TASK-BASED PARALLEL PROGRAMMING FOR SCALABLE ALGORITHMS AND ILLUSTRATION WITH MATRIX MULTIPLICATION

JCAD 2024

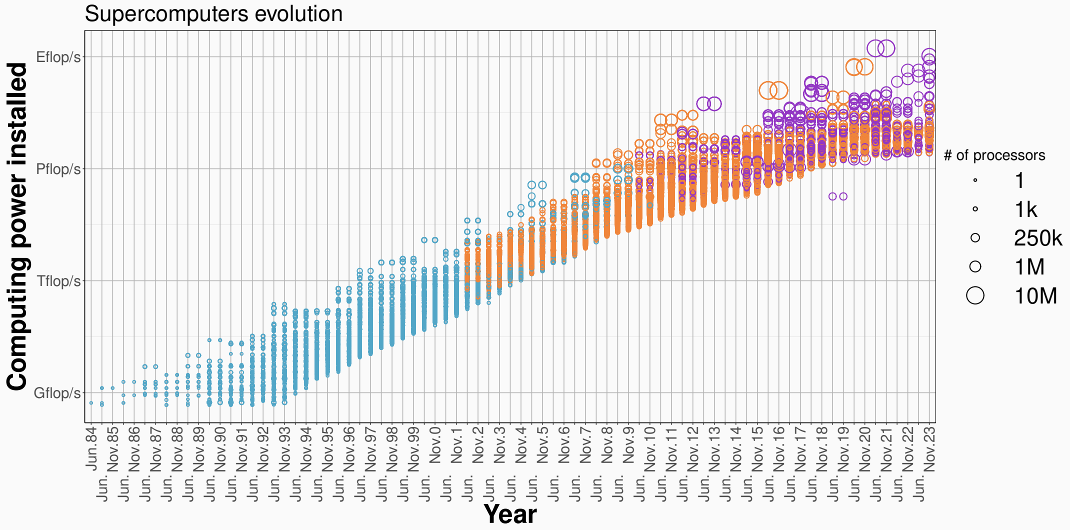
Antoine Jégo

Supervised by Alfredo Buttari, Emmanuel Agullo & Abdou Guermouche

5 novembre 2024

BACKGROUND

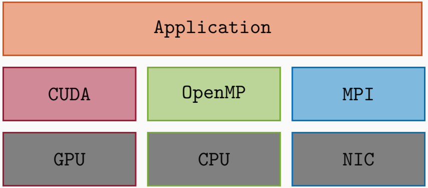
Computer architecture and programming models



