LU Decompositions over DAGuE

Friday Lunch Talk

Omar ZENATI, MsC Candidate

Supervisors: M. Faverge, E. Agullo, A. Bouteiller,
G. Bosilca, P. Ramet

May 25, 2012

ICL🔵ur

Inría

Summary

LU Decomposition Algorithms

DAGuE Runtime System

Static Pivoting

A generic update engine for dynamic pivoting

Partial Pivoting

Performance

Summary

LU Decomposition Algorithms

DAGuE Runtime System

Static Pivoting

A generic update engine for dynamic pivoting

Partial Pivoting

Performance

LU Decomposition Algorithms
Introduction

LU decomposition algorithm :

- ► Used to solve linear equation
- ► Used in Linpack Benchmark and HPL
- ► Implemented in most of mathematic library

Why implementing a new LU decomposition ?

- ► Trend to use Runtime
- ► Three layer architecture: Algorithm, Runtime and Hardware

→ performant and portable code

LU Decomposition Algorithms
Introduction

LU decomposition algorithm :

- ▶ Used to solve linear equation
- ▶ Used in Linpack Benchmark and HPL
- ▶ Implemented in most of mathematic library

Why implementing a new LU decomposition ?

- ▶ Trend to use Runtime
- ▶ Three layer architecture: Algorithm, Runtime and Hardware
- → performant and portable code

LU Decomposition Algorithms
The Challenge

DAGuE provide static DAG

The LU algorithm may be static:

- ▶ Without pivoting
- ▶ Criteria substitution
- ▶ Incremental pivoting

Or dynamic:

- ▶ Partial pivoting

How to programm dynamic application with static task flow ?

LU Decomposition Algorithms
The Challenge

DAGuE provide static DAG

The LU algorithm may be static:

- ► Without pivoting
- ► Criteria substitution
- ► Incremental pivoting

Or dynamic:

- ► Partial pivoting

How to programm dynamic application with static task flow ?

Summary

LU Decomposition Algorithms
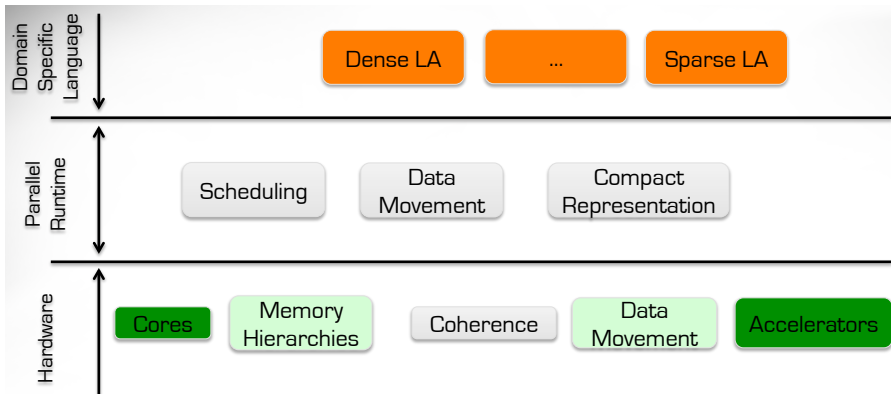
DAGuE Runtime System

Static Pivoting

A generic update engine for dynamic pivoting

Partial Pivoting

Performance

DAGuE
Quick presentation

DAGuE

Quick presentation

DAGuE is a Direct Acyclic Graph scheduler Engine based on task flow model where :

- ▶ nodes are tasks
- ▶ edges are dependencies

```
TRSM(k, n)
// Execution space
k = 0..SIZE-1
n = k+1..SIZE-1
: A(n, k) // Parallel Partitionning
READ  T <- T POTRF(k)
RW    C <- (k == 0) ? A(n, k)
                   : C GEMM(k-1, n, k)
      -> A SYRK(k, n)
      -> A GEMM(k, n+1..SIZE-1, n)
      -> B GEMM(k, n, k+1..n-1)
      -> A(n, k)

BODY*
```


