Inria

siam 2022 | Conference on Parallel Processing for Scientific Computing

Interactions between Task-Based Runtime Systems and the Communication Layer

Philippe SWARTVAGHER[1]

SIAM-PP22 - *Task-Based Programming for Distributed Memory Systems*

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

Introduction

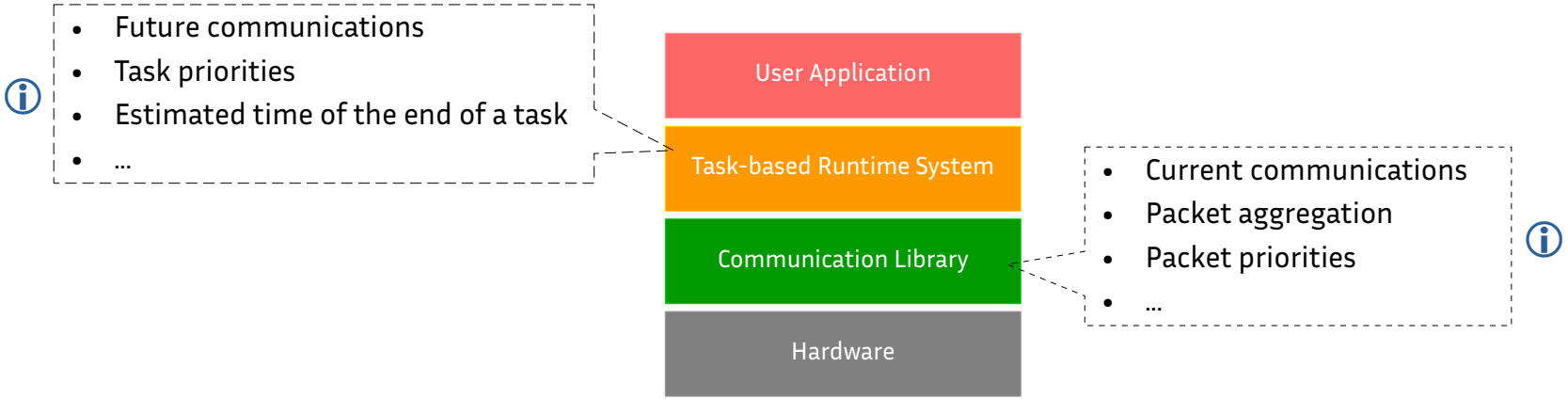
- PhD Student
- INRIA Bordeaux – Sud-Ouest, France
- Supervised by Alexandre DENIS and Emmanuel JEANNOT

- Increasing complexity of HPC machines
  > **Task-based runtime sytem** emergence
- **Communications**: one of key factors for scalability
  > But... often a bottleneck

How to improve interactions between task-based runtime systems and network communication libraries ?

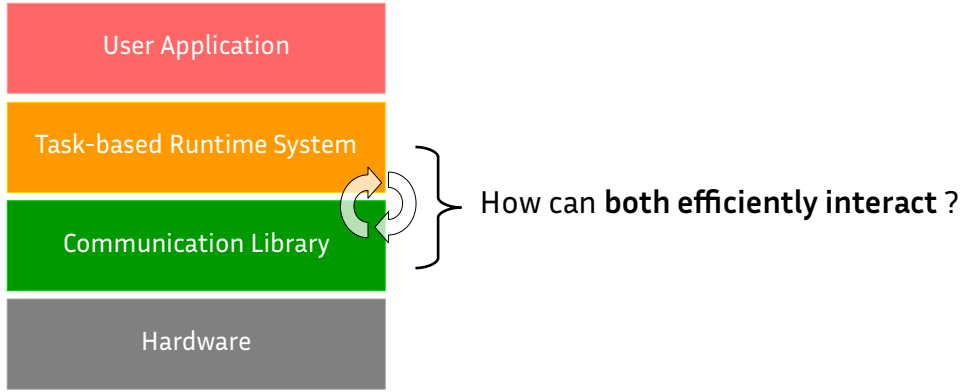
Runtime System and Communication Library

