Performance Analysis and Optimization of the Tiled Cholesky Factorization on NUMA Machines

Emmanuel Jeannot

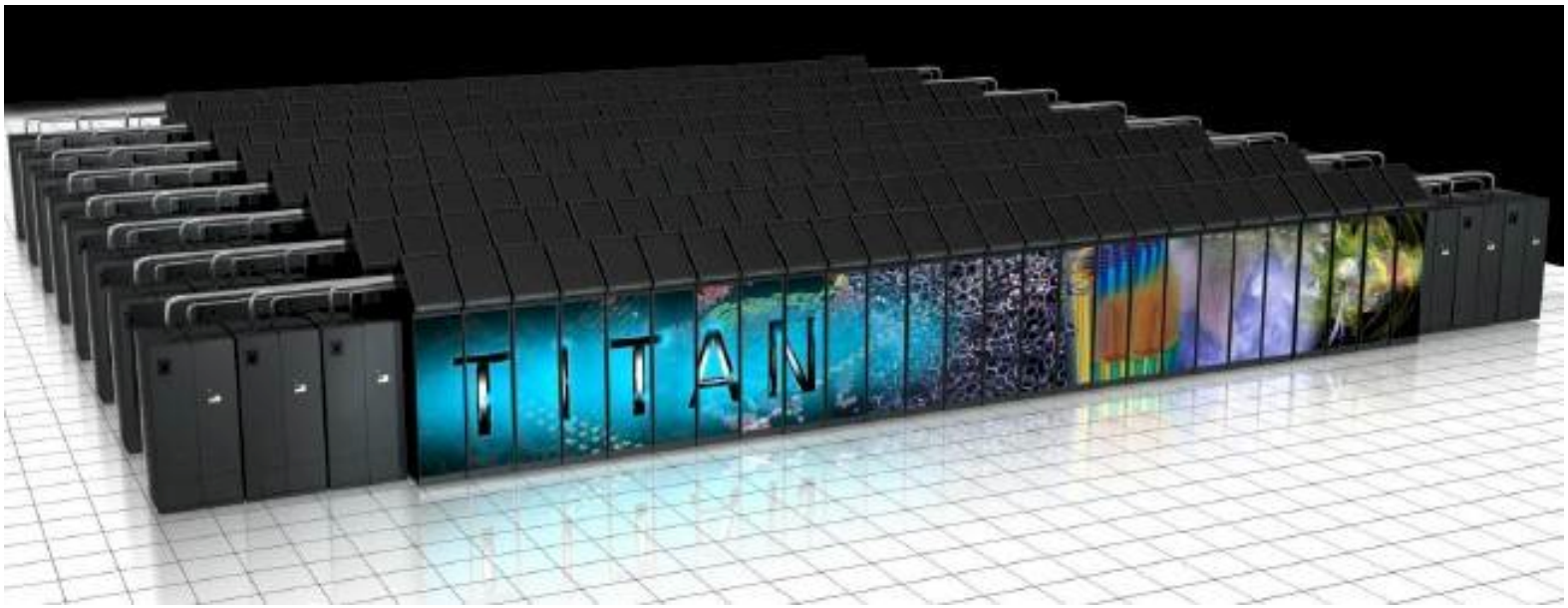
Inria Bordeaux Sud-Ouest, France

September 24, 2013

Context

High Performance Computing

- Treating very large problem
- on very large machine
- as fast as possible



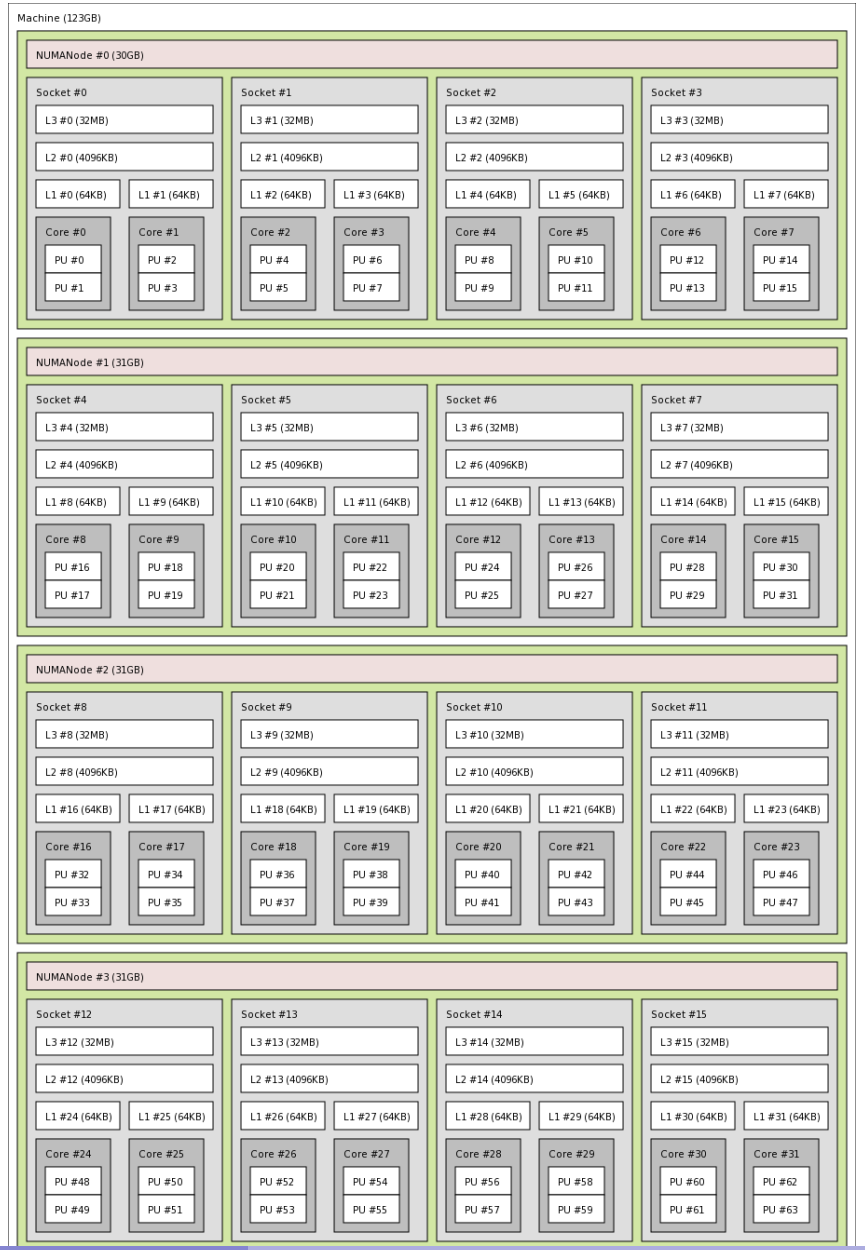
Oak-ridge Titan supercomputer. 200 cabinets, 18 688 nodes, AMD 16-cores + NVIDIA TESLA, 20+ Pflops

NUMA machines

Hierarchical with several levels

- machine
- node
- socket
- caches
- cores

But one address space!