Distributed + **Multicore** + **Accelerated**

Runtime systems

Classical approach



Complex supercomputers

$\bigoplus$

Advanced scalability techniques

$\Downarrow$

Complex software

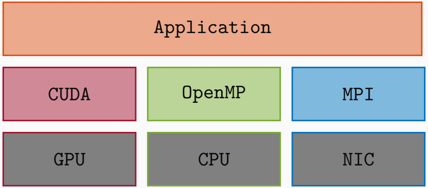
Runtime system approach

General-purpose runtimes

- High-level interface
- Data management
- Heterogeneity management

Runtime systems

Classical approach



Complex supercomputers

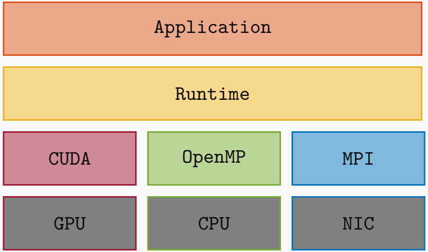
$\oplus$

Advanced scalability techniques

$\Downarrow$

Complex software

Runtime system approach

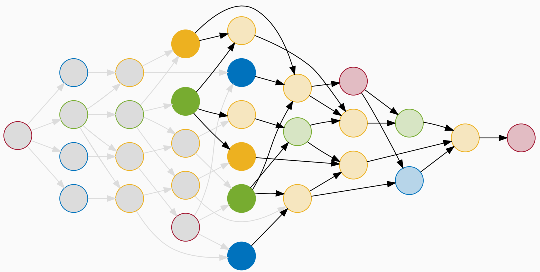


General-purpose runtimes

- High-level interface
- Data management
- Heterogeneity management

Task-based paradigm

Application → `taskify` → Directed Acylic Graph

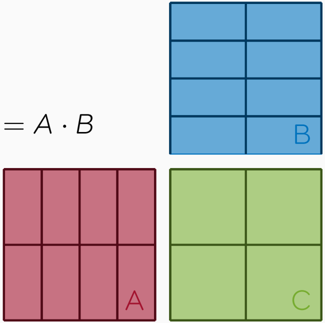


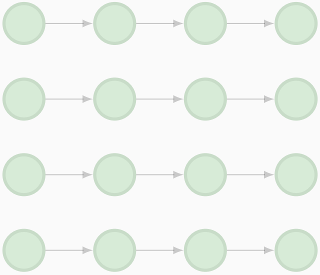
✓ Approach validated on heterogeneous single-node machine

- SOLHAR among multiple scientific projects

- Multiple ways to program the high-level description of the DAG
  - Parametrized Task Graph (PTG): PaRSEC
  - Sequential Task Flow (**STF**): PaRSEC, StarPU
  - ...

$C = A \cdot B$





DAG of GEMM

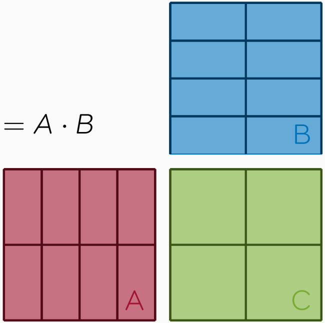
Algorithm 1: Sequential blocked GEMM.

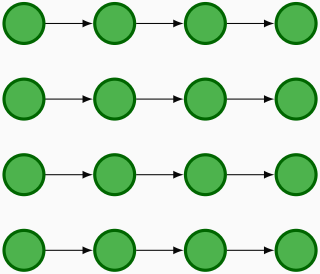
for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      $C_{ij} \mathrel{+}= A_{il} \cdot B_{lj}$

Algorithm 2: Shared-memory STF GEMM.

for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      `insert_task`(gemm, A_{il}:R, B_{lj}:R, C_{ij}:RW)
`task_wait()`

$C = A \cdot B$





DAG of GEMM

Algorithm 1: Sequential blocked GEMM.

for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      $C_{ij} \mathrel{+}= A_{il} \cdot B_{lj}$

Algorithm 2: Shared-memory STF GEMM.

for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      `insert_task`(gemm, A_{il}:`R`, B_{lj}:`R`, C_{ij}:`RW`)
`task_wait()`

Pros of runtime systems:

- Increased portability
- Improved separation of concerns

Pros of the STF model:

- Productive
- Clear translation from sequential

In practice, scientific computing software use **hand-tuned**, **application-tailored advanced** communication patterns, data layouts, workload mappings, . . .

Is STF expressive enough to transparently schedule these techniques and reach existing and new scalable algorithms ?

ScaLAPACK GEMM
$\mathcal{O}(1k)$ lines of code
legacy package

$\rightarrow ?$

Algorithm 3: STF GEMM.

```
for i = 1...m do
    for j = 1...n do
        for l = 1...k do
            insert_task(gemm, ?)
```

Three portable nested loops

Contributions

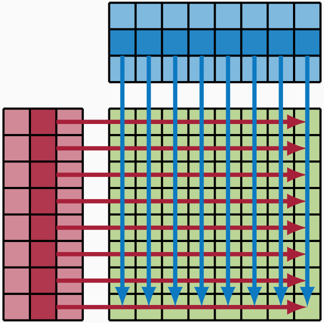
- Extension of the STF programming model
    - Data replication features (reduction, replication)
    - Validated over dense linera algebra operations (GEMM, SYMM, POTRF)

STF SCALABLE MATRIX-MATRIX MULTIPLICATION

Matrix-matrix multiplication – C-stationary

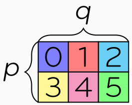
Scalable **U**niversal **M**atrix **M**ultiplication **A**lgorithms (SUMMA)

A-, **B**- and **C**-stationary depending on matrices' dimensions.



