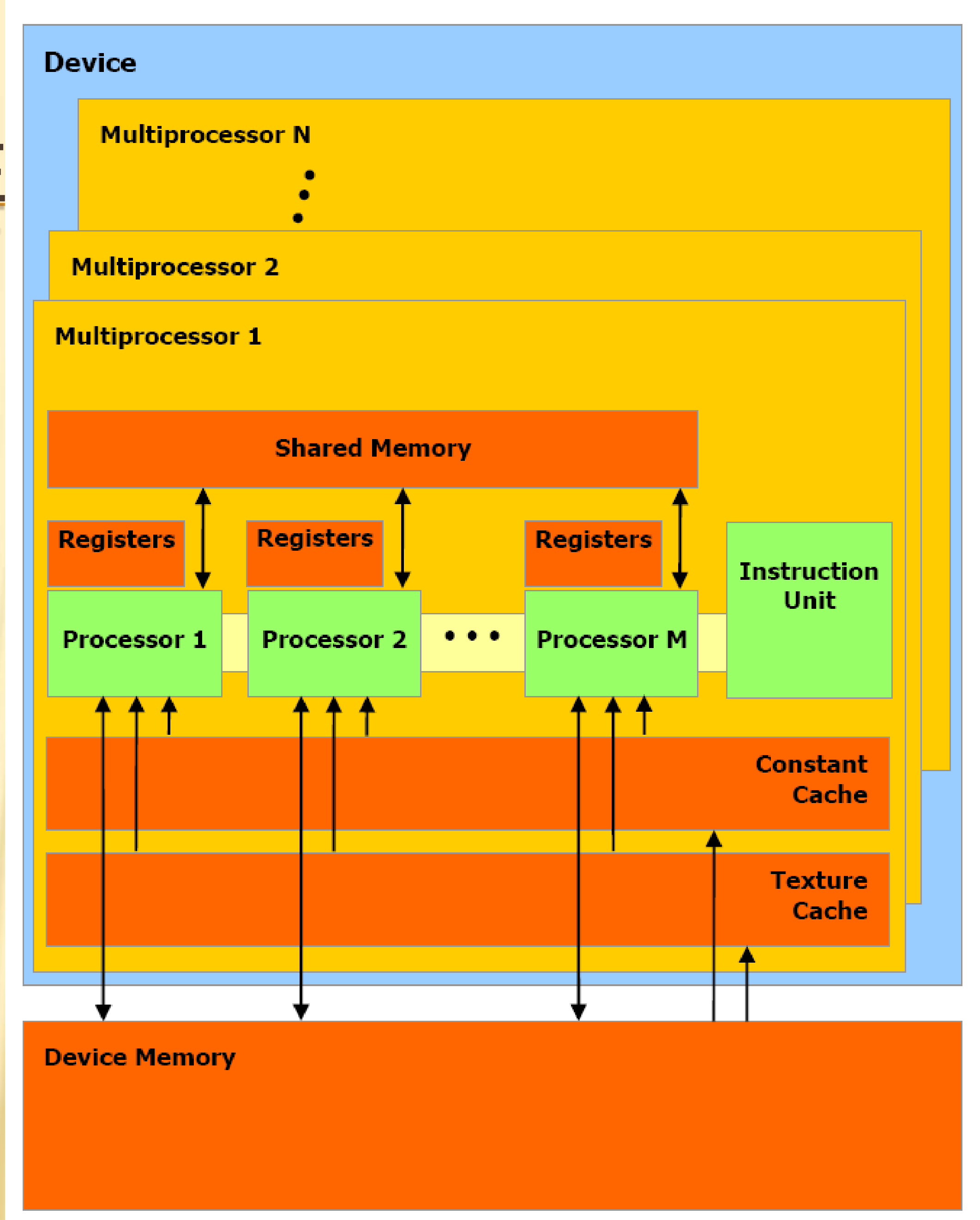


Pictures used *without permission* from NVIDIA CUDA Programming Guide v 2.2.1 . Copyrights NVIDIA Corporation

PERFORMANCE SCALING



THE BIG PICTURE



Pictures used *without permission* from NVIDIA CUDA Programming Guide v 2.2.1 . Copyrights NVIDIA Corporation