Today's talk

Obtaining peek performance is not easy on modern NUMA machine

This study

- What can be done statically
- What needs to be done dynamically
- Interaction between static and dynamic optimization
- Importance of the data allocation

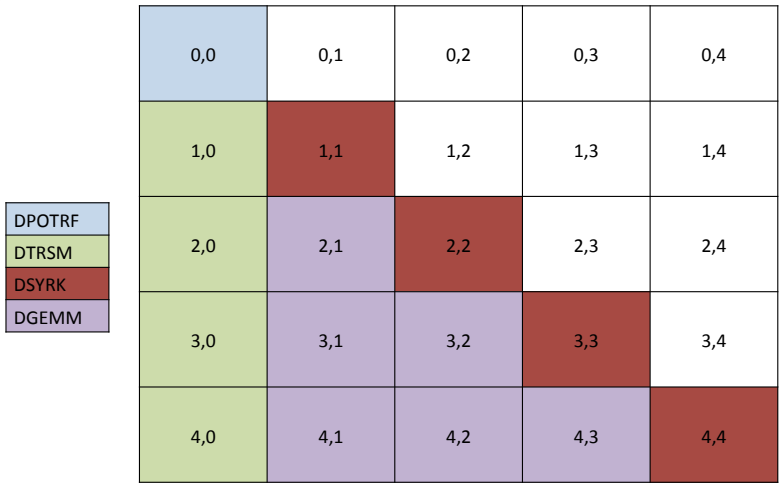
Outline

Outline

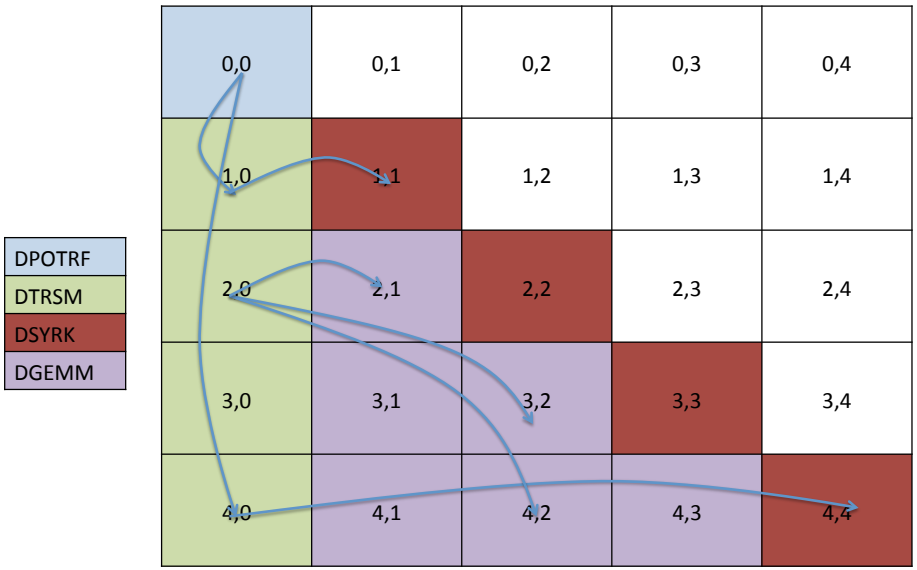
Algorithm

The tiled version. A square matrix, symmetric definite positive decomposed in squares tiles. Compute L, such that $A = LL^T$

for $k = 0 \ldots T-1$ **do**
  $A[k][k] \leftarrow$ DPOTRF$(A[k][k])$
  for $m = k+1 \ldots T-1$ **do**
    $A[m][k] \leftarrow$ DTRSM$(A[k][k], A[m][k])$
  end
  for $n = k+1 \ldots T-1$ **do**
    $A[n][n] \leftarrow$ DSYRK$(A[n][k], A[n][n])$
    for $m = n+1 \ldots T-1$ **do**
      $A[m][n] \leftarrow$ DGEMM$(A[m][k], A[n][k], A[m][n])$
    end
  end
end



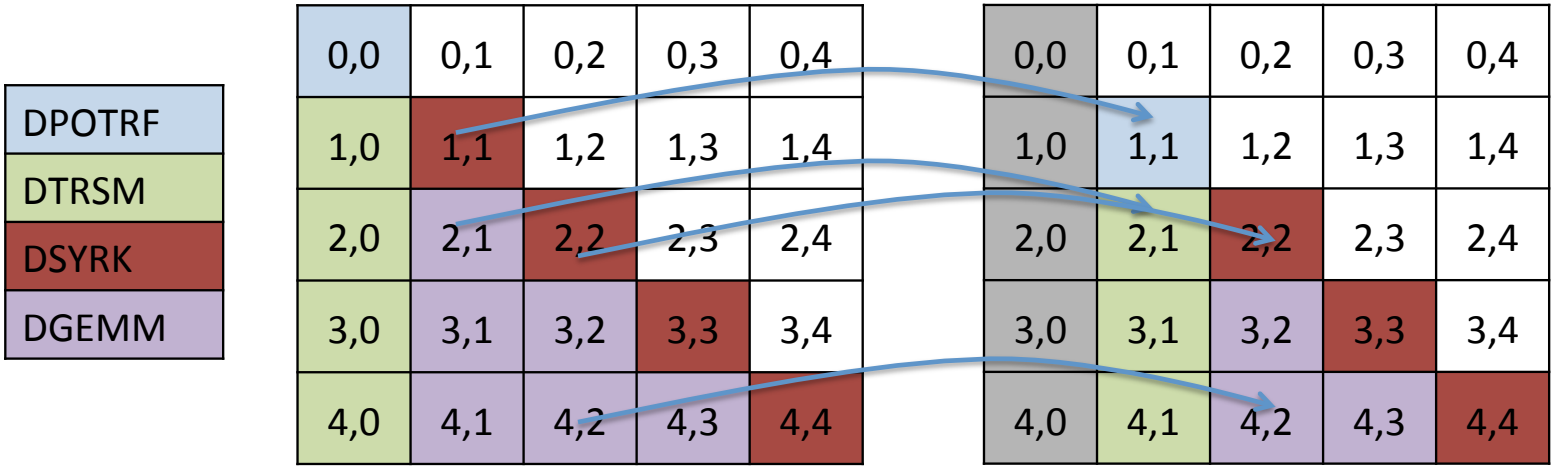
Dependencies analysis within an iteration



If $T \geq 1$:

- $\{\text{DPOTRF}(k) | 0 \leq k \leq T-1\} \rightarrow \{\text{DTRSM}(k; j) | k+1 \leq j \leq T-1\} : \{A[k][k]\}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DSYRK}(k; n) | \} : \{A[n][k] | \}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k; j; n) | n+1 \leq j \leq T-1\} : \{A[n][k] | \}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k; n; j) | k+1 \leq j \leq n-1\} : \{A[n][k] | \}$

Dependencies analysis between iterations



If $T \geq 1$:

- $\{\text{DSYRK}(k; m) | 0 \leq k \leq T-1; m = k+1\} \rightarrow \{\text{DPOTRF}(k+1) | \} : \{A[m][m] | \}$
- $\{\text{DSYRK}(k; m) | 0 \leq k \leq T-1; k+2 \leq m \leq T-1\} \rightarrow \{\text{DSYRK}(k+1; m) | \} : \{A[m][m] | \}$
- $\{\text{DGEMM}(k; m; n) | 0 \leq k \leq T-2; k+2 \leq n \leq T-1; m = k+1\} \rightarrow \{\text{DTRSM}(k+1; n) | \} : \{A[n][m] | \}$
- $\{\text{DGEMM}(k; m; n) | 0 \leq k \leq T-2; k+2 \leq n \leq T-1; k+2 \leq m \leq n-1\} \rightarrow \{\text{DGEMM}(k+1; m; n) | \} : \{A[n][m] | \}$