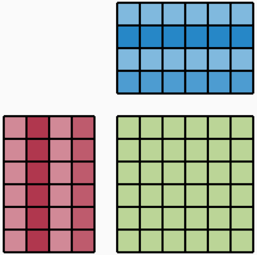
SUMMA – 2D C-stationary[1]

- 2D block-cyclic distribution



- owner of C computes
- **Communications**
  - row-wise broadcast of A
  - column-wise broadcast of B

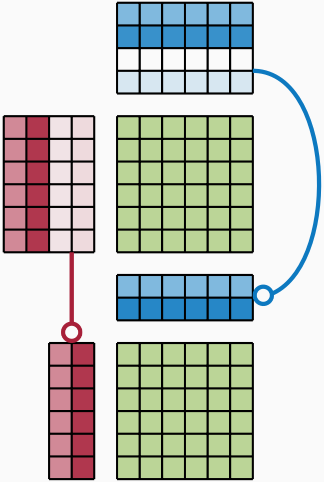
[1] R. v. d. Geijn and Watts, 1997



SUMMA – 3D C-stationary[2]

- 2D block-cyclic (first layer)
- owner of C's aisle (across layers) compute
- **Communications**
  - row-wise scatter+broadcast of A
  - column-wise scatter+broadcast of B
  - aisle-wise reduce of C

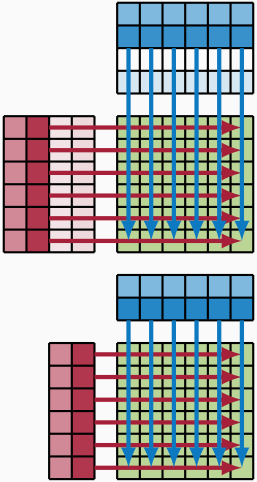
[2] Schatz, R. A. v. d. Geijn, and Poulson, 2016



SUMMA – 3D C-stationary[2]

- 2D block-cyclic (first layer)
- owner of C's aisle (across layers) compute
- **Communications**
  - row-wise **scatter**+broadcast of A
  - column-wise **scatter**+broadcast of B
  - aisle-wise reduce of C

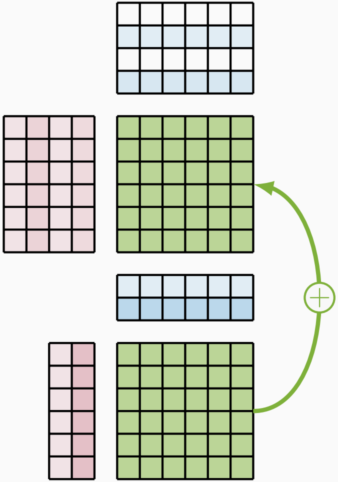
[2] Schatz, R. A. v. d. Geijn, and Poulson, 2016



SUMMA – 3D C-stationary[2]

- 2D block-cyclic (first layer)
- owner of C's aisle (across layers) compute
- **Communications**
  - row-wise scatter+**broadcast** of A
  - column-wise scatter+**broadcast** of B
  - aisle-wise reduce of C

[2] Schatz, R. A. v. d. Geijn, and Poulson, 2016



SUMMA – 3D C-stationary[2]

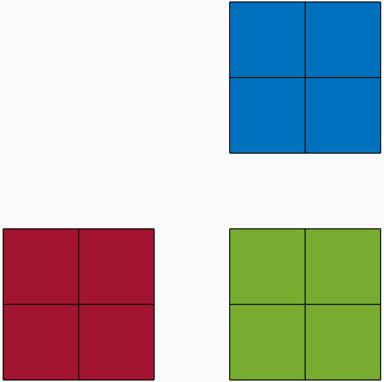
- 2D block-cyclic (first layer)
- owner of C's aisle (across layers) compute
- **Communications**
  - row-wise scatter+broadcast of A
  - column-wise scatter+broadcast of B
  - aisle-wise **reduce** of C

[2] Schatz, R. A. v. d. Geijn, and Poulson, 2016

Baseline approach

Algorithm 4: Baseline 2D GEMM.