- Software layers: specific knowledge

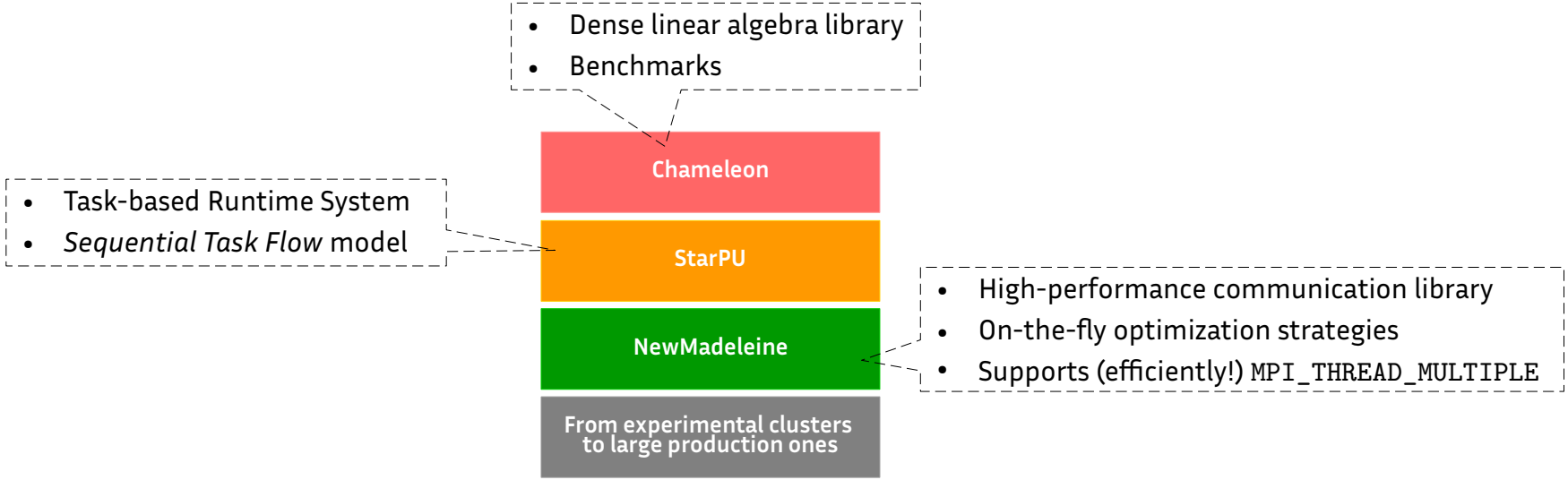


Runtime System and Communication Library

- Software layers: specific knowledge



Implementation



Two examples of interactions

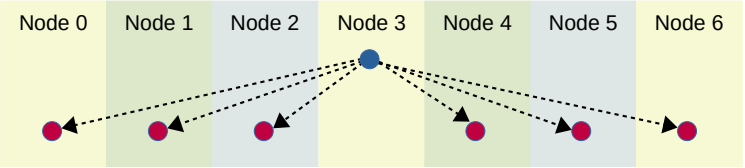
- **Positive:** dynamic broadcasts
- **Negative:** interferences between computations and communications

01

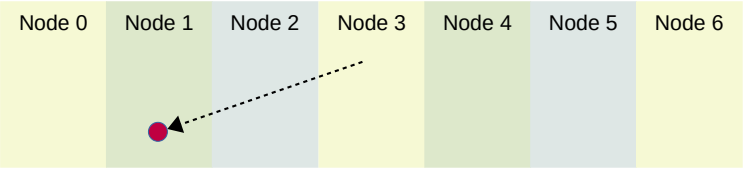
Dynamic Broadcasts

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

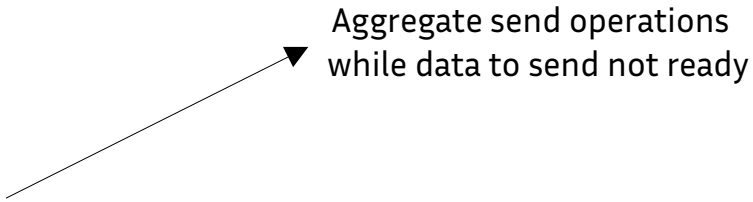
Dynamic Broadcasts

Our contributions:

- **Detection** of broadcasts
- Broadcasts with **efficient routing algorithms**
- Receive data from a broadcast **without knowing ahead of time**:
  - > Part of a broadcast
  - > The list of nodes in the 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⟩.

Dynamic Broadcasts



Our contributions:

- **Detection** of broadcasts
- Broadcasts with **efficient routing algorithms**
- Receive data from a broadcast **without knowing ahead of time**:
  - Part of a broadcast
  - The list of nodes in the 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⟩.

Dynamic Broadcasts

Aggregate send operations
while data to send not ready

Our contributions:

- **Detection** of broadcasts
- Broadcasts with **efficient routing algorithms**

Binomial trees, for instance

- Receive data from a broadcast **without knowing ahead of time**:
  - > Part of a broadcast
  - > The list of nodes in the 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⟩.

Dynamic Broadcasts

Our contributions:

- **Detection** of broadcasts
- Broadcasts with **efficient routing algorithms**
- Receive data from a broadcast **without knowing ahead of time**:
  - > Part of a broadcast
  - > The list of nodes in the broadcast

Aggregate send operations while data to send not ready

Binomial trees, for instance

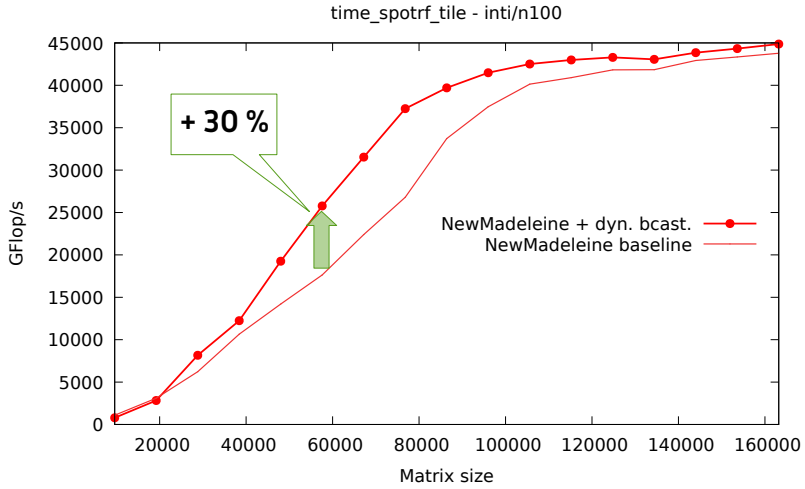
Source routing using active messages: **list of nodes** under the recipient nodes in the **message header**

