Latency hiding of global reductions in pipelined Krylov methods

Wim Vanroose[1], Pieter Ghysels[2] & Bram Reps[1]

wim.vanroose@uantwerp.be pghysels@lbl.gov bram.reps@uantwerp.be

[1] University of Antwerp - Dept Math & Computer Science, Belgium

[2] LBNL - Future Technologies Group, Berkeley, CA, USA

CANUM 2014
March 31 - April 4, 2014

What are we working on?

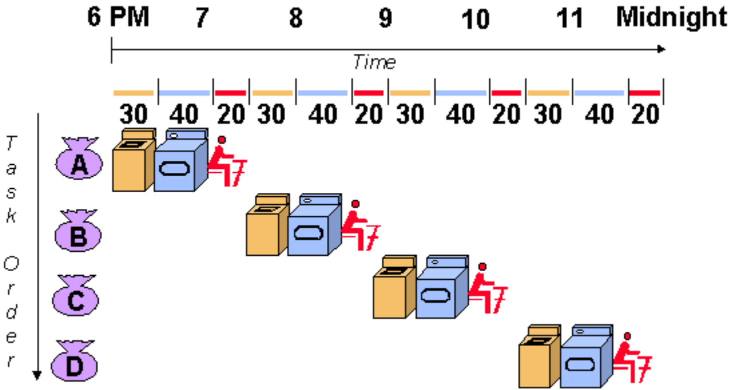


Figure: Latency hiding of global *drying* in pipelined *Laundry* methods

What are we working on?

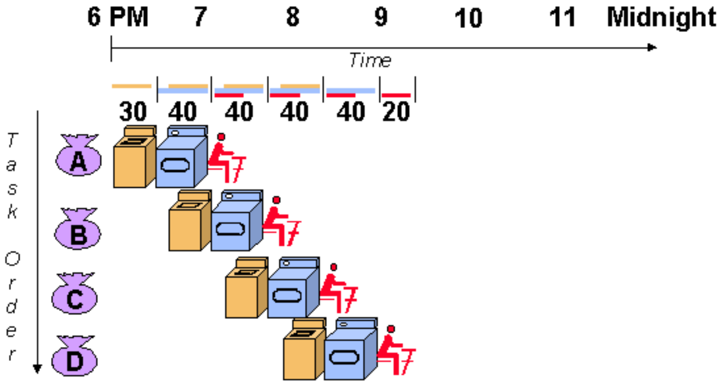
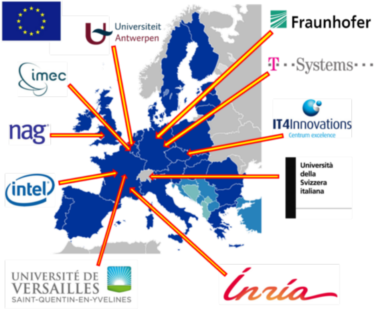


Figure: Latency hiding of global *drying* in pipelined *Laundry* methods

What EXA2CT-ly are we working on?



Increasing gap between computation and communication costs

- Floating point performance steadily increases
- Network latencies only go down marginally
- Memory latencies decline slowly
- Avoid communication by trading communication for computation
- Hide latency of communications

EXascale Algorithms and Advanced Computational Techniques

`https://projects.imec.be/exa2ct/`

Outline of the talk

Krylov subspace methods
(cf. Laundry methods)

Hiding global reductions
(cf. hiding drying time)

Increasing arithmetic intensity
(cf. piling up laundry)

Conclusions & future work
(cf. washing instructions and ecological detergents)

Outline of the talk

Krylov subspace methods
(cf. Laundry methods)

Hiding global reductions
(cf. hiding drying time)

Increasing arithmetic intensity
(cf. piling up laundry)

Conclusions & future work
(cf. washing instructions and ecological detergents)

General idea

Iteratively improve an approximate solution of linear system $Ax = b$,

$$x_i \in x_0 + \mathcal{K}_i(A, r_0) = x_0 + \mathrm{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{i-1} r_0\}$$

