Implementing a Blocked Aasen's Algorithm with a Dynamic Schedular on Multicore Architectures

Grey Ballard, Dulceneia Becker, James Demmel, Jack Dongarra, Alex Druinsky, Inon Peled, Oded Schwartz, Sivan Toledo, Ichitaro Yamazaki

University of Tennessee, Knoxville, USA
Tel-Aviv University, Israel
University of California, Berkeley, USA

International Parallel & Distributed Processing Symposium (IPDPS)
Boston, Massachusetts, 05/23/2013

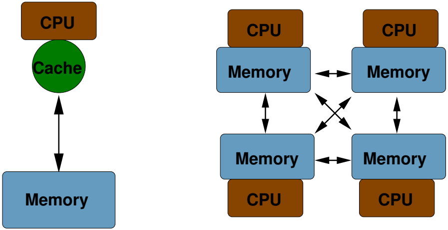
Communication-avoiding, direct linear algebra

- gaps between arithmetic and communication costs is increasing

$$\frac{\text{time}}{\text{flop}} \ll \frac{1}{\text{bandwidth}} \ll \text{latency}$$

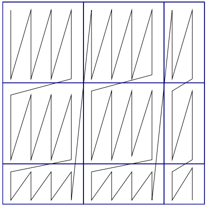
$\rightarrow$ computation-bound algorithm on a current machine could become communication-bound on a next machine.

- reduce runtime (or energy) by avoiding communication.
  - new algorithm with new numerical properties and bounds.



PLASMA: tiled-algorithm with DAG based dynamic scheduler

- **tiled algorithm**: consists of tasks on tiles
  - tile = block stored in contiguous memory
  - fine-grained parallelism and cache friendly.



- **QUARK**: QUeing And Runtime for Kernels
  - run a "sequential" code in parallel on a multicore
  - schedule task as soon as all dependencies are satisfied
    → synchronization avoiding

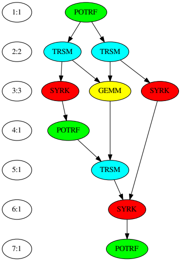
Cholesky factorization with QUARK

```
for (k = 0; k ¡ A.mt; k++) {
    QUARK_dpotrf(A(k, k))          (factor diagonal)
    for (m = k+1; m < A.mt; m++)        (compute off-diagonal)
        QUARK_dtrsm(A(k, k), A(m, k));

    for (m = k+1; m < A.mt; m++) {      (update trailing submatrix)
        QUARK_dsyrk(A(m, k), A(m, m));
        for (n = k+1; n < m; n++)
            QUARK_dgemm(A(m, k), A(n, k), A(m, n));
    }
}
```

Specifying dependencies with QUARK

```
void QUARK_dtrsm(double *L, double *B) {   (compute B := L^{-1}B)
    QUARK_Insert_Task(
        sizeof(double)*nb*nb, L, INPUT,
        sizeof(double)*nb*nb, B, INOUT );
}
```



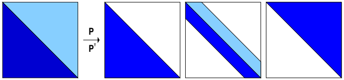
Problem description: direct linear algebra

- **setup**: given a matrix A that is

  dense ($a_{ij} \neq 0$), symmetric ($A = A^T$), and **indefinite** ($x^* Ax > 0 > y^* Ay$).

- **objective**: compute a permutation P for a "stable" factorization of A,

$$PAP^T = LBL^T,$$



  where L is unit-lower triangular and B is banded (on a shared-memory machine).

- **motivation**: used for solving

$$Ax = b.$$

  - needed in many scientific and engineering simulations:
    - discretized Maxwell equations with BEM, optimization problems for structural, acoustics, or electromagnetic physics, augmented linear least-squares problem, and etc. etc..

pivoting strategies for stable factorization of a dense symmetric indefinite matrix

