Interferences between Communications and Computations in Distributed HPC Systems

Alexandre DENIS, Emmanuel JEANNOT, Philippe SWARTVAGHER
Inria Bordeaux – Sud-Ouest, France

ICPP 2021 | TADaaM seminar

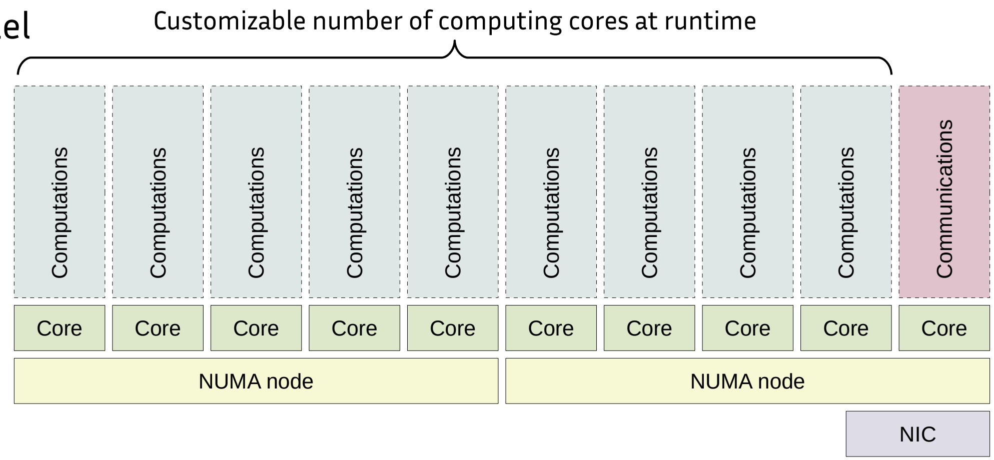
Introduction

- Communications: one of key factors for scalability
- Computations and communications in parallel: raising trend to get better performances
- Are there interferences between computations and communications ?
- Yes, communications can impact computations
  - Langguth, X. Cai, and M. Sourouri. 2018. Memory Bandwidth Contention: Communication vs Computation Tradeoffs in Supercomputers with Multicore Architectures. In 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). 497–506
  - T. Groves, R. E. Grant, and D. Arnold. 2016. NiMC: Characterizing and Eliminating Network-Induced Memory Contention. In 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 253–262.
- What about computations impacting communications ?

Context

- Distributed StarPU applications
  - Task-based runtime system for heterogeneous architectures

- Computations and communications in parallel
  - **One thread dedicated to communications**
  - Other threads perform computations
  - One thread bound per core



Origins of interferences ?

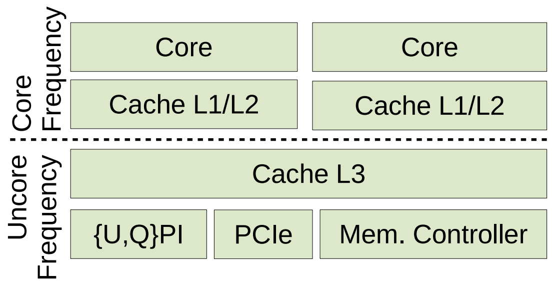
- Hypotheses:
  - Processor frequency variations
  - Memory contention
  - Runtime system