Dependencies analysis $\Longrightarrow$ parameterized task graph

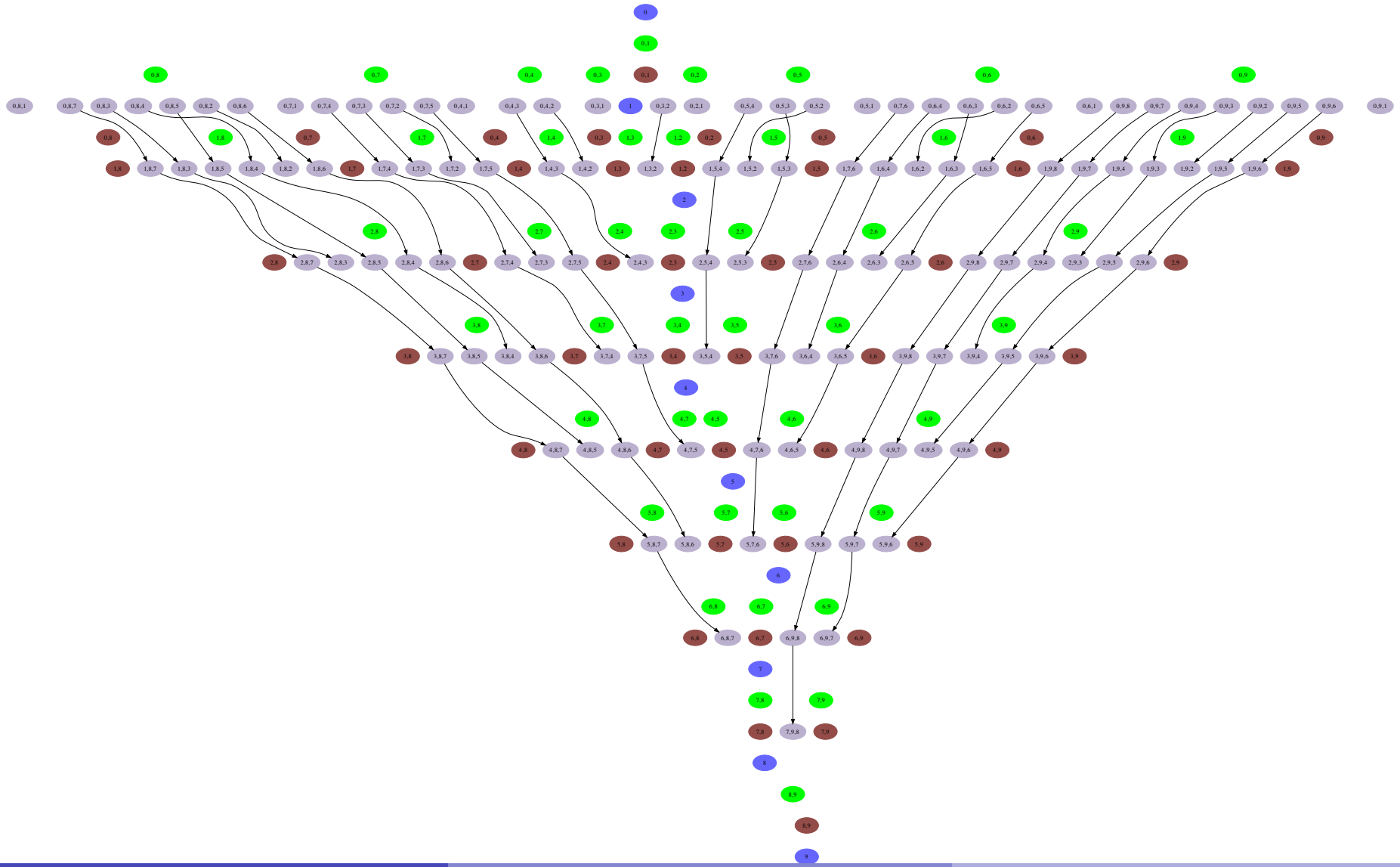
```
for k = 0...T − 1 do
    A[k][k] ← DPOTRF(A[k][k])
    for n = k + 1...T − 1 do
        A[n][k] ← DTRSM(A[k][k], A[n][k])
    end
    for m = k + 1...T − 1 do
        A[m][m] ← DSYRK(A[m][k], A[m][m])
        for n = m + 1...T − 1 do
            A[n][m] ← DGEMM(A[n][k], A[m][k], A[n][m])
        end
    end
end
```

If $T \geq 1$:

- $\{\text{DPOTRF}(k) | 0 \leq k \leq T-1\} \rightarrow \{\text{DTRSM}(k; j) | k+1 \leq j \leq T-1\} : \{A[k][k]\}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DSYRK}(k; n) | \} : \{A[n][k] | \}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k; j; n) | n+1 \leq j \leq T-1\} : \{A[n][k] | \}$
- $\{\text{DTRSM}(k; n) | 0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k; n; j) | k+1 \leq j \leq n-1\} : \{A[n][k] | \}$
- $\{\text{DSYRK}(k; m) | 0 \leq k \leq T-1; m = k+1\} \rightarrow \{\text{DPOTRF}(k+1) | \} : \{A[m][m] | \}$
- $\{\text{DSYRK}(k; m) | 0 \leq k \leq T-1; k+2 \leq m \leq T-1\} \rightarrow \{\text{DSYRK}(k+1; m) | \} : \{A[m][m] | \}$
- $\{\text{DGEMM}(k; m; n) | 0 \leq k \leq T-2; k+2 \leq n \leq T-1; m = k+1\} \rightarrow \{\text{DTRSM}(k+1; n) | \} : \{A[n][m] | \}$
- $\{\text{DGEMM}(k; m; n) | 0 \leq k \leq T-2; k+2 \leq n \leq T-1; k+2 \leq m \leq n-1\} \rightarrow \{\text{DGEMM}(k+1; m; n) | \} : \{A[n][m] | \}$

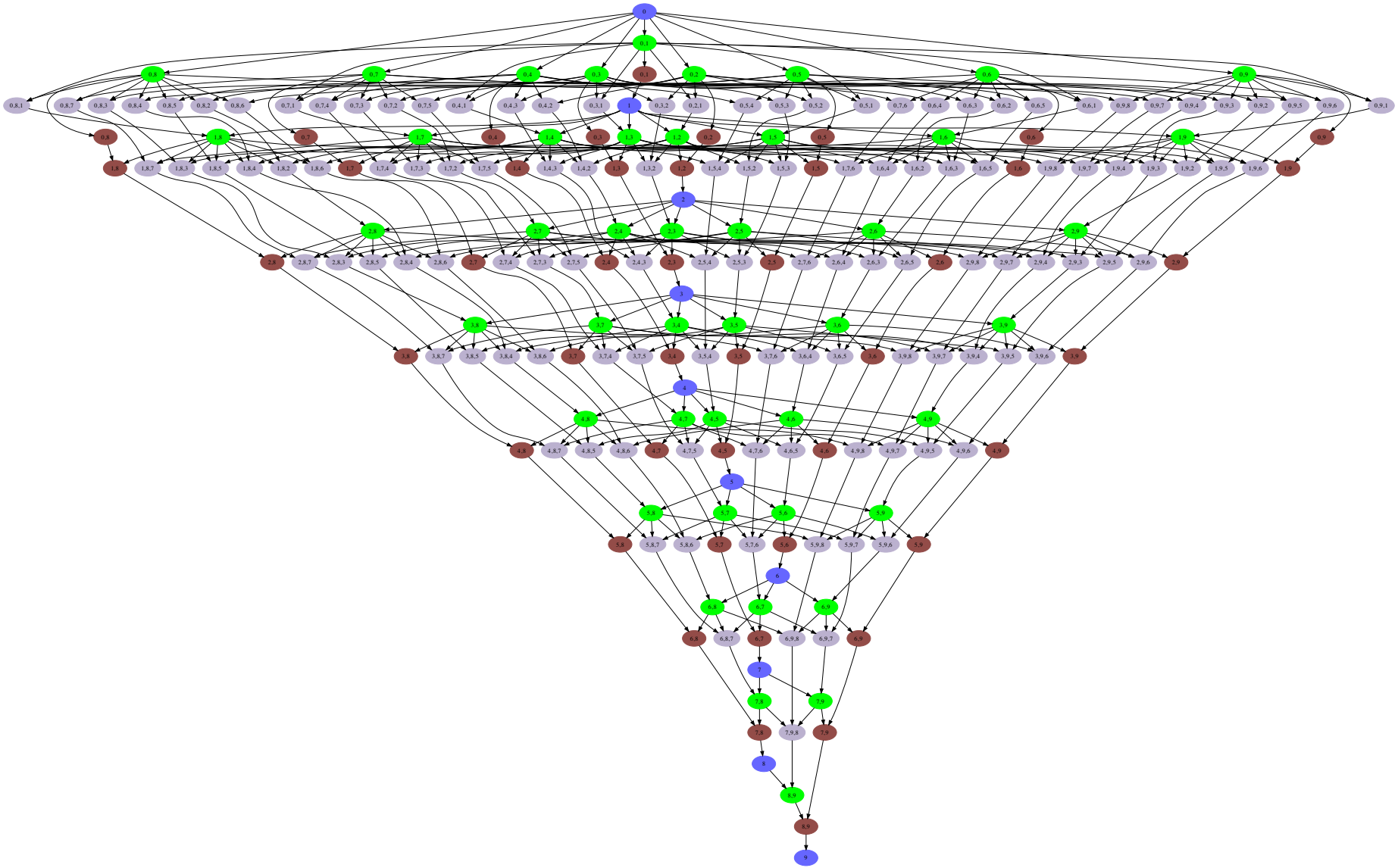
Rule 8, $T = 10$

$\{\text{DGEMM}(k; m; n) | 0 \leq k \leq T-2; k+2 \leq n \leq T-1; k+2 \leq m \leq n-1\} \rightarrow \{\text{DGEMM}(k+1; m; n) | \} : \{A[n][m] | \}$



