Parallel Tridiagonal Solvers

Fabian Prada, Eric Biagioli

IMPA

Tridiagonal Systems. Motivation

Serial Solution to Tridiagonal Systems

Parallel solution to linear recurrences

Parallel solution to linear recurrences

1. First Order No Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.

Parallel solution to linear recurrences

1. First Order No Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.
2. First Order With Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

Parallel solution to linear recurrences

1. First Order No Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1}$ for $i = 2 \dots n$.
2. First Order With Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \dots n$.
3. Second Order No Constant Term:
   - x_1, x_2 are given.
   - Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \dots n$.

Parallel solution to linear recurrences

1. First Order No Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.
2. First Order With Constant Term:
   - x_1 is given.
   - Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.
3. Second Order No Constant Term:
   - x_1, x_2 are given.
   - Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.
4. Second Order With Constant Term:
   - x_1, x_2 are given.
   - Find $x_i = a_i x_{i-1} + b_i + x_{i-2} + c_i$ for $i = 3 \ldots n$.

First Order No Constant Term

First Order No Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.

First Order No Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.

Solution:

First Order No Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1}$ for $i = 2 \dots n$.

Solution:

$$x_i = a_i a_{i-1} \dots a_2 x_1$$

First Order No Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1}$ for $i = 2 \ldots n$.

Solution:

$$x_i = a_i a_{i-1} \ldots a_2 x_1$$

Scan!!

First Order No Constant Term

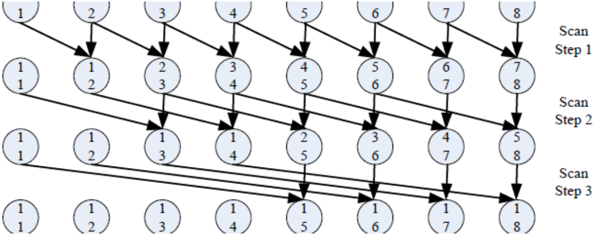
Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1}$ for $i = 2 \dots n$.

Solution:

$$x_i = a_i a_{i-1} \dots a_2 x_1$$

Scan!!



First Order With Constant Term

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

Solution:

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ 1 \end{pmatrix}}_{X_i}$$

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \dots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_1 = \binom{x1}{1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 2 \dots n$.

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_1 = \binom{x1}{1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 2 \ldots n$.

Then

$$X_i = A_i A_{i-1} \ldots A_2 X_1$$

First Order With Constant Term

Problem:

- x_1 is given.
- Find $x_i = a_i x_{i-1} + b_i$ for $i = 2 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_1 = \binom{x1}{1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 2 \ldots n$.

Then

$$X_i = A_i A_{i-1} \ldots A_2 X_1$$

Scan!!

Second Order No Constant Term

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Solution:

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 1 & 0 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \end{pmatrix}}_{X_i}$$

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 1 & 0 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \binom{x_2}{x_1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \ldots n$.

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 1 & 0 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \binom{x_2}{x_1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \ldots n$.

Then

$$X_i = A_i A_{i-1} \ldots A_2 X_1$$

Second Order No Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2}$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i \\ 1 & 0 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \binom{x_2}{x_1}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \ldots n$.

Then

$$X_i = A_i A_{i-1} \ldots A_2 X_1$$

Scan!!

Second Order With Constant Term

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \dots n$.

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \ldots n$.

Solution:

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \dots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}}_{X_i}$$

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \begin{pmatrix} x_2 \\ x_1 \\ 1 \end{pmatrix}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \ldots n$.

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \ldots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \begin{pmatrix} x_2 \\ x_1 \\ 1 \end{pmatrix}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \ldots n$.

Then

$$X_i = A_i A_{i-1} \ldots A_2 X_1$$

Second Order With Constant Term

Problem:

- x_1, x_2 are given.
- Find $x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$ for $i = 3 \dots n$.