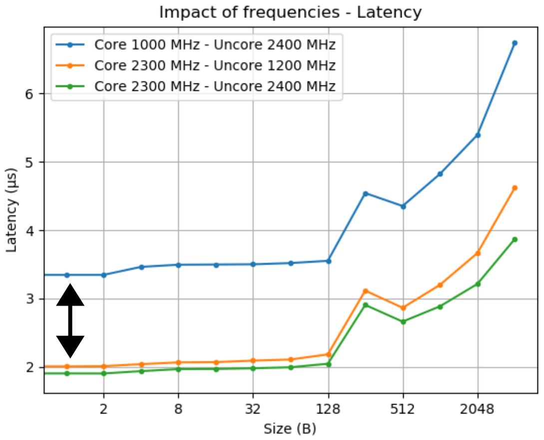
Frequency variations

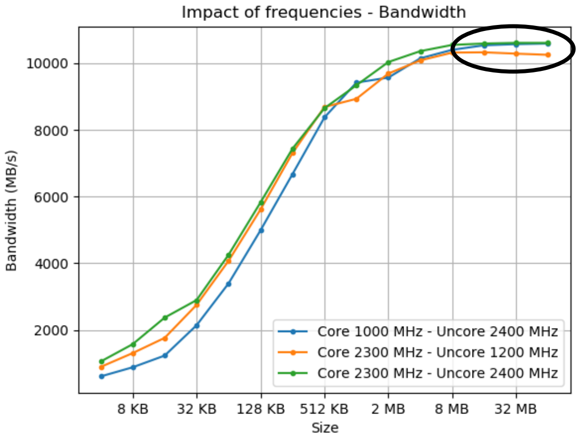
- Network performances are comprised of:
  - > Software overhead, to set up the communication
  - > Memory transfer time, to move the data between main memory and NIC
- Duration of these two steps: influenced by processor frequencies
- Dynamic frequency scaling of processors:
  - > **Change processor frequency according to workload** to avoid overheating
  - > AVX instructions (for computations) can force the core to lower its frequency

Frequency variations

- With alone communications and constant frequencies:
  - > *Core* frequency → has an impact on **network latency**
  - > *Uncore* frequency → has an impact on **network bandwidth**





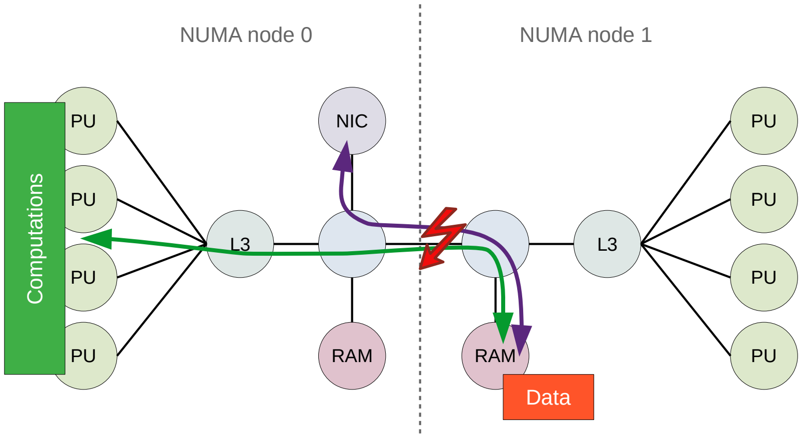


Frequency variations

- However, with computations in parallel of communications:
  - > CPU-bound computations
  - > Computing cores can change their frequency...
  - > ... but not the communicating core
    - → negligible impact on communications

Memory contention

- For computations, data move between RAM and cores
- For 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
- 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 → Compare them

- Computations:
  - Memory-bound: STREAM kernels:

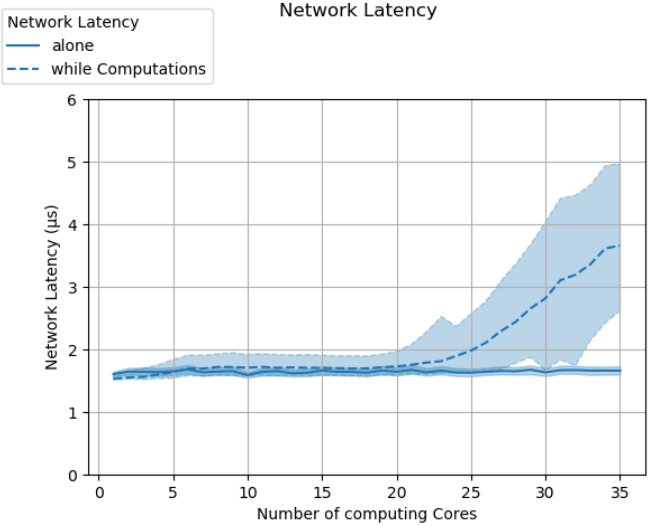
```
COPY : for(i=0; i<ARRAY_SIZE; i++) A[i]=B[i]
TRIAD: for(i=0; i<ARRAY_SIZE; i++) C[i]=A[i]+3.14*B[i]
```