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

E.g.: Conjugate Gradients

$r^{(0)} \leftarrow b - Ax^{(0)}$
$p^{(0)} \leftarrow r^{(0)}$
for $i = 0, \dots$ **do**
  $w \leftarrow Ap^{(i)}$
  $\alpha_i \leftarrow (r^{(i)}, r^{(i)})/(w, p^{(i)})$
  $x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
  $r^{(i+1)} \leftarrow r^{(i)} - \alpha_i w$
  $\beta_i \leftarrow (r^{(i+1)}, r^{(i+1)})/(r^{(i)}, r^{(i)})$
  $p^{(i+1)} \leftarrow r^{(i+1)} + \beta_i p^{(i)}$
end for

General idea

Iteratively improve an approximate solution of linear system $Ax = b$,

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

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

E.g.: Conjugate Gradients

$r^{(0)} \leftarrow b - Ax^{(0)}$
$p^{(0)} \leftarrow r^{(0)}$
for $i = 0, \ldots$ **do**
  $w \leftarrow Ap^{(i)}$
  $\alpha_i \leftarrow (r^{(i)}, r^{(i)})/(w, p^{(i)})$
  $x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
  $r^{(i+1)} \leftarrow r^{(i)} - \alpha_i w$
  $\beta_i \leftarrow (r^{(i+1)}, r^{(i+1)})/(r^{(i)}, r^{(i)})$
  $p^{(i+1)} \leftarrow r^{(i+1)} + \beta_i p^{(i)}$
end for

Communication patterns in the building blocks

i. axpy
  - no dependencies on other vector elements (no communication)
  - scales well

ii. SpMVM
  - dependencies given by matrix/vector partition (one-to-one communication)
  - bandwidth limited
  - scales

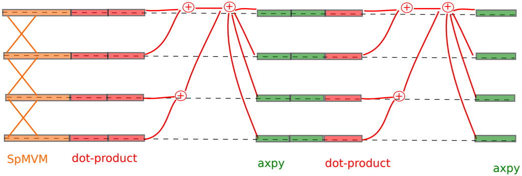
iii. dot-product
  - dependency on all vector elements (global reduction)
  - latency dominated
  - scales as $log_2(\#partitions)$

E.g.: Conjugate Gradients

1: $r^{(0)} \leftarrow b - Ax^{(0)}$
2: $p^{(0)} \leftarrow r^{(0)}$
3: **for** $i = 0, \ldots$ **do**
4: $\quad w \leftarrow Ap^{(i)}$
5: $\quad \alpha_i \leftarrow (r^{(i)}, r^{(i)})/(w, p^{(i)})$
6: $\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
7: $\quad r^{(i+1)} \leftarrow r^{(i)} - \alpha_i w$
8: $\quad \beta_i \leftarrow (r^{(i+1)}, r^{(i+1)})/(r^{(i)}, r^{(i)})$
9: $\quad p^{(i+1)} \leftarrow r^{(i+1)} + \beta_i p^{(i)}$
10: **end for**

Case study: Conjugate Gradients

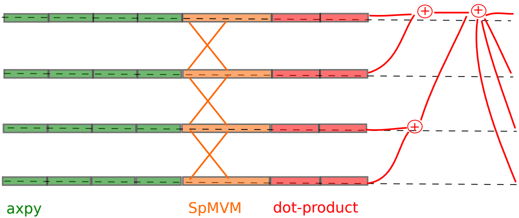
Hestenes and Stiefel (1952)



$r^{(0)} \leftarrow b - Ax^{(0)}$
$p^{(0)} \leftarrow r^{(0)}$
for $i = 0, \ldots$ **do**
  $w \leftarrow Ap^{(i)}$
  $\alpha_i \leftarrow (r^{(i)}, r^{(i)})/(w, p^{(i)})$
  $x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
  $r^{(i+1)} \leftarrow r^{(i)} - \alpha_i w$
  $\beta_i \leftarrow (r^{(i+1)}, r^{(i+1)})/(r^{(i)}, r^{(i)})$
  $p^{(i+1)} \leftarrow r^{(i+1)} + \beta_i p^{(i)}$
end for

Case study: Conjugate Gradients

Chronopoulos and Gear (1989)