Expansion of the rules, T=10

All rules



Outline

Task/Data Allocation

Parameterized Task Graph (*PTG*)

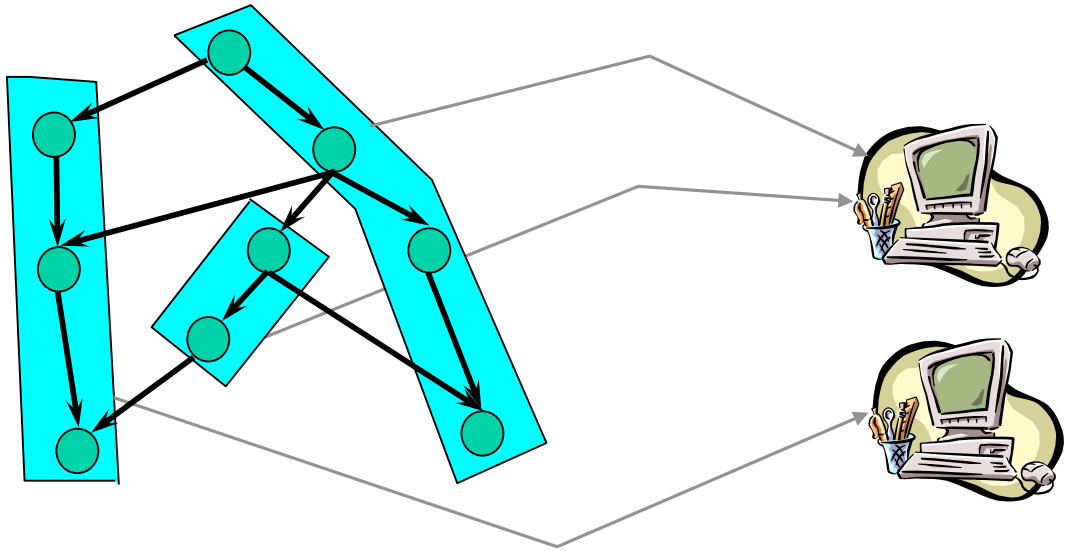
- Set of rules:

$$T_a(\vec{u}) \rightarrow T_b(\vec{v}) : D(\vec{y})|P$$

- $\forall \vec{u}$ $\vec{v}$ and $\vec{y}$ in P task $T_a(\vec{u})$ sends data $D(\vec{y})$ to task $T_b(\vec{v})$.
- A set of parameter. If we instantiate the parameter: we obtain the full DAG
- Symbolic allocation find a function that maps task/data to resources

Symbolic allocation

- Using the *PTG* determine a **function** that gives the (virtual) **processor** where a task has to be **executed** and where data needs to be **stored**
- **Principle :** put tasks that communicate on the same processor
- **Problem :** suppress all communications $\Longrightarrow$ no more parallelism
- **Trade-off** between supressing comunication and getting parallelism



Overview of SMA

Symbolic Mapping and Allocation (SMA) algorithm

3 steps:

- Select point to point (**bijective**) communication rules
- Suppress **conflicting** rules
- Perform mapping by expressing constraints: compute **symbolic allocation**

Overview of SMA

Symbolic Mapping and Allocation (SMA) algorithm

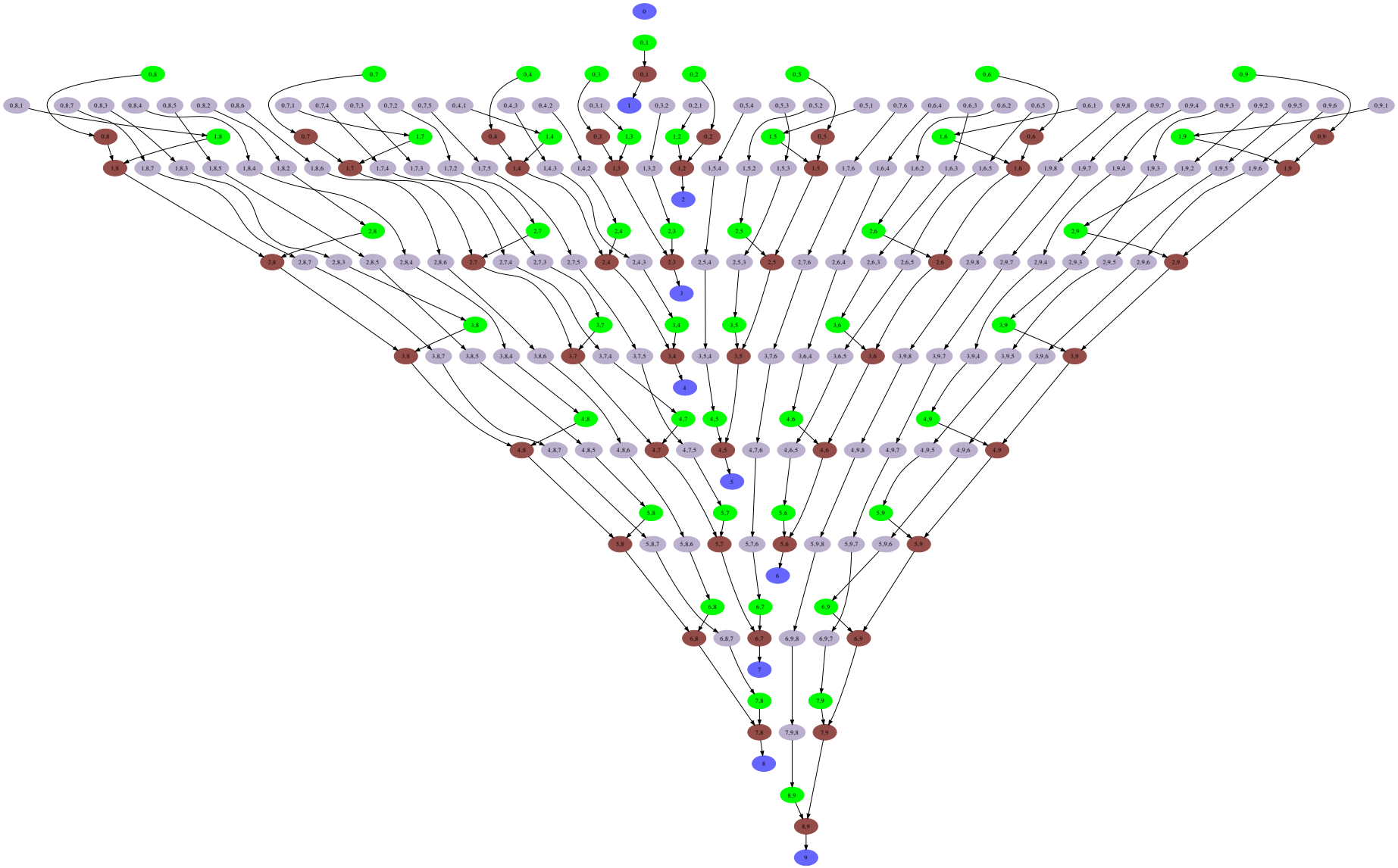
3 steps:

- Select point to point (**bijective**) communication rules
- Suppress **conflicting** rules
- Perform mapping by expressing constraints: compute **symbolic allocation**

5 bijection rules:

- x $\{\text{DPOTRF}(k)|0 \leq k \leq T-1\} \rightarrow \{\text{DTRSM}(k;j)|k+1 \leq j \leq T-1\} : \{A[k][k]\}$
- $\{\text{DTRSM}(k;n)|0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DSYRK}(k;n)|\} : \{A[n][k]|\}$
- x $\{\text{DTRSM}(k;n)|0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k;j;n)|n+1 \leq j \leq T-1\} : \{A[n][k]|\}$
- x $\{\text{DTRSM}(k;n)|0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DGEMM}(k;n;j)|k+1 \leq j \leq n-1\} : \{A[n][k]|\}$
- $\{\text{DSYRK}(k;m)|0 \leq k \leq T-1; m = k+1\} \rightarrow \{\text{DPOTRF}(k+1)|\} : \{A[m][m]|\}$
- $\{\text{DSYRK}(k;m)|0 \leq k \leq T-1; k+2 \leq m \leq T-1\} \rightarrow \{\text{DSYRK}(k+1;m)|\} : \{A[m][m]|\}$
- $\{\text{DGEMM}(k;m;n)|0 \leq k \leq T-2; k+2 \leq n \leq T-1; m = k+1\} \rightarrow \{\text{DTRSM}(k+1;n)|\} : \{A[n][m]|\}$
- $\{\text{DGEMM}(k;m;n)|0 \leq k \leq T-2; k+2 \leq n \leq T-1; k+2 \leq m \leq n-1\} \rightarrow \{\text{DGEMM}(k+1;m;n)|\} : \{A[n][m]|\}$

All bijection rules



Conflicting Rules?

Two types of conflicts

1. **fork conflict :** if two rules describe the same sending task instance for different receiving tasks instances
2. **join conflict :** if two rules describe the same receiving task instance for different sending tasks instances

1 fork conflict

- c $\{\text{DTRSM}(k;n)|0 \leq k \leq T-1; k+1 \leq n \leq T-1\} \rightarrow \{\text{DSYRK}(k;n)|\} : \{A[n][k]|\}$
- $\{\text{DSYRK}(k;m)|0 \leq k \leq T-1; m = k+1\} \rightarrow \{\text{DPOTRF}(k+1)|\} : \{A[m][m]|\}$
- c $\{\text{DSYRK}(k;m)|0 \leq k \leq T-1; k+2 \leq m \leq T-1\} \rightarrow \{\text{DSYRK}(k+1;m)|\} : \{A[m][m]|\}$
- $\{\text{DGEMM}(k;m;n)|0 \leq k \leq T-2; k+2 \leq n \leq T-1; m = k+1\} \rightarrow \{\text{DTRSM}(k+1;n)|\} : \{A[n][m]|\}$
- $\{\text{DGEMM}(k;m;n)|0 \leq k \leq T-2; k+2 \leq m \leq n-1\} \rightarrow \{\text{DGEMM}(k+1;m;n)|\} : \{A[n][m]|\}$

To ensure as much parallelism as possible

Only non conflicting rules need to be clustered (selected)

Identification

Find a **functions** that gives the **cluster number** of task $T_a(\vec{u})$ for any valid value of $\vec{u}$ and the mapping of tile $A[i][j]$

1. Let $\mathcal{Z}$ a set of non conflicting rules
2. $\vec{p}$ the vector of parameters
3. Each rule: : $T_a(\vec{u}) \longrightarrow T_b(\vec{v}) : D(\vec{y})|P$
4. Affine [fea94] data mapping function $\mu(D, \vec{y}) = \vec{\alpha_D}\vec{y} + \vec{\beta_D} + \vec{\gamma_D}\vec{p}$
5. Affine clustering function $\kappa(T_a, \vec{u}) = \vec{\alpha_a}\vec{u} + \vec{\beta_a} + \vec{\gamma_a}\vec{p}$
6. Build system of equations for all rules in $\mathcal{Z}$
7. Solve the system

Example

Data mapping (owner-compute rule)

- From the code, we have: $T_a(\vec{u})$ writes $D(\vec{y})$
- Therefore $\kappa(T_a, \vec{u}) = \mu(D, \vec{y})$
- Hence, $\vec{\alpha_a}\vec{u} + \vec{\beta_a} + \vec{\gamma_a}\vec{p} = \vec{\alpha_D}\vec{y} + \vec{\beta_D} + \vec{\gamma_D}\vec{p}$

Clustering

- Each rule clustered rule: $T_a(\vec{u}) \longrightarrow T_b(\vec{v}) : D(\vec{y})|P$
- $\kappa(T_a, \vec{u}) = \kappa(T_b, \vec{v})$
- Therefore: $\vec{\alpha_a}\vec{u} + \vec{\beta_a} + \vec{\gamma_a}\vec{p} = \vec{\alpha_b}\vec{v} + \vec{\beta_b} + \vec{\gamma_b}\vec{p}$

Cholesky Case

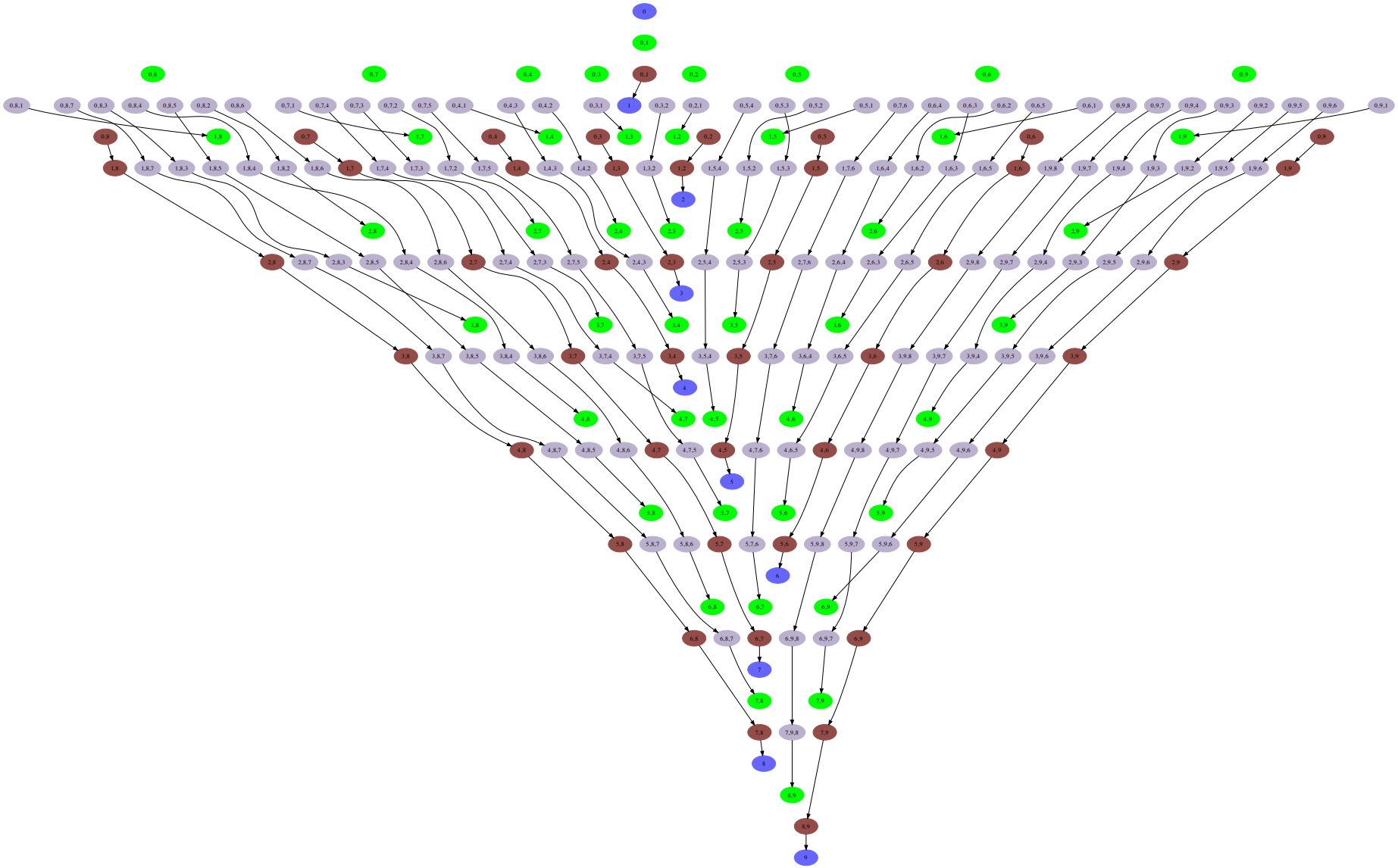
If we select the first conflicting rule:

- $\mu(A[i][j]) = 0$
- $\kappa(\text{DPOTRF}(k)) = k$
- $\kappa(\text{DTRSM}(k, n)) = k + n$
- $\kappa(\text{DSYRK}(k, m)) = m$
- $\kappa(\text{DGEMM}(k, m, n)) = m + n$

If we select the second conflicting rule:

- $\mu(A[i][j]) = i + j$
- $\kappa(\text{DPOTRF}(k)) = 2 \times k$
- $\kappa(\text{DSYRK}(k, n)) = 2 \times n$
- $\kappa(\text{DTRSM}(k, n)) = k + n$
- $\kappa(\text{DGEMM}(k, m, n)) = m + n$

Selected clustering



Outline

Memory node?

Solution: put tiles $A[i][j]$ on cluster $i + j$

In practice: tile $A[i][j]$ is put on memory:

$$i + j \mod G$$

G?

- NUMA machine: memory hierarchy
- Which granularity is the most efficient?
  - G = 1 (shared memory)?
  - G = number of memory bank?
  - G = number of core (cache)?

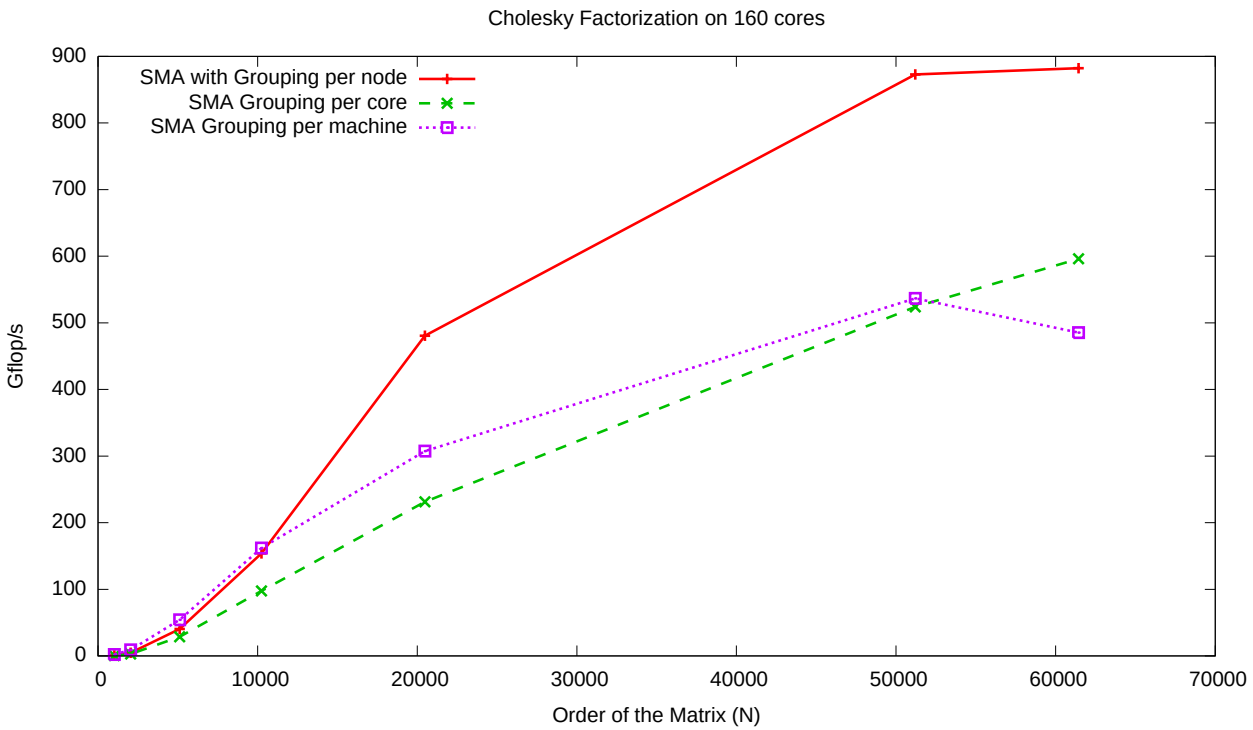
Experiment conditions

- 160 cores NUMA machine composed of 20 nodes of one 8 cores socket Intel Nehalem Eagleton (E7–8837) at 2.67GHz
- icc 11.1-075
- MKL 11.1-075 (comparison and compute kernels)
- Plasma 1.4.5 + Quark runtime system

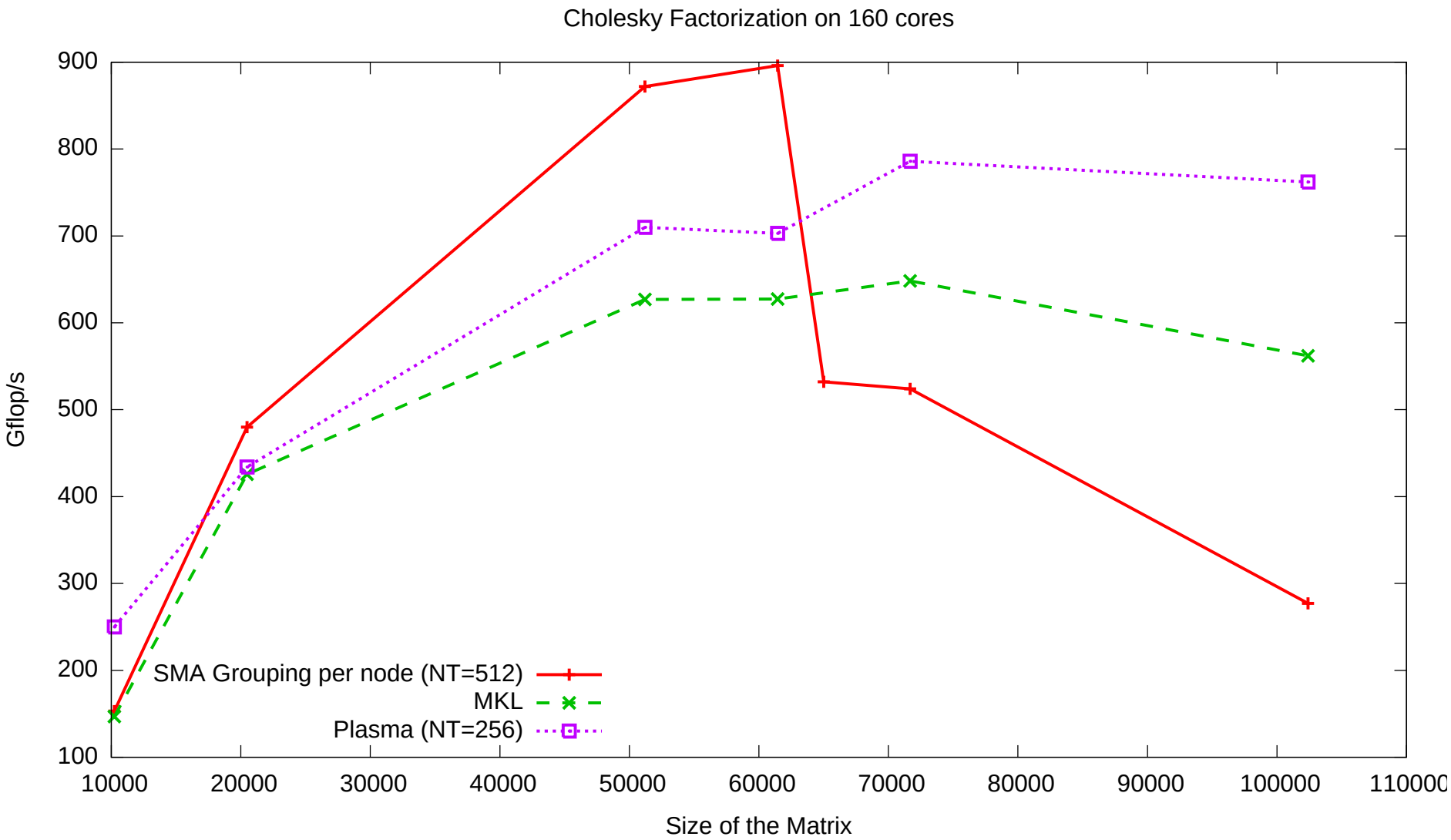
$i + j \mod G$

G?