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

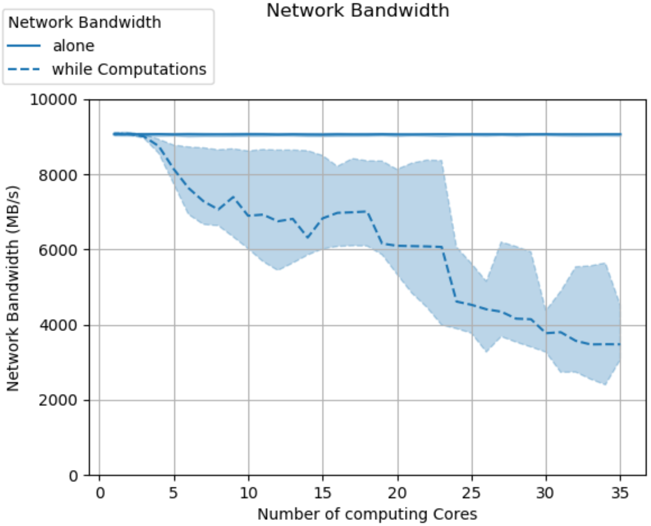
- Communications:
  - 2 MPI processes (one per node)
  - Ping-pongs to measure network latency (with 4 B) and bandwidth (with 64 MB)