Solution:

$$\underbrace{\begin{pmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{pmatrix}}_{X_{i-1}} = \underbrace{\begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}}_{X_i}$$

The recurrence can be written in matrix form as:

- $X_2 = \begin{pmatrix} x_2 \\ x_1 \\ 1 \end{pmatrix}$ given.
- $X_i = A_i X_{i-1}$ for $i = 3 \dots n$.

Then

$$X_i = A_i A_{i-1} \dots A_2 X_1$$

Scan!!

Analysis Parallel Recurrence Computation using Scan

Analysis Parallel Recurrence Computation using Scan

- FN : $x_i = a_i x_{i-1}$.
- FC : $x_i = a_i x_{i-1} + b_i$.
- SN : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2}$.
- SC : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$.

Analysis Parallel Recurrence Computation using Scan

- FN : $x_i = a_i x_{i-1}$.
- FC : $x_i = a_i x_{i-1} + b_i$.
- SN : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2}$.
- SC : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$.

	FN	FC	SN	SC
Total Steps:	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$

Analysis Parallel Recurrence Computation using Scan

- FN : $x_i = a_i x_{i-1}$.
- FC : $x_i = a_i x_{i-1} + b_i$.
- SN : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2}$.
- SC : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$.

	FN	FC	SN	SC
Total Steps:	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$
Active Threads:	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$

Analysis Parallel Recurrence Computation using Scan

- FN : $x_i = a_i x_{i-1}$.
- FC : $x_i = a_i x_{i-1} + b_i$.
- SN : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2}$.
- SC : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$.

	FN	FC	SN	SC
Total Steps:	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$
Active Threads:	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$
Floaps/Thread/Step:	1	3	12	20

Analysis Parallel Recurrence Computation using Scan

- FN : $x_i = a_i x_{i-1}$.
- FC : $x_i = a_i x_{i-1} + b_i$.
- SN : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2}$.
- SC : $\rightarrow x_i = a_i x_{i-1} + b_i x_{i-2} + c_i$.

	FN	FC	SN	SC
Total Steps:	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$	$\log_2(n)$
Active Threads:	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$	$n \rightarrow n/2$
Floaps/Thread/Step:	1	3	12	20
Memory per Thread:	1	2	4	6

Recursive Doubling (LU Form by Stone(1973))

1) Express A in LU form

Recursive Doubling (LU Form by Stone(1973))

1) Express A in LU form

$$A = \begin{pmatrix} 1 & & & & & \\ m_2 & 1 & & & & \\ & m_3 & 1 & & & \\ & & & \ddots & \ddots & \\ & & & m_{n-1} & 1 & \\ & & & & m_n & 1 \end{pmatrix} \begin{pmatrix} u_1 & c_1 & & & & \\ & u_2 & c_2 & & & \\ & & u_3 & c_3 & & \\ & & & \ddots & \ddots & \\ & & & & u_{n-1} & c_{n-1} \\ & & & & & u_n \end{pmatrix}$$

Recursive Doubling (LU Form by Stone(1973))

1) Express A in LU form

$$A = \begin{pmatrix} 1 & & & & & \\ m_2 & 1 & & & & \\ & m_3 & 1 & & & \\ & & & \ddots & \ddots & \\ & & & m_{n-1} & 1 & \\ & & & & m_n & 1 \end{pmatrix} \begin{pmatrix} u_1 & c_1 & & & & \\ & u_2 & c_2 & & & \\ & & u_3 & c_3 & & \\ & & & \ddots & \ddots & \\ & & & & u_{n-1} & c_{n-1} \\ & & & & & u_n \end{pmatrix}$$

- $u_1 = b_1$

Recursive Doubling (LU Form by Stone(1973))