for $i = 1 \ldots m$ **do**
  for $j = 1 \ldots n$ **do**
    for $l = 1 \ldots k$ **do**
      `insert_task` (gemm, A_{il}:R, B_{lj}:R, C_{ij}:RW)



Baseline model[a]

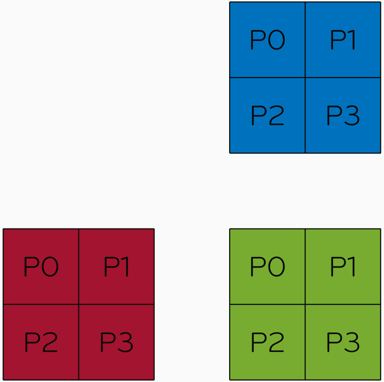
1. Data mapping
2. Task mapping **inferred** from data mapping
3. Point-to-point comms. **inferred** from task mapping

[a] Agullo, Aumage, Faverge, Furmento, Pruvost, Sergent, and Thibault, 2017.

Baseline approach

Algorithm 4: Baseline 2D GEMM.

for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      `insert_task` (`gemm`, A_{il}:`R`, B_{lj}:`R`, C_{ij}:`RW`)



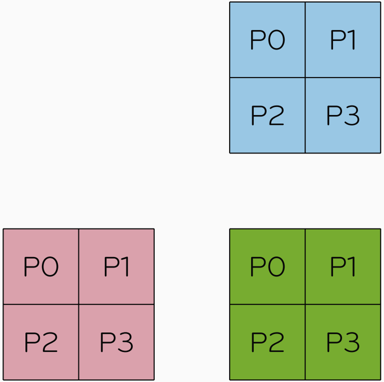
Baseline model[a]

1. Data mapping
2. Task mapping **inferred** from data mapping
3. Point-to-point comms. **inferred** from task mapping

[a] Agullo, Aumage, Faverge, Furmento, Pruvost, Sergent, and Thibault, 2017.

Algorithm 4: Baseline 2D GEMM.

```
for i = 1...m do
    for j = 1...n do
        for l = 1...k do
            insert_task (gemm, A_il:R, B_lj:R,
              C_ij:RW)
```



Baseline model[a]

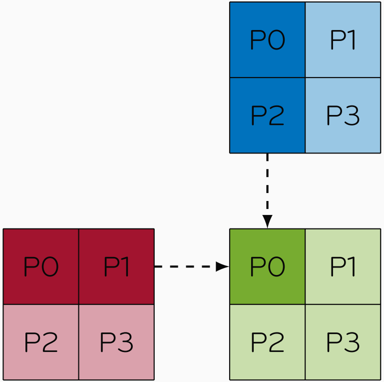
1. Data mapping
2. Task mapping **inferred** from data mapping
3. Point-to-point comms. **inferred** from task mapping

[a] Agullo, Aumage, Faverge, Furmento, Pruvost, Sergent, and Thibault, 2017.

Baseline approach

Algorithm 4: Baseline 2D GEMM.

for $i = 1 \ldots m$ **do**
  for $j = 1 \ldots n$ **do**
    for $l = 1 \ldots k$ **do**
      `insert_task` (gemm, A_{il}:R, B_{lj}:R, C_{ij}:RW)



Baseline model[a]

1. Data mapping
2. Task mapping **inferred** from data mapping
3. Point-to-point comms. **inferred** from task mapping

[a] Agullo, Aumage, Faverge, Furmento, Pruvost, Sergent, and Thibault, 2017.

Baseline STF model cannot **productively** provide an expression of SUMMA algorithms

- ✗ Task mapping is bound to data mapping
- ✗ Point-to-point communications patterns are inefficient
- ✗ Reduction patterns would burden the expression

Key features – task mapping

Algorithm 5: 3D GEMM (I/III).

for $i = 1 \dots m$ **do**
  for $j = 1 \dots n$ **do**
    for $l = 1 \dots k$ **do**
      `insert_task` (gemm, A_{il}:R, B_{lj}:R, C_{ij}:RW)

Advanced model

1. Data mapping
2. Implicit task mapping
3. Point-to-point comms. inferred from task mapping

Key features – task mapping

Algorithm 5: 3D GEMM (I/III).