Inria

Impacts of memory contention on communications



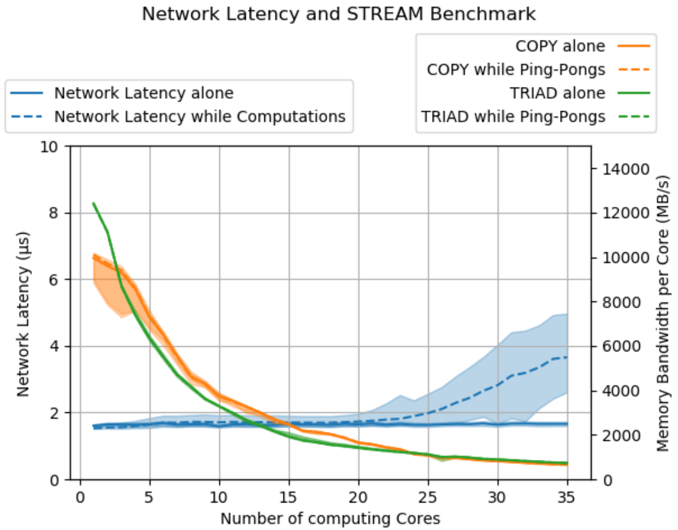
Network Latency Benchmark: **4 B**



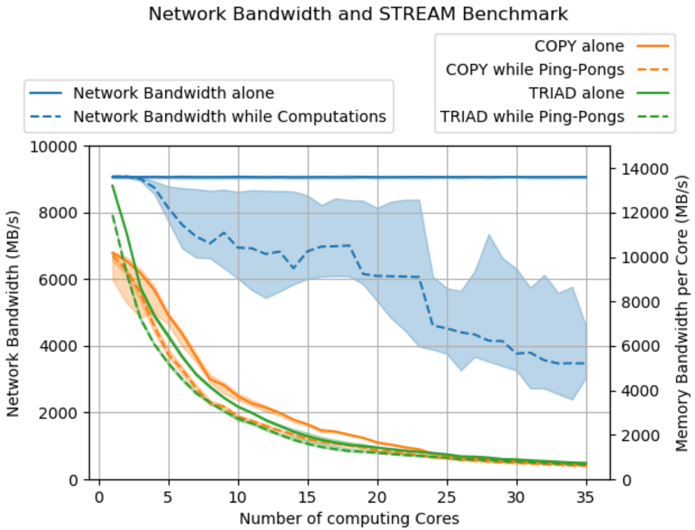
Network Bandwidth Benchmark: **64 MB**

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

Impacts of memory contention on communications & computations



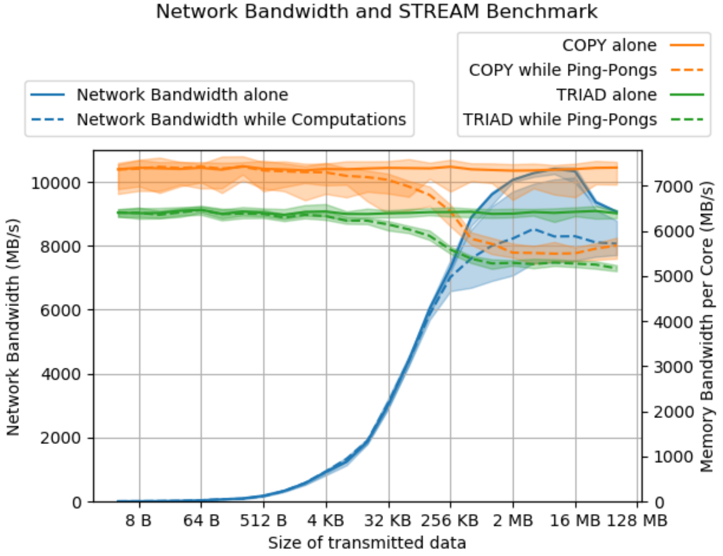
Network Latency Benchmark: **4 B**



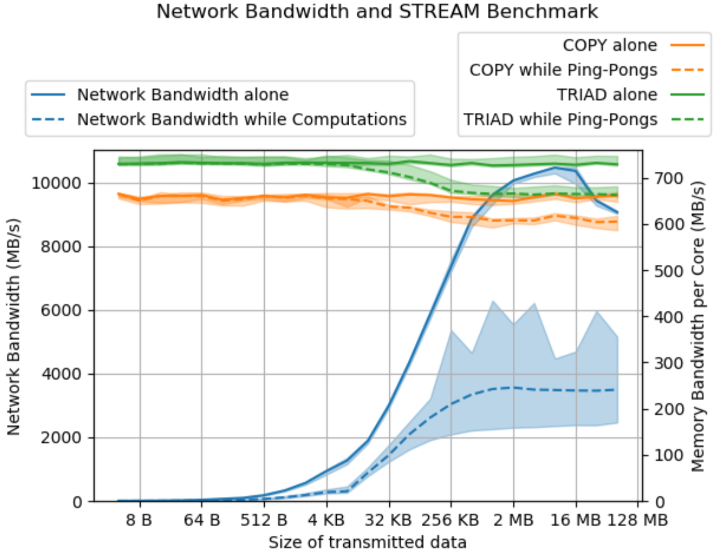
Network Bandwidth Benchmark: **64 MB**