$r_{(0)} \leftarrow b - Ax^{(0)}$

... (loop-unrolling)

for $i = 1, \ldots$ **do**

$\quad p^{(i)} \leftarrow r^{(i)} + \beta_i p^{(i-1)}$

$\quad s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$

$\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$

$\quad r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$

$\quad w^{(i+1)} \leftarrow Ar^{(i+1)}$

$\quad \gamma_{i+1} \leftarrow (r^{(i+1)}, r^{(i+1)})$

$\quad \delta \leftarrow (w^{(i+1)}, r^{(i+1)})$

$\quad \beta_{i+1} \leftarrow \gamma_{i+1}/\gamma_i$

$\quad \alpha_{i+1} \leftarrow \gamma_{i+1}/(\delta - \beta_{i+1}\gamma_{i+1}/\alpha_i)$

end for

Krylov subspace methods

Case study: Conjugate Gradients

Chronopoulos and Gear (1989)

- Equivalent to CG (in infinite precision)
- Extra recurrence relation for $s^{(i)} = Ap^{(i)}$
- Two dot-products are grouped in one global reduction
- Communication avoiding

$r_{(0)} \leftarrow b - Ax^{(0)}$
. . . (loop-unrolling)
for $i = 1, \ldots$ **do**
    $p^{(i)} \leftarrow r^{(i)} + \beta_i p^{(i-1)}$
    $s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$
    $x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
    $r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$
    $w^{(i+1)} \leftarrow Ar^{(i+1)}$
    $\gamma_{i+1} \leftarrow (r^{(i+1)}, r^{(i+1)})$
    $\delta \leftarrow (w^{(i+1)}, r^{(i+1)})$
    $\beta_{i+1} \leftarrow \gamma_{i+1}/\gamma_i$
    $\alpha_{i+1} \leftarrow \gamma_{i+1}/(\delta - \beta_{i+1}\gamma_{i+1}/\alpha_i)$
end for

Outline of the talk

Krylov subspace methods
(cf. Laundry methods)

Hiding global reductions
(cf. hiding drying time)

Increasing arithmetic intensity
(cf. piling up laundry)

Conclusions & future work
(cf. washing instructions and ecological detergents)

- Dot-products are latency dominated
- Dot-products block all other (local) work
- Other (local) operations (SpMVM/axpy) scale well

Objective

Rewrite Krylov solvers such that latency of dot-products (global reductions) can be overlapped with application of the SpMVM and/or the preconditioner.

- Use non-blocking asynchronous global communication
- MPI-3 standard introduces `MPI_Iallreduce()`
- GPI-2 introduces `gaspi_allreduce()` + uses PGAS (partitioned global address space)

Hiding global reductions
Pipelined Conjugate Gradients

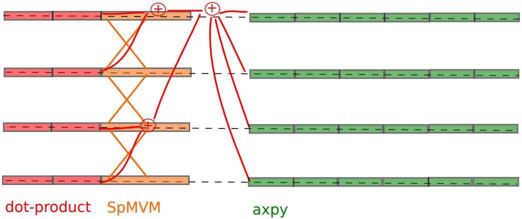
Ghysels and Vanroose (2013)

- Equivalent to CG (in infinite precision)
- Extra recurrence relations for $s^{(i)} = Ap^{(i)}$ and $z = As^{(i)}$
- Two dot-products are grouped in one global reduction
- Communication avoiding
- Overlap global communication with local computations: line 4 + 5 + 6
- Communication avoiding
  \+ communication hiding

1: $r_{(0)} \leftarrow b - Ax^{(0)}$
2: $\ldots$ (loop-unrolling)
3: **for** $i = 1, \ldots$ **do**
4: $\quad \gamma_i \leftarrow (r^{(i)}, r^{(i)})$
5: $\quad \delta \leftarrow (w^{(i)}, r^{(i)})$
6: $\quad q^{(i)} \leftarrow Aw^{(i)}$
7: $\quad \beta_i \leftarrow \gamma_i / \gamma_{i-1}$
8: $\quad \alpha_i \leftarrow \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
9: $\quad z^{(i)} \leftarrow q^{(i)} + \beta_i z^{(i-1)}$
10: $\quad s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$
11: $\quad p^{(i)} \leftarrow r^{(i)} + \beta_i p^{(i-1)}$
12: $\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
13: $\quad r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$
14: $\quad w^{(i+1)} \leftarrow w^{(i)} - \alpha_i z^{(i)}$
15: **end for**

