Assignment 3
MPI Point-to-Point and One-Sided Communication

Smith Agarwal
Shyam Arumugaswamy
Siddhesh Kandarkar

December 20, 2018

3 Setting a Baseline

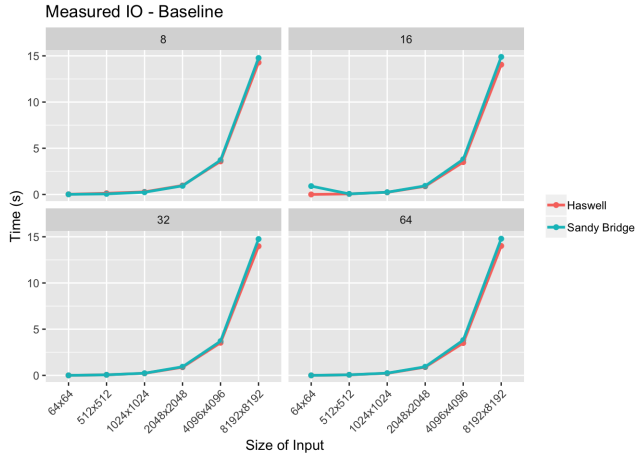
3.1 Required submission files

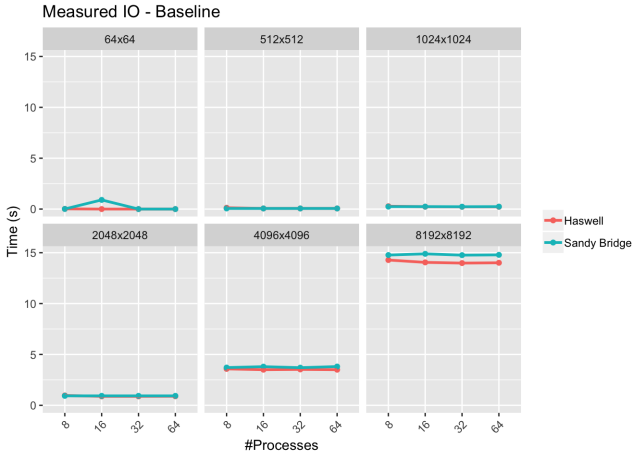
1. **The updated Load-Leveler batch script.**

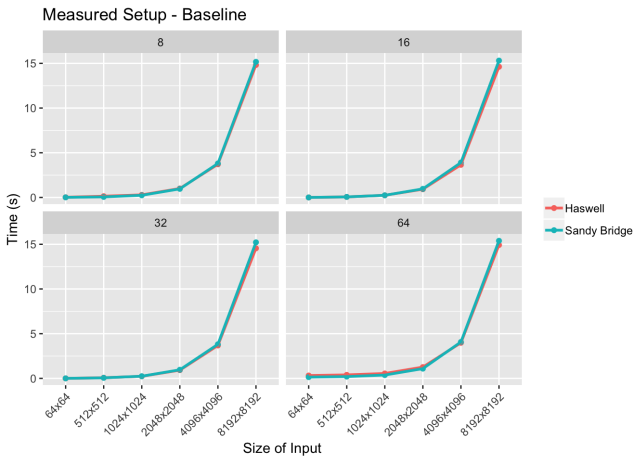
   Refer job_updated.ll file from code scripts of Task 3.1

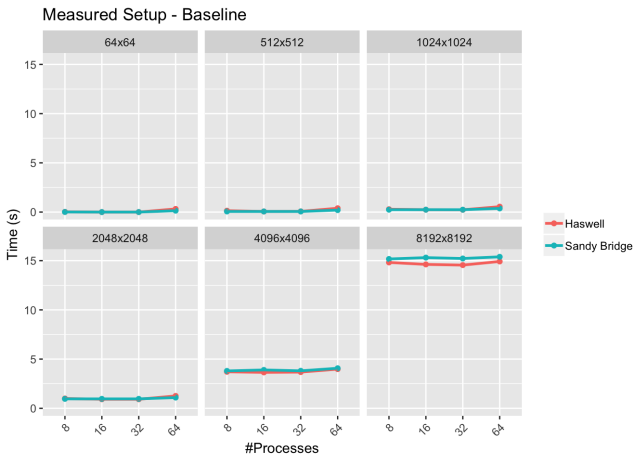
2. **The performance plots and description in the report.**