- Computations impacted by ping-pongs to measure network bandwidth

Impacts of message size



With **5** computing cores

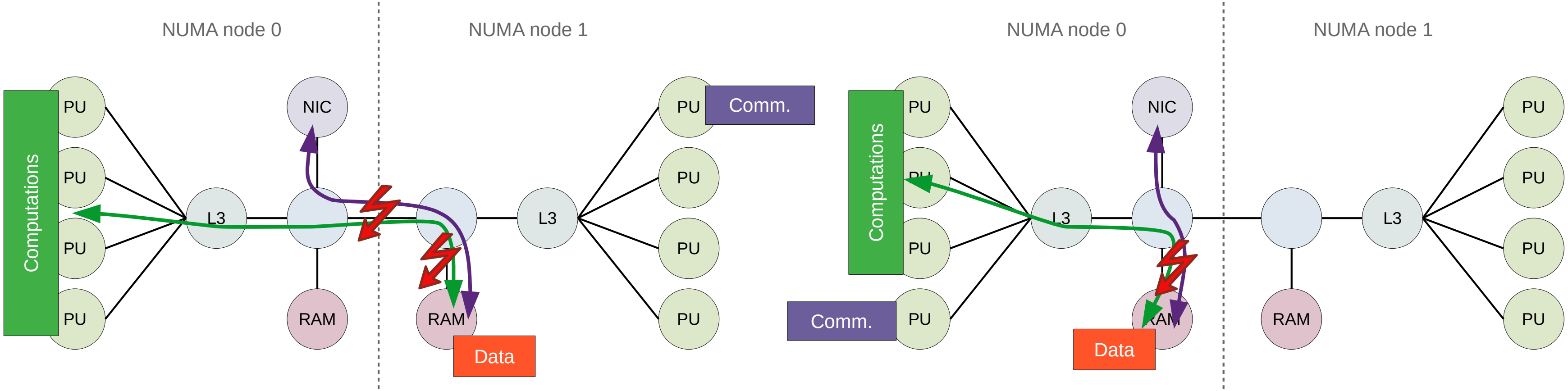


With **35** computing cores

- Large number of computing cores impacts a **wide range of message sizes**
- Large message sizes can disturb even a small number of computing cores

Impacts of placements

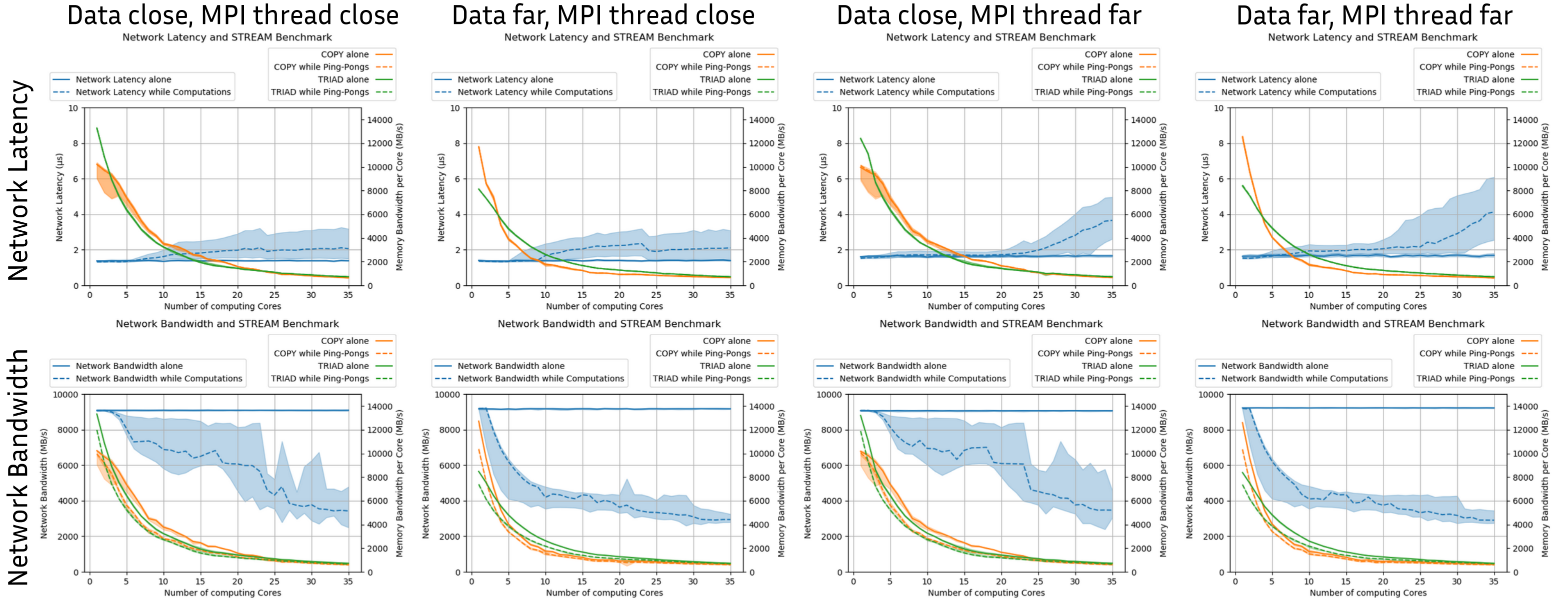
- **Placement** of data and communication thread **regarding the NIC**
  - > Change the path taken by the data and thus change the memory contention



