Performance of dense linear algebra software on multicore architectures

Emmanuel AGULLO
Jack DONGARRA
Bilel HADRI
Jakub KURZAK
Hatem LTAIEF
Piotr LUSZCZEK

CScADS Workshop on Libraries and Autotuning for Petascale Applications, Tahoe City, TN, August 10-12, 2009
1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
Outline

1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
**Tile Cholesky Factorization**

FOR $k = 0..\text{TILES}-1$
FOR $n = 0..k-1$
    $A[k][k] \leftarrow \text{DSYRK}(A[k][n], A[k][k])$
    $A[k][k] \leftarrow \text{DPOTRF}(A[k][k])$
FOR $m = k+1..\text{TILES}-1$
    FOR $n = 0..k-1$
        $A[m][k] \leftarrow \text{DGEMM}(A[k][n], A[m][n], A[m][k])$
        $A[m][k] \leftarrow \text{DTRSM}(A[k][k], A[m][k])$

- Basically identical to the block algorithm (LAPACK).
- Input matrix stored and processed by square tiles.
- Complex DAG.
Tile QR (&LU) Factorization

\[
\text{FOR } k = 0..\text{TILES-1} \\
A[k][k], T[k][k] \leftarrow \text{DGRQRT}(A[k][k]) \\
\text{FOR } m = k+1..\text{TILES-1} \\
A[k][k], A[m][k], T[m][k] \leftarrow \text{DTSQRT}(A[k][k], A[m][k], T[m][k]) \\
\text{FOR } n = k+1..\text{TILES-1} \\
A[k][n] \leftarrow \text{DLARFB}(A[k][k], T[k][k], A[k][n]) \\
\text{FOR } m = k+1..\text{TILES-1} \\
A[k][n], A[m][n] \leftarrow \text{DSSRFB}(A[m][k], T[m][k], A[k][n], A[m][n])
\]

- Different from the block algorithm \(\rightarrow\) new serial kernels.
- Derived from out-of-core algorithm.
- Input matrix stored and processed by square tiles.
- (Even more) Complex DAG.

\[\text{DGEQRT} \quad \text{DLARFB} \quad \text{DLARFB}\]

\[\text{DTSQRT} \quad \text{DSSRFB} \quad \text{DSSRFB}\]

\[\text{DTSQRT} \quad \text{DSSRFB} \quad \text{DSSRFB}\]

\[\text{DTSQRT} \quad \text{DSSRFB} \quad \text{DSSRFB}\]

\[\text{DTSQRT} \quad \text{DSSRFB} \quad \text{DSSRFB}\]
Expected behaviour of tile algorithms

(Expected behaviour)

_possible_overheads:

- extra-flops;
- kernels not optimized.

(expected_benefits):

- better data reuse;
- better scheduling opportunities (parallelism brought to the fore).
Impact of scheduling model on performance (Cholesky)
1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
Libraries

★ LAPACK:
  ▶ LAPACK 3.2 on Intel machine;
  ▶ LAPACK 3.1.1 on IBM machine;

★ SCALAPACK:
  ▶ SCALAPACK 1.8.0;

★ Vendor libraries:
  ▶ Intel MKL 10.1;
  ▶ IBM ESSL 4.3;
  ▶ IBM PESSL 3.3;

★ Tile algorithms:
  ▶ PLASMA ;
  ▶ TBLAS.
Libraries

★ LAPACK:
  ▶ LAPACK 3.2 on Intel machine;
  ▶ LAPACK 3.1.1 on IBM machine;

★ SCALAPACK:
  ▶ SCALAPACK 1.8.0;

★ Vendor libraries:
  ▶ Intel MKL 10.1;
  ▶ IBM ESSL 4.3;
  ▶ IBM PE SSL 3.3;

★ Tile algorithms:
  ▶ PLASMA;
  ▶ TBLAS.
Intel Xeon - 16 cores machine

- Node:
  - Quad-socket quad-core Intel64 processors (16 cores).

- Intel Xeon processor:
  - Quad-core (two dual-core units);
  - L1: 32 kB data + 32 kB instructions;
  - L2: 4 MB per dual core;
  - Frequency: 2.4 GHz.

- Theoretical peak:
  - 9.6 Gflop/s/core;
  - 153.6 Gflop/s/node.