	POTRF
	TRSM
	SYRK
	GEMM
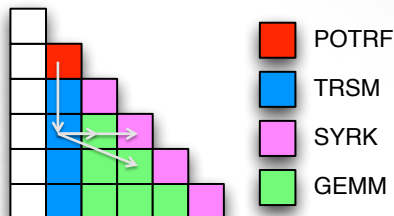
Compared with the serial code this code lost all control flow

DAGuE

Advantages :

- ▶ Independence between performances and computers
- ▶ Provide multicore parallelism
- ▶ Good reactivity for load imbalance
- ▶ Natural look ahead

Problems :

- ▶ DAG is a static representation of a task flow

DAGuE

Advantages :

- ► Independence between performances and computers
- ► Provide multicore parallelism
- ► Good reactivity for load imbalance
- ► Natural look ahead

Problems :

- ► DAG is a static representation of a task flow

Summary

LU Decomposition Algorithms

DAGuE Runtime System

Static Pivoting

A generic update engine for dynamic pivoting
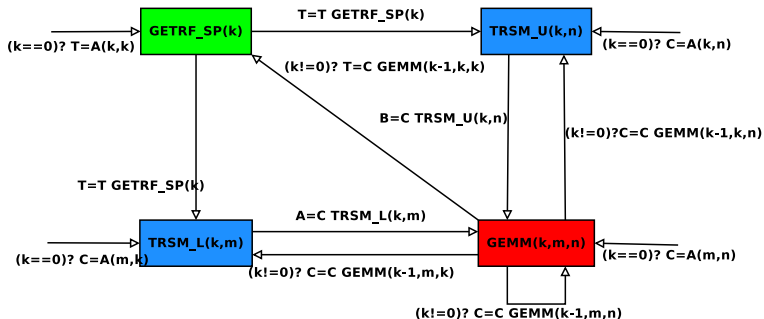
Partial Pivoting

Performance

Static Pivoting
Motivation

- ▶ Static pivoting matches the task flow programming model
- ▶ Good efficiency
- ▶ Stable for several problems
- ▶ Preprocess can improve stability
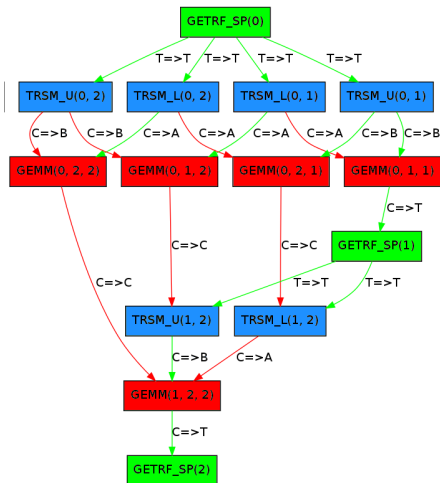- ▶ Good upper bound for the partial pivoting

Static Pivoting

Algebraic Representation

Static Pivoting

DAG for a matrix 3*3

Summary

LU Decomposition Algorithms

DAGuE Runtime System
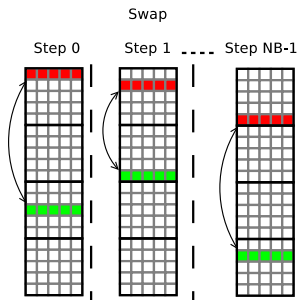
Static Pivoting

A generic update engine for dynamic pivoting

Partial Pivoting

Performance

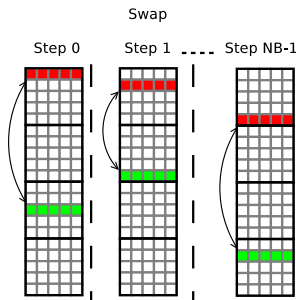# **A generic update engine for dynamic pivoting**
Update Issue

The tile U exchange swap rows with other concerned tile.

Problem

▸ A dynamic decision for a static DAG
  $\rightarrow$ Prepare tasks for all possible communications?

