Hiding Global Reduction Latency in Pipelined Krylov Methods

Parallel Performance and Numerical Accuracy of Communication-Hiding Krylov Subspace Methods for Solving Large Scale Linear Systems

17th GAMM Workshop, University of Cologne, Germany, September 7-8, 2017

University of Antwerp* [BE] & INRIA Bordeaux† [FR]

Siegfried Cools*, J. Cornelis*, W. Vanroose*, E. F. Yetkin†, E. Agullo†, L. Giraud†

Contact: siegfried.cools@uantwerp.be

Universiteit Antwerpen
Introduction

What are we working on?

Classical Krylov subspace method

Washing, drying and ironing in classical ‘Laundry method’

vs.

Pipelined Krylov subspace method

Latency hiding of global drying in pipelined ‘Laundry method’
Work initiated under: 
EXascale Algorithms and Advanced Computational Techniques
http://exa2ct.eu/
EU FP7 Project - Horizon 2020 - Ran from 2013 to 2016

Fellowship 12H4617N
(2016-2019)

Observation: increasing gap between computation and communication costs
- Floating point performance steadily increases
- Network latencies only go down marginally
- Memory latencies decline slowly
- Avoid communication: trade communication for computations
- Hide communication: overlap communication with computations
Krylov subspace methods

General idea

Iteratively improve an approximate solution of linear system \( Ax = b \),

\[
x_i \in x_0 + \mathcal{K}_i(A, r_0) = x_0 + \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{i-1}r_0\}
\]

- minimize an error measure over expanding Krylov subspace \( \mathcal{K}_i(A, r_0) \)
- usually in combination with sparse linear algebra/stencil application
- three building blocks:
  i. dot-product
  ii. SpMV
  iii. axpy

\[E.g.\ Conjugate Gradients\]

\begin{algorithm}
\caption{CG}
\begin{algorithmic}[1]
\Procedure{CG}{$A, b, x_0$}
\State \( r_0 := b - Ax_0; p_0 = r_0 \)
\For{$i = 0, \ldots$}
\State \( s_i := Ap_i \)
\State \( \alpha_i := \frac{r_i \cdot r_i}{s_i \cdot p_i} \)
\State \( x_{i+1} := x_i + \alpha_i p_i \)
\State \( r_{i+1} := r_i - \alpha_i s_i \)
\State \( \beta_{i+1} := \frac{(r_{i+1} \cdot r_{i+1})}{(r_i \cdot r_i)} \)
\State \( p_{i+1} := r_{i+1} + \beta_{i+1} p_i \)
\EndFor
\EndProcedure
\end{algorithmic}
\end{algorithm}
Algorithm 1 CG

1: procedure CG(A, b, x₀)
2: \( r₀ := b - Ax₀; p₀ = r₀ \)
3: for \( i = 0, \ldots \) do
4: \( sᵢ := Apᵢ \)
5: \( \alphaᵢ := (rᵢ, rᵢ) / (sᵢ, pᵢ) \)
6: \( xᵢ₊₁ := xᵢ + \alphaᵢpᵢ \)
7: \( rᵢ₊₁ := rᵢ - \alphaᵢsᵢ \)
8: \( \betaᵢ₊₁ := (rᵢ₊₁, rᵢ₊₁) / (rᵢ, rᵢ) \)
9: \( pᵢ₊₁ := rᵢ₊₁ + \betaᵢ₊₁pᵢ \)
10: end for
11: end procedure

- **Hestenes & Stiefel (1952)**

### i. 3 dot-products
- 2 global reduction phases
- latency dominated
- scales as \( \log₂(\#\text{partitions}) \)

### ii. 1 SpMV
- scales well (minor commun.)
- non-overlapping (sequential to dot-product)

### iii. 3 axpy’s
- recurrences to avoid SpMV’s
- perfectly scalable (no commun.)
Krylov subspace methods

Pipelined CG

Re-organized version of classical CG for improved parallel performance

- equivalent to CG in exact arithmetic
- Communication avoiding: dot-products are grouped in one global reduction phase (line 4+5)
- Communication hiding: overlap global commun. (line 4+5) with computations (SpMV, line 6)
- 3 extra recurrences for \( s_i = Ap_i, w_i = Ar_i, z_i = As_i \) (line 12+13+17)