Year	factorization (authors)	flops, $\frac{n^3}{3}$	compare	backward stable	$misses	algorithm/implementation
1970	LTL^T (Parlett-Reid)	2	$O(n^2)$	conditional	$O(\frac{n^3}{B})$	column-wise, right-look
1971	LDL^T (Bunch-Parlett)	1	$O(n^3)$	stable	$O(\frac{n^3}{B})$	column-wise, right-look
1971	LTL^T (Aasen) ✓	1	$O(n^2)$	conditional	$O(\frac{n^3}{B})$	column-wise, left-look PR
1977	LDL^T (Bunch-Kaufman) ✓	1	$O(n^2)$	conditional	$O(\frac{n^3}{BM/n})$	left-look panel, right-look submatririx-update, LAPACK
1998	LTL^T (Ashcraft-Grimes-Lewis)	1	$O(n^3)$	stable	$O(\frac{n^3}{B})$	fast BP
1998	LDL^T (Ashcraft-Grimes-Lewis)	1	$O(n^3)$	stable	$O(\frac{n^3}{B})$	stable BK (Rook pivot) LAPACK
2010	LTL^T (Rozloznik-Shklarski-ST)	$1+\frac{1}{n_b}$	$O(n^2)$	conditional	$O(\frac{n^3}{BM/n})$	PR on panel, Aasen to update
2012	LBL^T (AD,IP,ST,GB,JDem,OS) ✓	1	$O(n^2)$	conditional	$O(\frac{n^3}{B\sqrt{M}})$	blocked Aasen
2012	RBT (Baboulin,DB,JDon) ✓	1	0	probablistic	$O(\frac{n^3}{\sqrt{M}})$	right-look, tiled PLASMA

Difficult to develop an efficient/scalable implementation that both

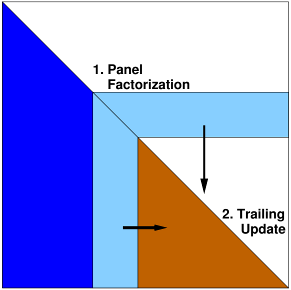
- takes advantage of symmetry and
- guarantees numerical stability through pivoting.

Outline:

1. algorithms
   - Bunch-Kaufman (LAPACK)
   - blocked Aasen
2. tiled implementation with a dynamic scheduler (QUARK/PLASMA)
3. performance and numerical results
4. final remarks

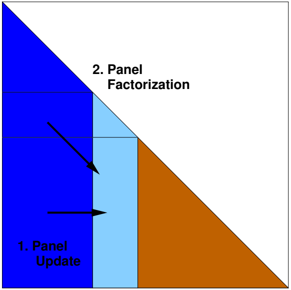
LAPACK: partitioned factorization

right-looking update (LAPACK Bunch-Kaufman)



- high parallelism
- poor locality for write

left-looking update (PLASMA blocked Aasen)

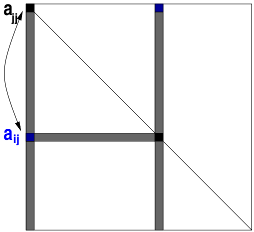


- limited parallelism
- good locality for write

LAPACK: Bunch-Kaufman algorithm to pick j-th pivot

$i := \text{argmax}\{|\mathbf{a}_{j:n,j}|\},\ \gamma_j = |a_{i,j}|$
if $\gamma_j == 0$ then $(\mathbf{a}_{j:n,j} = 0)$
  break (nothing to do)
else if $|a_{j,j}| \geq \alpha\gamma_j$ then
  pivot $a_{j,j}$
else
  $k := \text{argmax}\{|\mathbf{a}_{j:n,i}|\},\ \gamma_i = |a_{k,i}|$
  if $|a_{j,j}| \geq \alpha\gamma_j(\gamma_j/\gamma_i)$
    pivot $a_{j,j}$
  else if $|a_{i,i}| \geq \alpha\gamma_i$ then
    pivot $a_{i,i}$
  else
    pivot $\begin{pmatrix} a_{j,j} & a_{j,i} \\ a_{i,j} & a_{i,i} \end{pmatrix}$
  end if
end if

accept pivot $a_{j,j}$ if large enough compared with $a_{i,j} = \max_{r \neq j} a_{r,j}$

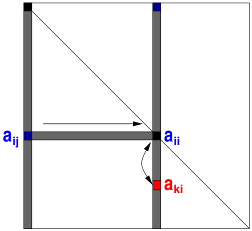


look for a large diagonal relative to its off-diagonals.

