WORKSHEET 3: MPI POINT-TO-POINT AND ONE-SIDED COMMUNICATION

Programming of Supercomputers [IN2190]

Group 09: Gerasimos Chourdakis, Nathaniel Knapp, Walter Simson

December 23, 2015

Fakultät für Informatik - Technische Universität München

AGENDA

TASK 3: BASE CASE

CANNON'S ALGORITHM

- Parallel algorithm for multiplying 2D matrices.
- Suiable for $N \times N$ grids of processes.
- The input matrices **A** and **B** are divided to blocks and distributed to the processes.
- In each step, a partial result is computed.
- The **A**-blocks are rotated/cycled in rows of the grid of processes and the **B**-blocks in columns.
- In a grid of $N \times N$ processes, N steps needed.
- Only read dependencies in **A** and **B**.

PROVIDED IMPLEMENTATION

- MPI-activated, P2P blocking communication.
- Communicators `cartesian_grid_communicator`, `row_communicator`, `column_communicator`.
- I/O handled by rank 0. (big overhead)
- Rank 0 uses blocking Send/Recv to distribute and collect data. (big overhead)
- Main loop (line 190):
  `for(cannon_block_cycle = 0; ...){...}`
- Computations (line 193):
  `for(C_index = 0, A_row = 0; ...){...}`

SYSTEM DESCRIPTION

We are working on SuperMUC Phase I and SuperMUC Phase II. Notable differences:

Property	Phase I - Sb	Phase II - Hw
Architecture	Sandybridge	Haswell
Cores per node	16	28 (we use only 16)
Nominal frequency	2.7 GHz	2.6 GHz
Memory bandwidth	102.4 GB/s	137 GB/s
L3 cache	2×20 MB	4×18 MB
L3 cache latency	30 cycles	36 cycles
Network	Infiniband FDR10	Infiniband FDR14

Both supercomputers use the same intra-island network topology (non-blocking tree).

JOBSCRIPT AND POST-PROCESSING

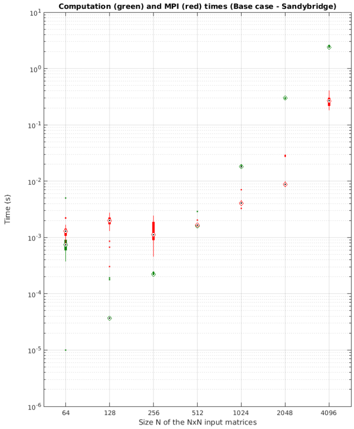
Jobscript:

- Repetitive execution of all the tests in the same job (for-loop) to deal with the high variance. Here results using 30 iterations.
- The same jobscript for every case, easy edit with variables.
- On-demand execution of the consistency checks.
- Extra output for verification and history.

After job execution, we use a bash script to extract the time data and we plot on Matlab.

We use boxplots and medians (instead of averages) to deal with outliers.

SCALABILITY (BASE CASE)



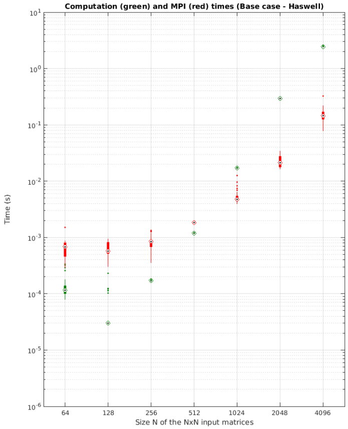
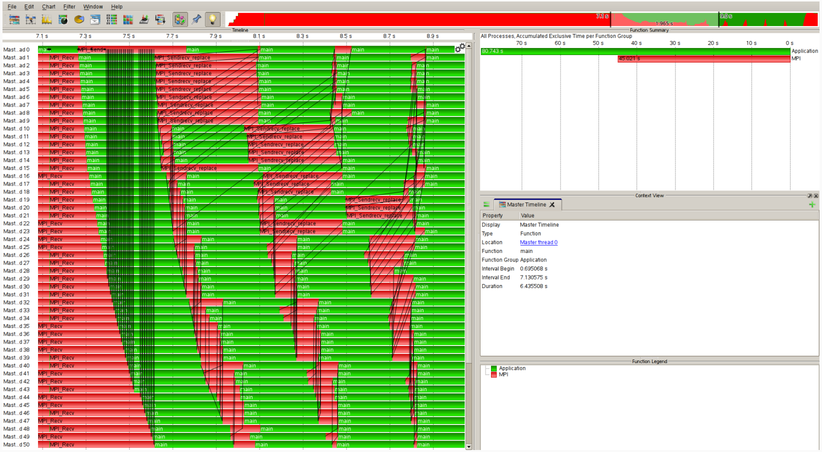