1) Express A in LU form

$$A = \begin{pmatrix} 1 & & & & & \\ m_2 & 1 & & & & \\ & m_3 & 1 & & & \\ & & & \ddots & \ddots & \\ & & & m_{n-1} & 1 & \\ & & & & m_n & 1 \end{pmatrix} \begin{pmatrix} u_1 & c_1 & & & & \\ & u_2 & c_2 & & & \\ & & u_3 & c_3 & & \\ & & & \ddots & \ddots & \\ & & & & u_{n-1} & c_{n-1} \\ & & & & & u_n \end{pmatrix}$$

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \ldots n$

Recursive Doubling (LU Form by Stone(1973))

1) Express A in LU form

$$A = \begin{pmatrix} 1 & & & & & \\ m_2 & 1 & & & & \\ & m_3 & 1 & & & \\ & & & \ddots & \ddots & \\ & & & m_{n-1} & 1 & \\ & & & & m_n & 1 \end{pmatrix} \begin{pmatrix} u_1 & c_1 & & & & \\ & u_2 & c_2 & & & \\ & & u_3 & c_3 & & \\ & & & \ddots & \ddots & \\ & & & & u_{n-1} & c_{n-1} \\ & & & & & u_n \end{pmatrix}$$

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$
- $m_i = \frac{a_i}{u_{i-1}}$ for $i = 2 \dots n$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \ldots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$
- $q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$ for $i = 2 \dots n$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \ldots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$
- $q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$ for $i = 2 \ldots n$

Second Order No Constant Term!!

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \ldots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$
- $q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$ for $i = 2 \ldots n$

Second Order No Constant Term!!

4) Compute u's and m's from q's

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$
- $q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$ for $i = 2 \dots n$

Second Order No Constant Term!!

4) Compute u's and m's from q's

- $u_i = \frac{q_i}{q_{i-1}}$ for $i = 1 \dots n$

Recursive Doubling (LU Form by Stone(1973))

2) Solve the recurrence:

- $u_1 = b_1$
- $u_i = b_i - \frac{a_i c_{i-1}}{u_{i-1}}$ for $i = 2 \dots n$

Trick!!: Express $u_i = \frac{q_i}{q_{i-1}}$.

$$\Rightarrow q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$$

3) Solve the recurrence:

- $q_0 = 1, q_1 = u_1$
- $q_i = b_i q_{i-1} - a_i c_{i-1} q_{i-2}$ for $i = 2 \dots n$

Second Order No Constant Term!!

4) Compute u's and m's from q's

- $u_i = \frac{q_i}{q_{i-1}}$ for $i = 1 \dots n$
- $m_i = \frac{a_i}{u_{i-1}}$ for $i = 2 \dots n$

Recursive Doubling (LU Form by Stone(1973))

The solution to the original system, $Ax = LUx = d$, is calculated in two sequential steps:

Recursive Doubling (LU Form by Stone(1973))

The solution to the original system, $Ax = LUx = d$, is calculated in two sequential steps:

5) Find y such that $Ly = d$

Recursive Doubling (LU Form by Stone(1973))

The solution to the original system, $Ax = LUx = d$, is calculated in two sequential steps:

5) Find y such that $Ly = d$

6) Find x such that $Ux = y$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \ldots n$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \ldots n$

First Order With Constant Term!!

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \dots n$

First Order With Constant Term!!

6)Find x such that $Ux = y$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \dots n$

First Order With Constant Term!!

6)Find x such that $Ux = y$

- $x_n = y_n/u_n$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \dots n$

First Order With Constant Term!!

6)Find x such that $Ux = y$

- $x_n = y_n / u_n$
- $x_i = (-\frac{c_i}{u_i})x_{i-1} + y_i$ for $i = n - 1 \dots 1$

Recursive Doubling (LU Form by Stone(1973))

5)Find y such that $Ly = d$

- $y_1 = d_1$
- $y_i = (-m_i)y_{i-1} + d_i$ for $i = 2 \dots n$

First Order With Constant Term!!

6)Find x such that $Ux = y$

- $x_n = y_n / u_n$
- $x_i = (-\frac{c_i}{u_i})x_{i-1} + y_i$ for $i = n - 1 \dots 1$

First Order With Constant Term!!

