About Interactions in Distributed Task-Based Runtime Systems for HPC Systems

Philippe SWARTVAGHER[1]

AFoDIB - Sémi-doc

[1]Inria Bordeaux – Sud-Ouest, 33405 Talence, France

Introduction

- PhD Student in 3rd year
- INRIA Bordeaux – Sud-Ouest, France
  - > TADaaM team (*Topology-Aware System-Scale Data Management for High-Performance Computing Applications*)
- Supervised by Alexandre DENIS and Emmanuel JEANNOT

- Thesis topic: **On the Interactions between Task-based Runtime Systems and Communication Libraries**
  - > HPC
  - > Task-based Runtime Systems
  - > Network Communications

Agenda

1) Quick introduction to HPC
2) Task-based Runtime Systems
3) Focus on my thesis:
   a) Interferences between communications and computations
   b) Dynamic broadcasts

01

Quick introduction to HPC

HPC

High Performance Computing

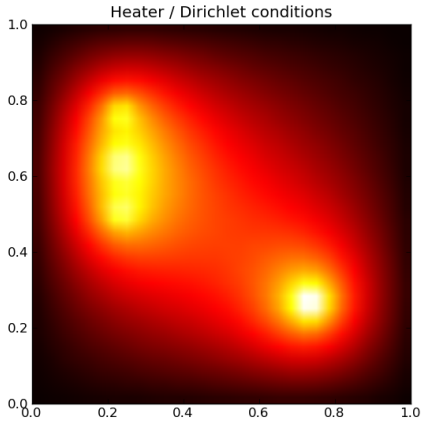
HPC

High Performance Computing

Use powerful computers to solve complex numerical problems

Numerical problems

- Usualy simulation
  - > Weather, climate forecasting
  - > Physical phenomena
  - > Earthquakes
  - > Nuclear physics
  - > Biology
  - > Chemistry
  - > ...



Simulation of heat diffusion

Numerical problems

- Require intensive computations
- Big problems: long executions
  > Several hours, days, weeks...
- Require lot of memory
  > A dense matrix 300 000 x 300 000 = 335 GB (simple precision)

Numerical problems

- Require intensive computations
- Big problems: long executions
  > Several hours, days, weeks...
- Require lot of memory
  > A dense matrix 300 000 x 300 000 = 335 GB (simple precision)

Not for your laptop!

Powerful computers

- Super-computers / Clusters / Machines / ...
- "Regular" servers equipped with cutting-edge technology
- Dedicated to HPC applications
- Concretely: many powerful servers connected with a network

Occigen cluster – CINES, Montpellier
Penalva, CC BY-SA 4.0, via Wikimedia Commons

Comparison

- Your laptop:
  - > CPU: 4 computing cores
  - > 8 GB RAM memory
  - > (GPU)
  - > Ethernet (Latency : ~ ms – Bandwidth: 1 GB/s)
  - > 200 Gflops

- A HPC computing node:
  - > CPU: 64 computing cores
  - > 256 GB RAM memory
  - > (GPU, FPGA, Xeon Phi, ...)
  - > HPC network (Latency : ~ µs – Bandwidth: 20 GB/s)
  - > 3000 Gflops

x 2 – 3000 !

But shared with other users: biggest jobs use ~ 200 nodes

Programming model

- Job: program executed on a supercomputer (on one or several nodes)
- Distribute problem input on job nodes
- Each node in a job has a rank [0, number of nodes in the job[
- Work on problem input / data, according to rank
- Send and receive data required, but located on another node
  > MPI standard for HPC communications

```
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &nb_nodes);

if (rank == 0) {
  dest = 1; // to right node
  source = nb_nodes-1; // from last node
  buffer = 0;

  MPI_Send(&buffer, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
  MPI_Recv(&buffer, 1, MPI_INT, source, 0, MPI_COMM_WORLD, NULL);

  fprintf(stdout, "Ring completed [%d]\n", buffer);
}
else {
  source = (rank - 1) % nb_nodes; // from left node
  dest = (rank + 1) % nb_nodes; // to right node

  MPI_Recv(&buffer, 1, MPI_INT, source, 0, MPI_COMM_WORLD, NULL);
  buffer++; // work
  MPI_Send(&buffer, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);

}
MPI_Finalize();
```

Programming model

- Instructions for communications
- Instructions for computations
  - > Depends on where computations will take place !
  - > CPU? OpenMP, pthread, ...
  - > GPU? Cuda, OpenCL, ...
  - > Xeon Phi?
  - > FPGA?