- System and compilers:
  - Linux 2.6.25;
  - Intel Compilers 11.0.
IBM Power6 - 32 cores machine

★ Node:
  ▶ 16 dual-core Power6 processors (32 cores).

★ Power6 processor:
  ▶ dual-core;
  ▶ L1: 64kB data + 64 kB instructions;
  ▶ L2: 4 MB per core, accessible by the other core;
  ▶ L3: 32 MB per processor, one controller per core (80 GB/s).
  ▶ Frequency: 4.7 GHz.

★ Theoretical peak:
  ▶ 18.8 Gflop/s/core;
  ▶ 601.6 Gflop/s/node.

★ System and compilers:
  ▶ AIX 5.3;
  ▶ xlf version 12.1;
  ▶ xlc version 10.1.
Performance metrics (How to read the graphs)

- Performance: Gflop/s (y-axis).
- Plots scaled to the theoretical peak.
- Parallel DGEMM.
- Upper bound: computational intensive serial kernel running embarrassingly parallel:
  - DPOTRF \( (LL^T) \rightarrow \text{dgemm-seq}; \)
  - DGEQRF \( (QR) \rightarrow \text{dssrfb-seq}; \)
  - DGETRF \( (LU) \rightarrow \text{dssssm-seq}. \)
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM
6. Conclusion and current work
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM
6. Conclusion and current work
**Problem**

**Degrees of freedom (search space)**

Input parameters of the computational intensive serial kernel:

- **NB**: tile size;
- **IB**: internal blocking (for dssrfb and dssssm only).

**Objective**

For a given matrix size $N$, find the (NB,IB) value that maximizes the performance by experimenting.
Sampling step

Problem

Degrees of freedom (search space)

Input parameters of the computational intensive serial kernel:

- **NB**: tile size;
- **IB**: internal blocking (for dssrfb and dssssm only).

Objective

For a given matrix size $N$, find the (NB,IB) value that maximizes the performance by experimenting.
Outline

1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
Impact of NB - DPOTRF - Intel64 - 16 cores

Matrix size

0 2000 4000 6000 8000 10000 12000

0 20 40 60 80 100 120 140

16xdgemm-seq

NB=196
NB=168
NB=120
NB=84
NB=60

Sampling step
Impact of (NB,IB)
Impact of NB - DPOTRF - Power 6 - 32 cores

Matrix size vs. Performance

- NB=288
- NB=224
- NB=220
- NB=192
- NB=160
- NB=120
- NB=80
- NB=60

32xdgemm-seq

Sampling step
Impact of (NB,IB)
Impact of (NB,IB) - DGEQRF - Intel64 - 16 cores

Matrix size vs. Performance (16xdssrfb-seq)

- NB=256 IB=64
- NB=200 IB=40
- NB=168 IB=56
- NB=120 IB=40
- NB=84 IB=28
- NB=60 IB=20

Sampling step

Impact of (NB,IB)
Impact of (NB,IB) - DGEQRF - Power6 - 32 cores

Matrix size

Sampling step

Impact of (NB,IB)

0 2000 4000 6000 8000 10000 12000

0 50 100 150 200 250 300

NB=480 IB=96
NB=340 IB=68
NB=300 IB=60
NB=256 IB=64
NB=168 IB=56
NB=160 IB=40
NB=120 IB=40
NB=80 IB=40

16xdssrfb-seq
Sampling step

Impact of (NB,IB)

Impact of (NB,IB) - DGETRF - Intel64 - 16 cores

Matrix size

NB=252 IB=28
NB=196 IB=28
NB=168 IB=28
NB=120 IB=24
NB=84 IB=28
NB=60 IB=20

16xdsssssm-seq

Performance of dense linear algebra software on multicore architectures
Impact of (NB,IB) - DGETRF - Power6 - 32 cores

Matrix size vs Performance

- NB=480 IB=60
- NB=340 IB=68
- NB=280 IB=56
- NB=240 IB=60
- NB=196 IB=28
- NB=168 IB=28
- NB=120 IB=40
- NB=80 IB=20

32xdsssssm-seq
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM
6. Conclusion and current work
Exhaustive Search

For “each” matrix size and number of cores:

1. **Time** PLASMA on all (NB,IB) samples;
2. Select the best sample.