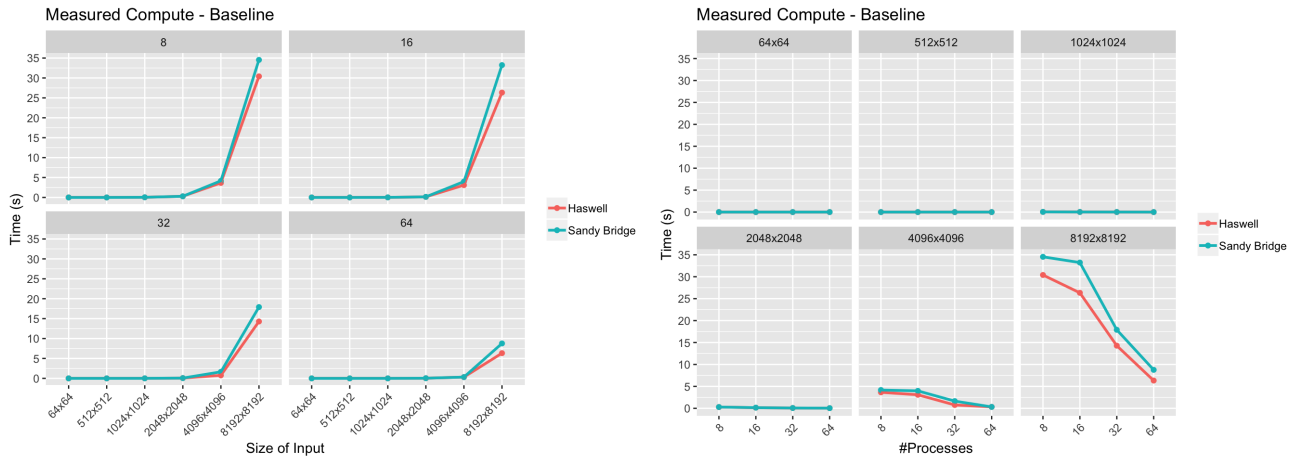
   The various performance plots for the baseline are:

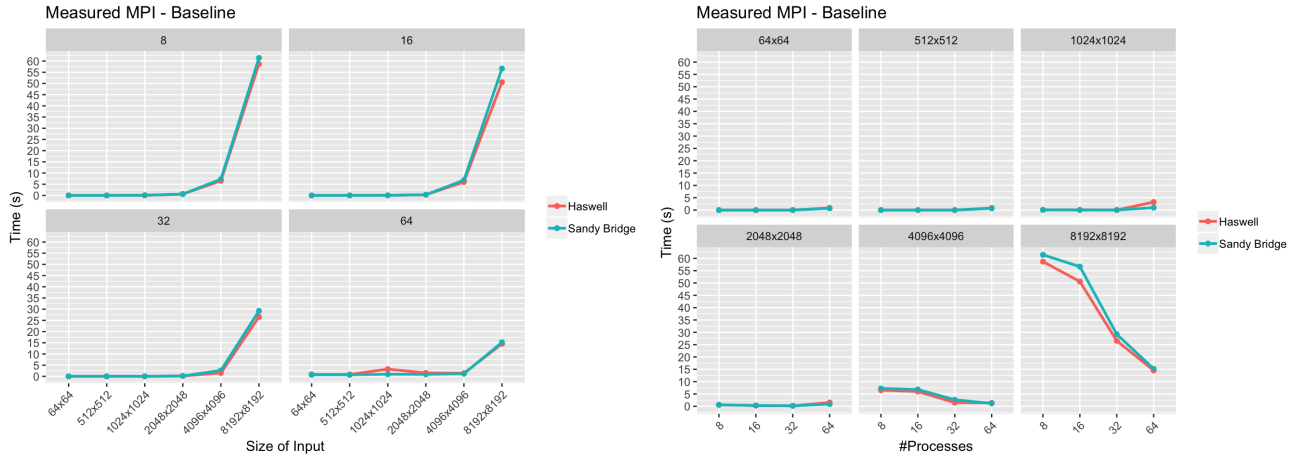


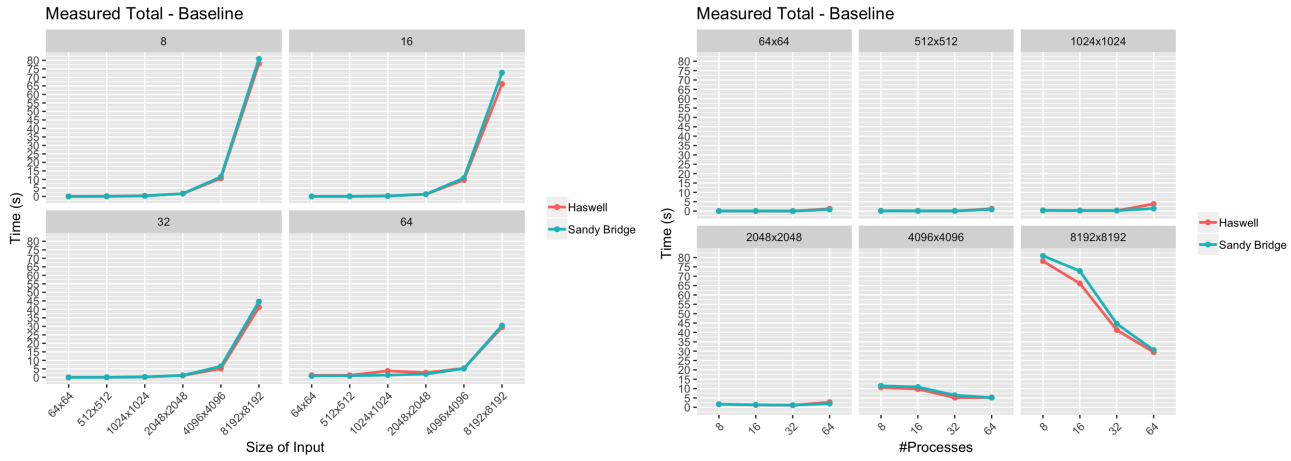












3.2 Questions

1. **Briefly describe the Gaussian Elimination and the provided implementation.**

   [https://en.wikipedia.org/wiki/Gaussian_elimination]
   Gaussian elimination (also known as row reduction) is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. There are three steps of elementary row transformation:

   (a) Swapping two rows

   (b) Multiplying a row by a nonzero number

   (c) Adding a multiple of one row to another row

Using these operations, a matrix can always be transformed into an upper triangular matrix. Our implementation divides the matrix into rows and then provides a set of them to each process. Each row performs the various operations mentioned above and sets the corresponding column values to zero thus resulting in a upper triangular matrix.

2. **How is data distributed among the processes?**

   All the rows in the data should be divided equally among the processes in such a way that each of the process can take a max of row/processes number of rows of the matrix.