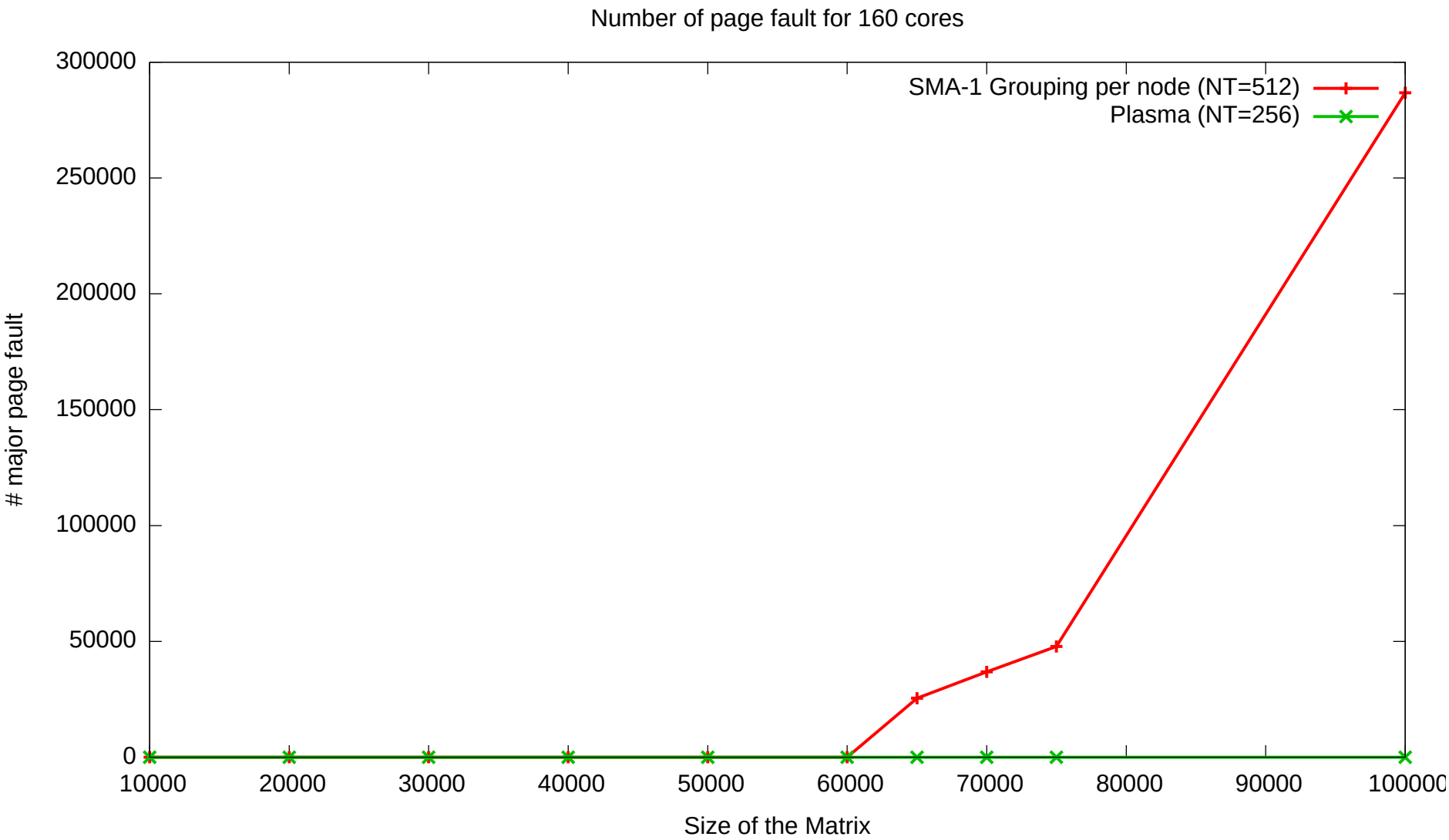
- Best compromise:
  - number of memory bank
- bind threads to memory node (hwloc)
- bind tile to memory node (`numa_alloc_onnode()`)



Pure performance



Page fault


Number of page fault for 160 cores
SMA-1 Grouping per node (NT=512)
Plasma (NT=256)
major page fault
Size of the Matrix
300000
250000
200000
150000
100000
50000
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

Using swap on a 600 Gb machine?

Memory usage

- N=64000: more than 30 Gb to store the matrix
- 20 nodes: 30 Gb/node

Using swap on a 600 Gb machine?

Memory usage

- N=64000: more than 30 Gb to store the matrix
- 20 nodes: 30 Gb/node

Memory allocation

- With `malloc` pages are put on the first node that writes it
- Sequential allocation: everything is put on the same node
- $N > 64000$ not enough memory on a node

Using swap on a 600 Gb machine?