**Number of samples**

- \( |\{(IB, NB) \mid IB|NB, 40 \leq NB \leq 500, 4 \leq IB \leq NB\}| = 1352; \)
- All combinations cannot be explored on large executions;

→ Need for a Pruned Search.
Exhaustive VS Pruned Search

**Exhaustive Search**

For "each" matrix size and number of cores:

1. Time \texttt{PLASMA} on all \((NB,IB)\) samples;
2. Select the best sample.

**Number of samples**

- \(|\{(IB,NB) \mid IB|NB, 40 \leq NB \leq 500, 4 \leq IB \leq NB\}| = 1352;\)
- All combinations cannot be explored on large executions;

→ Need for a Pruned Search.
Sampling step

Exhaustive VS Pruned Search

Pruned Search

Method

1. Time serial core kernels (dgemm, dssrfb, dssssm).

2. Pick up the "best" NB or (NB,IB) samples (pruning);

3. Select one per matrix size and number of cores.

Intel64 - dgemm

Power6 - dssrfb
Pruned Search

Method

1. Time serial core kernels (dgemm, dssrfb, dssssm).

2. Pick up the "best" NB or (NB,IB) samples (pruning);

3. Select one per matrix size and number of cores.
Pruned Search

Method

1. Time serial core kernels (dgemm, dssrfb, dssssm).

2. Pick up the "best" NB or (NB,IB) samples (pruning);

3. Select one per matrix size and number of cores.
Exhaustive Search VS Pruned Search

Intel64 - 16 cores - DPOTRF

Performance of dense linear algebra software on multicore architectures
Exhaustive Search VS Pruned Search

Intel64 - 16 cores - DGEQRF

Performance of dense linear algebra software on multicore architectures

EXPLASMA

PLASMA

16xdssrfb-seq
Exhaustive Search VS Pruned Search

Intel64 - 16 cores - DGETRF

Performance of dense linear algebra software on multicore architectures
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM
6. Conclusion and current work
Other software

- **PLASMA**: Pruned Search.
- **TBLAS**: Exhaustive Search.
- **SCALAPACK, PESSL**: Exhaustive Search.
- **LAPACK, MKL, ESSL**: tuned by vendor.
Comparison against other libraries

DPOTRF - Intel64 - 16 cores

Matrix size

PLASMA
TBLAS
MKL
SCALAPACK
LAPACK

Performance of dense linear algebra software on multicore architectures
Comparison against other libraries

DPOTRF - Power6 - 32 cores

Performance of dense linear algebra software on multicore architectures
Comparison against other libraries

**DGEQRF - Intel64 - 16 cores**

![Graph showing performance comparison of dense linear algebra software](image)

- **DGEMM**
- **PLASMA**
- **TBLAS**
- **MKL**
- **SCALAPACK**
- **LAPACK**

Matrix size vs. Performance

Related readings:

- Performance of dense linear algebra software on multicore architectures
Comparison against other libraries

DGEQRF - Power6 - 32 cores

Matrix size

- DGEMM
- PLASMA
- TBLAS
- ESSL
- PESSL
- SCALAPACK
- LAPACK

Performance of dense linear algebra software on multicore architectures
Comparison against other libraries

**DGETRF - Intel64 - 16 cores**

- **DGEMM**
- **PLASMA**
- **MKL**
- **SCALAPACK**
- **LAPACK**

Performance of dense linear algebra software on multicore architectures

- **Matrix size**
  - 0, 2000, 4000, 6000, 8000, 10000, 12000

- **16xdsssssm-seq**
Comparison against other libraries

DGETRF - Power6 - 32 cores

Matrix size vs. Performance for various libraries:
- DGEMM
- PLASMA
- ESSL
- PESSL
- SCALAPACK
- LAPACK

Graph shows performance of dense linear algebra software on multicore architectures.
Outline

1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM
6. Conclusion and current work
Problem

Degrees of freedom

Input parameters of the computational intensive serial kernel:

- NB: tile size;
- IB: internal blocking (for dssrfb and dssssm only).

Objective

For a given matrix size $N$, find the (NB,IB) value that maximizes the performance from samples obtained on other matrix sizes.

Preliminary Work

Establish an accurate enough performance model.
Problem

Degrees of freedom

Input parameters of the computational intensive serial kernel:

- **NB**: tile size;
- **IB**: internal blocking (for dssrfb and dssssm only).