LAPACK: Bunch-Kaufman algorithm to pick j-th pivot

```
i := argmax{|a_{j:n,j}|}, γ_j = |a_{i,j}|
if γ_j == 0 then (a_{j:n,j} = 0)
    break (nothing to do)
else if |a_{j,j}| ≥ αγ_j then
    pivot a_{j,j}
else
    k := argmax{|a_{j:n,i}|}, γ_i = |a_{k,i}|
    if |a_{j,j}| ≥ αγ_j(γ_j/γ_r)
        pivot a_{j,j}
    else if |a_{i,i}| ≥ αγ_i then
        pivot a_{i,i}
    else
        pivot ( a_{j,j}  a_{j,i} ; a_{i,j}  a_{i,i} )
    end if
end if
```

accept pivot $a_{i,i}$ if large enough
compared with $a_{k,i} = \max_{r \neq j, r \neq i} a_{r,i}$



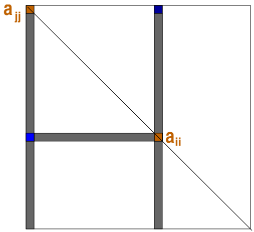
look for a large diagonal relative to its off-diagonals.

LAPACK: Bunch-Kaufman algorithm to pick j-th pivot

$i := \text{argmax}\{|\mathbf{a}_{j:n,j}|\}$, $\gamma_j = |a_{i,j}|$
if $\gamma_j == 0$ then $(\mathbf{a}_{j:n,j} = 0)$
    break (nothing to do)
else if $|a_{j,j}| \geq \alpha\gamma_j$ then
    pivot $a_{j,j}$
else
    $k := \text{argmax}\{|\mathbf{a}_{j:n,i}|\}$, $\gamma_i = |a_{k,i}|$
    if $|a_{j,j}| \geq \alpha\gamma_j(\gamma_j/\gamma_r)$
        pivot $a_{j,j}$
    else if $|a_{i,i}| \geq \alpha\gamma_i$ then
        pivot $a_{i,i}$
    else
        pivot $\begin{pmatrix} a_{j,j} & a_{j,i} \\ a_{i,j} & a_{i,i} \end{pmatrix}$
    end if
end if

look for a large diagonal relative to its off-diagonals.

form 2-by-2 pivot if both $a_{j,j}$ and $a_{i,i}$ were too small

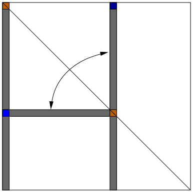


- compute $PAP^T = LDL^T$, where
  - D is block-diagonal with 1-by-1 or 2-by-2 diagonal blocks.
- is normwise backward stable (conditionally).

LAPACK: Bunch-Kaufman algorithm (implementational challenges)

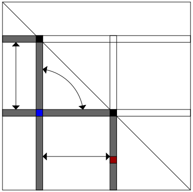
1. pivot selection

   - two reduction operations
     2nd column unknown till run-time and
     anywhere in trailing submatrix.
   - additional run-time dependency
     → global synchronization with a dynamic scheduler.
   - symmetric storage
     → irregular (additional) dependency/memory access.



2. symmetric swap (both columns and rows swapped)

   - two columns of length n are swapped
     ↔ only triangular part is stored and updated
   - symmetric storage
     - irregular memory access
     - row and col dependencies (swapped at once).



difficult to develop a scalable implementation

fork-join paradigm of LAPACK → panel becomes bottleneck.

column-wise Aasen's algorithm:

Aasen's idea: reduction to tridiagonal T,

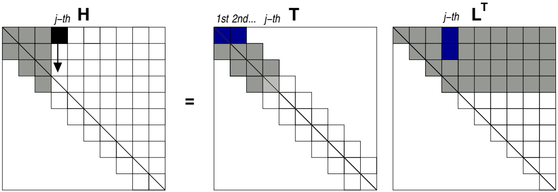
$$PAP^T = LTL^T = LH.$$

using auxiriary Hessenberg matrix $H = TL^T$ and left-looking algorithm.