Algorithm 3 Pipelined CG

1: procedure PIPE-CG(\( A, b, x_0 \))
2: \( r_0 := b - Ax_0; w_0 := Ar_0 \)
3: for \( i = 0, \ldots \) do
4: \( \gamma_i := \langle r_i, r_i \rangle \)
5: \( \delta := \langle w_i, r_i \rangle \)
6: \( q_i := Aw_i \)
7: if \( i > 0 \) then
8: \( \beta_i := \gamma_i / \gamma_{i-1}; \alpha_i := (\delta / \gamma_i - \beta_i / \alpha_{i-1})^{-1} \)
9: else
10: \( \beta_i := 0; \alpha := \gamma_i / \delta \)
11: end if
12: \( z_i := q_i + \beta_i z_{i-1} \)
13: \( s_i := w_i + \beta_i s_{i-1} \)
14: \( p_i := r_i + \beta_i p_{i-1} \)
15: \( x_{i+1} := x_i + \alpha_i p_i \)
16: \( r_{i+1} := r_i - \alpha_i s_i \)
17: \( w_{i+1} := w_i - \alpha_i z_i \)
18: end for
19: end procedure

Ghysels & Vanroose (2013)
Bi-Conjugate Gradients Stabilized
BiCGStab

Algorithm 4 Standard BiCGStab

1: function BICGSTAB(A, b, x₀)
2:   r₀ := b - Ax₀; p₀ := r₀
3:   for i = 0,... do
4:     sᵢ := Apᵢ
5:     compute (r₀, sᵢ)
6:     αᵢ := (r₀, rᵢ) / (r₀, sᵢ)
7:     qᵢ := rᵢ - αᵢsᵢ
8:     yᵢ := Aqᵢ
9:     compute (qᵢ, yᵢ); (yᵢ, yᵢ)
10:    ωᵢ := (qᵢ, yᵢ) / (yᵢ, yᵢ)
11:   xᵢ+1 := xᵢ + αᵢpᵢ + ωᵢqᵢ
12:   rᵢ+1 := qᵢ - ωᵢyᵢ
13:   compute (r₀, rᵢ+1)
14:   βᵢ := (αᵢ/ωᵢ)(r₀, rᵢ+1) / (r₀, rᵢ)
15:   pᵢ+1 := rᵢ+1 + βᵢ(pᵢ - ωᵢsᵢ)
16: end for
17: end function

Traditional BiCGStab:
(non-preconditioned)

Global communication
► 3 global reduction phases

Local communication
► 2 non-overlapping SpMVs

No communication
► 4 recurrences

General two-step framework for deriving pipelined Krylov methods:

Step 1. Avoiding communication: merge global reductions

Step 2. Hiding communication: overlap SpMVs & global reductions
Bi-Conjugate Gradients Stabilized

Step 1. Avoiding communication

(a) **Identify** two global comm. phases for merger (lines 5-6 & 13-14)

(b) **Rewrite** SpMV as recurrence:

\[ s_i = Ap_i = w_i + \beta_{i-1} (s_{i-1} - \omega_{i-1} z_{i-1}) , \]

**define** \( w_i := Ar_i, z_i := As_i \) and note that \( y_i := w_i - \alpha_i z_i \)

(c) **Rewrite** dot-product using (b):

\[ (r_0, s_i) = (r_0, w_i + \beta_{i-1} (s_{i-1} - \omega_{i-1} z_{i-1})) , \]

independent of interlying variables

(d) **Move** dot-product (lines 5-6) upward and merge with existing global comm. phase (lines 13-14)
Bi-Conjugate Gradients Stabilized
CA-BiCGStab

Algorithm 5 Communication avoiding BiCGStab

1: function CA-BiCGSTAB(A, b, x₀)
2: \[ r₀ := b - Ax₀; \quad w₀ := Ar₀; \quad α₀ := (r₀, r₀) / (r₀, w₀); \quad β₋₁ := 0 \]
3: for \( i = 0, \ldots \) do
4: \[ pᵢ := rᵢ + β₋₁ (p₋₁ - ω₋₁ s₋₁) \]
5: \[ sᵢ := wᵢ + β₋₁ (s₋₁ - ω₋₁ z₋₁) \]
6: \[ zᵢ := Asᵢ \]
7: \[ qᵢ := rᵢ - αᵢ sᵢ \]
8: \[ yᵢ := wᵢ - αᵢ zᵢ \]
9: compute \((qᵢ, yᵢ) / (yᵢ, yᵢ)\)
10: \[ ωᵢ := (qᵢ, yᵢ) \]
11: \[ xᵢ₊₁ := xᵢ + αᵢ pᵢ + ωᵢ qᵢ \]
12: \[ rᵢ₊₁ := qᵢ - ωᵢ yᵢ \]
13: \[ wᵢ₊₁ := Arᵢ₊₁ \]
14: compute \((r₀, rᵢ₊₁) ; (r₀, wᵢ₊₁) ; (r₀, sᵢ) ; (r₀, zᵢ)\)
15: \[ βᵢ := (αᵢ / ωᵢ) (r₀, rᵢ₊₁) / (r₀, rᵢ) \]
16: \[ αᵢ₊₁ := (r₀, rᵢ₊₁) / ((r₀, wᵢ₊₁) + βᵢ (r₀, sᵢ) - βᵢ ωᵢ (r₀, zᵢ)) \]
17: end for
18: end function