Objective

For a given matrix size $N$, find the (NB,IB) value that maximizes the performance from samples obtained on other matrix sizes.

Preliminary Work

Establish an accurate enough performance model.
Problem

Degrees of freedom
Input parameters of the computational intensive serial kernel:
- \textbf{NB}: tile size;
- \textbf{IB}: internal blocking (for dssrfb and dssssm only).

Objective
For a given matrix size \( N \), find the (NB,IB) value that maximizes the performance from samples obtained on other matrix sizes.

Preliminary Work
Establish an accurate enough performance model.
Building blocks

Given a tile size $NB$ and a number of cores, the matrix size $N$ varying:

- Serial computational intensive kernel
  → Upper bound performance;
- $N/NB$ → DAG → Degree of parallelism
Performance Model

Building blocks

Given a tile size $NB$ and a number of cores, the matrix size $N$ varying:

- Serial computational intensive kernel → Upper bound performance;
- $N/NB$ → DAG → Degree of parallelism

Cholesky degree of parallelism:
Extrapolation step

Performance Model

Building blocks

Given a tile size \( NB \) and a number of cores, the matrix size \( N \) varying:

- Serial computational intensive kernel
  \( \rightarrow \) Asymptotic performance;
- \( N/NB \) \( \rightarrow \) DAG \( \rightarrow \) Degree of parallelism

DGEMM: \( C \leftarrow C + A \times B \)
\( \rightarrow \) No dependencies
Outline

1. Tile Algorithms for 1-sided dense factorizations
2. Experimental environment
3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search
4. Comparison against other libraries
5. Extrapolating step (preliminary ideas)
   - Problem
     - Tentative experiments with DGEMM
6. Conclusion and current work
Serial DGEMM $c \leftarrow c + a \times b$ - Impact of flushing

![Graph showing the performance of dense linear algebra software on multicore architectures](image-url)
Serial DGEMM $c \leftarrow c + a \times b$ - Impact of flushing

![Graph showing performance of dense linear algebra software on multicore architectures](image-url)
Extrapolation step

Tentative experiments with DGEMM

DGEMM- 1 cores

Matrix size

NB=480
NB=360
NB=240
NB=196
NB=168
NB=120
NB=84
NB=60

1xdgemm-seq

Performance of dense linear algebra software on multicore architectures
Extrapolation step
Tentative experiments with DGEMM

**DGEMM - 16 cores**

Performance of dense linear algebra software on multicore architectures
Extrapolation step  
Tentative experiments with DGEMM

DGEMM- 16 cores

Matrix size

0  2000  4000  6000  8000  10000  12000

0  20  40  60  80  100  120  140

MKL 10.1
PLASMA
PLASMABDL

16xdgemm-seq

Performance of dense linear algebra software on multicore architectures
1. Tile Algorithms for 1-sided dense factorizations

2. Experimental environment

3. Sampling step
   - Problem
   - Impact of (NB,IB)
   - Exhaustive VS Pruned Search

4. Comparison against other libraries

5. Extrapolating step (preliminary ideas)
   - Problem
   - Tentative experiments with DGEMM

6. Conclusion and current work
Conclusion and current work

Conclusion

1. Efficient sampling through pruned search.
2. Building blocks for extrapolation:
   - Serial computational intensive kernel as an asymptote / as an upper bound;
   - Degree of parallelism as a proportion of this asymptotic value / of this upper bound.

Current work

★ Turn these ideas into an autotuning component for PLASMA;
★ Parallelism within a panel (Tile CAQR);
★ PLASMA 2.0: http://icl.cs.utk.edu/plasma/;
★ Extension to distributed memory machines;
★ MAGMA 0.1 (multicore + GPU): http://icl.cs.utk.edu/magma/.
Conclusion and current work

Conclusion

1. Efficient sampling through pruned search.
2. Building blocks for extrapolation:
   - How much can we (should we) rely on the model?
   - Enough to look for crossover points.