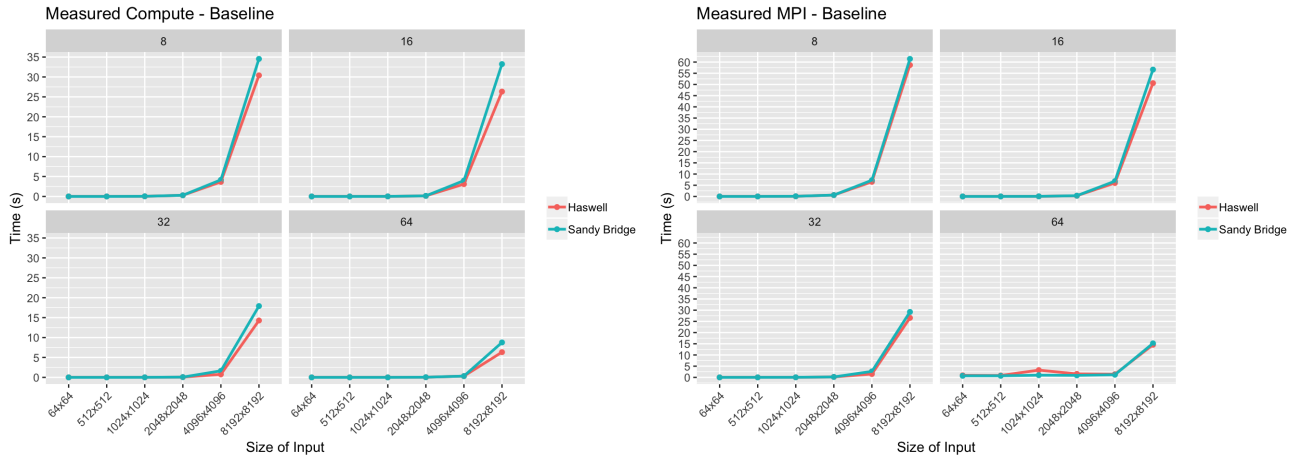
3. **Explain the changes applied to the provided Load-Leveler batch script.**

   To address the variance between the runs, we ran each of the instance multiple times. For each combination of processes and input file size, the job was executed 3 times to find an average value.

4. **What were the challenges in getting an accurate baseline time for Gaussian Elimination.**

   To get accurate baseline time for Gaussian Elimination, we had to ensure less waiting time for each of the measured times. The factors affecting the various measured times are explained as follows. IO communication takes place sequentially, so the number of processes does not affect the time. Setup time is the time required by each process to receive the rows of the matrix. From the performance plots, we observe that setup time increases with increase in domain input size. Also, even for high number of processes, the setup time is low for lower domain size. Setup time is relatively constant when processes is increased keeping domain size constant. This may probably be due to communication overhead. Compute time measures the time interval between the start and end of gaussian elimination. It also includes the MPI time. Compute time should be constant for a domain size even though number of processes increase. MPI time is communication time. MPI time increases with increase domain size and for 1024 domain size, it decreases with increase in processes. Total time is time required for entire Gaussian elimination algorithm to execute.

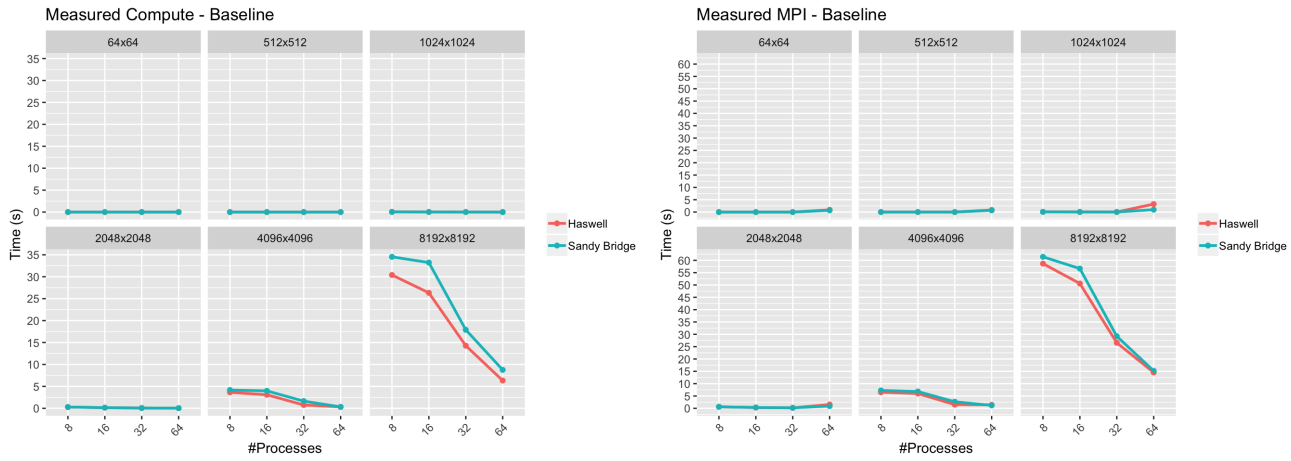
5. **Describe the compute and MPI times scalability with fixed process counts and varying size of input files for the Sandy Bridge and Haswell nodes. Did you observe any differences?**