Processed by the communication library, **outside of the application flow**

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



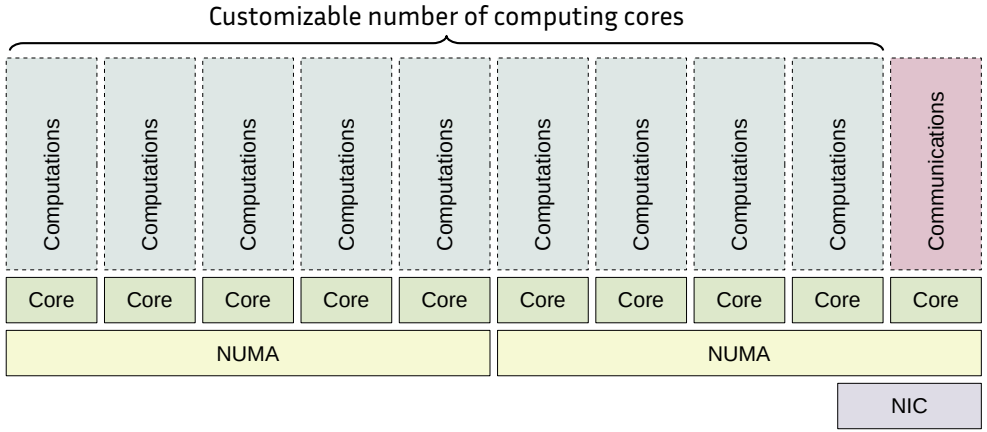
Performance gain of tiled Cholesky decomposition with our *dynamic broadcasts* on 100 nodes (1600 cores).

02

Interferences between Computations and Communications

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*?)



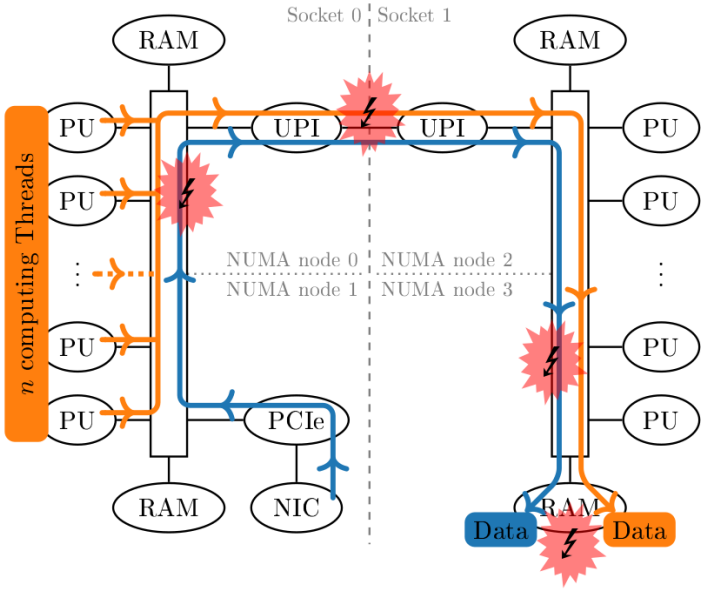
Interferences between Communications and Computations

- 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** !



Experimental protocol

- **Goal**: compare performances of computations and communications alone and in parallel
  - → benchmarking program

Experimental protocol

- **Goal**: compare performances of computations and communications alone and in parallel
  > → benchmarking program
- 3 steps:
  1) Computations alone
  2) Communications alone
  3) Computations and communications in parallel → to see the impact of one on each other

Experimental protocol

- **Goal**: compare performances of computations and communications alone and in parallel
  > → benchmarking program
- 3 steps:
  1) Computations alone
  2) Communications alone

  } to get their nominal performances → Compare them

  3) Computations and communications in parallel → to see the impact of one on each other

Experimental protocol

- **Goal**: compare performances of computations and communications alone and in parallel
  - → benchmarking program
- 3 steps:
  1) Computations alone
  2) Communications alone

  } to get their nominal performances → Compare them
  3) Computations and communications in parallel → to see the impact of one on each other
- Computations:
  - Memory-bound: STREAM kernels:

    ```
    COPY : for(i=0; i<ARRAY_SIZE; i++) A[i]=B[i]
    TRIAD: for(i=0; i<ARRAY_SIZE; i++) C[i]=A[i]+3.14*B[i]
    ```

    → memory bandwidth per core (higher is better)
  - Embarrassingly parallel, independant from communications