Current work

- Turn these ideas into an autotuning component for PLASMA;
- Parallelism within a panel (Tile CAQR);
- PLASMA 2.0: http://icl.cs.utk.edu/plasma/;
- Extension to distributed memory machines;
- MAGMA 0.1 (multicore + GPU): http://icl.cs.utk.edu/magma/.
Conclusion

1. Efficient sampling through pruned search.
2. Building blocks for extrapolation:
   ▶ How much can we (should we) rely on the model?
   ▶ Enough to look for crossover points.

Current work

★ Turn these ideas into an autotuning component for PLASMA;
★ Parallelism within a panel (Tile CAQR);
★ PLASMA 2.0: http://icl.cs.utk.edu/plasma/;
★ Extension to distributed memory machines;
★ MAGMA 0.1 (multicore + GPU):
  http://icl.cs.utk.edu/magma/.
Questions?
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Tile Cholesky Factorization - Static pipeline

- Work partitioned in one dimension (by block-rows).
- Cyclic assignment of work across all steps of the factorization (pipelining of factorization steps).
- Process tracking by a global progress table.
- Stall on dependencies (busy waiting).

```c
void dsyrk(double *A, double *T);
void dpotrf(double *T);
void dgemm(double *A, double *B, double *C);
void dtrsm(double *T, double *C);

k = 0; m = my_core_id;
while (m >= TILES) {
    k++; m = m-TILES+k;
} n = 0;

while (k < TILES && m < TILES) {
    next_n = n; next_m = m; next_k = k;

    next_n++;
    if (next_n > next_k) {
        next_m += cores_num;
        while (next_m >= TILES && next_k < TILES) {
            next_k++; next_m = next_m-TILES+next_k;
        } next_n = 0;
    }

    if (m == k) {
        if (n == k) {
            dpotrf(A[k][k]);
            core_progress[k][k] = 1;
        } else {
            while(core_progress[k][n] != 1);
            dsyrk(A[k][n], A[k][k]);
        }
    } else {
        if (n == k) {
            dtrsm(A[k][k], A[m][k]);
            core_progress[m][k] = 1;
        } else {
            while(core_progress[k][n] != 1);
            while(core_progress[m][n] != 1);
            dgemm(A[k][n], A[m][n], A[m][k]);
        }
    }
}
```
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Tile QR Factorization - Static pipeline

```c
void dgeqrt(double *R1, double *T);
void dtqrt(double *R, double *V2, double *T);
void dlqrt(double *V1, double *T, double *C1);
void dssqrt(double *V2, double *T, double *C1, double *C2);

k = 0; n = my_core_id;
while (n >= TILES) {
    k++; n = n-TILES+k;
} m = k;
while (k < TILES && n < TILES) {
    next_n = n; next_m = m; next_k = k;
    next_m++;
    if (next_m == TILES) {
        next_n += cores_num;
        while (next_n >= TILES && next_k < TILES) {
            next_k++; next_n = next_n-TILES+next_k;
        } next_m = next_k;
    }
    if (n == k) {
        if (m == k) {
            while(progress[k][k] != k-1);
            dgeqrt(A[k][k], T[k][k]);
            progress[k][k] = k;
        }
        else {
            while(progress[m][k] != k-1);
            dtqrt(A[k][k], A[m][k], T[k][k]);
            progress[m][k] = k;
        }
    }
    else {
        while(progress[k][k] != k);
        while(progress[k][n] != k-1);
        dlqrt(A[k][k], T[k][n], A[k][n]);
    }
    else {
        while(progress[m][k] != k);
        while(progress[m][n] != k-1);
        dssqrt(A[m][k], T[m][n], A[m][n]);
    }
    n = next_n; m = next_m; k = next_k;
}
```

- **Work partitioned in one dimension (by block-rows).**
- **Cyclic assignment of work across all steps of the factorization (pipelining of factorization steps).**
- **Process tracking by a global progress table.**
- **Stall on dependencies (busy waiting).**
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL-ESSL
   - SCALAPACK-PESSL
   - LAPACK

5. DGEMM on Intel64
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. **PLASMA** scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Comparison against other libraries on few cores

**DPOTRF**

**DPOTRF - Intel64 - 4 cores**

Matrix size vs. Performance of dense linear algebra software on multicore architectures

- **DGEMM**
- **PLASMA**
- **TBLAS**
- **MKL**
- **SCALAPACK**
- **LAPACK**

**4xdgemm-seq**

Performance levels range from 0 to 40000.
Comparison against other libraries on few cores