Figure: Compute time (green) and communication time (red) of Rank0 for Sandybridge (left) and Haswell (right). The points are medians. 30 iterations.

SCALABILITY (BASE CASE)

- N=512 divides each plot to a communication-bounded area (left) and a computation-bounded (right).
- Higher variance in smaller sizes.
- Almost constant communication time for N < 512.
- Computation time increases logarithmically. Higher for N=64.
- Haswell performs a bit better in communication. Faster network, higher memory bandwidth, bigger caches. Compiled only with the default optimization (no flags).
- Non-blocking tree network topology - maybe we could see some extra variance because of different allocation of the nodes if we had data from multiple job submissions. We can still see clear tendencies with our approach though.

TRACING WITH SCORE-P AND VAMPIR (BASE CASE)


File Edit Chart Filter Window Help
Timeline
Function Summary
7.1 s 7.3 s 7.5 s 7.7 s 7.9 s 8.1 s 8.3 s 8.5 s 8.7 s 8.9 s
Mast...ad 0 … Mast...d 50
main
MPI_Recv
MPI_Send
MPI_Sendrecv_replace
All Processes, Accumulated Exclusive Time per Function Group
70 s 60 s 50 s 40 s 30 s 20 s 10 s 0 s
80.743 s
45.021 s
Application
MPI
Context View
Master Timeline
Function Legend

Property	Value
Display	Master Timeline
Type	Function
Location	Master thread 0
Function	main
Function Group	Application
Interval Begin	0.695068 s
Interval End	7.130575 s
Duration	6.435508 s

🟩 Application
🟥 MPI

Figure: Tracing for the base case, N=4096, Haswell. Note the peaks of communication activity and the synchronization between senders-receivers.

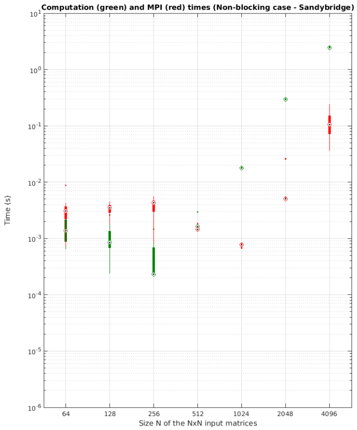
TASK 4: NON-BLOCKING P2P

APPROACH

We aim to overlap communication with computation in the main loop. For this, we drop the `MPI_Sendrecv()` calls and we use a combination of `MPI_Isend()`, `MPI_IRecv()`, `MPI_Test()` and `MPI_Wait()` calls instead.

- Goal: Post Send and Receive requests as early as possible. We test the previous communication for completion (buffer re-usage). We wait only if absolutely necessary.
- We use another pair of pointers as receive buffers. As soon as we receive, we copy their contents to the local blocks and re-post receive requests immediately.
- We try to receive whichever of the A, B blocks is first available.
- Several attempts to send:
  - As soon as the buffer copy is complete.
  - In every iteration of the outer loop of the actual computations.
  - If everything fails, wait and send after the loop.
- First and last cycles peeled-off in order to better manage requests and minimize branches.

SCALABILITY (NONBLOCKING P2P)



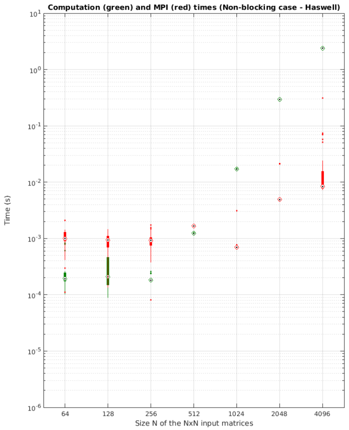
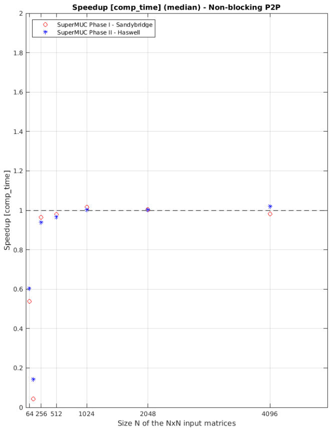


Figure: Compute time (green) and communication time (red) of Rank0 for Sandybridge (left) and Haswell (right). The points are medians. 30 iterations.