COMMUNICATION VS COMPUTATION

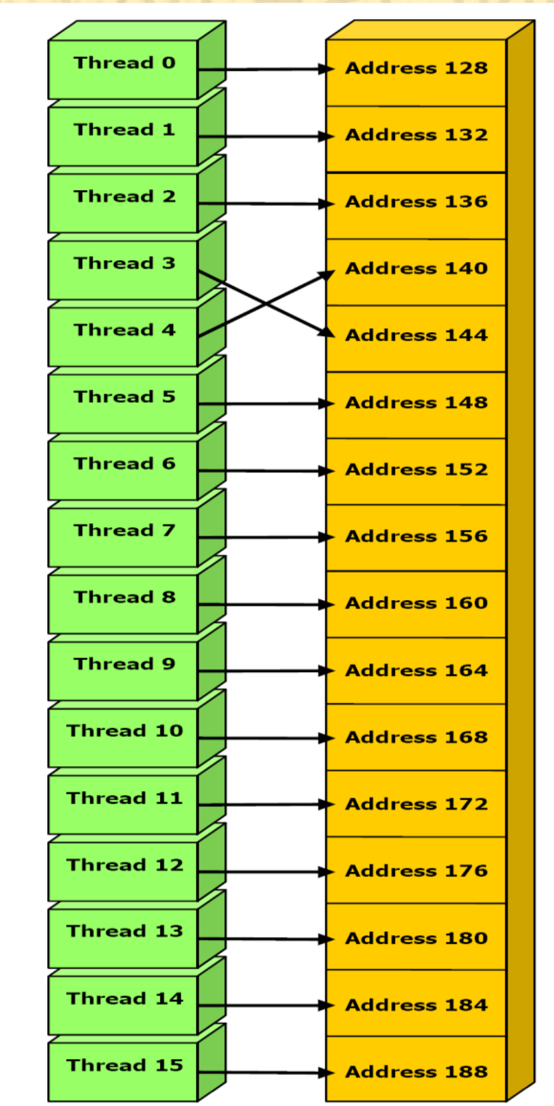
•Parallel Computation Time = T(Execution) + T (Memory Transfers).

•Computation can be reduced by a factor p, Number of Processors.

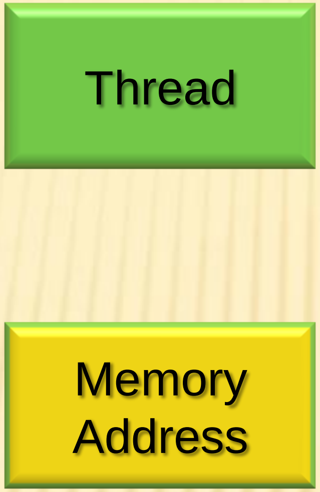
•Communication is problem dependent.

•Important to get more flops per byte (or per float).

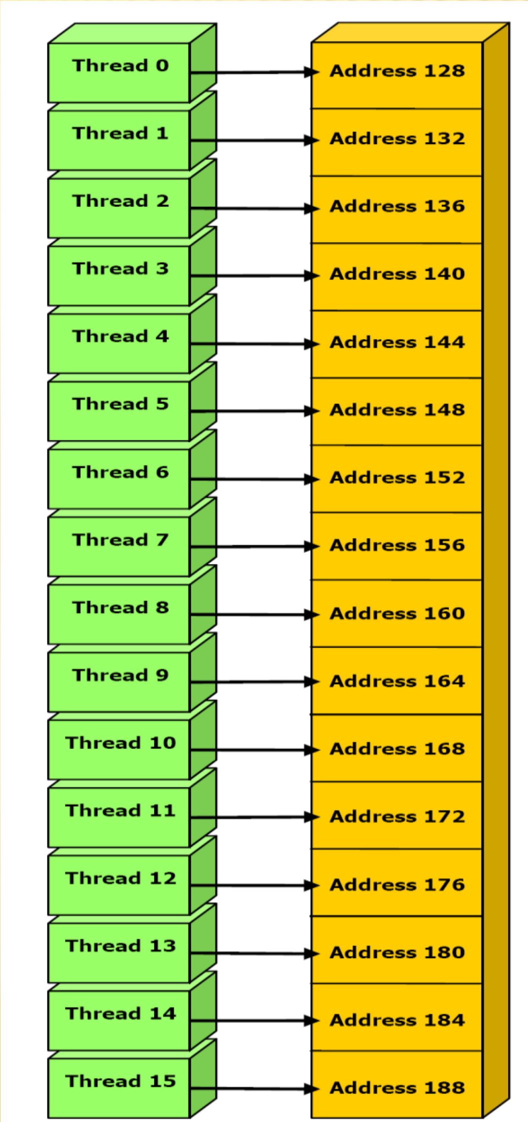
COALESCING



Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
Thread 9
Thread 10
Thread 11
Thread 12
Thread 13
Thread 14
Thread 15
Address 128
Address 132
Address 136
Address 140
Address 144
Address 148
Address 152
Address 156
Address 160
Address 164
Address 168
Address 172
Address 176
Address 180
Address 184
Address 188