Recursive Doubling (LU Form by Stone(1973))

Summary:

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 12

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 12
2. Solve First Order With Constant Term (Ly=x)

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 12
2. Solve First Order With Constant Term (Ly=x)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 3

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 12
2. Solve First Order With Constant Term (Ly=x)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 3
3. Solve First Order With Constant Term (Ux=y)

Recursive Doubling (LU Form by Stone(1973))

Summary:

1. Solve Second Order No Constant Term (A=LU)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 12
2. Solve First Order With Constant Term (Ly=x)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 3
3. Solve First Order With Constant Term (Ux=y)
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 3

Recursive Doubling (Scan Form by Egecioglu(1989))

1)Express the system as Second Order Linear Recurrence With Constant Term:

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

Recursive Doubling (Scan Form by Egecioglu(1989))

1)Express the system as Second Order Linear Recurrence With Constant Term:

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

$$x_{i+1} = -\frac{b_i}{c_i} x_i - \frac{a_i}{c_i} x_{i-1} + \frac{d_i}{c_i} \text{ for } i = 2 \ldots n-1$$

Recursive Doubling (Scan Form by Egecioglu(1989))

1)Express the system as Second Order Linear Recurrence With Constant Term:

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

$$x_{i+1} = -\frac{b_i}{c_i}x_i - \frac{a_i}{c_i}x_{i-1} + \frac{d_i}{c_i} \text{ for } i = 2 \ldots n-1$$

The initial conditions, x_1 and x_2, are missing!!

Recursive Doubling (Scan Form by Egecioglu(1989))

2) Transform to an equivalent system with boundary conditions

Recursive Doubling (Scan Form by Egecioglu(1989))

2) Transform to an equivalent system with boundary conditions

$$\begin{pmatrix} 1 & b_1 & c_1 & & & & & \\ & a_2 & b_2 & c_2 & & & & \\ & & a_3 & b_3 & c_3 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} & \\ & & & & & a_n & b_n & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

s.a. $x_0 = x_{n+1} = 0$

Recursive Doubling (Scan Form by Egecioglu(1989))

2) Transform to an equivalent system with boundary conditions

$$\begin{pmatrix} 1 & b_1 & c_1 & & & & & \\ & a_2 & b_2 & c_2 & & & & \\ & & a_3 & b_3 & c_3 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} & \\ & & & & & a_n & b_n & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

s.a. $x_0 = x_{n+1} = 0$

$$x_{i+1} = -\frac{b_i}{c_i} x_i - \frac{a_i}{c_i} x_{i-1} + \frac{d_i}{c_i} \text{ for } i = 1 \ldots n$$

Recursive Doubling (Scan Form by Egecioglu(1989))

2) Transform to an equivalent system with boundary conditions

$$
\begin{pmatrix}
1 & b_1 & c_1 & & & & & \\
 & a_2 & b_2 & c_2 & & & & \\
 & & a_3 & b_3 & c_3 & & & \\
 & & & \ddots & \ddots & \ddots & & \\
 & & & & a_{n-1} & b_{n-1} & c_{n-1} & \\
 & & & & & a_n & b_n & 1
\end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \end{pmatrix}
=
\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}
$$

s.a. $x_0 = x_{n+1} = 0$

$$x_{i+1} = -\frac{b_i}{c_i} x_i - \frac{a_i}{c_i} x_{i-1} + \frac{d_i}{c_i} \text{ for } i = 1 \dots n$$

Instead of initial conditions we have boundary conditions!!