With fixed process count and a varying size of input files, we observe that the compute time increases with input size. When comparing the performance of the Haswell and Sandy Bridge architectures, we observe that Haswell nodes generally have less compute times compared to Sandy Bridge when the domain size is between 2048 and 8192. However, the Sandy Bridge nodes have less compute times at small domain sizes.

At the same time, MPI time increases with domain size. We also see that Haswell nodes have shorter MPI times compared to Sandy Bridge nodes for when the domain size is above 4096. When the domain size is below 4096, Sandy Bridge nodes have shorter MPI times for all process count 64. Thus we observe that, the Haswell nodes perform slightly better.

6. **Describe the compute and MPI times scalability with fixed input sets and varying process counts for the Sandy Bridge and Haswell nodes. Did you observe any differences?**



With fixed input size and increasing the process count, we observe that the compute time decreases with process count.The compute time is completely negligible in comparison to the communication time for small input size of 64. At higher domain size, we see that the Haswell nodes has comparatively shorter compute times for any processor count. However, the Sandy Bridge nodes still seem to scale better with a larger number of processes at a large domain size.

For smaller domain size, MPI time is longer as number of processes increase. But for larger domain size, the MPI time decreases and we observe Haswell has comparatively less MPI time than Sandy Bridge as processes increase. Overall, Haswell has better performance architecture.