SCALABILITY (NONBLOCKING P2P)



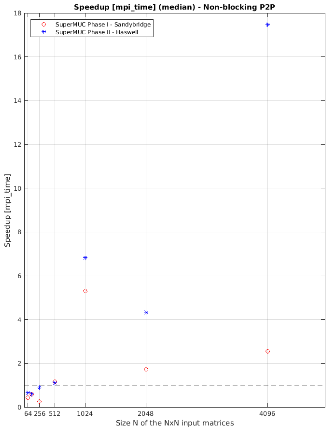


Figure: Speedup with respect to compute time (left) and communication time (right) of Rank0 for Sandybridge (circles) and Haswell (stars). Speedup computed using the medians of 30 iterations.

scalability (nonblocking p2p)

- Critical point still at N=512.
- This communication scheme helps much for larger N but gives worse results at small N.
- Similar general behavior in large N as in base case. A bit different in small.
- Computation time increases logarithmically. Higher for N=64 and N=128.
- This scheme favors the faster network of SuperMUC Phase II.

POTENTIAL TO OVERLAP

In the base case (e.g. for N=4096), the region of interest is from approximately 8s to 11s. In this, the accumulated exclusive time spent in the application is 166.6s and in the MPI calls 28.7s (including some communication out of the loop).

Using these data, we estimate that communication in the loop takes at most 14% of the accumulated time. In a perfect overlap, i.e. with zero communication time, we could gain up to 14% in this region.

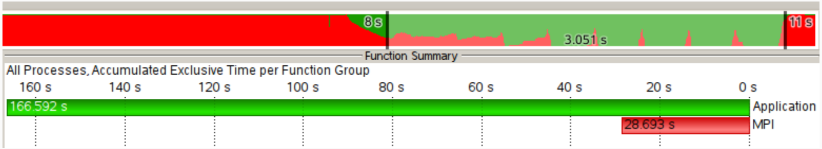


Figure: Application and communication time in the region of interest - base case, N=4096, Haswell.

TRACING WITH SCORE-P AND VAMPIR (NONBLOCKING P2P)

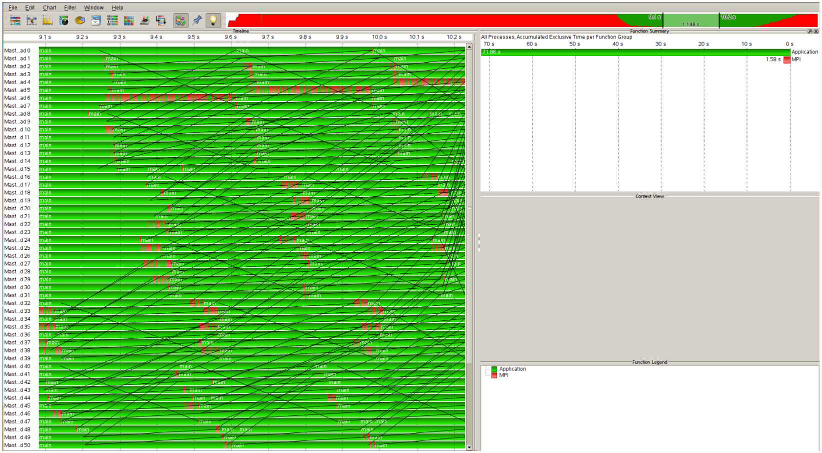


Figure: Tracing for the nonblocking P2P case, N=4096, Haswell. Much better communication-computation overlap is achieved.

OVERLAP ACHIEVED (NONBLOCKING P2P)

In the non-blocking P2P case we see that, in an inner part of the the region of interest, the communication time corresponds only to approximately to 2% of the total accumulated time. In other words, the 98% of the time is used for useful work.