One placement configuration

Another placement configuration

Impacts of placements

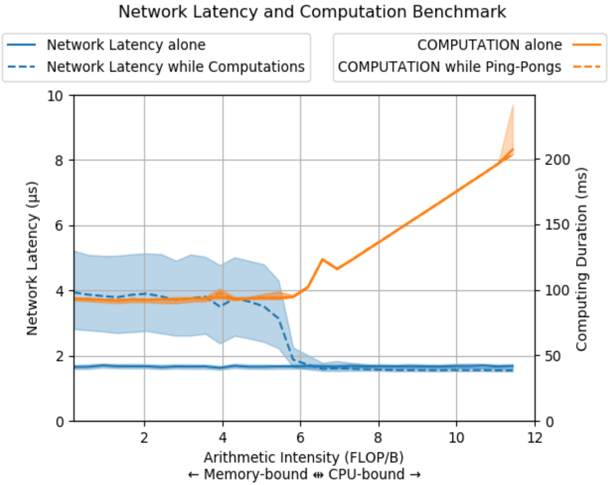


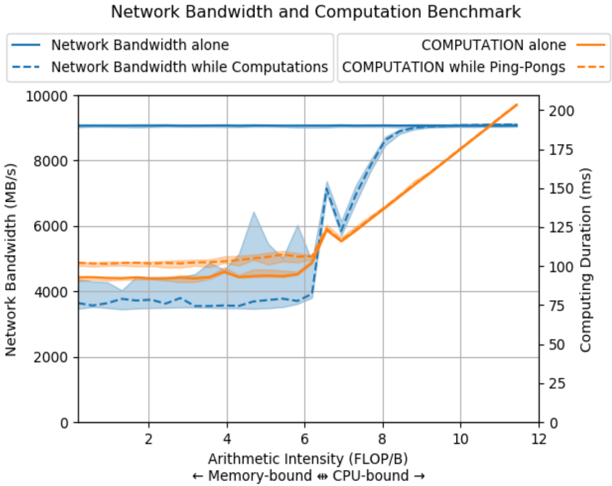
- Communication thread far from the NIC: network latency is more impacted by contention
- Data far from the NIC: network bandwidth is more impacted by contention
- STREAM always more impacted with network bandwidth benchmark in parallel