Swap

Step 0   Step 1 ---- Step NB-1

A generic update engine for dynamic pivoting
Update Issue

The tile U exchange swap rows with other concerned tile.
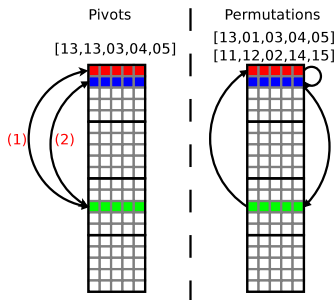
Problem

- ▸ A dynamic decision for a static DAG
  → Prepare tasks for all possible communications?

A generic update engine for dynamic pivoting
Solutions

Ideas:

- ▶ Avoiding useless swap to increase parallelism
  → Use of permutations instead of pivots indexes

- ▶ Updating the main tile is more urgent
  → Parallelize the swap **from** and the swap **into** the tile U

- ▶ Limiting the number of communication (not the volume)
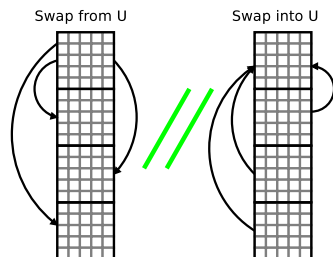 → Gather communications of all rows over two buffers

Pivots

[13,13,03,04,05]

(1) (2)

Permutations
[13,01,03,04,05]
[11,12,02,14,15]

A generic update engine for dynamic pivoting
Solutions

Ideas:

- ▶ Avoiding useless swap to increase parallelism
  $\rightarrow$ Use of permutations instead of pivots indexes

- ▶ Updating the main tile is more urgent
  $\rightarrow$ Parallelize the swap **from** and the swap **into** the tile U

- ▶ Limiting the number of communication (not the volume)
  $\rightarrow$ Gather communications of all rows over two buffers

Swap from U   Swap into U

A generic update engine for dynamic pivoting
### **Solutions**

Ideas:

- ▶ Avoiding useless swap to increase parallelism
  $\rightarrow$ Use of permutations instead of pivots indexes

- ▶ Updating the main tile is more urgent
  $\rightarrow$ Parallelize the swap **from** and the swap **into** the tile U

- ▶ Limiting the number of communication (not the volume)
  $\rightarrow$ Gather communications of all rows over two buffers

$\rightarrow$ Five kinds of tasks : COPY, COLLECT, RECEIVE, SEND and PASTE.

A generic update engine for dynamic pivoting
Solutions

Ideas:

- ▶ Avoiding useless swap to increase parallelism
  $\rightarrow$ Use of permutations instead of pivots indexes
- ▶ Updating the main tile is more urgent
  $\rightarrow$ Parallelize the swap **from** and the swap **into** the tile U
- ▶ Limiting the number of communication (not the volume)
  $\rightarrow$ Gather communications of all rows over two buffers

$\rightarrow$ Five kinds of tasks : COPY, COLLECT, RECEIVE, SEND and PASTE.

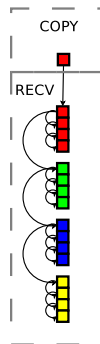# A generic update engine for dynamic pivoting
Swap into U

- ▶ COLLECT: Collecting the rows needed by the tile U into a buffer.
- ▶ SEND: Gather the buffers collected by COLLECT of each node.
- ▶ PASTE: Overwrite the tile U with the buffer.
- ▶ COPY
- ▶ RECEIVE
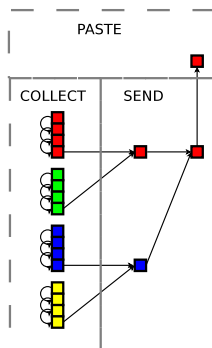
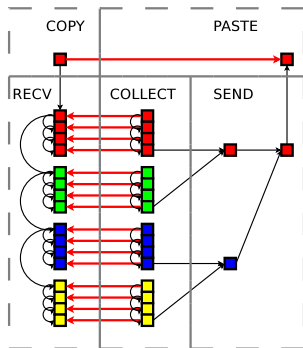# A generic update engine for dynamic pivoting
Swap from U

- COLLECT
- SEND
- PASTE
- COPY: Copy tile U into a buffer.
- RECEIVE: Receive the buffer U and make the swap from it.