Programming model

- Instructions for communications
- Instructions for computations
  - > Depends on where computations will take place !
    - > CPU? OpenMP, pthread, ...
    - > GPU? Cuda, OpenCL, ...
    - > Xeon Phi?
    - > FPGA?

  - Computation source code different
  - Copy of data for computations between memories
    - > Data copy very costly
    - > Is the copy amortized by the performance of the device?
  - Which computations on which device ?
    - > Some devices better for some type of computations
    - > Scheduling challenges

Programming model

- Instructions for communications
- Instructions for computations
  - > Depends on where computations will take place !
    - > CPU? OpenMP, pthread, ...
    - > GPU? Cuda, OpenCL, ...
    - > Xeon Phi?
    - > FPGA?

  - Computation source code different
  - Copy of data for computations between memories
    - > Data copy very costly
    - > Is the copy amortized by the performance of the device?
  - Which computations on which device ?
    - > Some devices better for some type of computations
    - > Scheduling challenges

Hard to use all computational power manually!

02

Task-based runtime systems

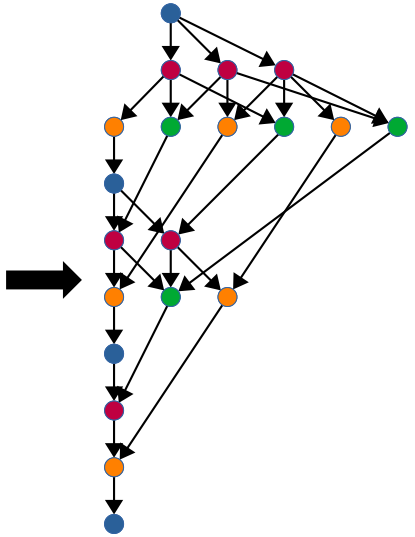
Task-based runtime systems

- Programming model based on a **Directed Acyclic Graph (DAG)**
  - > **Tasks** (small parts of the program) are vertices
  - > **Data dependencies** between tasks are edges
- The user provides:
  - > Dependencies between tasks
  - > Task implementation for each targeted architecture
- The runtime system :
  - > Schedules and executes tasks on computing units
  - > Performs required data transfers between computing units
  - > (Does all needed boring work)
- Examples : StarPU, PaRSEC, OmpSs, ...

The (famous) Cholesky example

- Cholesky decomposition:
  - > given a symetric positive definite matrix A,
  - > compute the lower triangular matrix L such that $A=LL^T$
  - > (very common in application using linear algebra)
- Tiled matrix: submatrices of A
  - > Divide the work
  - > Subprograms in the algorithm
    → tasks

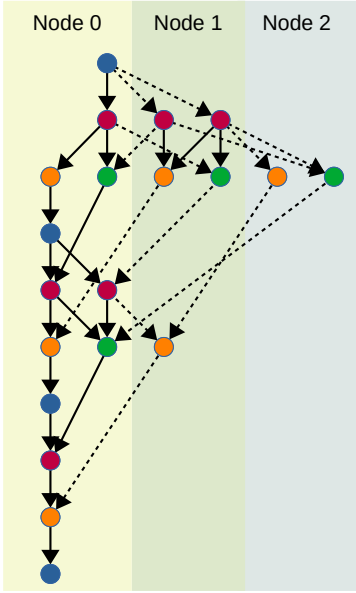
```
for j = 0; j < N; j++ do
    POTRF(RW, A[j][j])
    for i = j+1; i < N; i++ do
        TRSM(RW, A[i][j], R, A[j][j])
    end for
    for i = j+1; i < N; i++ do
        SYRK(RW, A[i][i], R, A[i][j])
        for k = j+1; k < i; k++ do
            GEMM(RW, A[i][k], R, A[i][j], R, A[k][j])
        end for
    end for
end for
```