Communication-avoiding BiCGStab:
(non-preconditioned)

Global communication
- 2 global red. phases (vs. 3)

Local communication
- 2 non-overlapping SpMVs

No communication
- 6 recurrences (vs. 4)

Status after Step 1:
number of global comm. phases reduced from 3 to 2, at the cost of 2 axpys

Note: further reduction from 2 to 1 global comm. phase possible, but not recommended (see later).
Bi-Conjugate Gradients Stabilized

Step 2. Hiding communication

(a) **Identify** SpMV / global reduction pairs (lines 6 & 9 and 13 & 14)

(b) **Rewrite** SpMVs as recurrences:

\[
\begin{align*}
  z_i & := As_i = t_i + \beta_{i-1} (z_{i-1} - \omega_{i-1} v_{i-1}) \\
  w_{i+1} & := Ar_{i+1} = y_i - \omega_i (t_i - \alpha_i v_i),
\end{align*}
\]

**define** \( t_i := Aw_i, v_i := Az_i \)

(c) **Check** SpMV / global reduction pairwise dependencies:

- line 9 independent of \( v_i \)? **yes**
- line 14 independ. of \( t_{i+1} \)? **yes**

(d) **Insert** new SpMVs **below** corresponding global comm. phases

---

**Algorithm 5** Communication avoiding BiCGStab

1: function CA-BICGSTAB(\( A, b, x_0 \))
2: \( r_0 \) := \( b - Ax_0 \); \( w_0 \) := \( Ar_0 \); \( \alpha_0 \) := \( r_0, r_0 \) / \( r_0, w_0 \); \( \beta_{-1} \) := 0
3: for \( i = 0, \ldots \) do
4: \( p_i \) := \( r_i + \beta_{i-1} (p_{i-1} - \omega_{i-1} s_{i-1}) \)
5: \( s_i \) := \( w_i + \beta_{i-1} (s_{i-1} - \omega_{i-1} z_{i-1}) \)
6: \( z_i \) := \( As_i \)
7: \( q_i \) := \( r_i - \alpha_i s_i \)
8: \( y_i \) := \( w_i - \alpha_i z_i \)
9: compute \((q_i, y_i) ; (y_i, y_i)\)
10: \( \omega_i \) := \((q_i, y_i) / (y_i, y_i)\)
11: \( x_{i+1} \) := \( x_i + \alpha_i p_i + \omega_i q_i \)
12: \( r_{i+1} \) := \( q_i - \omega_i y_i \)
13: \( w_{i+1} \) := \( Ar_{i+1} \)
14: compute \((r_0, r_{i+1}) ; (r_0, w_{i+1}) ; (r_0, s_i) ; (r_0, z_i)\)
15: \( \beta_i \) := \((\alpha_i / \omega_i) (r_0, r_{i+1}) / (r_0, r_i)\)
16: \( \alpha_{i+1} \) := \((r_0, r_{i+1}) / (r_0, w_{i+1}) + \beta_i (r_0, s_i) - \beta_i \omega_i (r_0, z_i)\)
17: end for
18: end function
Bi-Conjugate Gradients Stabilized
pipe-BiCGStab

**Algorithm 6** Pipelined BiCGStab

1: function PIPE-BICGSTAB($A$, $b$, $x_0$)
2:     $r_0 := b - Ax_0$; $w_0 := Ar_0$; $t_0 := Aw_0$;
3:      for $i = 0 \ldots$ do
4:         $p_i := r_i + \beta_{i-1} (p_{i-1} - \omega_{i-1}s_{i-1})$
5:         $s_i := w_i + \beta_{i-1} (s_{i-1} - \omega_{i-1}z_{i-1})$
6:         $z_i := t_i + \beta_{i-1} (z_{i-1} - \omega_{i-1}v_{i-1})$
7:         $q_i := r_i - \alpha_is_i$
8:         $y_i := w_i - \alpha_iz_i$
9:         **compute** ($q_i, y_i$) ; ($y_i, y_i$)
10:        $\omega_i := (q_i, y_i) / (y_i, y_i)$
11:       **overlap** $v_i := Az_i$
12:      $x_{i+1} := x_i + \alpha_ip_i + \omega_iq_i$
13:      $r_{i+1} := q_i - \omega_iy_i$
14:      $w_{i+1} := y_i - \omega_i(t_i - \alpha_iw_i)$
15:      **compute** ($r_0, r_{i+1}$) ; ($r_0, w_{i+1}$) ; ($r_0, s_i$) ; ($r_0, z_i$)
16:      $\beta_i := (\alpha_i/\omega_i) (r_0, r_{i+1}) / (r_0, r_i)$
17:      $\alpha_{i+1} := (r_0, r_{i+1}) / ((r_0, w_{i+1}) + \beta_i (r_0, s_i) - \beta_i \omega_i (r_0, z_i))$
18:      **overlap** $t_{i+1} := Aw_{i+1}$
19:    end for
20: end function