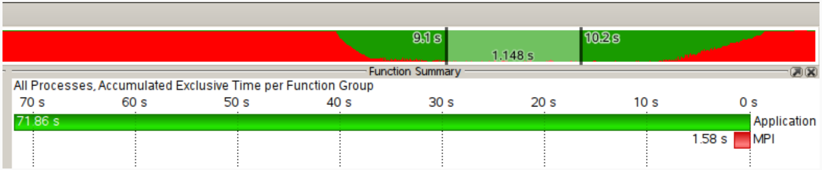


Figure: Application and communication time in the region of interest - nonblocking P2P case, N=4096, Haswell.

TASK 5: ONE-SIDED COMMUNICATION

APPROACH (1)

Our initial (and simpler) approach was:

- Create one window for each of A,B that point to the (new) pointers `A_shared_block` and `B_shared_block`. They are used in the same way as the receive buffers in the nonblocking case.
- Synchronize the windows with `MPI_Win_fence()`.
- Copy the shared blocks to the local blocks.
- Synchronize the windows again.
- Put the local blocks to the shared blocks of the neighbors using `MPI_Put()`.
- Perform the actual computations with the local blocks.

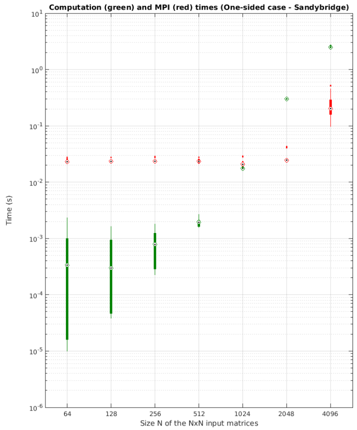
APPROACH (2)

Trying to improve the overlap, we extended our initial approach to use two windows, alternately. The idea is to fence the individual windows less frequently, in order to allow more time for communication completion.

For this, we create two windows for each matrix, that each corresponds to another pointer. After filling both pointers, the second one is always one step ahead than the first.

Again, we peel-off the first and last cycles, in order to better balance the communication operations without adding even more branches. Inside the loop, we use the loop index to swap between the windows.

SCALABILITY (ONE-SIDED)



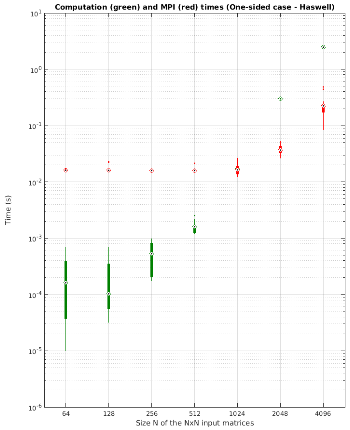
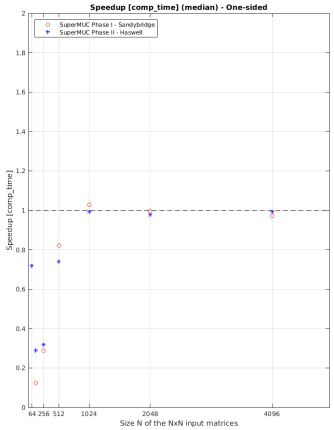


Figure: Compute time (green) and communication time (red) of Rank0 for Sandybridge (left) and Haswell (right). The points are medians. 30 iterations.

SCALABILITY (ONE-SIDED)



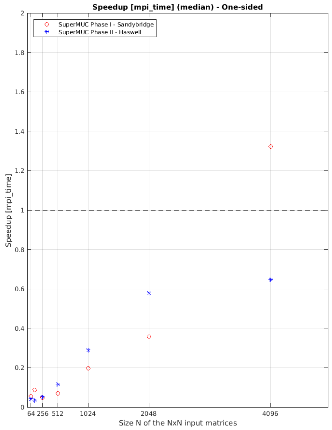


Figure: Speedup with respect to compute time (left) and communication time (right) of Rank0 for Sandybridge (circles) and Haswell (stars). Speedup computed using the medians of 30 iterations.