Code and DAG of the Cholesky decomposition

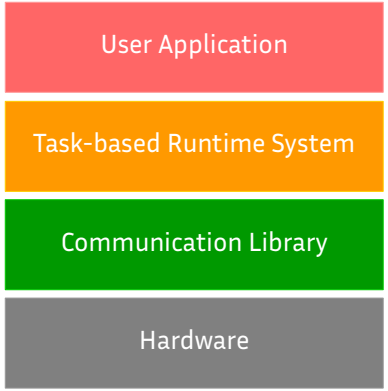
Distributed Task-Based Runtime Systems

- Run your program on several nodes
- Tasks are distributed among nodes (distributed memory)
- Required data exchanges can be **implicit** : inferred from the DAG
- Runtime systems often rely on an external communication library (e.g. based on MPI)



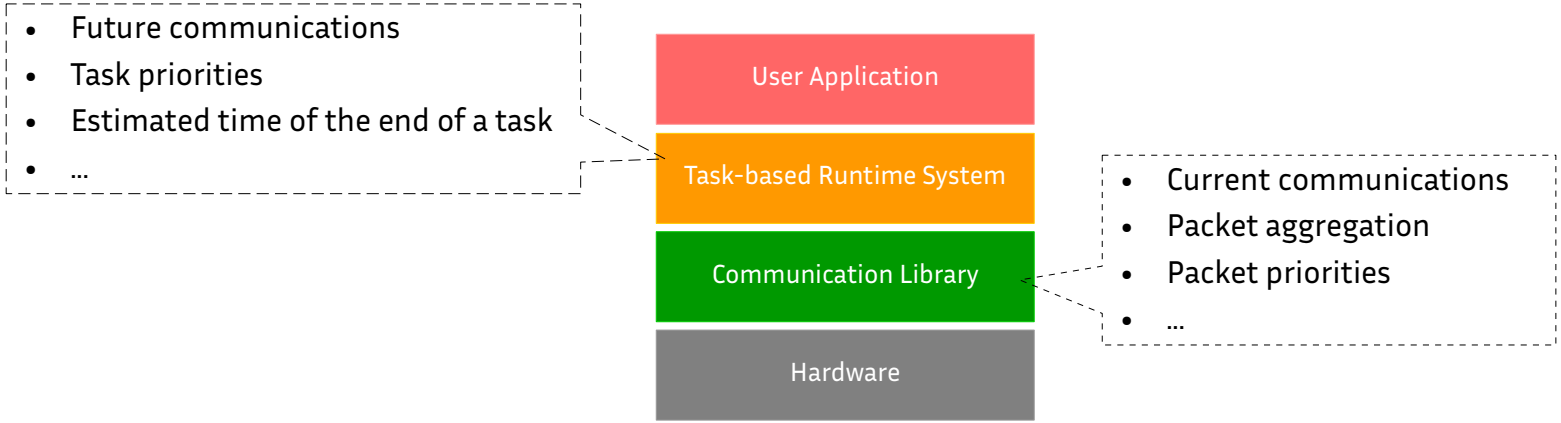
DAG of the Cholesky's decomposition executed on 3 nodes

Software stack



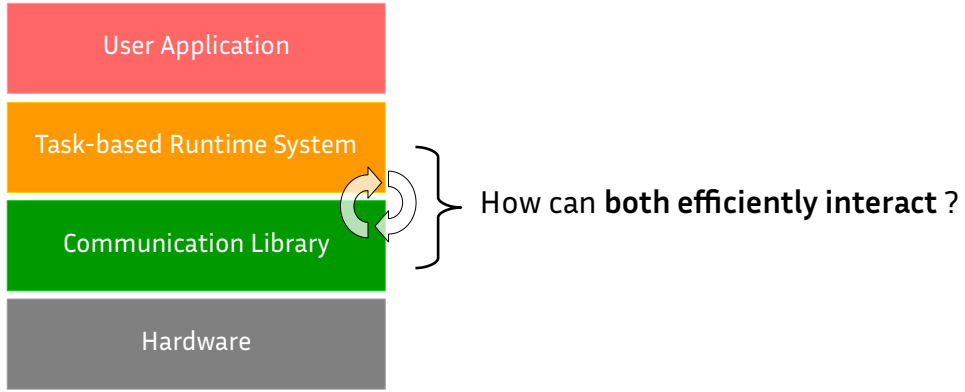
Runtime System and Communication Library

