Interactions between communications and computations within task-based runtime systems

Solharis / HPC-scalable-ecosystem days

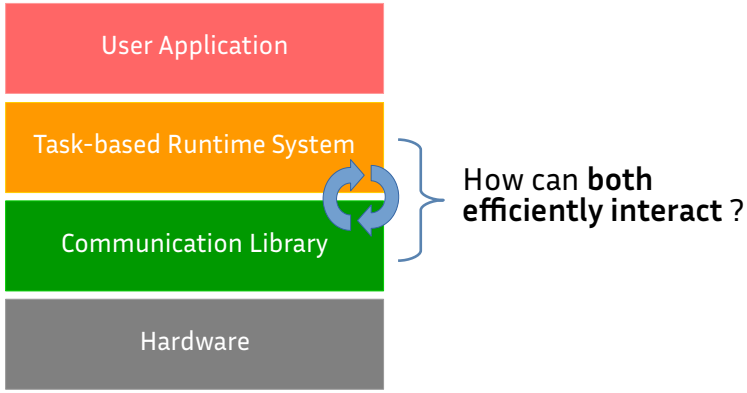
Philippe SWARTVAGHER
TADaaM

Introduction

- PhD student in the TADaaM team
- Supervised by Alexandre DENIS and Emmanuel JEANNOT
- Thesis topic: *interactions between task-based runtime systems and communication library*

  >To reach a better scalability on many nodes

- Collaboration with the STORM team about StarPU



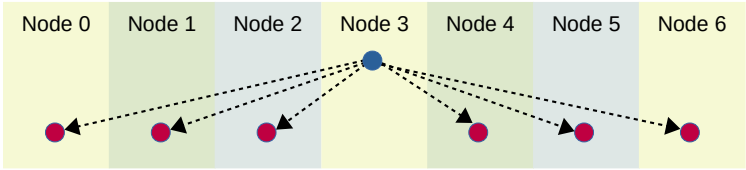
Agenda

- Dynamic broadcasts published ✓
- Performance analysis
- Interaction between computations and communications published soon... hopefully !

Efficient broadcasts

- In asynchronous dynamic task-based runtime system:
  - A data owned by a node can be needed on several other nodes: a **broadcast**
  - Only the sender node knows all recipients
  - Recipient node ignores if received data is part of a broadcast
  - → `MPI_Bcast` not usable in this case

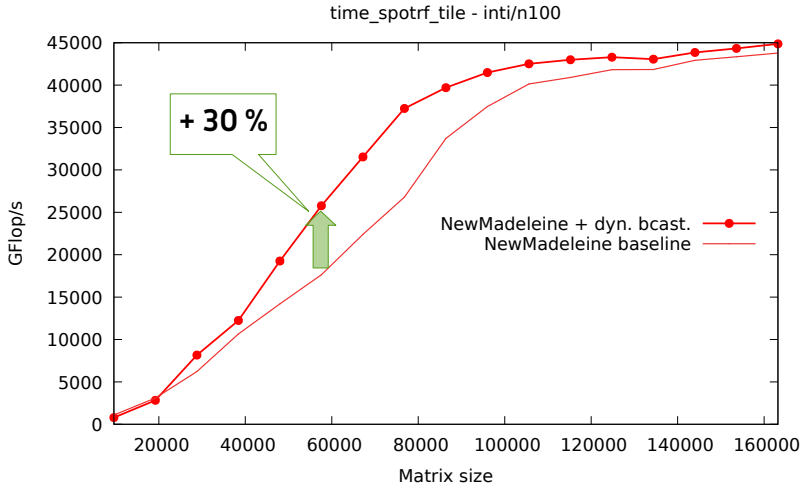
- We developed **dynamic broadcasts**:
  - Use efficient broadcast algorithms
  - Routing informations are stored in message header



A broadcast pattern in the DAG

Efficient broadcasts: results

- We developed **dynamic broadcasts**:
  - > Impove performances up to 30% in Cholesky decomposition according to our benchmarks
  - > Leads to better scalability
  - > Article published at Euro-Par 2020
- Benchmarks:
  - > With Chameleon
  - > With qr_mumps: collaboration in progress with Antoine JEGO
  - > ... your application ?