For each j-th column of A,

1. compute j-th column $\mathbf{h}_j$ of H (three-term)

$$h_{i,j} = t_{i,i-1}\ell^T_{j,i-1} + t_{i,i}\ell^T_{j,i} + t_{i,i+1}\ell^T_{j,i+1} \qquad \text{for } i = 1, 2, \ldots, j.$$



$$\ell_{j,j}t_{j,j}\ell^T_{j,j} = a_{j,j} - \ell_{j,j}t_{j,j-1}\ell^T_{j,j-1} - \sum_{k=1}^{j-1}\ell_{j,k}h_{k,j}$$

column-wise Aasen's algorithm:

Aasen's idea: reduction to tridiagonal T, using auxiriary Hessenberg matrix $H = TL^T$ and left-looking algorithm;

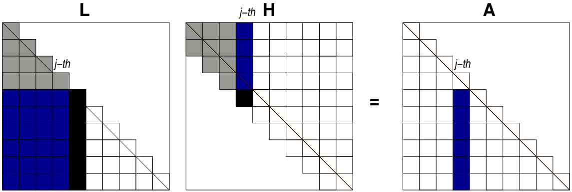
$$PAP^T = LTL^T = LH.$$

For each j-th column of A,

2. compute next column ℓ_{j+1} of L and $h_{j+1,j}$ (update+factor, just like LU)

$$\ell_{(j+1):n,j+1} h_{j+1,j} = \mathbf{a}_{(j+1):n,j} - \sum_{k=1}^{j} \ell_{(j+1):n,k} h_{k,j}.$$

- for numerical stablity, picks largest element as a pivot!!



column-wise Aasen's algorithm:

For each j-th column of A,

3. symmetrically pivot both rows and columns of $A_{j+1:n,j+1:n}$ (and rows of $L_{j+1:n,1:j}$).
4. extract $t_{j+1,j}$ from $h_{j+1,j}$ ($t_{j+1,j} = h_{j+1,j}\ell_{j,j}^{-T}$).

Left-looking Aasen's algorithm:
Advantages:
→ guarantees stability through a simple pivoting (just like LU).

→ updates only $\mathbf{a}_{j+1:n,j}$, performing total of $\frac{1}{3}n^3$ flops
(same as Bunch-Kauffman, and half of the right-looking version, Parlett-Reid).

Challenges:
→ exhibits limited parallelism (only one column $\mathbf{a}_j$ is updated at each step).

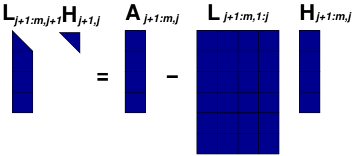
→ introduces a dependency (all the pivots must be applied to $\mathbf{a}_j$ before updating it).

blocked Aasen's algorithm:

Replace element-wise operations with block-wise operations: T is now banded.

1. compute the j-th block column $\mathbf{H}_j$ (three-term)
   - for stable factorization, symmetry of $T_{j,j}$ must be maintained through symmetric solve
2. compute the $(j+1)$-th column L_{j+1} (panel factorization, tall-skiny LU)

$$L_{(j+1):m,j+1} H_{j+1,j} = (A_{(j+1):n,j} - \sum_{k=1}^{j} L_{(j+1):n,k} H_{k,j}) P^{(j+1)}.$$



$\rightarrow$ depends only on panel, and can use any "communication-avoiding" LU.

3. pivot $L_{j+1:n,1:j}$ and $A_{j+1:n,j+1:n}$ (symmetric pivoting)
4. extract $T_{j+1,j}$ from $H_{j+1,j}$ ($T_{j+1,j} = H_{j+1,j} L_{j,j}^{-T}$)

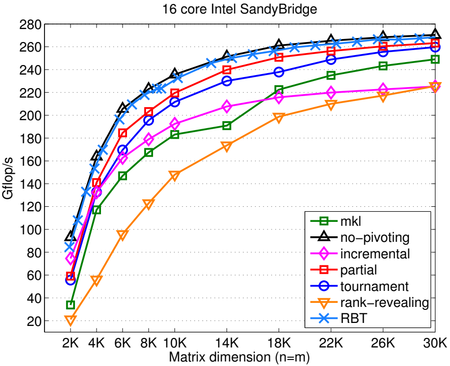
Comparing blocked Aasen's and Bunch-Kaufman algorithms

algorithm (factorization)	flops	backward stable	\$misses	algorithm/implementation
Bunch-Kaufman (LDL^T)	$\frac{1}{3}n^3 + O(n^2)$	conditional	$O(\frac{n^3}{BM/n})$	right-look, column-wise panel
blocked Aasen (LBL^T)	$\frac{1}{3}n^3 + O(n^2 n_b)$	conditional	$O(\frac{n^3}{B\sqrt{M}})$	left-look, TSLU panel

about the same number of flops but with less "communication"