- Software layers: specific knowledge

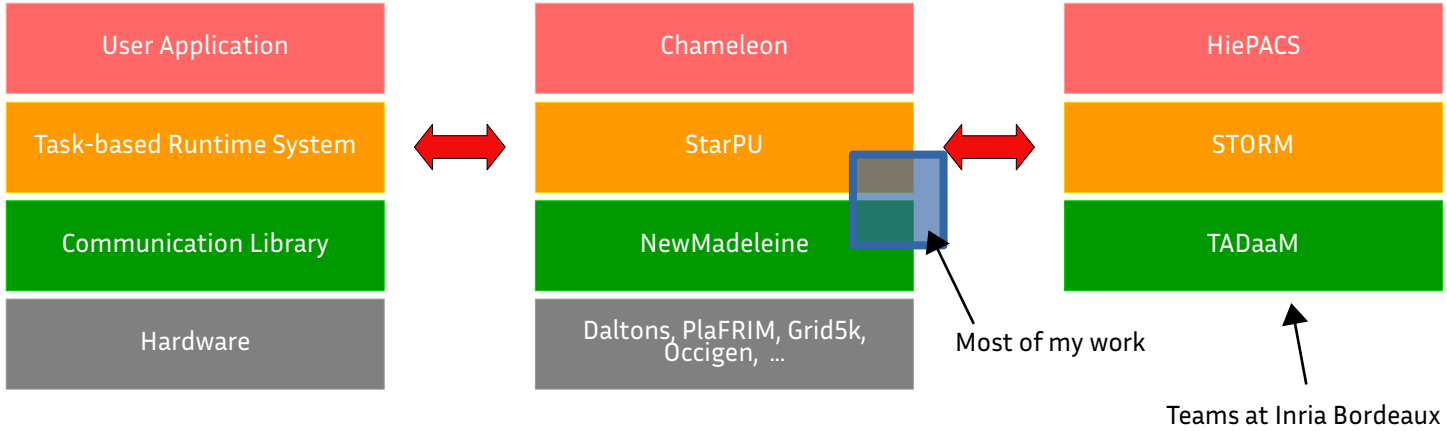


Runtime System and Communication Library

- My thesis topic



In practice


User Application
Task-based Runtime System
Communication Library
Hardware
Chameleon
StarPU
NewMadeleine
Daltons, PlaFRIM, Grid5k, Occigen, ...
HiePACS
STORM
TADaaM
Most of my work
Teams at Inria Bordeaux

03

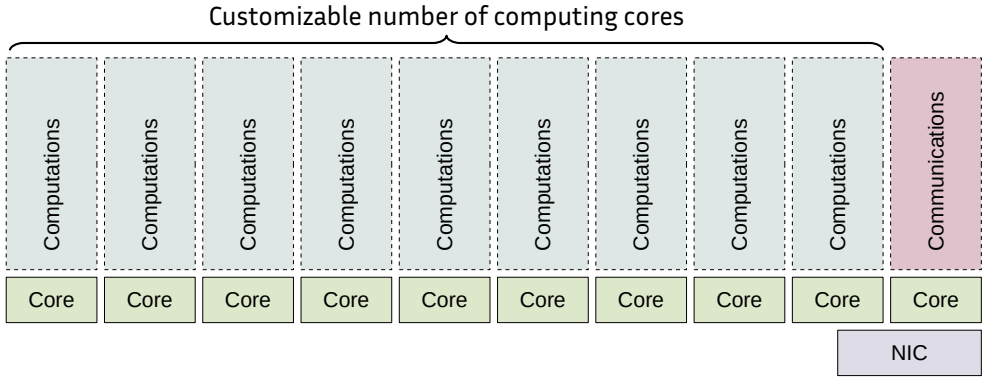
Contributions of my thesis

Two examples of interactions

- **Negative:** interferences between computations and communications
- **Positive:** dynamic broadcasts

Interferences between Communications and Computations

- In parallel computations and communications:
  - > StarPU's configuration: one thread for communications, other threads for computations
  - > Can computations impact communication performances? (and *vice-versa*?)