**Pipelined BiCGStab:**
(non-preconditioned)

**Global communication**
- 2 global red. phases (vs. 3)

**Local communication**
- 2 overlapping SpMVs

**No communication**
- 8 recurrences (vs. 4)

---

**Status after Step 2:**
both global comm. phases are overlapped with SpMV computations (‘hidden’),
at the cost of 4 additional axpys compared to standard BiCGStab
Like for any pipelined method, including a preconditioner is - in theory - easy.

**Pipelined BiCGStab:**
(preconditioned)

- **Global communication**
  - 2 global red. phases (vs. 3)

- **Local communication**
  - 2 overlapping Prec + SpMVs

- **No communication**
  - 11 recurrences (vs. 4)
Bi-Conjugate Gradients Stabilized

Pipe-BiCGStab vs. IBiCGStab vs. s-step CA-BiCGStab

<table>
<thead>
<tr>
<th></th>
<th>GLRED</th>
<th>SPMV</th>
<th>Flops (APXY + DOT-PROD)</th>
<th>Time (GLRED + SPMV)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGStab</td>
<td>3</td>
<td>2</td>
<td>20</td>
<td>3 GLRED + 2 SPMV</td>
<td>7</td>
</tr>
<tr>
<td>IBiCGStab</td>
<td>1</td>
<td>2</td>
<td>30</td>
<td>1 GLRED + 2 SPMV</td>
<td>10</td>
</tr>
<tr>
<td>p-BiCGStab</td>
<td>2</td>
<td>2*</td>
<td>38</td>
<td>2 max(GLRED, SPMV)</td>
<td>11</td>
</tr>
<tr>
<td>s-step CA-BiCGStab</td>
<td>1/s</td>
<td>4</td>
<td>32s+45</td>
<td>1/s GLRED + 4 SPMV</td>
<td>4s+5</td>
</tr>
</tbody>
</table>

Theoretical speed-up factors over classical BiCGStab:

- Pipe-BiCGStab
  - C., Vanroose, 2017
  - \( \times 2.5 \)

- IBiCGStab
  - Yang & Brent, 2002
  - \( \times 1.67 \)

- CA-BiCGStab
  - Carson, 2015
  - \( \times 1.25 \)

\[
\text{if } \text{time(GLRED)} \approx \text{time(SPMV)} \quad \times 2.5 \\
\text{if } \text{time(GLRED)} \gg \text{time(SPMV)} \quad \times 1.5
\]

Is algorithm with 1 GLRED overlapped with all SPMVs possible? Yes; however...

- no. axyps is much larger → algorithm robustness decreases (rounding errors)
- one extra SpMV required
Krylov subspace methods

Other pipelined Krylov methods

- **Pipelined GMRES**
  \[ V_{i-\ell+1} = [v_0, v_1, \ldots, v_{i-\ell}] \]
  \[ Z_{i+1} = [z_0, z_1, \ldots, z_{i-\ell}, z_{i-\ell+1}, \ldots, z_i] \]
  - compute \( \ell \) new basis vectors for Krylov subspace (SpMVs) during global communication (dot-products).
  - deeper/variable pipelining possible: \( p(\ell) \)-GMRES

- **Pipelined CG**
  - deeper pipelining also possible: \( p(\ell) \)-CG

- **Pipelined BiCGStab**
  - non-symmetric operators

- **Preconditioned** pipelined variants are available
  - prec-pipe-CG
  - prec-pipe-GMRES
  - prec-pipe-BiCGStab

- **Augmented and deflated** pipelined methods are available

Ghysels et al. (2012)

Ghysels et al. (2013)

C., Vanroose (2017)

see talk by W. Vanroose
Motivation: pipe-CG loses max. attainable accuracy compared to classical CG

- Model problem: small 2D Laplacian with 2,500 unknowns
- Loss of attainable accuracy is more pronounced for larger systems/longer pipelines
Motivation: *pipe-CG loses max. attainable accuracy compared to classical CG*

- Model problem: medium-sized 2D Laplacian with 250,000 unknowns
- Loss of attainable accuracy is more pronounced for larger systems/longer pipelines
Conjugate Gradients