SCALABILITY (ONE-SIDED)

- Worse results than both the base and the non-blocking case for these problem sizes.
- Still, the two-window method performed a bit better than our initial one-window approach.
- Significantly longer communication time, almost constant and very reproducible.
- Critical point moved at N=1024.
- Much worse for small N, unkown behavior above N=4096.
- Large variance in computation time below N=1024. Maybe energy management? Again reproducible in the compute-bounded area.
- In the most cases, Haswell performs better.

TRACING WITH SCORE-P AND VAMPIR (ONE-SIDED)

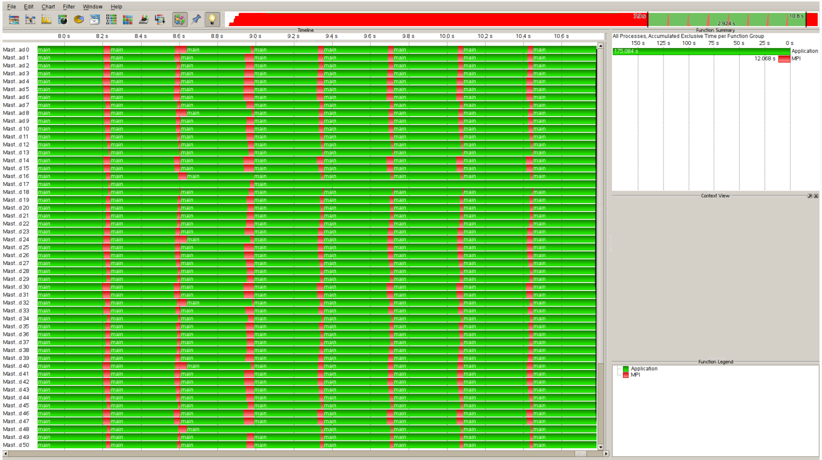

File Edit Chart Filter Window Help
7.9 s
2.924 s
10.8 s
Timeline
8.0 s 8.2 s 8.4 s 8.6 s 8.8 s 9.0 s 9.2 s 9.4 s 9.6 s 9.8 s 10.0 s 10.2 s 10.4 s 10.6 s
Mast...ad 0 … Mast...d 50
main
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
150 s 125 s 100 s 75 s 50 s 25 s 0 s
175.084 s
Application
12.068 s
MPI
Context View
Function Legend
🟩 Application
🟥 MPI

Figure: Tracing for the one-sided case, N=4096, Haswell.

OVERLAP ACHIEVED (ONE-SIDED)

In the one-sided case we observe that, it the region of interest (that is here easier to define), the communication time corresponds to the 6% of the overall accumulated time. Not as good as in the non-blocking case, but better than the base case.

Please note that in our implementation we include in the `mpi_time` also non-mpi regions that we added because of our approach to the communication scheme.

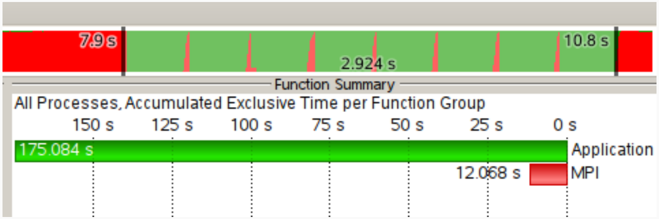


Figure: Application and communication time in the region of interest - nonblocking P2P case, N=4096, Haswell.

CONCLUSIONS

CONCLUSIONS

- There is some optimization potential for the provided Cannon's algorithm implementation.
- The application is communication-bounded under N=512 or N=1024 and compute-bounded above.
- Non-blocking P2P communication achieves good overlap and helps more in the case of SuperMUC Phase II.
- Our one-sided communication did not turn to improve the performance according neither to the non-blocking or the base case. But there seems to be a tendency to help in bigger problems.
- SuperMUC Phase II seems to perform generally better for this application.
- Tracing proved to be very useful during our investigations.

FUTURE WORK

FUTURE WORK

To improve the performance of this application more, one could try:

- Implement a parallel input scheme, to reduce the initial overhead.
- Change the blocking communication between Rank0 and the rest of the processes with non-blocking P2P or collectives.
- Investigate the scaling up to bigger matrix sizes, especially for the one-sided communication.
- Investigate the effect of compiler optimizations.