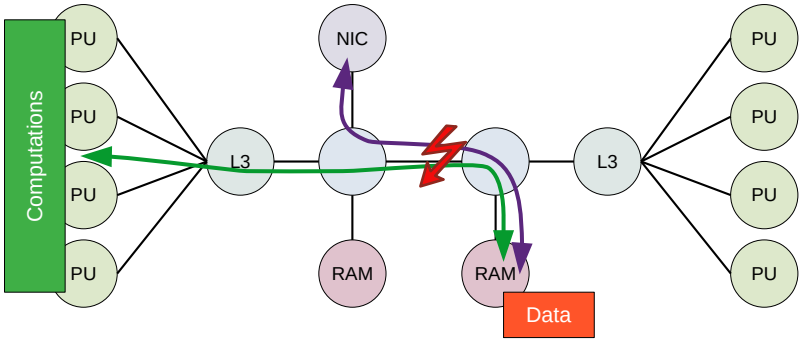
Interferences between Communications and Computations

- In parallel computations and communications:
  - > StarPU's configuration: one thread for communications, other threads for computations
  - > Can computations impact communication performances? (and *vice-versa*?)
- Yes!
  - > **Memory contention**

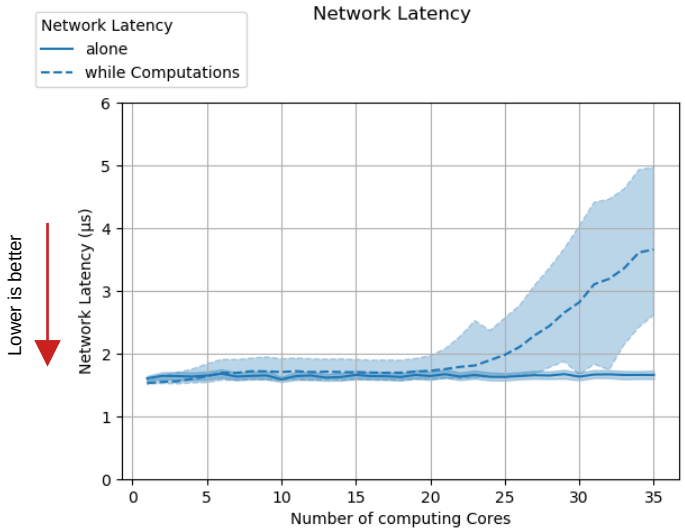
Alexandre Denis, Emmanuel Jeannot, Philippe Swartvagher. *Interferences between Communications and Computations in Distributed HPC Systems.* ICPP 2021 - 50th International Conference on Parallel Processing, Aug 2021, Chicago / Virtual, United States. ⟨10.1145/3472456.3473516⟩.

Memory contention

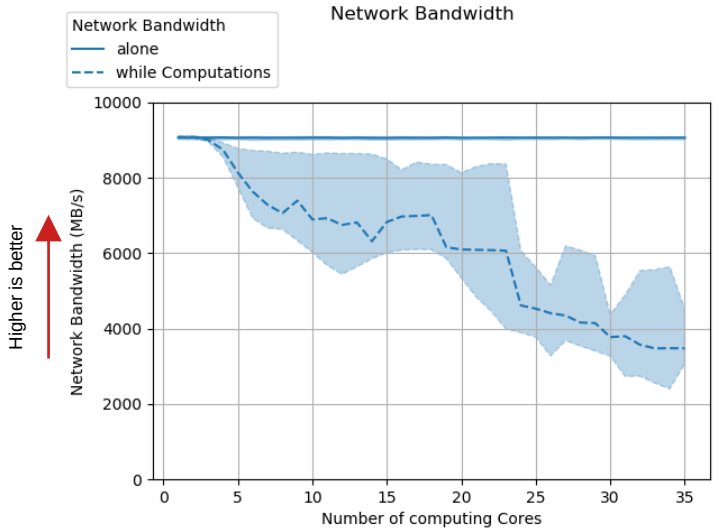
> Computations: data move between RAM and cores
> Communications: data move between RAM and NIC
> → **Contention on memory bus** !