Rounding error propagation in CG

Rounding errors due to recursive definition of residual (and auxiliary variables)

\[
\begin{align*}
\bar{p}_{i+1} &= \bar{u}_{i+1} + \beta_{i+1} \bar{p}_i + \delta^p_i, \\
\bar{x}_{i+1} &= \bar{x}_i + \alpha_i \bar{p}_i + \delta^x_i, \\
\bar{r}_{i+1} &= \bar{r}_i - \bar{x}_i A \bar{p}_i + \delta^r_i,
\end{align*}
\]

which deviates from the true residual \( b - A\bar{x}_i \) in finite precision arithmetics

\[
\begin{align*}
f_{i+1} &= (b - A\bar{x}_{i+1}) - \bar{r}_{i+1} \\
&= b - A(\bar{x}_i + \alpha_i \bar{p}_i + \delta^x_i) - (\bar{r}_i - \bar{x}_i A \bar{p}_i + \delta^r_i) \\
&= f_i - A\delta^x_i - \delta^r_i.
\end{align*}
\]

After \( i \) iterations:

\[
f_{i+1} = f_0 - \sum_{j=0}^{i} (A\delta^x_j + \delta^r_j).
\]

Only accumulation of local rounding errors in classical CG, no amplification.

**Observation:** rounding error propagation in pipe-CG may be much more dramatic due to additional recurrence relations that all induce rounding errors.

\[
\begin{align*}
\bar{x}_{i+1} &= \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \delta^x_i, \\
\bar{r}_{i+1} &= \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \delta^r_i, \\
\bar{p}_i &= \bar{u}_i + \bar{\beta}_i \bar{p}_{i-1} + \delta^p_i, \\
\bar{s}_i &= \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \delta^s_i, \\
\bar{w}_{i+1} &= \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \delta^w_i, \\
\bar{z}_i &= A\bar{m}_i + \bar{\beta}_i \bar{z}_{i-1} + \delta^z_i,
\end{align*}
\]

Residual gap is **coupled** with the gaps on the other auxiliary variables:

\[
\begin{align*}
f_i &= (b - A\bar{x}_i) - \bar{r}_i, \\
g_i &= A\bar{p}_i - \bar{s}_i, \\
h_i &= A\bar{u}_i - \bar{w}_i, \\
j_i &= A\bar{q}_i - \bar{z}_i
\end{align*}
\]

\[
\begin{bmatrix}
  f_{i+1} \\
g_i \\
h_{i+1} \\
j_i
\end{bmatrix} =
\begin{bmatrix}
  1 & -\bar{\alpha}_i \bar{\beta}_i & -\bar{\alpha}_i & 0 \\
  0 & \bar{\beta}_i & 1 & 0 \\
  0 & 0 & 1 & -\bar{\alpha}_i \bar{\beta}_i \\
  0 & 0 & 0 & \bar{\beta}_i
\end{bmatrix}
\begin{bmatrix}
f_i \\
g_{i-1} \\
h_i \\
j_{i-1}
\end{bmatrix} +
\begin{bmatrix}
-A\delta^x_i - \delta^r_i - \bar{\alpha}_i (A\delta^p_i - \delta^s_i) \\
A\delta^p_i - \delta^s_i \\
A\delta^u_i - \delta^w_i - \bar{\alpha}_i (A\delta^q_i - \delta^z_i) \\
A\delta^q_i - \delta^z_i
\end{bmatrix}.
\]

**Amplification** of local rounding errors possible, depending on \(\alpha_i\)'s and \(\beta_i\)'s.

---

C. & Vanroose (2017)
Error bounds: Local rounding errors $A\delta_i^\times + \delta_i^r$ can be bounded by

$$\|A\delta_i^\times + \delta_i^r\| \leq \left( \|A\| \|\bar{x}_i\| + (\mu \sqrt{n} + 4) |\bar{\alpha}_i| \|A\| \|\bar{p}_i\| + \|\bar{r}_i\| \right) \epsilon = e_i^f \epsilon.$$

In practice:
- characterizes extreme case rounding error effects
- often largely overestimates the actual errors

Error estimates: Local rounding errors can be approximated as

$$\|A\delta_i^\times + \delta_i^r\| \approx \sqrt{e_i^f} \epsilon$$

In practice:
- additional norm computations required
- include in existing global reduction phase to avoid overhead

(\(\epsilon = \text{machine precision}\))
Conjugate Gradients

Pipe-CG with automated residual replacement

Explicitly replace $\bar{r}_i$, $\bar{s}_i$, $\bar{w}_i$ and $\bar{z}_i$ by their true values in selected iterations:

$$
\bar{r}_{i+1} = \text{fl}(b - A\bar{x}_{i+1}), \quad \bar{w}_{i+1} = \text{fl}(A\bar{u}_{i+1}),
$$

$$
\bar{s}_i = \text{fl}(A\bar{p}_i), \quad \bar{z}_i = \text{fl}(A\bar{q}_i).
$$

Residual replacement criterion:

$$
\|f_{i-1}\| \leq \tau \|\bar{r}_{i-1}\| \quad \text{and} \quad \|f_i\| > \tau \|\bar{r}_i\|.
$$

with $\tau = \sqrt{\epsilon}$.