Pipelined Conjugate Gradients

Ghysels and Vanroose (2013)



$r_{(0)} \leftarrow b - Ax^{(0)}$
$\ldots$ (loop-unrolling)
for $i = 1, \ldots$ **do**
$\quad \gamma_i \leftarrow (r^{(i)}, r^{(i)})$
$\quad \delta \leftarrow (w^{(i)}, r^{(i)})$
$\quad q^{(i)} \leftarrow Aw^{(i)}$
$\quad \beta_i \leftarrow \gamma_i / \gamma_{i-1}$
$\quad \alpha_i \leftarrow \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
$\quad z^{(i)} \leftarrow q^{(i)} + \beta_i z^{(i-1)}$
$\quad s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$
$\quad p^{(i)} \leftarrow r^{(i)} + \beta_i p^{(i-1)}$
$\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
$\quad r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$
$\quad w^{(i+1)} \leftarrow w^{(i)} - \alpha_i z^{(i)}$
end for

Preconditioned pipelined Conjugate Gradients

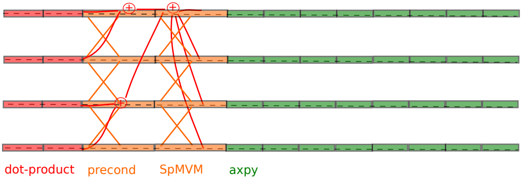
Ghysels and Vanroose (2013)

- Equivalent to CG (in infinite precision)
- Extra recurrence relations for $w^{(i)} = Au^{(i)}$, $s^{(i)} = Ap^{(i)}$ and $z = Aq^{(i)}$
- Two dot-products are grouped in one global reduction
- Overlap global communication with *extra* local computations: line 4 + 5 + 7 + 6
- Communication avoiding + communication hiding

1: $r_{(0)} \leftarrow b - Ax^{(0)}$
2: ... (loop-unrolling)
3: **for** $i = 1, \dots$ **do**
4: $\quad \gamma_i \leftarrow (r^{(i)}, u^{(i)})$
5: $\quad \delta \leftarrow (w^{(i)}, u^{(i)})$
6: $\quad m^{(i)} \leftarrow M^{-1} w^{(i)}$
7: $\quad n^{(i)} \leftarrow Am^{(i)}$
8: $\quad \beta_i \leftarrow \gamma_i / \gamma_{i-1}$
9: $\quad \alpha_i \leftarrow \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
10: $\quad z^{(i)} \leftarrow n^{(i)} + \beta_i z^{(i-1)}$
11: $\quad q^{(i)} \leftarrow m^{(i)} + \beta_i q^{(i-1)}$
12: $\quad s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$
13: $\quad p^{(i)} \leftarrow u^{(i)} + \beta_i p^{(i-1)}$
14: $\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
15: $\quad r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$
16: $\quad u^{(i+1)} \leftarrow u^{(i)} - \alpha_i q^{(i)}$
17: $\quad w^{(i+1)} \leftarrow w^{(i)} - \alpha_i z^{(i)}$
18: **end for**

Preconditioned pipelined Conjugate Gradients

Ghysels and Vanroose (2013)



$r_{(0)} \leftarrow b - Ax^{(0)}$
... (loop-unrolling)
for $i = 1, \ldots$ **do**
    $\gamma_i \leftarrow (r^{(i)}, u^{(i)})$
    $\delta \leftarrow (w^{(i)}, u^{(i)})$
    $m^{(i)} \leftarrow M^{-1} w^{(i)}$
    $n^{(i)} \leftarrow Am^{(i)}$
    $\beta_i \leftarrow \gamma_i / \gamma_{i-1}$
    $\alpha_i \leftarrow \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
    $z^{(i)} \leftarrow n^{(i)} + \beta_i z^{(i-1)}$
    $q^{(i)} \leftarrow m^{(i)} + \beta_i q^{(i-1)}$
    $s^{(i)} \leftarrow w^{(i)} + \beta_i s^{(i-1)}$
    $p^{(i)} \leftarrow u^{(i)} + \beta_i p^{(i-1)}$
    $x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
    $r^{(i+1)} \leftarrow r^{(i)} - \alpha_i s^{(i)}$
    $u^{(i+1)} \leftarrow u^{(i)} - \alpha_i q^{(i)}$
    $w^{(i+1)} \leftarrow w^{(i)} - \alpha_i z^{(i)}$
end for