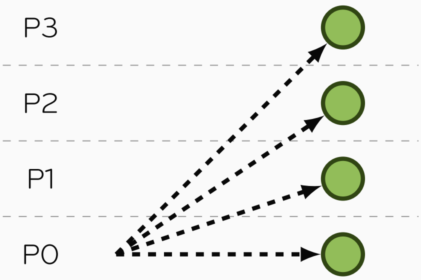
for $i = 1 \ldots m$ **do**
    for $j = 1 \ldots n$ **do**
        for $l = 1 \ldots k$ **do**
            rank = map(i,j,l,stat,s)
            `insert_task` (`gemm`, A_{il}:`R`, B_{lj}:`R`, C_{ij}:`RW`, **rank**:`ON_RANK`)

Advanced model

1. Data mapping
2. **Explicit** task mapping
3. Point-to-point comms. inferred from task mapping

Algorithm 6: 3D GEMM (II/III).

```
for i = 1 ... m do
    for j = 1 ... n do
        for l = 1 ... k do
            rank = map(i,j,l,stat,s)
            insert_task (gemm, A_il:R, B_lj:R,
              C_ij:RW, rank:ON_RANK)
```

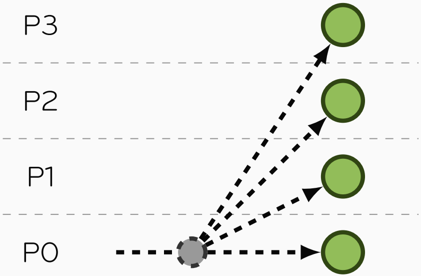


Advanced model

1. Data mapping
2. **Explicit** task mapping
3. Point-to-point comms. inferred from task mapping

Algorithm 6: 3D GEMM (II/III).

```
for i = 1...m do
    for j = 1...n do
        for l = 1...k do
            rank = map(i,j,l,stat,s)
            insert_task (gemm, A_il:R, B_lj:R,
              C_ij:RW, rank:ON_RANK)
```



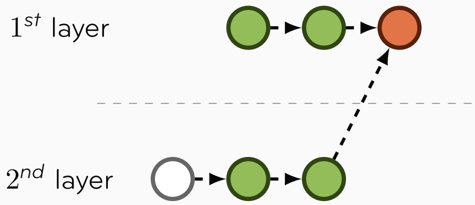
Advanced model

1. Data mapping
2. **Explicit** task mapping
3. **Collective** communications dynamically detected[a]

[a] Denis, Jeannot, Swartvagher, and Thibault, 2020.

Algorithm 7: 3D GEMM (III/III).

submit_initilization_tasks()
for $i = 1 \ldots m$ **do**
    for $j = 1 \ldots n$ **do**
        for $l = 1 \ldots k$ **do**
            rank = map(i,j,l,stat,s)
            `insert_task` (`gemm`, A_{il}:`R`, B_{lj}:`R`, $C_{ij}^{(\text{rank})}$:`RW`, rank:`ON_RANK`)
submit_reduction_tasks()

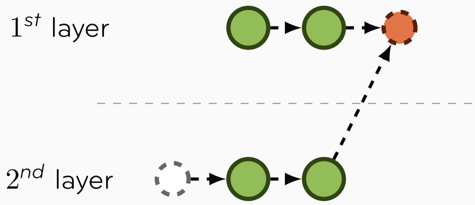


Advanced model

1. Data mapping
2. **Explicit** task mapping
3. **Collective** communications dynamically detected
4. Explicit reduction pattern

Algorithm 7: 3D GEMM (III/III).

```
bind_methods(C,redux,init)
for i = 1...m do
    for j = 1...n do
        for l = 1...k do
            rank = map(i,j,l,stat,s)
            insert_task (gemm, A_il:R, B_lj:R,
              C_ij:RANK_REDUX,
              rank:ON_RANK)
```



Advanced model

1. Data mapping
2. **Explicit** task mapping
3. **Collective** communications dynamically detected
4. **Implicit** reduction pattern

Algorithm 8: 3D GEMM.

```
bind_methods(C,redux,init)
for i = 1...m do
    for j = 1...n do
        for l = 1...k do
            rank = map(i,j,l,stat,s)
            insert_task (gemm, A_il:R, B_lj:R,
              C_ij:RANK_REDUX,
              rank:ON_RANK)
```

Conclusion I

Suitably extended STF model can **portably** express scalable GEMM algorithms.