- Sleijpen & Van der Vorst 1996
- Tong & Ye, 1999
- Van der Vorst & Ye, 2000

*Estimate for gap $\|f_i\|$ can be computed at runtime (without additional overhead), so fully automated replacement strategy is possible for pipe-CG.*
Example: 2D Laplacian - 2,500 unk (cont.)

- Pipe-CG-rr = pipe-CG with residual replacement based on rounding error model

Cost? 4 additional SpMV’s per replacement step

- Replacement criterion ensures:
  1. number of replacements is limited,
  2. only replace when $\|r_i\|$ is sufficiently large (Krylov convergence is not affected)

Tong & Ye, 1999
**MatrixMarket collection**: convergence tests on all non-diagonal SPD matrices

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Prec</th>
<th>$\kappa(A)$</th>
<th>$n$</th>
<th>#nnz</th>
<th>CG</th>
<th>CG-CG</th>
<th>p-CG</th>
<th>p-CG-rr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>iter</td>
<td>relres</td>
<td>iter</td>
<td>relres</td>
</tr>
<tr>
<td>bcsstk14</td>
<td>JAC</td>
<td>1.3e+10</td>
<td>1806</td>
<td>63,454</td>
<td>650</td>
<td>7.6e-16</td>
<td>658</td>
<td>7.1e-16</td>
</tr>
<tr>
<td>bcsstk15</td>
<td>JAC</td>
<td>8.0e+09</td>
<td>3948</td>
<td>117,816</td>
<td>772</td>
<td>3.7e-15</td>
<td>785</td>
<td>3.5e-15</td>
</tr>
<tr>
<td>bcsstk16</td>
<td>JAC</td>
<td>65</td>
<td>4884</td>
<td>290,378</td>
<td>298</td>
<td>3.5e-15</td>
<td>300</td>
<td>4.0e-15</td>
</tr>
<tr>
<td>bcsstk17</td>
<td>JAC</td>
<td>65</td>
<td>10,974</td>
<td>428,650</td>
<td>3547</td>
<td>1.0e-14</td>
<td>3428</td>
<td>1.7e-14</td>
</tr>
<tr>
<td>bcsstk18</td>
<td>JAC</td>
<td>65</td>
<td>11,948</td>
<td>149,090</td>
<td>2299</td>
<td>2.2e-15</td>
<td>2294</td>
<td>2.1e-14</td>
</tr>
<tr>
<td>bcsstk27</td>
<td>JAC</td>
<td>7.7e+04</td>
<td>1224</td>
<td>56,126</td>
<td>345</td>
<td>3.2e-15</td>
<td>345</td>
<td>4.0e-15</td>
</tr>
<tr>
<td>gr.30_30</td>
<td></td>
<td>3.8e+02</td>
<td>900</td>
<td>7744</td>
<td>56</td>
<td>2.7e-15</td>
<td>55</td>
<td>3.1e-15</td>
</tr>
<tr>
<td>nos1</td>
<td>*ICC</td>
<td>2.5e+07</td>
<td>237</td>
<td>1017</td>
<td>301</td>
<td>1.3e-14</td>
<td>338</td>
<td>1.2e-14</td>
</tr>
<tr>
<td>nos2</td>
<td>*ICC</td>
<td>6.3e+09</td>
<td>957</td>
<td>4137</td>
<td>3180</td>
<td>8.3e-14</td>
<td>3292</td>
<td>1.1e-13</td>
</tr>
<tr>
<td>nos3</td>
<td>ICC</td>
<td>7.3e+04</td>
<td>960</td>
<td>15,844</td>
<td>64</td>
<td>1.0e-14</td>
<td>63</td>
<td>1.1e-14</td>
</tr>
<tr>
<td>nos4</td>
<td>ICC</td>
<td>2.7e+03</td>
<td>100</td>
<td>594</td>
<td>31</td>
<td>1.9e-15</td>
<td>31</td>
<td>1.9e-15</td>
</tr>
<tr>
<td>nos5</td>
<td>ICC</td>
<td>2.9e+04</td>
<td>468</td>
<td>5172</td>
<td>63</td>
<td>3.2e-16</td>
<td>64</td>
<td>3.4e-16</td>
</tr>
<tr>
<td>nos6</td>
<td>ICC</td>
<td>8.0e+06</td>
<td>675</td>
<td>3255</td>
<td>34</td>
<td>5.1e-15</td>
<td>35</td>
<td>6.2e-15</td>
</tr>
<tr>
<td>nos7</td>
<td>ICC</td>
<td>4.1e+09</td>
<td>729</td>
<td>4617</td>
<td>29</td>
<td>4.0e-14</td>
<td>31</td>
<td>2.8e-14</td>
</tr>
<tr>
<td>slrmq2m1</td>
<td>ICC</td>
<td>1.8e+06</td>
<td>5489</td>
<td>262,411</td>
<td>122</td>
<td>4.3e-15</td>
<td>122</td>
<td>4.6e-15</td>
</tr>
<tr>
<td>slrmq3m1</td>
<td>ICC</td>
<td>2.5e+06</td>
<td>5489</td>
<td>217,651</td>
<td>229</td>
<td>9.3e-15</td>
<td>228</td>
<td>8.7e-15</td>
</tr>
<tr>
<td>s2mq4m1</td>
<td>*ICC</td>
<td>1.8e+08</td>
<td>5489</td>
<td>263,351</td>
<td>370</td>
<td>6.7e-15</td>
<td>387</td>
<td>7.3e-15</td>
</tr>
<tr>
<td>s2rtm3m1</td>
<td>ICC</td>
<td>2.5e+08</td>
<td>5489</td>
<td>217,681</td>
<td>285</td>
<td>8.7e-15</td>
<td>283</td>
<td>1.0e-14</td>
</tr>
<tr>
<td>s3rkq4m2</td>
<td>*ICC</td>
<td>1.9e+11</td>
<td>90,449</td>
<td>2,455,670</td>
<td>-</td>
<td>1.9e-08</td>
<td>-</td>
<td>2.1e-08</td>
</tr>
<tr>
<td>s3dtk3m2</td>
<td>*ICC</td>
<td>3.6e+11</td>
<td>90,449</td>
<td>1,921,955</td>
<td>-</td>
<td>2.9e-07</td>
<td>-</td>
<td>2.9e-07</td>
</tr>
<tr>
<td>s3rmq4m1</td>
<td>*ICC</td>
<td>1.8e+10</td>
<td>5489</td>
<td>262,943</td>
<td>1651</td>
<td>1.5e-14</td>
<td>1789</td>
<td>1.6e-14</td>
</tr>
<tr>
<td>s3rtm3m1</td>
<td>*ICC</td>
<td>2.5e+10</td>
<td>5489</td>
<td>217,669</td>
<td>2282</td>
<td>2.7e-14</td>
<td>2559</td>
<td>2.9e-14</td>
</tr>
<tr>
<td>s3rmt3m3</td>
<td>*ICC</td>
<td>2.4e+10</td>
<td>5357</td>
<td>207,123</td>
<td>2862</td>
<td>3.3e-14</td>
<td>2798</td>
<td>3.4e-14</td>
</tr>
</tbody>
</table>