$\rightarrow$ implemented in PLASMA (synchronization-avoiding)

LU factorizations in PLASMA:



- several LU algorithms are available
  - recursive partial, tournament, incremental, random-butterfly, no-pivoting

    A survery of recent parallel Gaussian elimination

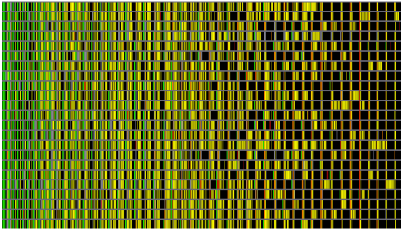
    Donfack, JDon, Faverge, Gates, Kurzak, Luszek, IY.
  - reank-revealing pivoting

    LU factorization with panel rank reveling pivoting

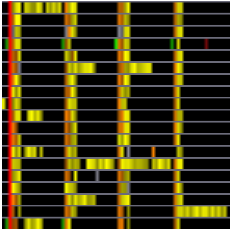
    Khabou, JDem, Grigori, Gu

Improving performance of blocked Aasen's:

Initial performance was not ideal:



- EzTrace on 24 core AMD Opteron ($n = 5000$, $n_b = 100$) -



j-th step performs reductions (left-looking)

$$\mathbf{a}_{i,j} := \mathbf{a}_{i,j} - \sum_{k=1}^{j-1} \ell_{i,k} h_{k,j} \quad \text{for } i = j, j+1, \ldots, n_t.$$

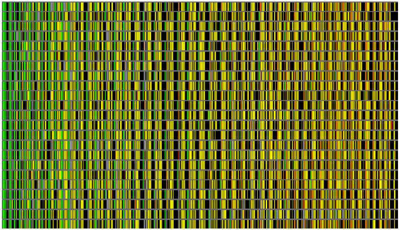
Initial performance of blocked Aasen's:

use workspaces to perform binary-reduction:

$$w_1 = \sum_{k=1}^{h} \ell_{i,k} h_{k,j} \qquad w_1 = w_1 + w_2$$

$$w_2 = \sum_{k=h+1}^{2h} \ell_{i,k} h_{k,j} \qquad \vdots$$

$$\vdots$$

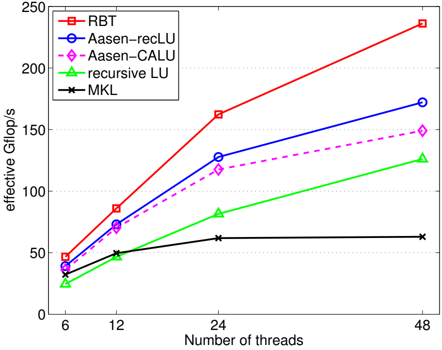


- breaks a reduction operation into independent tasks
- starts accumulating updates before destination block $a_{i,j}$ is ready

a few other techniques (e.g., symmetric pivoting) described in the paper.

Current performance of blocked Aasen's:

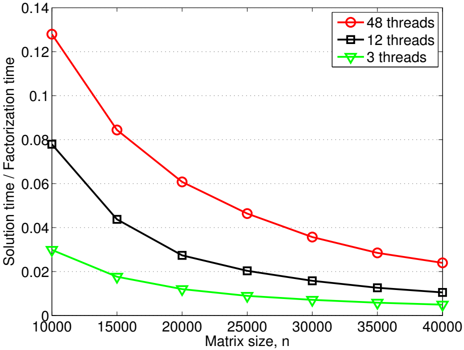
strong-scaling on eight 6-core 2.8MHz AMD Opteron (n=45K).



- On 6 and 48 cores, blocked Aasen with recursive-panel obtains
  - about 83% and 73% of RBT Gflop/s
  - speedups of about 1.6 and 1.4 over recursive LU.
- Block Aasen with tournament pivoting was slightly slower
  - difficult to overlap with left-looking update

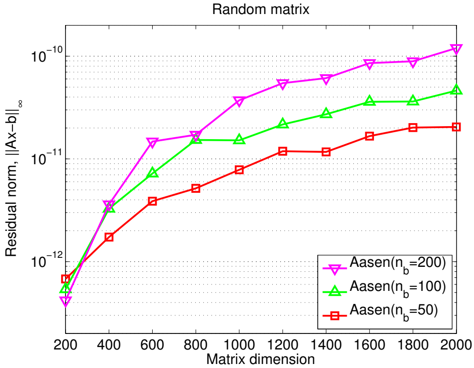
Current performance of blocked Aasen's:

blocked Aasen's computes $PAP^T = LTL^T$, where T is banded.