Preconditioned pipelined Conjugate Residuals

Ghysels and Vanroose (2013)

- Equivalent to CR (in infinite precision)
- Based on $(\cdot,\cdot)_A$-inner product
- Two dot-products are grouped in one global reduction
- Overlap global communication with local computations: line 5 + 6 + 7
- No overlap with preconditioner
- Only 3 additional axpy's save memory

1: $r_{(0)} \leftarrow b - Ax^{(0)}$
2: ... (loop-unrolling)
3: **for** $i = 1, \ldots$ **do**
4: $\quad m^{(i)} \leftarrow M^{-1} w^{(i)}$
5: $\quad \gamma_i \leftarrow (w^{(i)}, u^{(i)})$
6: $\quad \delta \leftarrow (m^{(i)}, w^{(i)})$
7: $\quad n^{(i)} \leftarrow Am^{(i)}$
8: $\quad \beta_i \leftarrow \gamma_i / \gamma_{i-1}$
9: $\quad \alpha_i \leftarrow \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
10: $\quad z^{(i)} \leftarrow n^{(i)} + \beta_i z^{(i-1)}$
11: $\quad q^{(i)} \leftarrow m^{(i)} + \beta_i q^{(i-1)}$
12: $\quad p^{(i)} \leftarrow u^{(i)} + \beta_i p^{(i-1)}$
13: $\quad x^{(i+1)} \leftarrow x^{(i)} + \alpha_i p^{(i)}$
14: $\quad u^{(i+1)} \leftarrow u^{(i)} - \alpha_i q^{(i)}$
15: $\quad w^{(i+1)} \leftarrow w^{(i)} - \alpha_i z^{(i)}$
16: **end for**

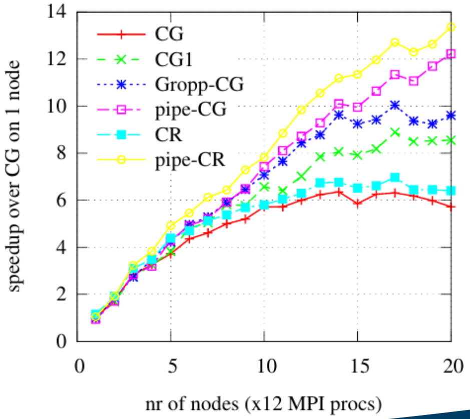
Comparison of CG variants

	flops	time (excl axpy's, dot's)	# syncs	mem
CG	10	2G + SpMVM + PC	2	4
Chron/Gear-CG	12	G + SpMVM + PC	1	5
Gropp-CG	14	max(G,SpMVM) + max(G,PC)	2	6
pipe-CG	20	max(G,SpMVM+PC)	1	9
CR	12	2G + SpMVM + PC	2	5
pipe-CR	16	max(G,SpMVM) + PC	1	7

- G: latency of global reduction
- SpMVM: sparse matrix-vector time
- PC: application of preconditioner

Strong scaling experiment

- Hydrostatic ice sheet flow, $100 \times 100 \times 50$ Q1 finite elements
- Line search Newton method (rtol$=10^{-8}$, atol$=10^{-15}$)
- CG preconditioned with block Jacobi with ICC(0) (rtol$=10^{-5}$, atol$=10^{-50}$)



- max pipe-CG/CG speedup: $2.14\times$
  max pipe-CG/CG1 speedup: $1.43\times$
- max pipe-CR/CR speedup: $2.09\times$

(CG1 = Chrono/Gear CG)