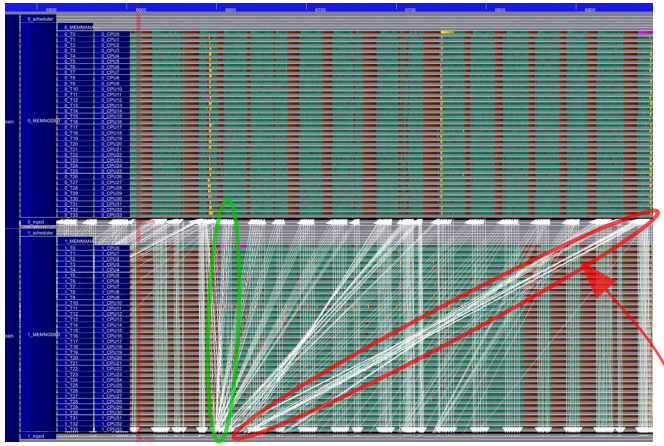


Performance gain of tiled Cholesky decomposition with our *dynamic broadcasts* on 100 nodes (1600 cores).

Paper : Denis A., Jeannot E., Swartvagher P., Thibault S. (2020) **Using Dynamic Broadcasts to Improve Task-Based Runtime Performances.** In: Malawski M., Rzadca K. (eds) Euro-Par 2020: Parallel Processing. Euro-Par 2020. Lecture Notes in Computer Science, vol 12247. Springer, Cham.

Presentation: https://youtu.be/hyC6oUEzD3k

Performance analysis

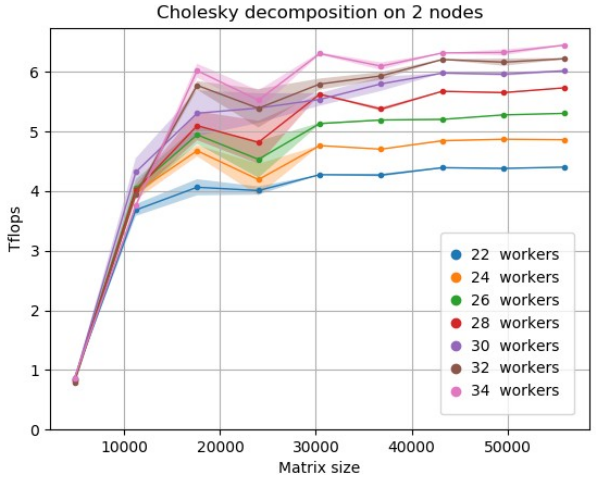


Trace of a Cholesky decomposition on 2 nodes:
some communications are longer than other

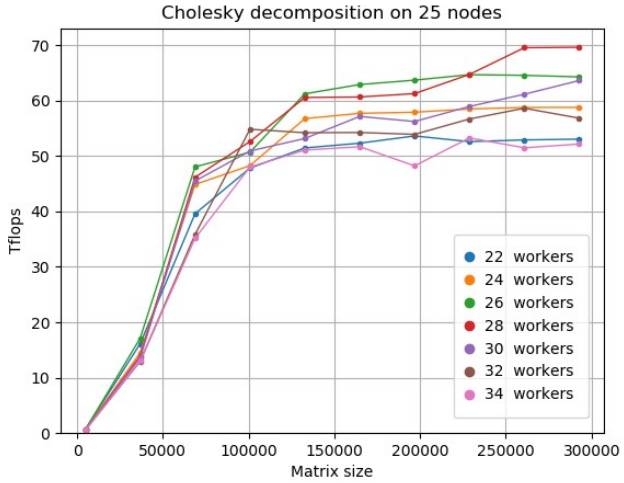
- Still some open questions:
  - >30% performance improvement:
    is it the theoretical expected gain ?
  - >No performance improvement for big matrices: is it normal ?
- Different performance improvements according to cluster characteristics:
  - >**Which characteristic is important ?**
- Traces showed that, even on two nodes, during task execution, **some communications can be very looong**

Worker scalability

- Workers: threads executing tasks (one worker per core)
- The number of workers can be changed before starting the benchmark