Recursive Doubling (Scan Form by Egecioglu(1989))

3)Express the recurrence in multiplicative form:

Recursive Doubling (Scan Form by Egecioglu(1989))

3)Express the recurrence in multiplicative form:

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{b_i}{c_i} & -\frac{a_i}{c_i} & -\frac{d_i}{c_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}$$

Recursive Doubling (Scan Form by Egecioglu(1989))

3)Express the recurrence in multiplicative form:

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{b_i}{c_i} & -\frac{a_i}{c_i} & -\frac{d_i}{c_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}$$

Define $C_i = A_i A_{i-1} \dots A_1$, then

Recursive Doubling (Scan Form by Egecioglu(1989))

3)Express the recurrence in multiplicative form:

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{b_i}{c_i} & -\frac{a_i}{c_i} & -\frac{d_i}{c_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix}$$

Define $C_i = A_i A_{i-1} \ldots A_1$, then

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = A_i A_{i-1} \ldots A_1 \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix} = C_i \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

4) Compute $C_i = A_i A_{i-1} \ldots A_1$ for $i = 1 \ldots n$ using **Scan!!**.

Recursive Doubling (Scan Form by Egecioglu(1989))

5) Find x_1

Recursive Doubling (Scan Form by Egecioglu(1989))

5) Find x_1

$$\begin{pmatrix} x_{n+1} \\ x_n \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix}}_{C_n} \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

Recursive Doubling (Scan Form by Egecioglu(1989))

5) Find x_1

$$\begin{pmatrix} x_{n+1} \\ x_n \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix}}_{C_n} \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

Since $x_0 = x_{n+1} = 0$ we get $x_1 = -\frac{c11}{c13}$.

Recursive Doubling (Scan Form by Egecioglu(1989))

5) Find x_1

$$\begin{pmatrix} x_{n+1} \\ x_n \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix}}_{C_n} \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

Since $x_0 = x_{n+1} = 0$ we get $x_1 = -\frac{c11}{c13}$.

6)Find x_i for $i = 2 \dots n$

Recursive Doubling (Scan Form by Egecioglu(1989))

5) Find x_1

$$\begin{pmatrix} x_{n+1} \\ x_n \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix}}_{C_n} \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

Since $x_0 = x_{n+1} = 0$ we get $x_1 = -\frac{c11}{c13}$.

6)Find x_i for $i = 2 \ldots n$

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = C_i \begin{pmatrix} x_1 \\ x_0 \\ 1 \end{pmatrix}$$

Recursive Doubling (Scan Form by Egecioglu(1989))

Summary:

Recursive Doubling (Scan Form by Egecioglu(1989))

Summary:

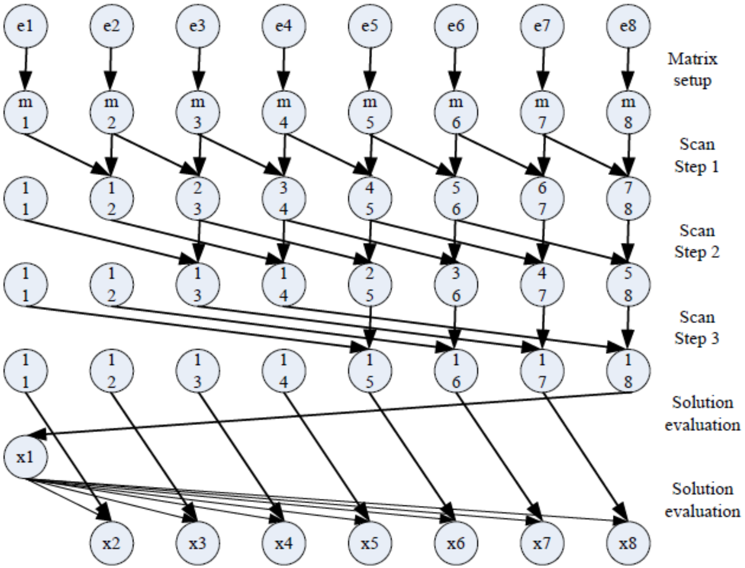
1. Solve Second Order With Constant Term

Recursive Doubling (Scan Form by Egecioglu(1989))

Summary:

1. Solve Second Order With Constant Term
   Total Steps: $\log_2 n$. Floaps/Thread/Step: 20

Recursive Doubling (Scan Form by Egecioglu(1989))



Could Recursive Doubling be improved?