Impacts of memory contention



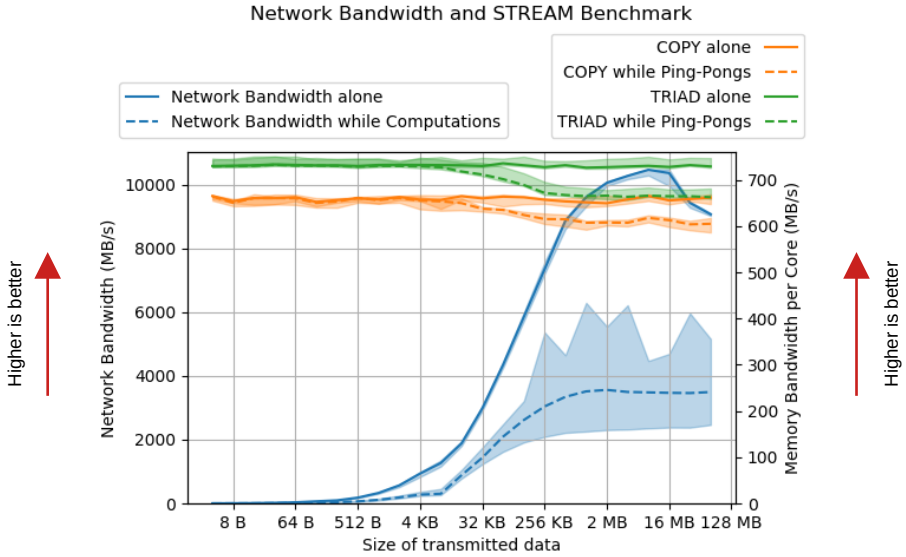
Network Latency Benchmark: **4 B**



Network Bandwidth Benchmark: **64 MB**

- Network latency impacted from 23 computing cores
- Network bandwidth impacted from 3 computing cores

Impacts of message size

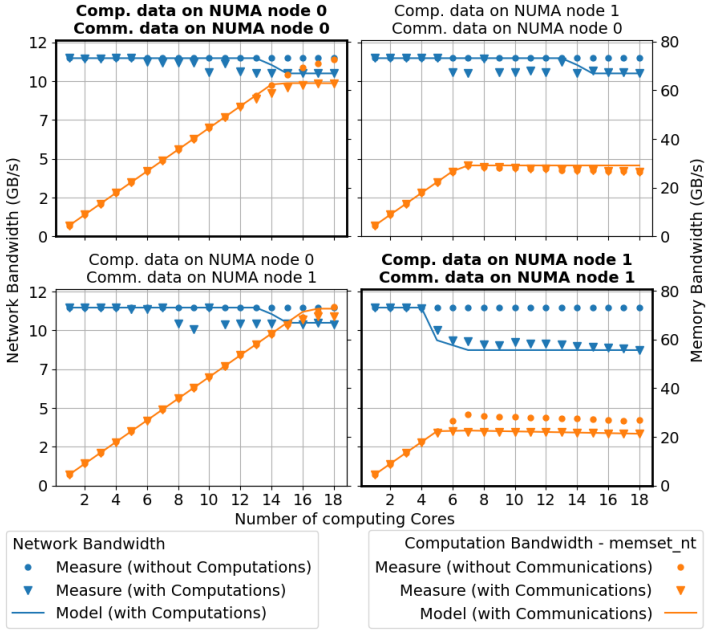


With 35 computing cores

- Large number of computing cores impacts a **wide range of message sizes**

Model of memory contention

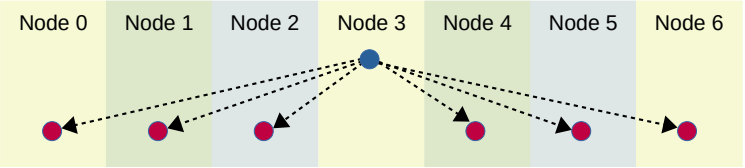
- Predict memory bandwidth for computations and communications
  - > When executed in parallel
  - > According to data locality
  - > Taking into account contention
- Allowed to understand locations of bottleneck
  - > Depends on behaviours not publicly documented
- Model error < 4%