On 2 nodes: worker scalability: ✓

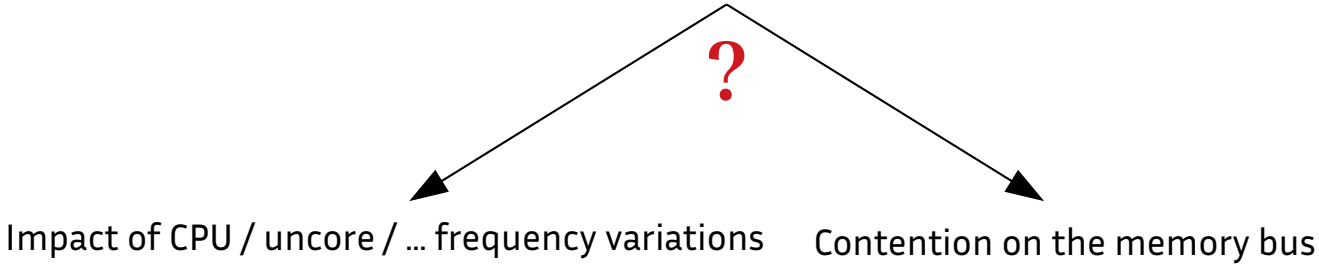


On 25 nodes: worker scalability: ✗

→ **Wrong worker scalability caused by communications ?**

Competition between communications and computations ?

- During task execution, some communications can be very long
- Using a lot of workers seems to degrade communication performances

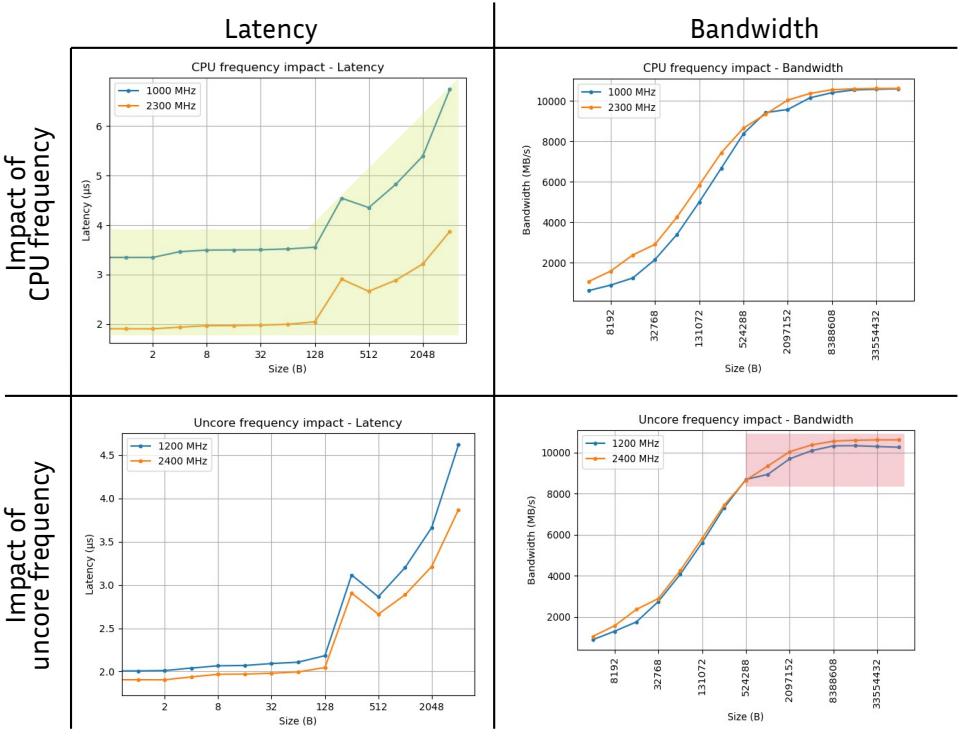


Frequency impact on communications

- Ping-pongs to measure communication performances
- Variation of CPU and uncore frequencies to observe their impact

- CPU frequency has an impact on network latency
- Uncore frequency has an impact on network bandwidth

- AVX instructions for computations:
  - > Force the CPU to reduce its frequency if several cores use them
  - > Has an impact on communication latency ?
  - > → No