Other pipelined Krylov methods

- Preconditioned pipelined GMRES

  Ghysels, Ashby, Meerbergen and Vanroose (2012)

$$
\begin{aligned}
V_{i-\ell+1} &= [v_0, v_1, \ldots, v_{i-\ell}] \\
Z_{i+1} &= [z_0, z_1, \ldots, z_{i-\ell}, \underbrace{z_{i-\ell+1}, \ldots, z_i}_{\ell}]
\end{aligned}
$$

  - Compute ℓ new basis vectors for Krylov subspace (SpMVMs) during global communication (dot-products).
  - Orthogonalization step when previous global reduction has finished
  - More technical, but deeper and variable pipelining possible p(ℓ)-GMRES
- Augmented and deflated Krylov subspace methods

Outline of the talk

Krylov subspace methods
(cf. Laundry methods)

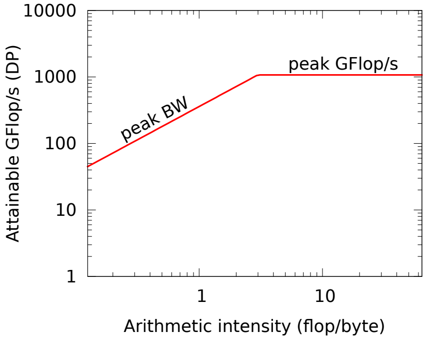
Hiding global reductions
(cf. hiding drying time)

Increasing arithmetic intensity
(cf. piling up laundry)

Conclusions & future work
(cf. washing instructions and ecological detergents)

Roofline Model

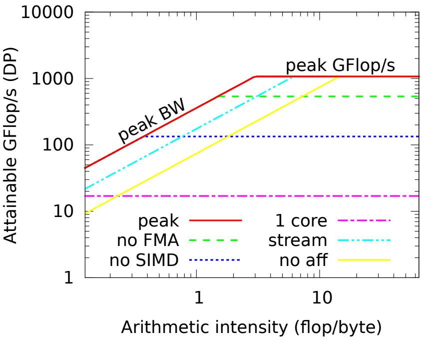
- Arithmetic intensity: $q = \dfrac{\text{floating-point operations}}{\text{byte off-chip memory traffic}}$
- High $q \rightarrow$ compute bound (dense algebra, fft, ...)
- Low $q \rightarrow$ bandwidth bound (sparse algebra, stencils, ...)
- Roofline gives upperbound for performance for given q



Williams, Waterman, Patterson (2008)

Roofline Model

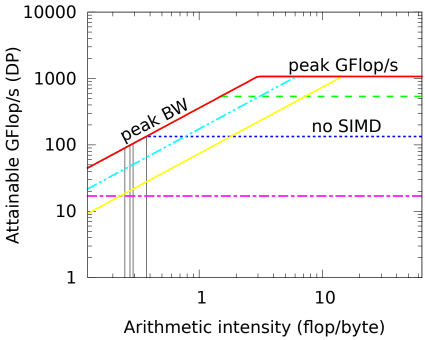
- Arithmetic intensity: $q = \dfrac{\text{floating-point operations}}{\text{byte off-chip memory traffic}}$
- High $q \rightarrow$ compute bound (dense algebra, fft, ...)
- Low $q \rightarrow$ bandwidth bound (sparse algebra, stencils, ...)
- Roofline gives upperbound for performance for given q



Williams, Waterman, Patterson (2008)

Roofline Model

- Arithmetic intensity: $q = \dfrac{\text{floating-point operations}}{\text{byte off-chip memory traffic}}$
- High $q \rightarrow$ compute bound (dense algebra, fft, ...)
- Low $q \rightarrow$ bandwidth bound (sparse algebra, stencils, ...)
- Roofline gives upperbound for performance for given q



Williams, Waterman, Patterson (2008)

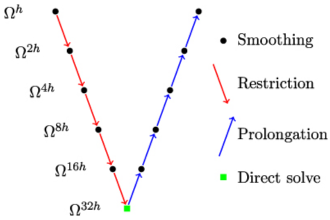
Arithmetic intensity of s dependent SpMVMs