**Conjugate Gradients**

**Numerical results**: attainable accuracy
Conjugate Gradients

Numerical results: strong scaling

- PETSc implementation using MPICH-3.1.3 communication
- Benchmark problem: 2D Laplacian model, 1,000,000 unknowns
- System specs: 20 nodes, two 6-core Intel Xeon X5660 Nehalem 2.8GHz CPUs/node

Speedup over single-node CG (12-240 cores)  
Accuracy i.f.o. total time spent (240 cores)
Conjugate Gradients
Numerical results: strong scaling

- PETSc implementation using MPICH-3.3a2 communication
- Benchmark problem: 3D ice sheet flow, $150 \times 150 \times 100 / 500 \times 500 \times 50$ Q1 FE
- System specs: 128 nodes, two 14-core Intel Xeon E5-2680v4 2.4GHz CPUs/node

Speedup over single-node CG
(2,250,000 unk)
Non-automated residual replacement every \( i \)-th iteration:

\[
\begin{align*}
    r_i &:= b - Ax_i, \\
    \hat{r}_i &:= M^{-1}r_i, \\
    w_i &:= A\hat{r}_i, \\
    s_i &:= A\hat{p}_i, \\
    \hat{s}_i &:= M^{-1}s_i, \\
    z_i &:= A\hat{s}_i.
\end{align*}
\]

⊕ increased maximal attainable accuracy: comparable to BiCGStab level
⊕ increased robustness: resets accumulated rounding errors
⊖ delay of convergence: increased number of iterations possible

Strakos & Tichy, 2002
Bi-Conjugate Gradients Stabilized
Numerical results: attainable accuracy

Multi-term auxiliary recurrences
↓
Local error propagation matrix
\[
\begin{bmatrix}
1 & -\bar{\alpha}_i\bar{\beta}_i & -\bar{\alpha}_i - \bar{\omega}_i & \bar{\alpha}_i\bar{\beta}_i(\bar{\omega}_i + \bar{\omega}_{i-1}) \\
0 & \bar{\beta}_i & 1 & -\bar{\beta}_i\bar{\omega}_{i-1} \\
0 & 0 & 1 & -\bar{\alpha}_i\bar{\beta}_i \\
0 & 0 & 0 & \bar{\beta}_i
\end{bmatrix}
\]