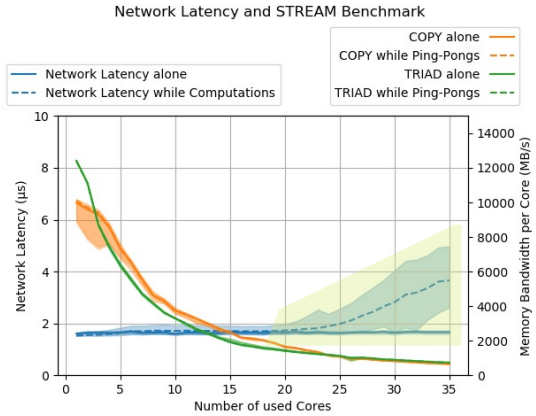
Network latency and bandwidth obtained with ping-pong benchmarks

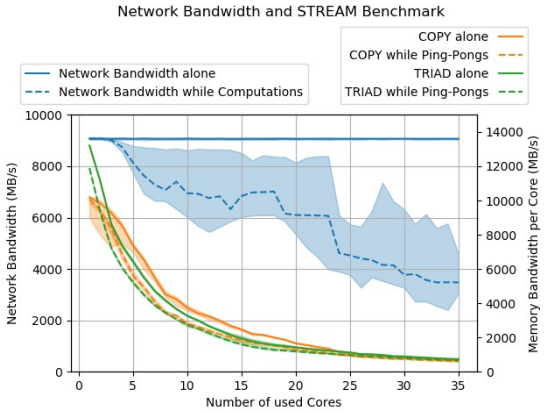
Memory contention

- Contention on the memory bus, caused by:
  - > Data moving between NIC and memory, for communications
  - > Data moving between cores and memory, for computations
- Benchmark:
  - > Computing: STREAM: memory-bound program to get the worst case

    `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]`
  - > Ping-pongs to measure network latency and bandwidth
  - > Independently to get expected performance (continuous curves), then simultaneously to get the impact (dashed curves)
  - > According to the number of cores executing STREAM
- → Impact on network latency
- → Impact on network bandwidth and STREAM performance

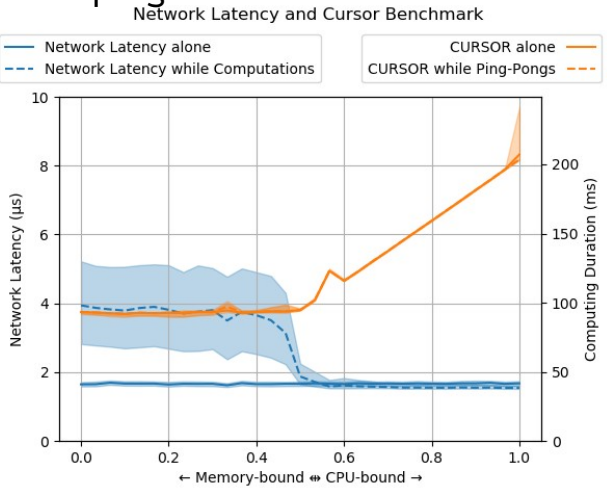


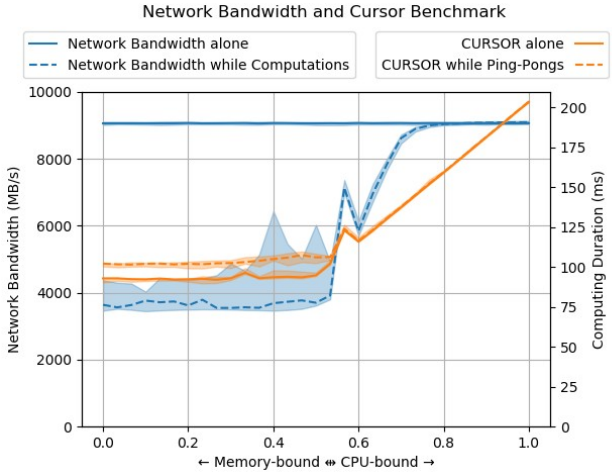


Between memory-bound and CPU-bound

- Repeat the arithmetic operation on one array item several time, before skipping to next array item
  - > Repeat one time: memory-bound program
  - > Repeat many times: CPU-bound program

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





→ **Interaction between communications and computations disappears when computations are more CPU-bound**

Conclusion

- These interactions between communications and computations are present:
  - > With StarPU (and the runtime system adds other interactions !)
  - > With different MPI libraries
  - > On different clusters
- Current goal: highlight the problem an article is written
- Future goal: implement some mechanisms in StarPU to avoid this problem your ideas are welcome !

- Your application uses StarPU and depends on communication performances ? Come talk with me !