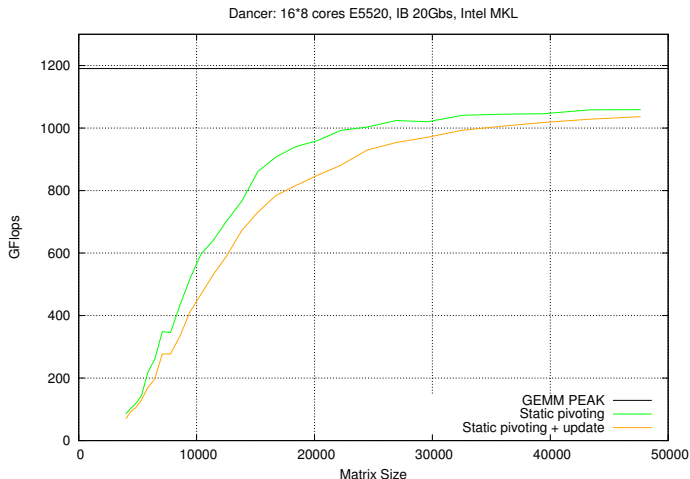# A generic update engine for dynamic pivoting
Update Tasks Synchronisation

- ► COLLECT
- ► SEND
- ► PASTE
- ► COPY
- ► RECEIVE

The red arrows prevent the **READ AFTER WRITE**.

A generic update engine for dynamic pivoting
Update Impact

Dancer: 16*8 cores E5520, IB 20Gbs, Intel MKL

Summary

LU Decomposition Algorithms

DAGuE Runtime System

Static Pivoting

A generic update engine for dynamic pivoting

Partial Pivoting
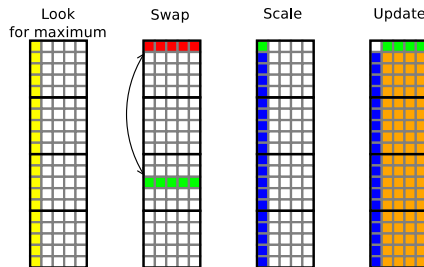
Performance

Partial Pivoting
Heuristic Factorization

Several heuristic to factorize the panel:

- ▶ Partial pivoting
- ▶ Threshold pivoting
- ▶ Tournament pivoting
  - ▶ Internal partial pivoting
  - ▶ QR rank revealing

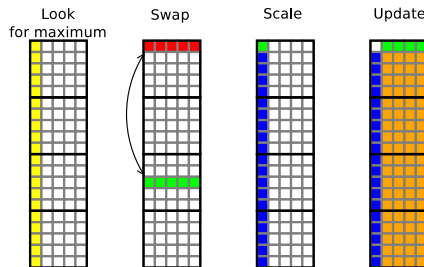# Partial Pivoting

Operations of Panel LU Decomposition

Problem for implementing with task flow model is based on

▸ Swap line is dynamically decided but the DAG is static

▸ Minimize latency for the panel

Partial Pivoting

Operations of Panel LU Decomposition

Look for maximum   Swap   Scale   Update

Problem for implementing with task flow model is based on
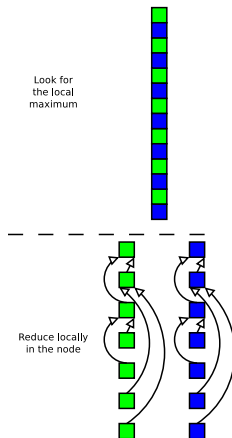
- ▶ Swap line is dynamically decided but the DAG is static
- ▶ Minimize latency for the panel

Partial Pivoting
Solutions

Solutions:

▶ Start looking for the maximum locally then reduce locally the result

▶ Share the global result by using Bruck's algorithm

▶ Use internal blocking

Look for the local maximum

Reduce locally in the node

Partial Pivoting
Implemented version

Solutions:

- ▶ Start looking for the maximum locally then reduce locally the result
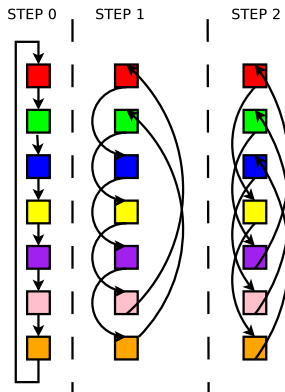- ▶ Share the global result by using Bruck's algorithm
- ▶ Use internal blocking

STEP 0   STEP 1   STEP 2

Partial Pivoting
Implemented version

Panel update

Optimizations:

▶ Start looking for the
  maximum locally then reduce
  locally the result

▶ Share the global result by
  using Bruck's algorithm

▶ Use internal blocking
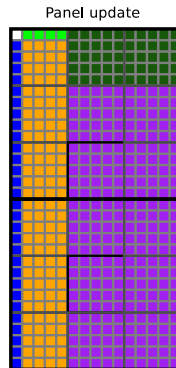
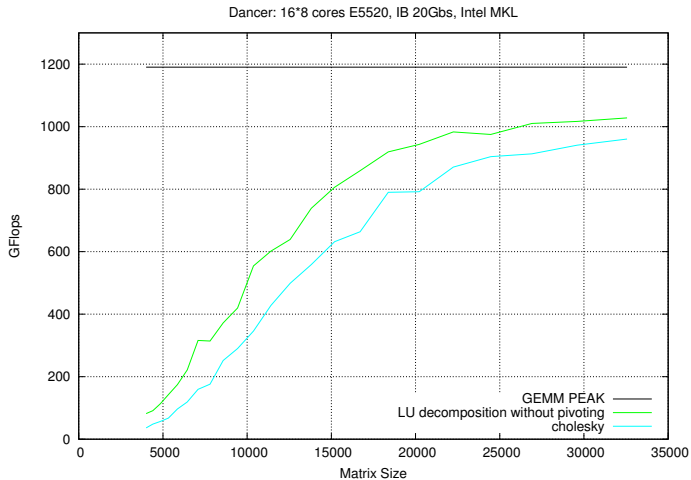# Summary

LU Decomposition Algorithms

DAGuE Runtime System

Static Pivoting

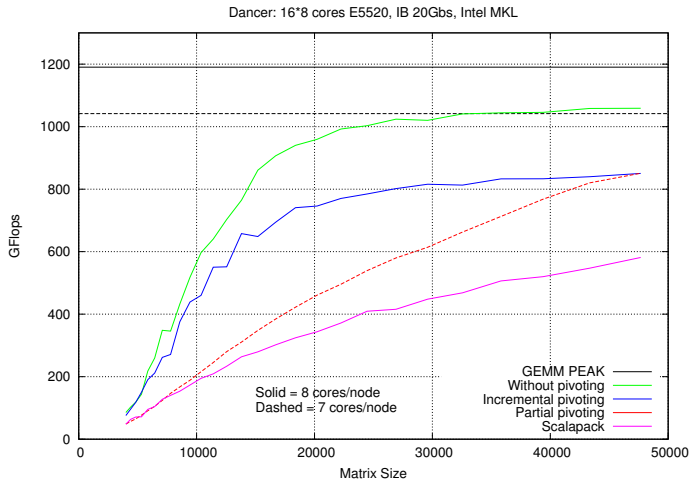A generic update engine for dynamic pivoting

Partial Pivoting

Performance

Performance

Dancer: 16*8 cores E5520, IB 20Gbs, Intel MKL

Performance

Dancer: 16*8 cores E5520, IB 20Gbs, Intel MKL

Solid = 8 cores/node
Dashed = 7 cores/node

Legend:
- GEMM PEAK
- Without pivoting
- Incremental pivoting
- Partial pivoting
- Scalapack

Conclusion and future work

Conclusion :
- ▶ Small impact with the update engine
- ▶ Hopeful result with partial pivoting

Future work :
- ▶ Improve the performance of the partial pivoting
- ▶ Implement other panel factorization

Conclusion

Thank you !