- Solution time does not scale as well as factorization time
  - about 80 — 90% of solution time spent in banded solver GBSV of LAPACK

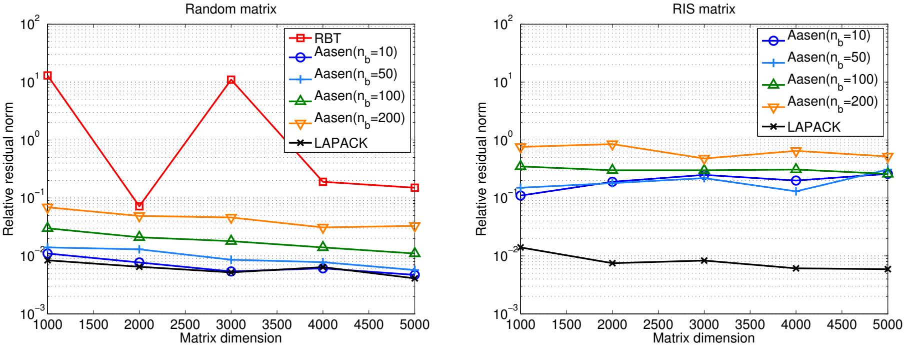
Numerical behavior of blocked Aasen's with partial pivoting LU:



- Residual norms increase slightly (proportionally) with the block size.

  - seems to be due to the growth in $\max_{i,j}(|L||T||L|^T)_{i,j}$.

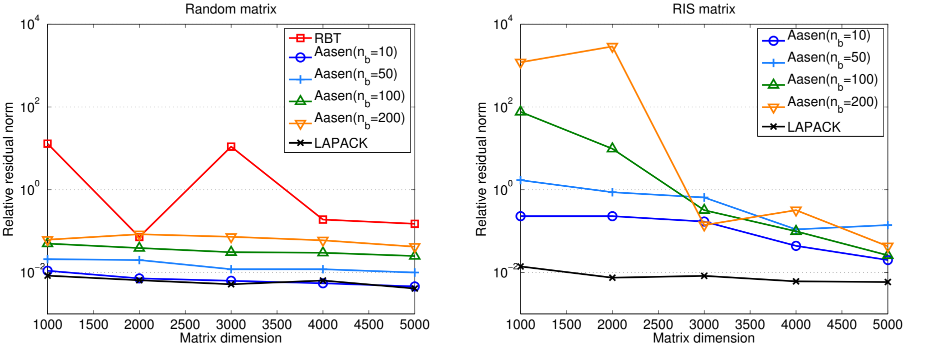
Numerical behavior of blocked Aasen's with partial pivoting LU:



relative residual norm $= \|b - Ax\| / (n\epsilon\|b\| + \|A\|\|x\|)$

- blocked Aasen obtained quite stable/robust performance
  - lost a couple of digits compared to LAPACK (proportional to block size)
  - was able to factorize "hard" matrix, where RBT failed
- iterative refinements would lower the residual norms of RBT
  (if the factorization is succesful).

Numerical behavior of blocked Aasen's with tournament pivoting LU:



relative residual norm $= \|b - Ax\| / (n\epsilon\|b\| + \|A\|\|x\|)$

- for some matrices, blocked Aasen became unstable with tournament pivoting
  - is due to low-rank/singular off-diagonal blocks (CALU panel lead to large growth-factor)
  - may be fixed using rank-revealing pivoting

Summary:

- blocked Aasen with a potential of being scalable and numerically stable.
- nice joint research between applied math and computer science
  - looking for applications.
- dynamic scheduler QUARK to speedup an efficient implementation/prototyping.

Current studies:

- other pivoting strategies (e.g., rank-revealing) for panel factorization.
- more performance profiling (e.g., "communication" or "energy" costs).
- larger-scale experiments (distributed-memory system with PaRSEC or QUARKd).
  - more scalable banded solver.
- more theoretial (stablity, etc.) understanding.

Thank you.