Experimental protocol

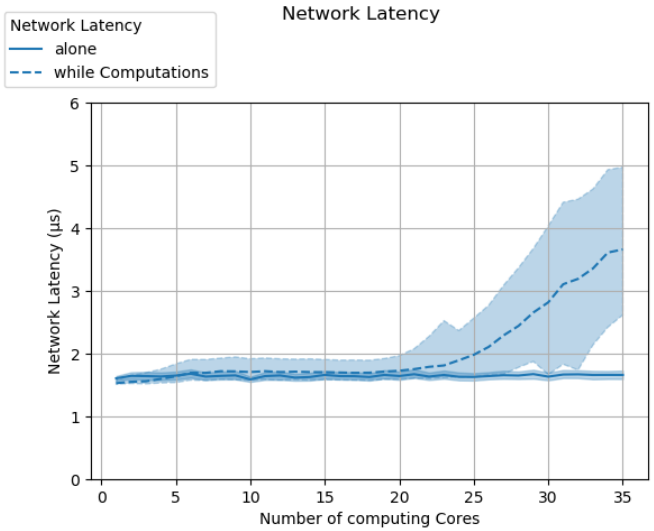
- **Goal**: compare performances of computations and communications alone and in parallel
  - → benchmarking program
- 3 steps:
  1) Computations alone
  2) Communications alone

  } to get their nominal performances → Compare them
  3) Computations and communications in parallel → to see the impact of one on each other
- Computations:
  - Memory-bound: STREAM kernels:

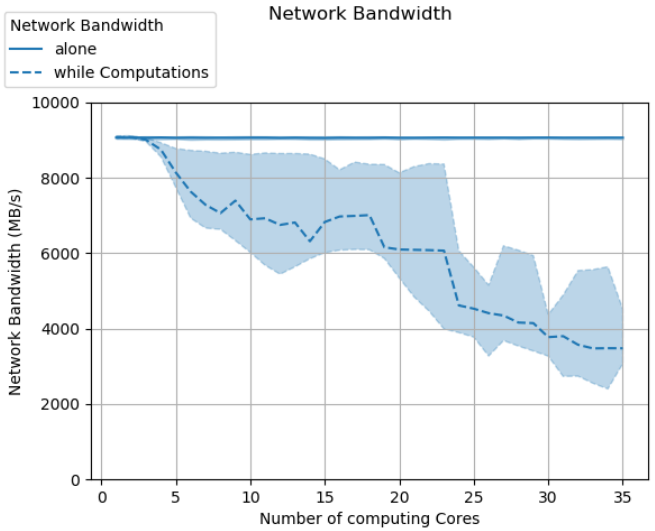
    ```
    COPY : for(i=0; i<ARRAY_SIZE; i++) A[i]=B[i]
    TRIAD: for(i=0; i<ARRAY_SIZE; i++) C[i]=A[i]+3.14*B[i]
    ```

    → memory bandwidth per core (higher is better)
  - Embarrassingly parallel, independant from communications
- Communications:
  - 2 MPI processes (one per node)
  - Ping-pongs to measure network latency (with 4 B) and bandwidth (with 64 MB)