	1 SpMVM	$s\times$ SpMVM	$s\times$ SpMVM in place
flops	$2n_{nz}$	$2s \cdot n_{nz}$	$2s \cdot n_{nz}$
words moved	$n_{nz} + 2n$	$sn_{nz} + 2sn$	$n_{nz} + 2n$
q	2	2	**2s**

See J. Demmel's course: CS 294-76 on Communication-Avoiding algorithms

V(ν_1,ν_2)-cycle multigrid

while $\|r^h\| > \texttt{tol}\|f^h\|$ **do**
  V-cycle(v^h, f^h)
end while

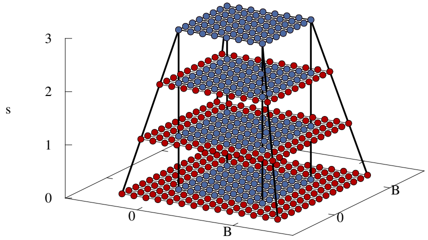


- I_h^{2h} Full weighting
- I_{2h}^h Linear interpolation

V-cycle(v^h, f^h)
  if Coarsest level **then**
    $v^h \leftarrow (A^h)^{-1} f^h$
  else
    for $k = 1, \ldots, \nu_1$ **do**
      $v^h \leftarrow (1 - \omega D^{-1} A^h) v^h + \omega D^{-1} f^h$
    end for
    $r^h \leftarrow f^h - A^h v^h$
    $r^{2h} \leftarrow I_h^{2h} r^h$
    $e^{2h} \leftarrow \text{V-cycle}^{2h}(0, r^{2h})$
    $e^h \leftarrow I_{2h}^h e^{2h}$
    $v^h \leftarrow v^h + e^h$
    for $k = 1, \ldots, \nu_2$ **do**
      $v^h \leftarrow (1 - \omega D^{-1} A^h) v^h + \omega D^{-1} f^h$
    end for
  end if

Consecutive smoothing steps

- A smoother is an SpMVM kernel with dependent vectors where only the last vector is required
  - Possibility to increase arithmetic intensity
  - Tiling over different smoother iterations
  - $q(\nu \times \omega\text{-Jac}) = \nu q_1(\omega\text{-Jac})$
- Divide the domain in tiles which fit in the cache
- Ground surface is loaded in cache and reused s $(= \nu)$ times
- Redundant work at the tile edges



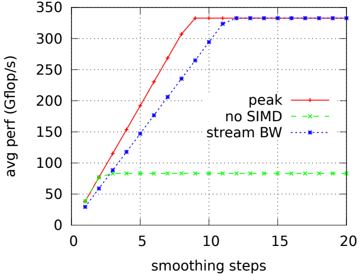
Cost of ν smoothing steps

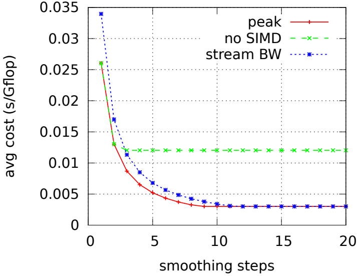
Since the arithmetic intensity increases for more smoothing steps

$$q(\nu \times \omega\text{-Jac}) = \nu q_1(\omega\text{-Jac})$$

according to the roofline:

performance increases & the average cost decreases



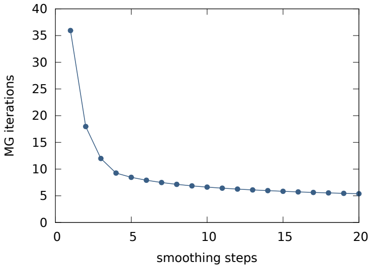


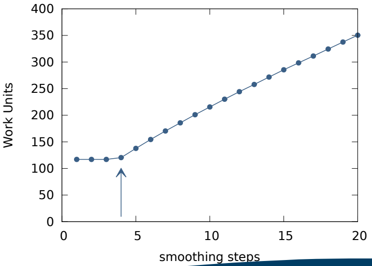
Work Unit Cost model

- Classical Work Unit cost model ignores memory bandwidth

$$1\,\mathrm{WU} = \text{smoother cost} = \mathcal{O}(n)$$

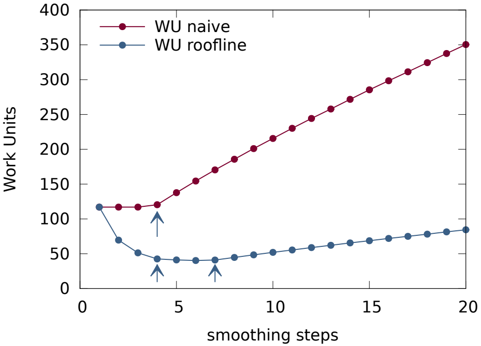
- Cost of multigrid to reach tolerance
$$= (9\nu + 19)\left(1 + \frac{1}{4} + \frac{1}{16} + \dots\right)\left\lceil \frac{\log(\texttt{tol})}{\log(\rho(\nu))} \right\rceil \mathrm{WU} \leq (9\nu + 19)\,\frac{4}{3}\left\lceil \frac{\log(\texttt{tol})}{\log(\rho(\nu))} \right\rceil \mathrm{WU}$$
- Optimum for low ν because computational cost increases with ν
- ... but communication overhead decreases!





Roofline Cost model

In contrast to naive model, the modified cost model suggests to repeat application of the smoother.

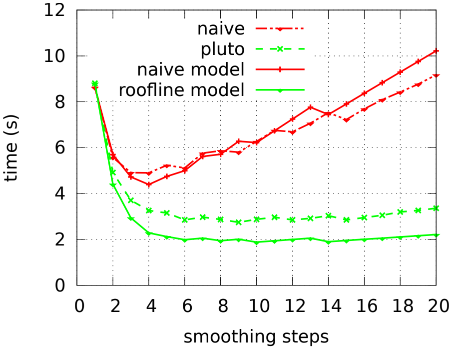


By tiling the smoother

- the optimal number of smoothing steps shifts to the right
- vectorization can be exploited

Roofline Cost model

In contrast to naive model, the modified cost model suggests to repeat application of the smoother.



By tiling the smoother

- the optimal number of smoothing steps shifts to the right
- vectorization can be exploited

Outline of the talk

Krylov subspace methods
(cf. Laundry methods)

Hiding global reductions
(cf. hiding drying time)

Increasing arithmetic intensity
(cf. piling up laundry)

Conclusions & future work
(cf. washing instructions and ecological detergents)

Summary

- **Krylov subspace methods**
  - 3 building blocks: axpy, SpMVM, dot-product
  - CG variants that group building blocks
  - Reduce global reduction steps
  - Communication avoiding
- **Hiding global reductions**
  - Pipelined CG and pipelined CR
  - Preconditioned versions
  - Overlap global reduction steps with other computational steps
  - Communication hiding (+ communication avoiding)
- **Increasing arithmetic intensity**
  - Tiling of smoother improves data locality and scalability
  - Trade-off between better convergence and increasing cost of smoother
  - Optimal number of smoothing steps increases
  - This allows exploiting of vector units
  - Still to be combined with an improved interpolation and restriction

References

- P. Ghysels, T.J. Ashby, K. Meerbergen & W. Vanroose, Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines, SIAM J. Sci. Comput., 35, 2013.
- P. Ghysels, W. Vanroose, Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing, 2013.
- P. Ghysels, P. Klosiewicz, W. Vanroose, Improving the arithmetic intensity of multigrid with the help of polynomial smoothers, Num. Linear Algebra Appl., 19, 2012.

References & FAQs

- P. Ghysels, T.J. Ashby, K. Meerbergen & W. Vanroose, Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines, SIAM J. Sci. Comput., 35, 2013.
- P. Ghysels, W. Vanroose, Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing, 2013.
- P. Ghysels, P. Klosiewicz, W. Vanroose, Improving the arithmetic intensity of multigrid with the help of polynomial smoothers, Num. Linear Algebra Appl., 19, 2012.

Q: *What's the difference between pipelined and s-step Krylov methods?*

- A: Global communication is hidden vs avoided
- A: Off-the-shelf preconditioning possible vs specialized preconditioning

Q: *Is the code available online?*

- A: Yes, pipe-CG, Gropp-CG, pipe-CR and p(ℓ)-GMRES are in the PETSc library