Could Recursive Doubling be improved?

- Yes!!.

Could Recursive Doubling be improved?

- Yes!!.
- Instead of computing all the n vectors $X_1, X_2, X_3, \ldots, X_n$, we just require half of them!!.

Could Recursive Doubling be improved?

- Yes!!.
- Instead of computing all the n vectors $X_1, X_2, X_3, \ldots, X_n$, we just require half of them!!.
- This can be achieved using the same previous scan structure but only updating matrix multiplications in even positions.

Could Recursive Doubling be improved?

- Yes!!.
- Instead of computing all the n vectors $X_1, X_2, X_3, \ldots, X_n$, we just require half of them!!.
- This can be achieved using the same previous scan structure but only updating matrix multiplications in even positions.
- If we assign two threads to compute the respective matrix multiplication at each step, then Floaps/Thread/Step ratio would reduce from 20 to 10!!.

Could Recursive Doubling be improved?

- Yes!!.
- Instead of computing all the n vectors $X_1, X_2, X_3, \ldots, X_n$, we just require half of them!!.
- This can be achieved using the same previous scan structure but only updating matrix multiplications in even positions.
- If we assign two threads to compute the respective matrix multiplication at each step, then Floaps/Thread/Step ratio would reduce from 20 to 10!!.
- This could improve PCR which takes 12 Floaps/Thread/Step, where 2 of such flops are divisions.

Hybrid Algorithms

Building Blocks:

Hybrid Algorithms

Building Blocks:

- CR $\rightarrow$ Work Efficient.

Hybrid Algorithms

Building Blocks:

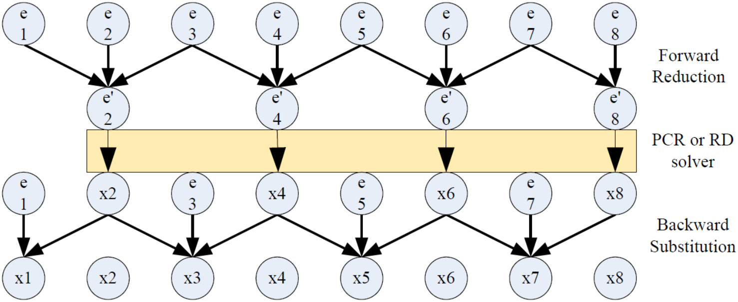
- CR $\rightarrow$ Work Efficient.
- PCR,RD $\rightarrow$ StepEfficient.

Hybrid Algorithms

Building Blocks:

- CR → Work Efficient.
- PCR,RD → StepEfficient.

Hybrid Algorithm Structure:



GPU Implementation

GPU Implementation

- Five arrays are initially allocated in Global Memory: three for the matrix diagonals, one for the right hand side, and one to save the solution.

GPU Implementation

- Five arrays are initially allocated in Global Memory: three for the matrix diagonals, one for the right hand side, and one to save the solution.
- The complete system is transferred to shared memory. This imposed a limit on the size of evaluated systems (up to 512x512).

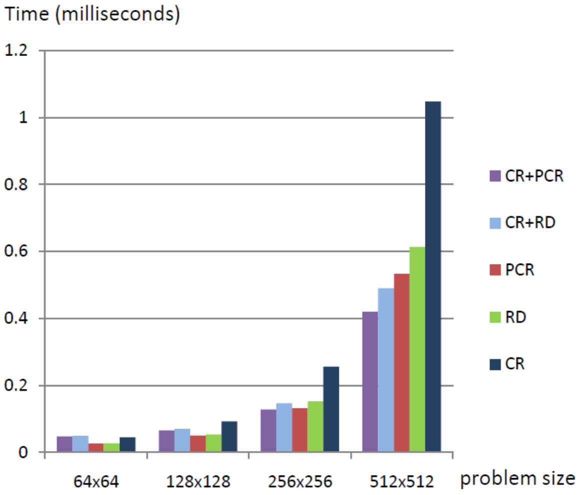
GPU Implementation

- Five arrays are initially allocated in Global Memory: three for the matrix diagonals, one for the right hand side, and one to save the solution.
- The complete system is transferred to shared memory. This imposed a limit on the size of evaluated systems (up to 512x512).
- For CR and RD active threads were contiguous threads to reduce divergence.