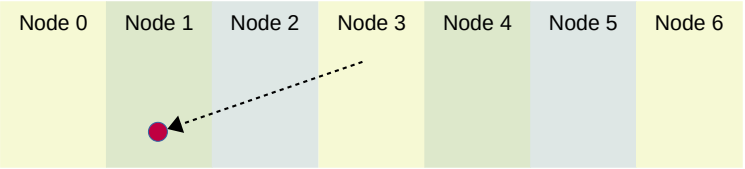
Alexandre Denis, Emmanuel Jeannot, Philippe Swartvagher. Modeling Memory Contention between *Communications and Computations in Distributed HPC Systems.* IPDPS - 2022 - IEEE International Parallel and Distributed Processing Symposium Workshops - APDCM, May 2022, Lyon / Virtual, France. *To appear.*

Broadcasts

- A data owned by a node can be needed on several other nodes: a **broadcast**
- **Detection of broadcasts:**
  - > Only from the DAG: communications are **inferred** from it
  - > Dynamic DAG: is the **recipient list complete** ?
- In StarPU, **local view** of the DAG per node:
  - > Only the sender node knows all recipients
  - > No information for recipient nodes
- → `MPI_Bcast` **not usable**



A broadcast pattern in the DAG, as seen by Node 3



The same broadcast pattern, as (not) seen by Node 1

Dynamic Broadcasts

Our contributions:

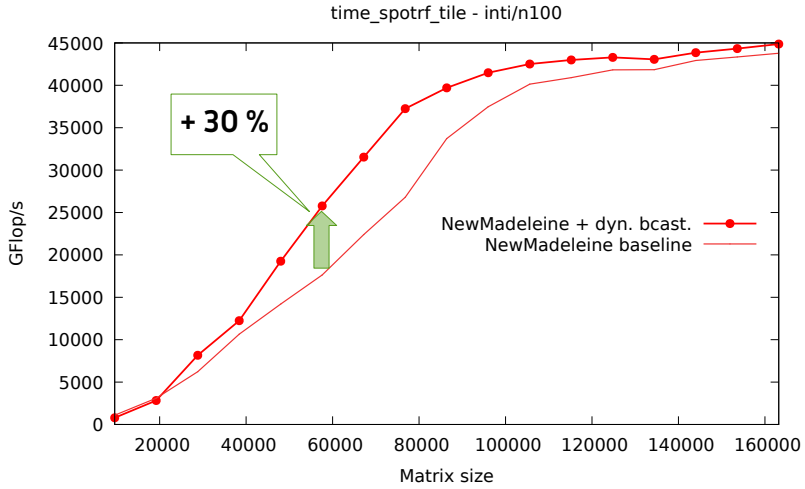
- **Detection** of broadcasts from the DAG
- Broadcasts with **efficient routing algorithms** (like `MPI_Bcast` does)
- Receive data from a broadcast **without knowing ahead of time**:
  - Part of a broadcast
  - The list of nodes in the broadcast

} Receiver nodes don't need to know it's a broadcast

Alexandre Denis, Emmanuel Jeannot, Philippe Swartvagher, Samuel Thibault. *Using Dynamic Broadcasts to improve Task-Based Runtime Performances.* Euro-Par - 26th International European Conference on Parallel and Distributed Computing, Rzadca and Malawski, Aug 2020, Warsaw, Poland. ⟨10.1007/978-3-030-57675-2_28⟩.

Results

- Impove performances up to 30% in Cholesky decomposition
  - >Leads to better scalability



Performance gain of tiled Cholesky decomposition with our *dynamic broadcasts* on 100 nodes (1600 cores).

Conclusion

- HPC: complex applications on complex powerful machines
- HPC machines: hard to use all computing power manually
- Task-based runtime systems: split program in small tasks and let the runtime system manage them
- Runtime systems and communication libraries have to work together
- Positive interaction: dynamic broadcasts
- Negative interactions: memory contention between computations and communications
  - > Memory system problem
  - > Taking it into account might be a positive interaction!