Non-automated residual replacement every \(i\)-th iteration:
\[
\begin{align*}
    r_i &:= b - Ax_i, \\
    \hat{r}_i &:= M^{-1}r_i, \\
    w_i &:= A\hat{r}_i, \\
    s_i &:= A\hat{r}_i, \\
    \hat{s}_i &:= M^{-1}s_i, \\
    z_i &:= A\hat{s}_i.
\end{align*}
\]

⊕ increased maximal attainable accuracy: comparable to BiCGStab level
⊕ increased robustness: resets accumulated rounding errors
⊕ delay of convergence: increased number of iterations possible

Strakos & Tichy, 2002
Bi-Conjugate Gradients Stabilized
Numerical results: strong scaling

- PETSc implementation using MPICH-3.1.3 communication
- Benchmark problem 2: 2D indefinite Helmholtz model, 1,000,000 unknowns
- System specs: 20 nodes, two 6-core Intel Xeon X5660 Nehalem 2.8GHz CPUs/node

\[ A_{2}^{stencil} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \]

Speedup over single-node BiCGStab

<table>
<thead>
<tr>
<th>nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGStab</td>
<td>1563</td>
<td>1805</td>
<td>1789</td>
<td>1875</td>
<td>1710</td>
</tr>
<tr>
<td>p-BiCGStab</td>
<td>1807</td>
<td>1614</td>
<td>1779</td>
<td>1787</td>
<td>1673</td>
</tr>
<tr>
<td>p-BiCGStab-rr</td>
<td>1857</td>
<td>1788</td>
<td>1728</td>
<td>1570</td>
<td>1677</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nodes</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGStab</td>
<td>1852</td>
<td>1789</td>
<td>1555</td>
<td>1641</td>
<td>1909</td>
</tr>
<tr>
<td>p-BiCGStab</td>
<td>1547</td>
<td>1668</td>
<td>1773</td>
<td>1640</td>
<td>1673</td>
</tr>
<tr>
<td>p-BiCGStab-rr</td>
<td>1721</td>
<td>1688</td>
<td>1283</td>
<td>1884</td>
<td>1718</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nodes</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGStab</td>
<td>1805</td>
<td>1715</td>
<td>1875</td>
<td>1717</td>
<td>1765</td>
</tr>
<tr>
<td>p-BiCGStab</td>
<td>1936</td>
<td>1811</td>
<td>1629</td>
<td>1642</td>
<td>1849</td>
</tr>
<tr>
<td>p-BiCGStab-rr</td>
<td>1845</td>
<td>1628</td>
<td>1562</td>
<td>1861</td>
<td>1650</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nodes</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiCGStab</td>
<td>1722</td>
<td>1657</td>
<td>2050</td>
<td>1778</td>
<td>1670</td>
</tr>
<tr>
<td>p-BiCGStab</td>
<td>1843</td>
<td>1726</td>
<td>1796</td>
<td>1713</td>
<td>1870</td>
</tr>
<tr>
<td>p-BiCGStab-rr</td>
<td>1647</td>
<td>1889</td>
<td>1750</td>
<td>1701</td>
<td>2112</td>
</tr>
</tbody>
</table>
In this talk

Pipelined Conjugate Gradients: algorithm and rounding error analysis
- pipe-CG is much more sensitive to rounding errors than standard CG
- rounding error model allows for real-time countermeasures: pipe-CG-rr

Pipelined BiCGStab: algorithm and numerical stability
- pipe-BiCGStab for non-symmetric operators (alternative to pipe-GMRES)

Related work

Longer pipelines: pipe(ℓ)-GMRES & pipe(ℓ)-CG  
- talk by W. Vanroose
  - useful when time(GLRED) is much larger than time(SPMV+PREC)
  - further reduced numerical stability (WIP)

Shifted pipelined CG
- introduce shift $\sigma$ in auxiliary variables: $s_i = (A - \sigma)p_i$, $w_i = (A - \sigma)s_i$
- stabilizing effect on rounding error propagation for the right shift choice

Soft error detection  
- rounding error model allows separation rounding errors vs. soft errors
- detect-and-correct strategy improves soft error resilience

with INRIA Bordeaux


Thank you for your attention!

Q: What is the difference between pipelined and s-step Krylov methods?
   Carson, Knight, Demmel, 2013
   A: Global communication is hidden vs. avoided
   A: Off-the-shelf preconditioning possible vs. specialized preconditioning required

Q: Is the code available online?
   A: The inclusion of pipe(ℓ)-CG and pipe(ℓ)-GMRES in PETSc is work in progress.