DPOTRF - Power6 - 2 cores

Performance of dense linear algebra software on multicore architectures
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Comparison against other libraries on few cores

DGEQRF

DGEQRF - Intel64 - 4 cores

Matrix size

0 2000 4000 6000 8000 10000 12000

0 5 10 15 20 25 30 35

DGEMM
PLASMA
TBLAS
MKL
SCALAPACK
LAPACK

4xdssrfb-seq

Performance of dense linear algebra software on multicore architectures
Comparison against other libraries on few cores

DGEQRF - Power6 - 2 cores

Performance of dense linear algebra software on multicore architectures
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Comparison against other libraries on few cores

DGETRF

DGETRF - Intel64 - 4 cores

Performance of dense linear algebra software on multicore architectures
Comparison against other libraries on few cores

DGETRF - Power6 - 2 cores

![Graph showing performance comparison of different libraries on Power6 architecture with 2 cores. The x-axis represents matrix size, while the y-axis represents performance. The libraries compared include DGEMM, PLASMA, ESSL, PESSL, SCALAPACK, and LAPACK. The graph highlights the performance of 2xdssssm-seq matrix size.](image-url)
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. **PLASMA scalability**

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Performance of dense linear algebra software on multicore architectures
PLASMA scalability

PLASMA-DGEQRF-Intel64

Matrix size

16 cores
14 cores
12 cores
10 cores
8 cores
6 cores
4 cores
2 cores
1 core

16xdssrfb-seq

Performance of dense linear algebra software on multicore architectures
Performance of dense linear algebra software on multicore architectures
Performance of dense linear algebra software on multicore architectures
Performance of dense linear algebra software on multicore architectures

PLASMA scalability

PLASMA-DGEQRF-Power6

Matrix size

32 cores
16 cores
8 cores
4 cores
2 cores
1 core

32xdssrfb-seq
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Scalability of other libraries

TBLAS-DPOTRF-Intel64

Matrix size

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

TBLAS - DGEQRF - Intel64

Matrix size

Performance of dense linear algebra software on multicore architectures

PLASMA group
Scalability of other libraries

TBLAS - DPOTRF - Power6

Matrix size

32 cores
16 cores
8 cores
4 cores
2 cores
1 core

32xdgemm-seq

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

TBLAS - DGEQRF - Power6

![Graph showing the performance of TBLAS-DGEQRF on a Power6 architecture with different core counts. The x-axis represents the matrix size, and the y-axis represents performance. The graph includes lines for 32, 16, 8, 4, 2, and 1 core counts, with each line showing the performance on a sequence of matrix sizes.](image)

TBLAS group

Performance of dense linear algebra software on multicore architectures
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Scalability of other libraries

MKL - ESSL

MKL - DPOTRF - Intel64

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

MKL - ESSL

MKL - DGEQRF - Intel64

Matrix size

16 cores
8 cores
4 cores
2 cores
1 core

16xdsrffb-seq

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

ESSL - DPOTRF - Power6

Matrix size

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

**ESSL - DGEQRF - Power6**

![Graph showing scalability of ESSL-DGEQRF on Power6](image)

Matrix size vs. Performance

- **32 cores**
- **16 cores**
- **8 cores**
- **4 cores**
- **2 cores**
- **1 core**

**Legend:**
- **Red** for 32 cores
- **Green** for 16 cores
- **Blue** for 8 cores
- **Purple** for 4 cores
- **Orange** for 2 cores
- **Gray** for 1 core

**Axes:**
- **Y-axis:** Performance (units not specified)
- **X-axis:** Matrix size (units not specified)

**Note:**
- The graph illustrates the performance of dense linear algebra software on multicore architectures.
Scalability of other libraries

ESSL - DGETRF - Power6

Performance of dense linear algebra software on multicore architectures
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
Scalability of other libraries

SCALAPACK - PESSL

SCALAPACK - DPOTRF - Intel 64

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

Scalapack vs Pessl

SCALAPACK - DPOTRF - Power6

Performance of dense linear algebra software on multicore architectures

Matrix size vs 32xdgemm-seq for 32, 16, 8, 4, 2, and 1 cores.
Scalability of other libraries