GPU Implementation

- Five arrays are initially allocated in Global Memory: three for the matrix diagonals, one for the right hand side, and one to save the solution.
- The complete system is transferred to shared memory. This imposed a limit on the size of evaluated systems (up to 512x512).
- For CR and RD active threads were contiguous threads to reduce divergence.
- Transformations are done in-place to save shared memory. This produces bank conflicts in CR at the last steps of forward reduction and first steps of backward substitution.

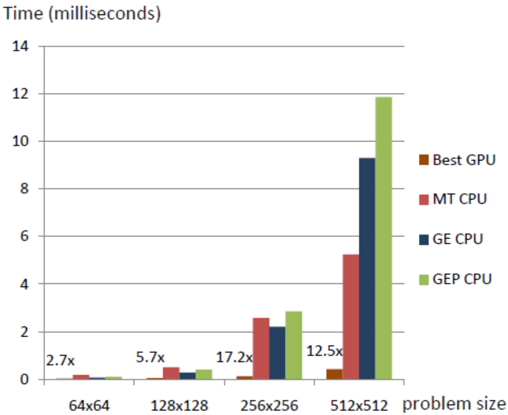
Performance Results:Parallel Algorithms in GPU



Hardware Specifications:

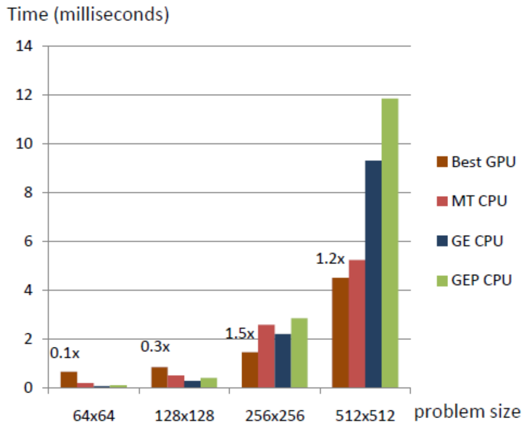
- GPU: GTX 280, 30 SM's, 8 cores per SM, 16kb shared memory. CUDA 2.0.

GPU vs CPU (Ignoring Transfer Time)



- GPU: GTX 280, 30 SM's, 8 cores per SM, 16kb shared memory. CUDA 2.0.
- CPU: 2.5 GHZ Intel Core 2 Q9300 quadcore.

GPU vs CPU (Regarding Transfer Time)

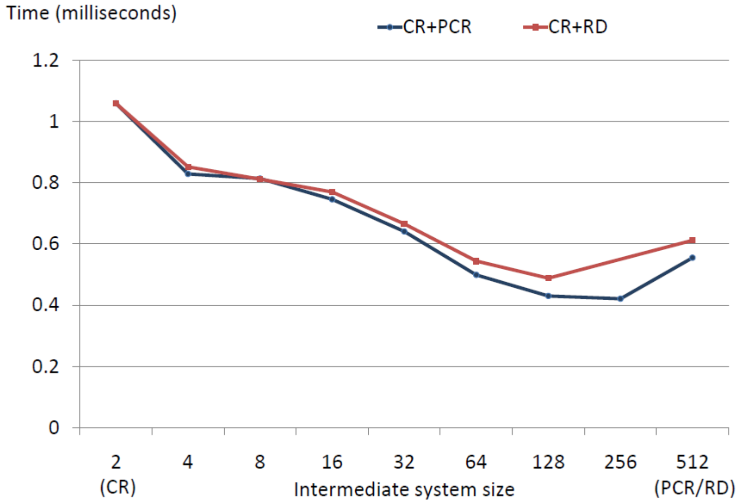


- GPU: GTX 280, 30 SM's, 8 cores per SM, 16kb shared memory. CUDA 2.0.
- CPU: 2.5 GHZ Intel Core 2 Q9300 quadcore.