Impacts of memory contention



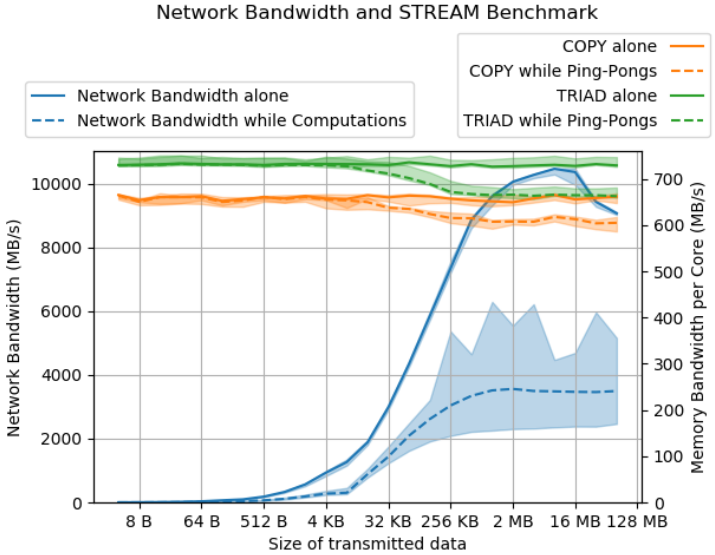
Network Latency Benchmark: **4 B**



Network Bandwidth Benchmark: **64 MB**

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

Impacts of message size



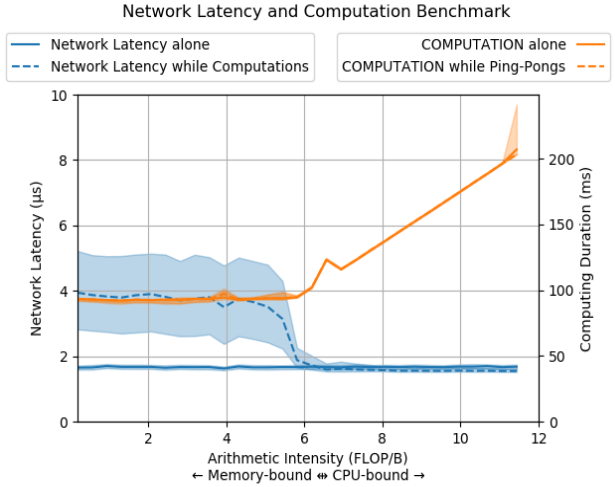
With 35 computing cores

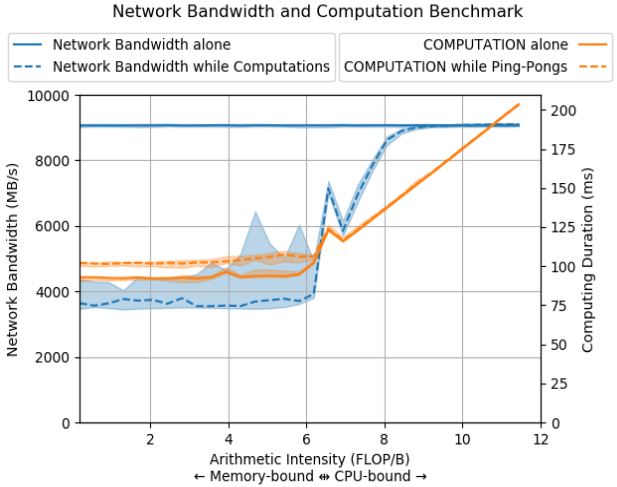
- Large number of computing cores impacts a **wide range of message sizes**

Impacts of arithmetic intensity

- *Arithmetic intensity*: number of flops per byte of moved data
- → `TRIAD` with tunable arithmetic intensity

```
for (i = 0; i < ARRAY_SIZE; i++)
    for (c = 0; c < cursor; c++)
        C[i] = A[i] + 3.14 * B[i]
```





- **Computations more CPU-bound → less trafic on memory bus → less contention**

Conclusion

- 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!
- Future work:
  - > Take into account memory contention (data placement strategies, number of computing cores, etc)
  - > What about GPUs? (memory traffic, faster tasks, etc)

This work is supported by the Agence Nationale de la Recherche, under grant ANR-19-CE46-0009.

This work is supported by the Région Nouvelle-Aquitaine, under grant 2018-1R50119 HPC scalable ecosystem.

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr/).

This work was granted access to the HPC resources of CINES under the allocation 2019-A0060601567 attributed by GENCI (Grand Equipement National de Calcul Intensif).

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

Use-cases: computational kernels

- Computational kernels:
  - > Dense conjugate gradient (`CG`)
  - > Dense general matrix-matrix multiplication (`GEMM`)
- Metrics:
  - > Impact on network performance
  - > Number of stalled cycles → lost cycles waiting for memory
- `CG` is **more memory-bound** than `GEMM`:
  - > `CG` has more stalls
  - > `CG` has a **network bandwidth more impacted**