Thread
Memory Address



Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
Thread 9
Thread 10
Thread 11
Thread 12
Thread 13
Thread 14
Thread 15
Address 128
Address 132
Address 136
Address 140
Address 144
Address 148
Address 152
Address 156
Address 160
Address 164
Address 168
Address 172
Address 176
Address 180
Address 184
Address 188

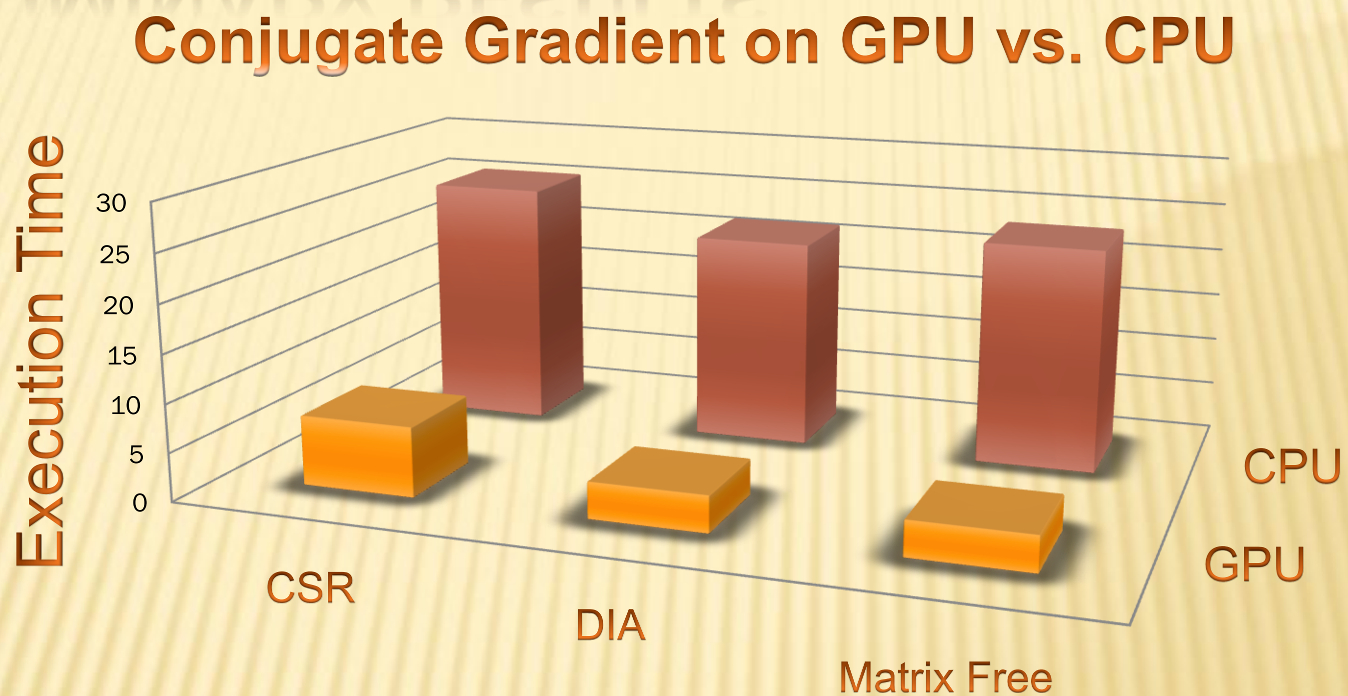
PARALLELIZATION OF ITERATIVE METHODS ON THE GPU

- NVIDIA's SpMV Library
- Work for Dense Matrices (Demmel, Volkov)
- Precision studies (baboulin, et.al)
- Modest Speed-Ups reported with Preconditioned CG

RESEARCH QUESTIONS

- Conjugate Gradient on the GPU
- Preconditioned Conjugate Gradient (different flavors)
- Precision (Mixed and Double)
- Deflation applied on PCG on the GPU
- Multiple GPUs for better performance

PRELIMINARY RESULTS



	CSR	DIA	Matrix Free
GPU	7.286702	3.800385	3.660142
CPU	25.711725	21.787317	23.362423
speedup	3.5x	5.7x	6.4x

Relative Error = 10^{-5} . Iterations on GPU = ~1422. Iterations on CPU=~1250.

Grid Size = 512 X 512

QUESTIONS AND SUGGESTIONS

Pictures used *without permission* from www.cartoonstock.com .