Is this approach scalable in practice ?

Advanced model = STF✦

1. Data mapping
2. **Explicit** task mapping
3. **Collective** communications dynamically detected
4. **Implicit** reduction pattern

Algorithm 8: 3D GEMM.

bind_methods(C,redux,init)
for $i = 1 \ldots m$ **do**
  for $j = 1 \ldots n$ **do**
    for $l = 1 \ldots k$ **do**
      rank = map(i,j,l,stat,s)
      `insert_task` (gemm, A_{il}:R, B_{lj}:R, C_{ij}:RANK_REDUX, rank:ON_RANK)

Conclusion I

Suitably extended STF model can **portably** express scalable GEMM algorithms.

Is this approach scalable in practice ?

Advanced model = STF✦

1. Data mapping
2. **Explicit** task mapping
3. **Collective** communications dynamically detected
4. **Implicit** reduction pattern

Implementation details

Runtime system used is `StarPU` – extended with improved reduction features.

STF✦ GEMM has been implemented in `qr_mumps`.

software package	programming model	2D C	2D A	2D B	3D all	GPU
Chameleon 1.1	STF✕, StarPU 1.3	✓	✗	✗	✗	✓
SLATE	MPI+OpenMP	✓	✓	✗	✗	✓
DPlasma	PTG, PaRSEC	✓	✓	✓	✗	✓
ScaLAPACK	SPMD, BLACS	✓	✓	✓	✗	✗
Elemental	SPMD	✓	✓	✓	✗	✗
this work	STF✦, StarPU 1.4	✓	✓	✓	✓	✓

Experiments – Platforms

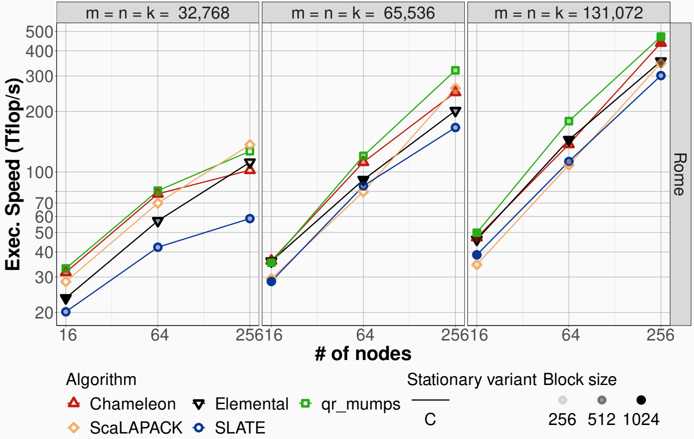
Machines hosted inside the Très Grand Centre de Calcul (TGCC).

A. **Irène-Rome** AMD Rome 7H12 2.6 GHz
   - 2,292 128-cores nodes with Infiniband HDR100 interconnect
   - $R_{\text{peak}} = 11.75$ PFlop/s
   - 256GB DDR4 RAM per node

B. **Irène-Skylake** Intel Skylake 8168 2.7 GHz
   - 1,656 48-cores nodes with Infiniband EDR interconnect
   - $R_{\text{peak}} = 6.86$ PFlop/s
   - 192GB DDR4 RAM per node

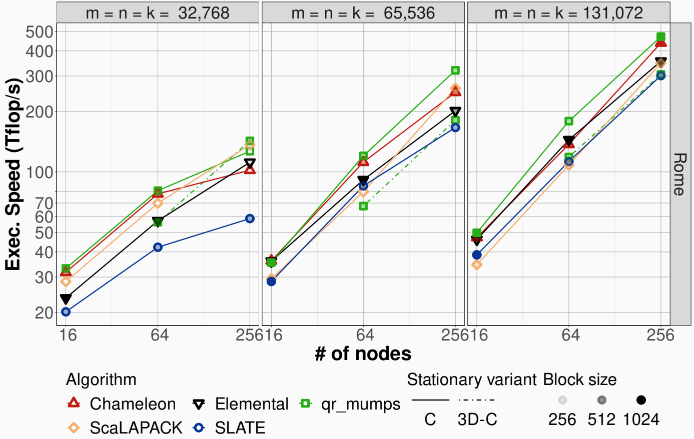
Overall about 1 million CPU hours consumed.