PESSL - DPOTRF - Power6

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

SCALAPACK - DGEQRF - Intel64

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

SCALAPACK - PESSL

SCALAPACK - DGETRF - Power6

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

PESSL - DGEQRF - Power 6

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

SCALAPACK- DGETRF- Intel64

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

SCALAPACK - PESSL

SCALAPACK - DGETRF - Power6

Performance of dense linear algebra software on multicore architectures
Performance of dense linear algebra software on multicore architectures

Matrix size vs. performance for different core counts.
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL-ESSL
   - SCALAPACK-PESSL
   - LAPACK

5. DGEMM on Intel64
Scalability of other libraries

LAPACK

LAPACK - DPOTRF

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

**LAPACK - DGEQRF**

Performance of dense linear algebra software on multicore architectures
Scalability of other libraries

LAPACK - DGETRF

Matrix size vs. Performance for different core counts.

- 32 cores
- 16 cores
- 8 cores
- 4 cores
- 2 cores
- 1 core

32xdssssm-seq
Outline

1. Static pipeline
   - DPOTRF
   - QR and LU

2. Comparison against other libraries on few cores
   - DPOTRF
   - DGEQRF
   - DGETRF

3. PLASMA scalability

4. Scalability of other libraries
   - TBLAS
   - MKL- ESSL
   - SCALAPACK- PESSL
   - LAPACK

5. DGEMM on Intel64
PLASMA-DGEMM- "Scalability"

The diagram shows the performance of dense linear algebra software on multicore architectures, specifically focusing on the DGEMM operation. The x-axis represents the matrix size, while the y-axis represents performance. Different lines signify various numbers of cores: 16, 8, 4, 2, and 1 core. The graph highlights scalability with increasing matrix size across different core counts, demonstrating the efficiency and performance characteristics of the PLASMA-DGEMM implementation on Intel64 architectures.
DGEMM on Intel64

DGEMM - 16 cores

Matrix size vs Performance

- MKL 10.1
- PLASMA
- PLASMABDL
- 16xdgemm-seq

Performance of dense linear algebra software on multicore architectures
Serial **DGEMM** - Impact of flushing

![Graph showing the impact of flushing on Serial DGEMM performance. The x-axis represents NB (a parameter), the y-axis represents performance (in ticks), and the graph compares No Flush and MultCallFlushLRU on A and B. The graph indicates that MultCallFlushLRU generally outperforms No Flush, especially as NB increases.](image-url)
Serial DGEMM - Impact of flushing

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM- 8 cores

Matrix size

0 2000 4000 6000 8000 10000 12000

PLASMA group
Performance of dense linear algebra software on multicore architectures

MKL 10.1
PLASMA
PLASMA BDL
8xdgemm-seq
DGEMM- 4 cores

Matrix size

MKL 10.1
PLASMA
PLASMA\text{\textsc{bdl}}

4xdgemma-seq

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM - 2 cores

Matrix size

PLASMA

PLASMABDL

MKL 10.1

2xdgemm-seq

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM - 1 cores

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM- 16 cores

Matrix size vs Performance for different block sizes (NB) on 16 cores.

- NB=480
- NB=360
- NB=240
- NB=196
- NB=168
- NB=120
- NB=84
- NB=60

Performance of dense linear algebra software on multicore architectures.
DGEMM on Intel64

DGEMM- 8 cores

![Graph showing performance of DGEMM on Intel64 with 8 cores. The graph plots matrix size against performance, with lines for different NB values (480, 360, 240, 196, 168, 120, 84, 60). The x-axis represents matrix size, and the y-axis represents performance. The graph shows that performance increases as matrix size increases for each NB value.]
DGEMM on Intel64

DGEMM - 4 cores

Matrix size

NB=480
NB=360
NB=240
NB=196
NB=168
NB=120
NB=84
NB=60

4xdgemm-seq

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM - 2 cores

Matrix size vs. Performance

- NB=480
- NB=360
- NB=240
- NB=196
- NB=168
- NB=120
- NB=84
- NB=60

2xdgemm-seq

Performance of dense linear algebra software on multicore architectures
DGEMM on Intel64

DGEMM- 1 cores

Matrix size

NB=480
NB=360
NB=240
NB=196
NB=168
NB=120
NB=84
NB=60

1xdgmm-seq

PLASMA group

Performance of dense linear algebra software on multicore architectures