EXTRA SLIDES

Operator Splitting
Dy/dt (K) = L(K)

And L(k) can be split up into L(k1) + L(k2)

$$
\begin{array}{ll}
\text{Dirichlet Condition} & u(x) = \phi(x) \\
\text{Neumann Condition} & \dfrac{\partial u}{\partial \vec{n}}(x) = 0 \\
\text{Cauchy Condition} & \dfrac{\partial u}{\partial \vec{n}}(x) + \alpha(x)u(x) = \gamma(x)
\end{array}
$$

The vector $\vec{n}$ refers to a unit vector that is normal to Γ and directed outwards. For a given vector $\vec{v}$, with components $v1$ and $v2$, the directional derivative $\frac{\partial u}{\partial \vec{v}}$ is defined by

$$
\begin{aligned}
\frac{\partial u}{\partial \vec{v}}(x) &= \lim_{h \to 0} \frac{u(x + h\vec{v}) - u(x)}{h} \\
&= \frac{\partial u}{\partial x_1}(x)v_1 + \frac{\partial u}{\partial x_2}(x)v_2 \\
&= \nabla u.\vec{v}
\end{aligned}
$$

EXTRA SLIDES II

ALGORITHM 1.1: Gram-Schmidt

1. Compute $r_{11} := \|x_1\|_2$. If $r_{11} = 0$ Stop, else compute $q_1 := x_1/r_{11}$.
2. For $j = 2, \ldots, r$ Do:
3. Compute $r_{ij} := (x_j, q_i)$, for $i = 1, 2, \ldots, j-1$
4. $\hat{q} := x_j - \sum_{i=1}^{j-1} r_{ij} q_i$
5. $r_{jj} := \|\hat{q}\|_2$,
6. If $r_{jj} = 0$ then Stop, else $q_j := \hat{q}/r_{jj}$
7. EndDo

ARNOLDI

Arnoldi's procedure is an algorithm for building an orthogonal basis of the Krylov subspace $\mathcal{K}_m$. In exact arithmetic, one variant of the algorithm is as follows:

ALGORITHM 6.1: Arnoldi

1. *Choose a vector* v_1 *of norm 1*
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. Compute $h_{ij} = (Av_j, v_i)$ for $i = 1, 2, \ldots, j$
4. Compute $w_j := Av_j - \sum_{i=1}^{j} h_{ij} v_i$
5. $h_{j+1,j} = \|w_j\|_2$
6. *If* $h_{j+1,j} = 0$ *then Stop*
7. $v_{j+1} = w_j / h_{j+1,j}$
8. *EndDo*

At each step, the algorithm multiplies the previous Arnoldi vector v_j by A and then orthonormalizes the resulting vector w_j against all previous v_i's by a standard Gram-Schmidt procedure. It will stop if the vector w_j computed in line 4 vanishes.

CG Derivation

Gradient algorithm which we now derive. The vector x_{j+1} can be expressed as

$$x_{j+1} = x_j + \alpha_j p_j.$$

Therefore, the residual vectors must satisfy the recurrence

$$r_{j+1} = r_j - \alpha_j A p_j.$$

If the r_j's are to be orthogonal, then it is necessary that $(r_j - \alpha_j A p_j, r_j) = 0$ and as a result

$$\alpha_j = \frac{(r_j, r_j)}{(A p_j, r_j)}.$$

Also, it is known that the next search direction p_{j+1} is a linear combination of r_{j+1} and p_j, and after rescaling the p vectors appropriately, it follows that

$$p_{j+1} = r_{j+1} + \beta_j p_j.$$

Thus, a first consequence of the above relation is that

$$(A p_j, r_j) = (A p_j, p_j - \beta_{j-1} p_{j-1}) = (A p_j, p_j)$$

CG DERIVATION

because Ap_j is orthogonal to p_{j-1}. Then, becomes $\alpha_j = (r_j, r_j)/(Ap_j, p_j)$. In addition, writing that p_{j+1} as defined by ( ) is orthogonal to Ap_j yields

$$\beta_j = -\frac{(r_{j+1}, Ap_j)}{(p_j, Ap_j)}.$$

Note that from

$$Ap_j = -\frac{1}{\alpha_j}(r_{j+1} - r_j)$$

and therefore,

$$\beta_j = \frac{1}{\alpha_j}\frac{(r_{j+1}, (r_{j+1} - r_j))}{(Ap_j, p_j)} = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}.$$

Putting these relations together gives the following algorithm.

DOMAIN DECOMPOSITION