Experiments – Performance (C-stat)



Conclusion II

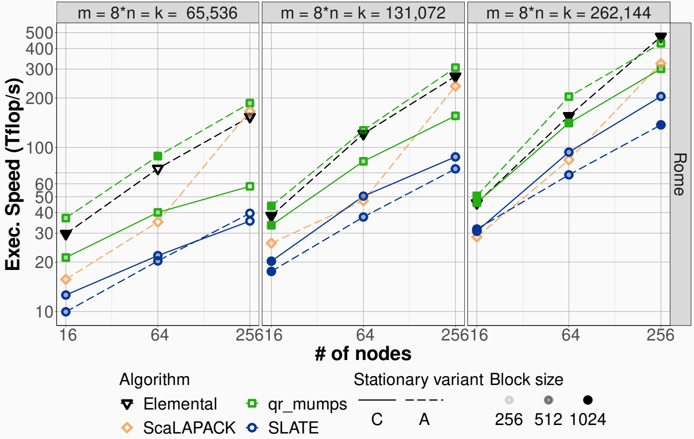
STF✦ offers competitive performance w.r.t reference libraries



Conclusion II

STF✦ offers competitive performance w.r.t reference libraries

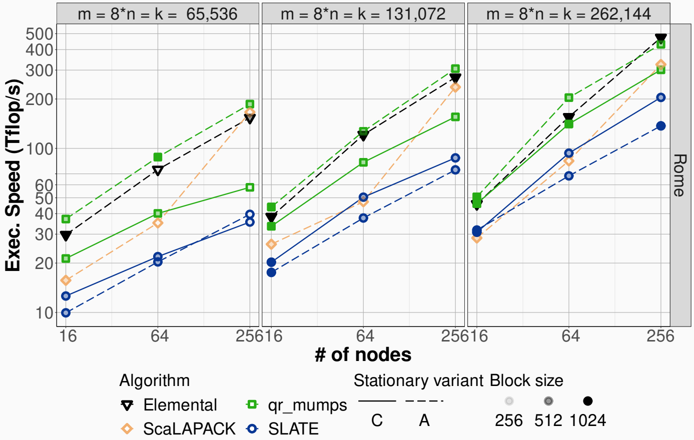
Experiments – Performance (A-stat)



Conclusion III

STF✦ brings versatility together with performance

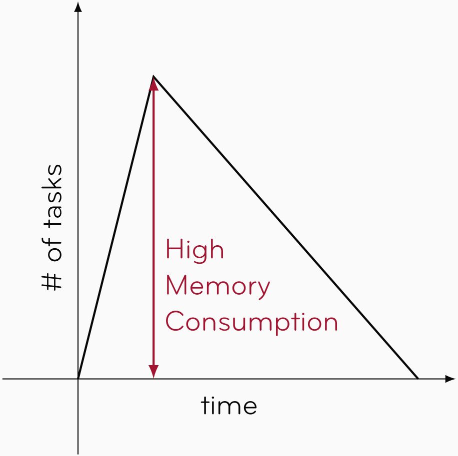
Experiments – Performance (A-stat)