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]
```

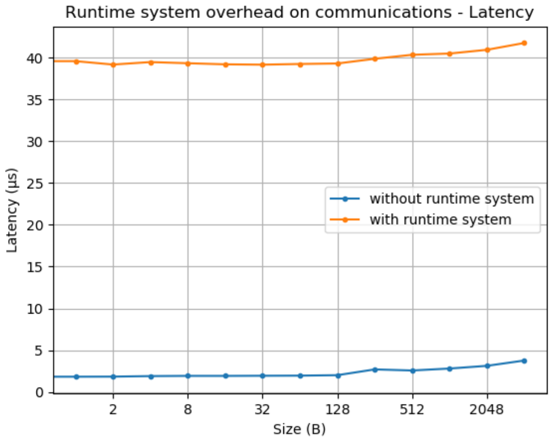


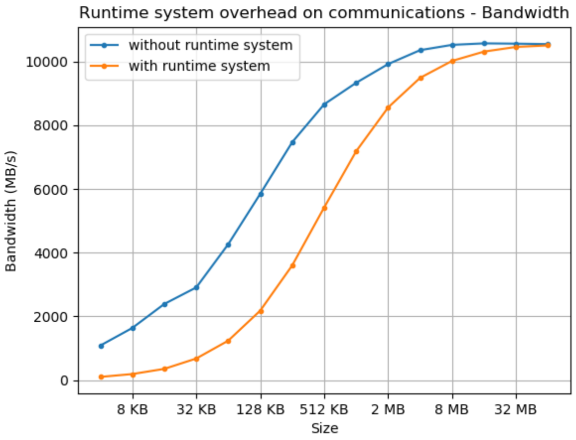


- **Computations more CPU-bound → less trafic on memory bus → less contention**

Impact of a runtime system

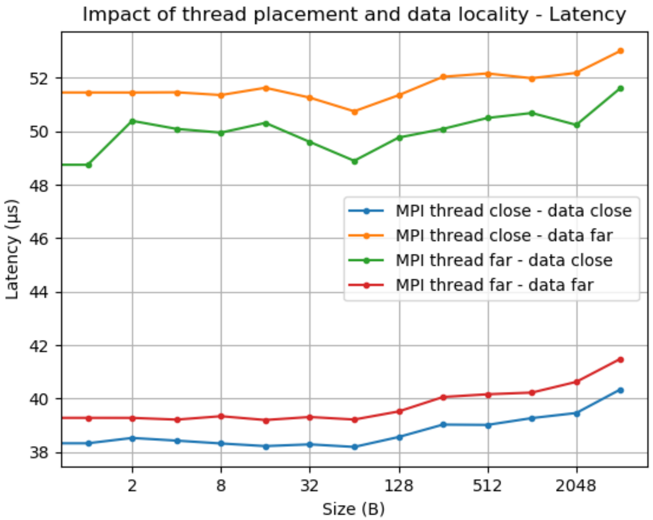
- Runtime system: **StarPU**
  - > Distributed task-based runtime system
  - > Abstracts network communications
- StarPU's overhead on lonely communications:
  - > Without runtime system: `MPI_Send()` / `MPI_Recv()`
  - > With runtime system: `starpu_mpi_send()` / `starpu_mpi_recv()`

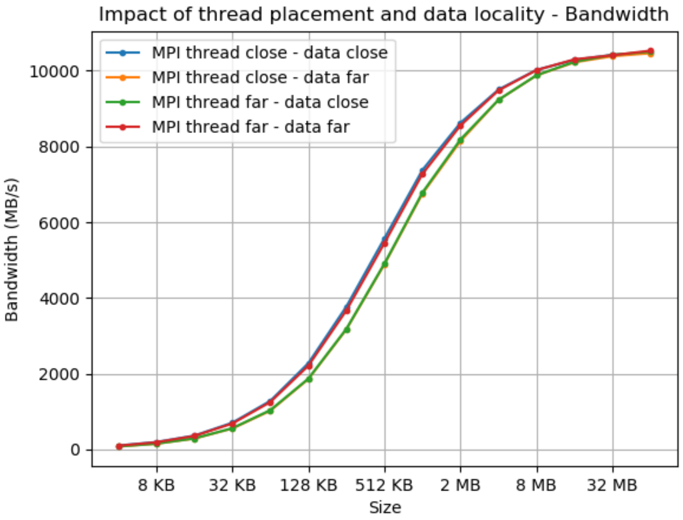




Impacts of placements with StarPU

- One thread dedicated to communication progression

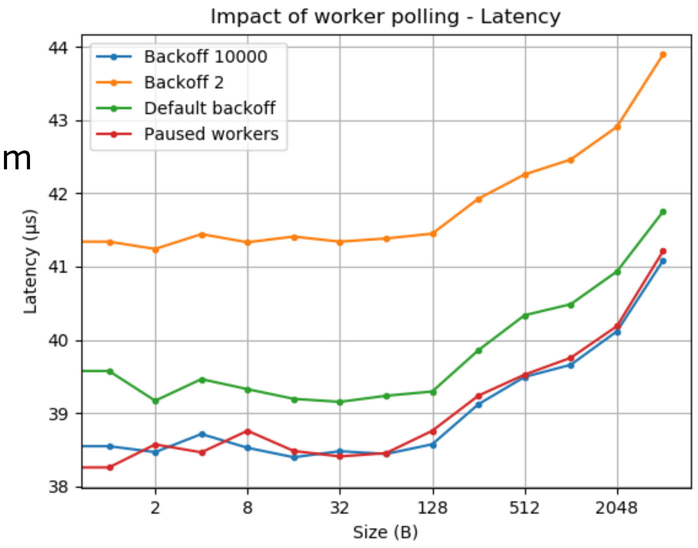




- Data to communicate has to be on **the same NUMA node** as the core running the communication thread.

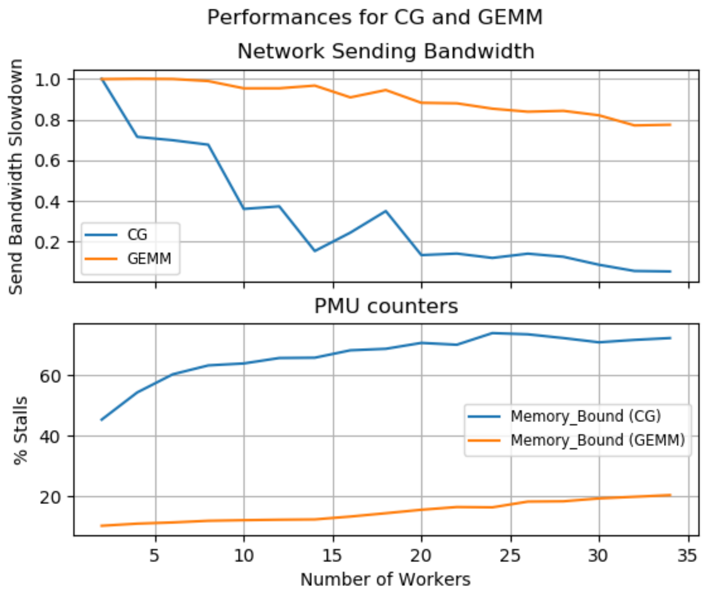
Impact of worker polling

- Each StarPU worker gets the next task to execute from a list
  - To be reactive enough: active wait by **polling**
  - Wait for few `nops` between each peek
  - Number of `nops` defined by an **exponential backoff** algorithm
- One parameter for this algorithm:
  - High value: workers poll rarely
  - Low value: workers poll frequently
- **Frequently polling workers impact communication latency**



Use-cases: computational kernels

- Computational kernels:
  - > Dense conjugate gradient (CG)
  - > Dense general matrix-matrix multiplication (GEMM)
  - > On top of StarPU
- 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**



Conclusion

- Computations and communications in parallel to get better performances in distributed HPC applications
- Side-by-side computations and communications
  - > Can disturb computations
  - > Can **highly impact communications**
- Main factor of interferences: **memory contention**, influenced by placement, message size, arithmetic intensity, runtime system overhead
- Behaviours also observed with real-world computational kernels
- Future work:
  - > Model these interactions
  - > Take into account these interactions in runtime systems to minimize them
  - > Same study with GPUs

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).

Inria