Performance Analysis

- "We use a differental method to measure the time for each part of the algorithm. We first comment out the whole code and uncomment it incrementally in program order and measure excecution time."
- "To estimate shared memory access time, we replaced shared memory accesses with register accesses, and calculate the shared memory access time as the difference between this program and the original program."

Hybrid Algorithms Performance



Why hybrid algorithms outperform the others?

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Intuition:

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Intuition:

- In a system with 512 variables 8 warps execute the instruction.

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Intuition:

- In a system with 512 variables 8 warps execute the instruction.
- In system with 256 variables 4 warps execute the instruction.

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Intuition:

- In a system with 512 variables 8 warps execute the instruction.
- In system with 256 variables 4 warps execute the instruction.
- Since warp execution is serial in each SM, a system with 256 variables would take half of time per step!!.

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

Intuition:

- In a system with 512 variables 8 warps execute the instruction.
- In system with 256 variables 4 warps execute the instruction.
- Since warp execution is serial in each SM, a system with 256 variables would take half of time per step!!.
- Conclusion: Using CR for the first iterations reduce the per step time required by PCR or RD to solve the intermediate system.

Why hybrid algorithms outperform the others?

They achieve a better trade off between number of steps and amount of work per step.

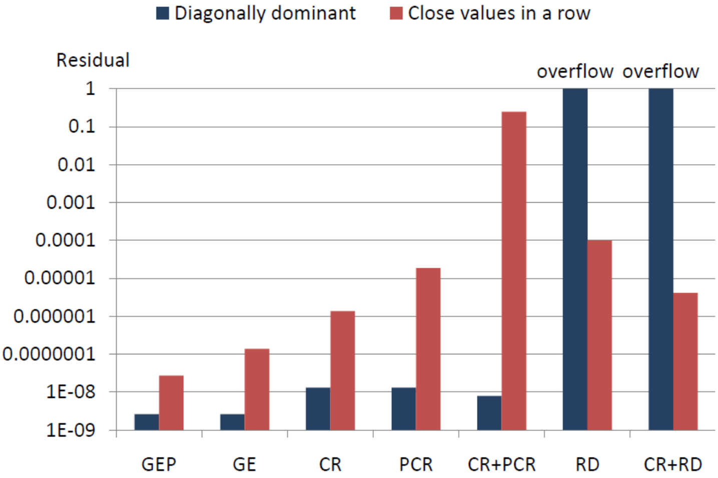
Intuition:

- In a system with 512 variables 8 warps execute the instruction.
- In system with 256 variables 4 warps execute the instruction.
- Since warp execution is serial in each SM, a system with 256 variables would take half of time per step!!.
- Conclusion: Using CR for the first iterations reduce the per step time required by PCR or RD to solve the intermediate system.

PCR and RD are preferred to solve the intermediate system since they require less steps and are free of bank conflicts.

Accuracy Experiments

$||Ax - b||$ error in Tridiagonal Systems



Bibliography

1. Zhang Y.,Cohen J.,Owens J..*Fast Tridiagonal Solvers on th GPU*, 2010.
2. Stone H..*An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations*,1973.
3. Kogge P.. Stone H.,*A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations*, 1973.
4. Hockney R.W.. Jesshope C.R., *Parallel Computers*,1981.
5. Egecioglu O.,Koc C.K., Laub A.J..*A recursive doubling algorithm for solution of tridiagonal systems on hypercube multiprocessors*, 1989.