Conclusion III

STF✦ brings versatility together with performance

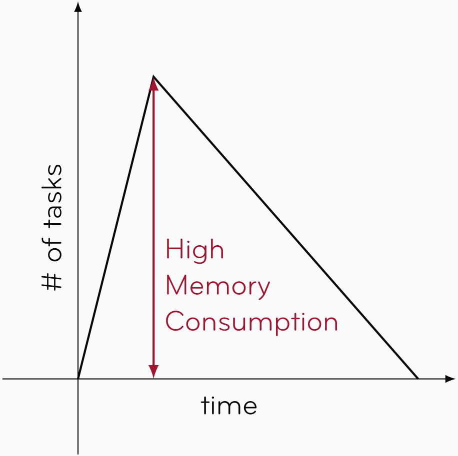
Unlimited graph unrolling



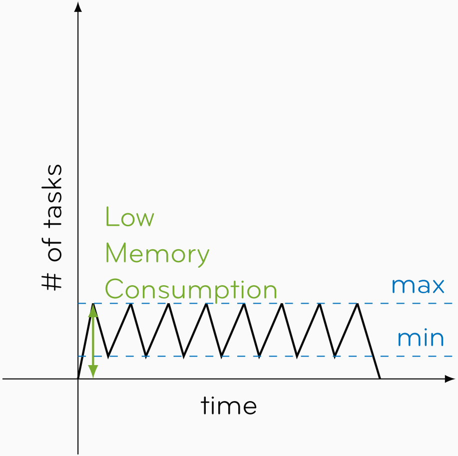
User-controlled graph unrolling

Memory consumption control through limits on tasks' flow.

Unlimited graph unrolling

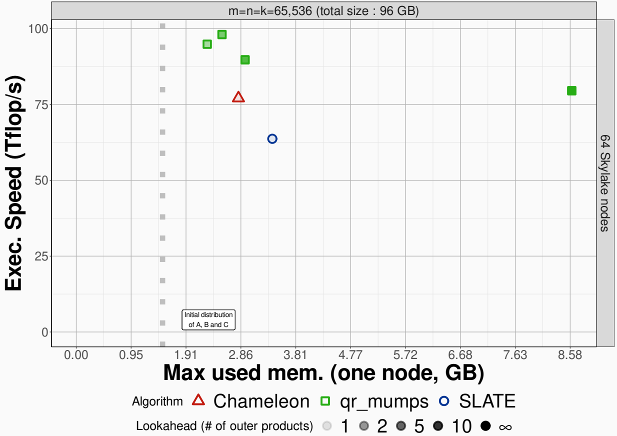


User-controlled graph unrolling



Memory consumption control through limits on tasks' flow.

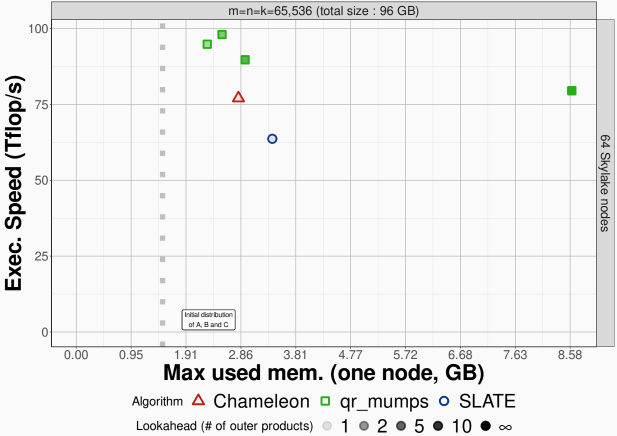
Experiments – Memory consumption



Conclusion IV

STF✦ incorporates tunable indirect method to control memory consumption

Experiments – Memory consumption



Conclusion IV

STF✦ incorporates tunable indirect method to control memory consumption

CONCLUSIONS

Contributions

STF✦ assessment over linear algebra operations

I **Suitably extended** STF model can **portably** express scalable GEMM algorithms

II STF✦ offers competitive performance w.r.t reference libraries

III STF✦ brings versatility together with performance

Additional contributions (not discussed here)

- Implementation of scalable SYMM, POTRF algorithms
- Implementation of a communication-avoiding 2D 5-points stencil prototype

Software contributions

- Implementation of additional features in StarPU
  - Distributed-memory reduction patterns
  - Data write replication (not discussed here, MR 66 and 68)
- Implementation of dense scalable routines in `qr_mumps`

Transfer

- Chameleon 1.3
  - Used by INRAE colleagues to analyse biodiversity datasets