Memory usage

- N=64000: more than 30 Gb to store the matrix
- 20 nodes: 30 Gb/node

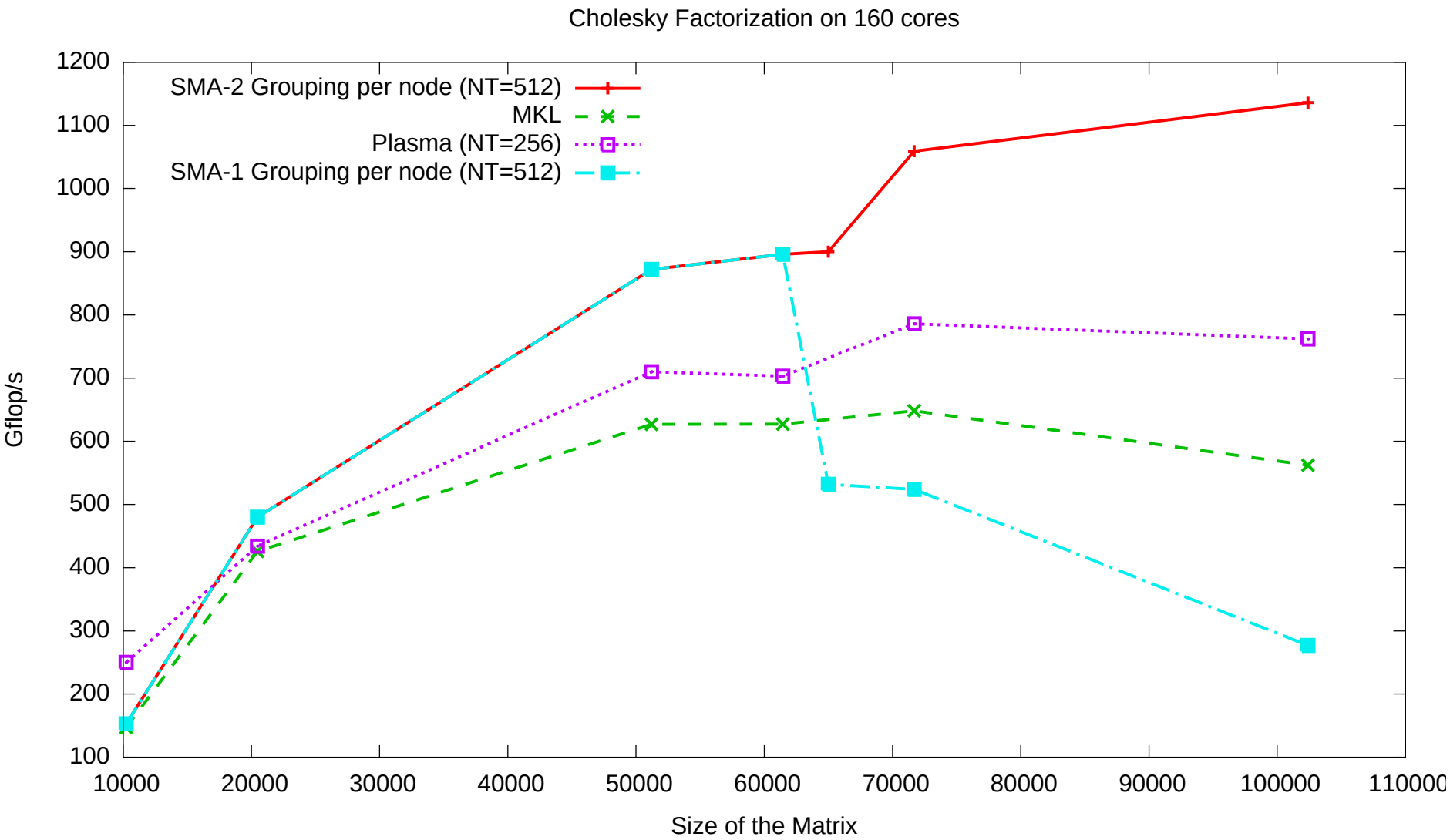
Memory allocation

- With `malloc` pages are put on the first node that writes it
- Sequential allocation: everything is put on the same node
- $N > 64000$ not enough memory on a node

Solutions

- directly store data in tile layout
- have a multithreaded I/O for creating the matrix in lapack format
- allocate pages of matrix in round robin fashion onto the nodes (`numa_alloc_interleaved`)

Final result



Comparion with Block-Cyclic Mapping

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}	a_{49}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}	a_{58}	a_{59}
a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	a_{66}	a_{67}	a_{68}	a_{69}
a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}	a_{78}	a_{79}
a_{81}	a_{82}	a_{83}	a_{84}	a_{85}	a_{86}	a_{87}	a_{88}	a_{89}
a_{91}	a_{92}	a_{93}	a_{94}	a_{95}	a_{96}	a_{97}	a_{98}	a_{99}

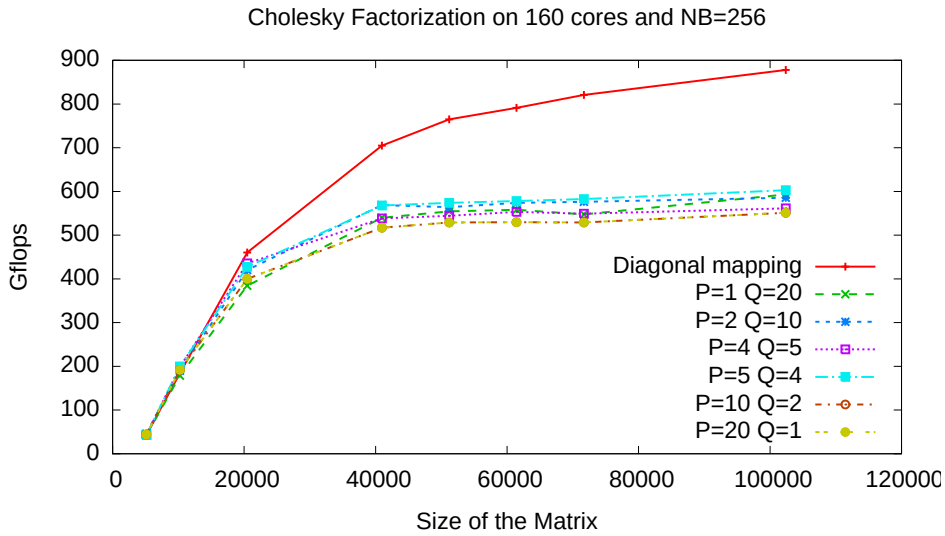
Global View

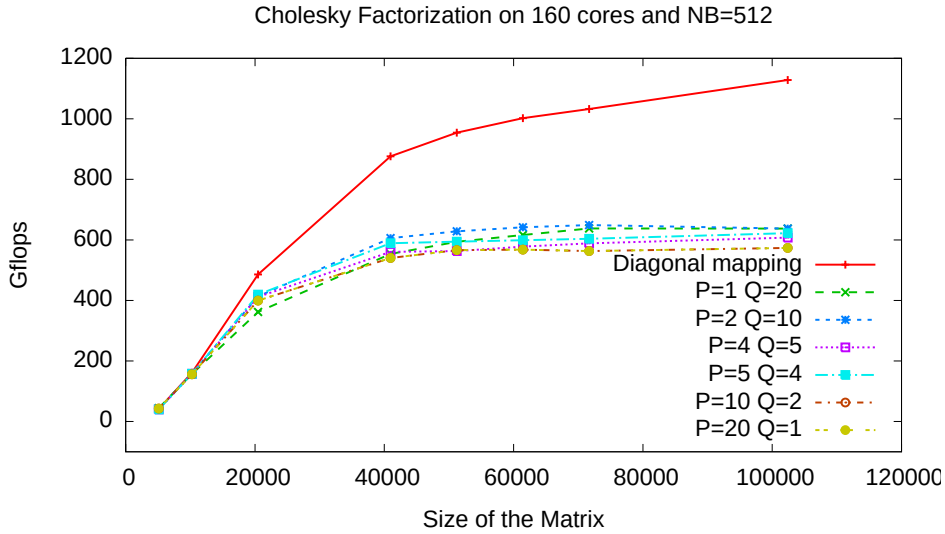
	0				1			2	
	a_{11}	a_{12}	a_{17}	a_{18}	a_{13}	a_{14}	a_{19}	a_{15}	a_{16}
	a_{21}	a_{22}	a_{27}	a_{28}	a_{23}	a_{24}	a_{29}	a_{25}	a_{26}
0	a_{51}	a_{52}	a_{57}	a_{58}	a_{53}	a_{54}	a_{59}	a_{55}	a_{56}
	a_{61}	a_{62}	a_{67}	a_{68}	a_{63}	a_{64}	a_{69}	a_{65}	a_{66}
	a_{91}	a_{92}	a_{97}	a_{98}	a_{93}	a_{94}	a_{99}	a_{95}	a_{96}
	a_{31}	a_{32}	a_{37}	a_{38}	a_{33}	a_{34}	a_{39}	a_{35}	a_{36}
	a_{41}	a_{42}	a_{47}	a_{48}	a_{43}	a_{44}	a_{49}	a_{45}	a_{46}
1	a_{71}	a_{72}	a_{77}	a_{78}	a_{73}	a_{74}	a_{79}	a_{75}	a_{76}
	a_{81}	a_{82}	a_{87}	a_{88}	a_{83}	a_{84}	a_{89}	a_{85}	a_{86}

Local (distributed) View

P=2, Q=3

Comparion with Block-Cyclic Mapping





Conclusion

Shared memory machine

Shared memory machine gives the illusion of a flat address space but:

- data allocation
- data movement

have a huge impact on the performance

Conclusion

Shared memory machine

Shared memory machine gives the illusion of a flat address space but:

- data allocation
- data movement

have a huge impact on the performance

Scientific challenges

Low-level system commands can help to solve:

- data allocation
- data (re)distribution

Outline

Conclusion

Wrap-up

- SMA can be applied to new tiled algorithms: results are promising
- Memory is not flat. Data allocation have a huge impact on performance
- Many thing can be compiled statically
- But a good runtime system is necessary to achieve efficiency!

Acknowledgement: Emmanuel Agullo, George Bosilca, Michel Cosnard, Brice Goglin, *Morse associated team*, Tao Yang, ...