4 MPI Point-to-Point Communication

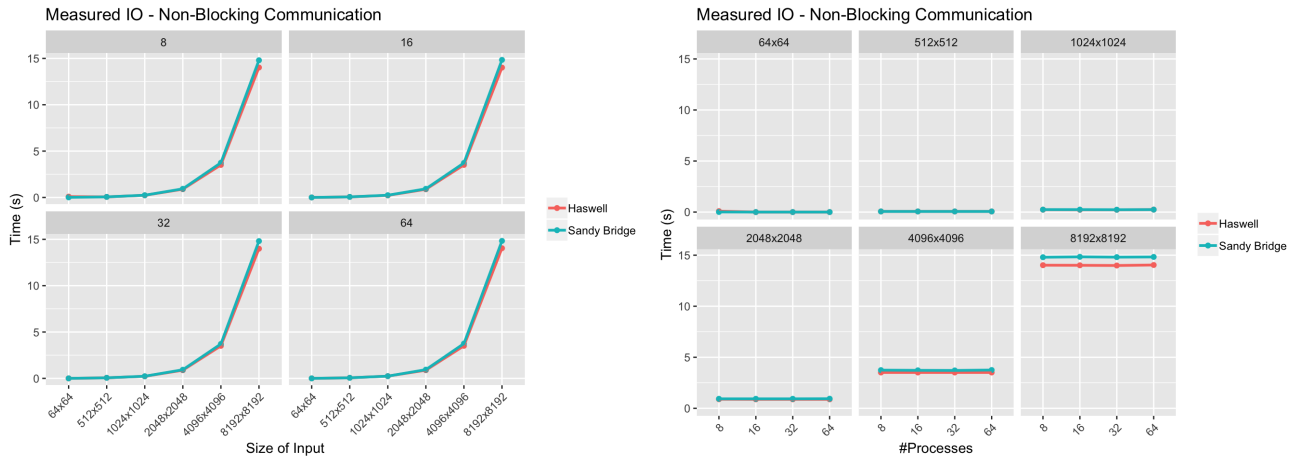
4.1 Required submission files

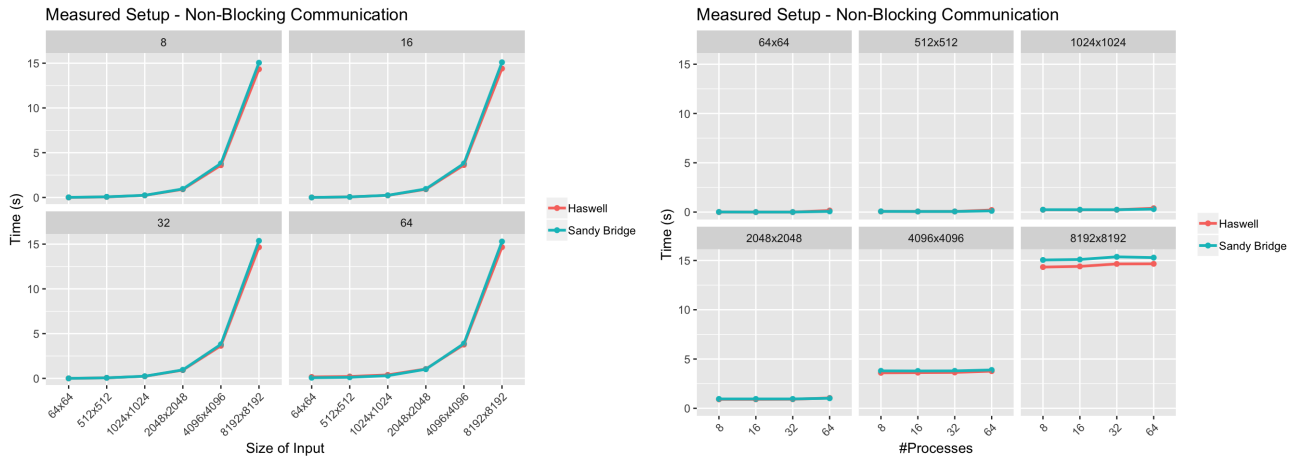
1. **The updated gauss.c file.**

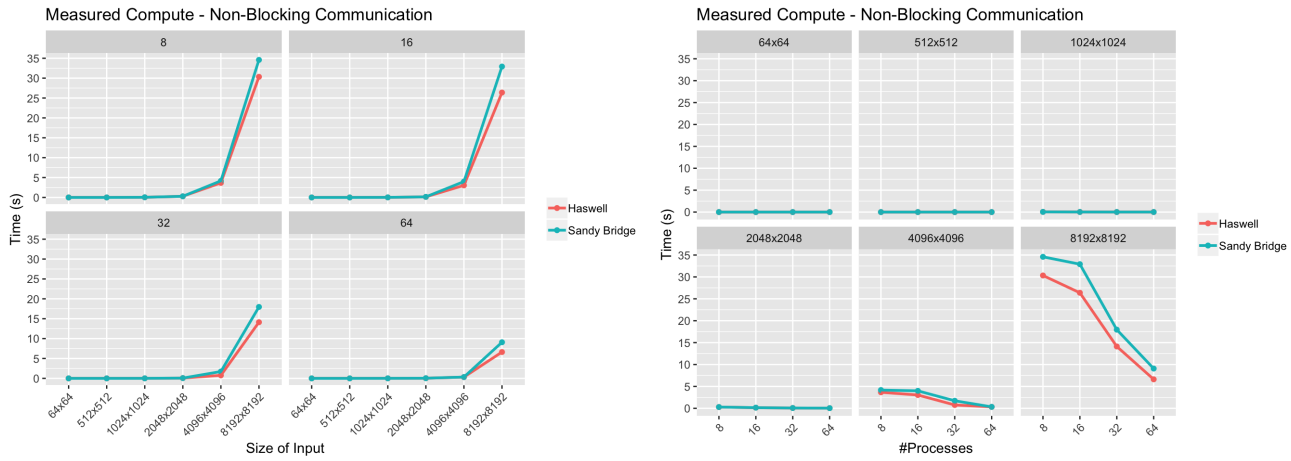
   Refer gauss_nonblock.c file from code scripts of Task 4.1

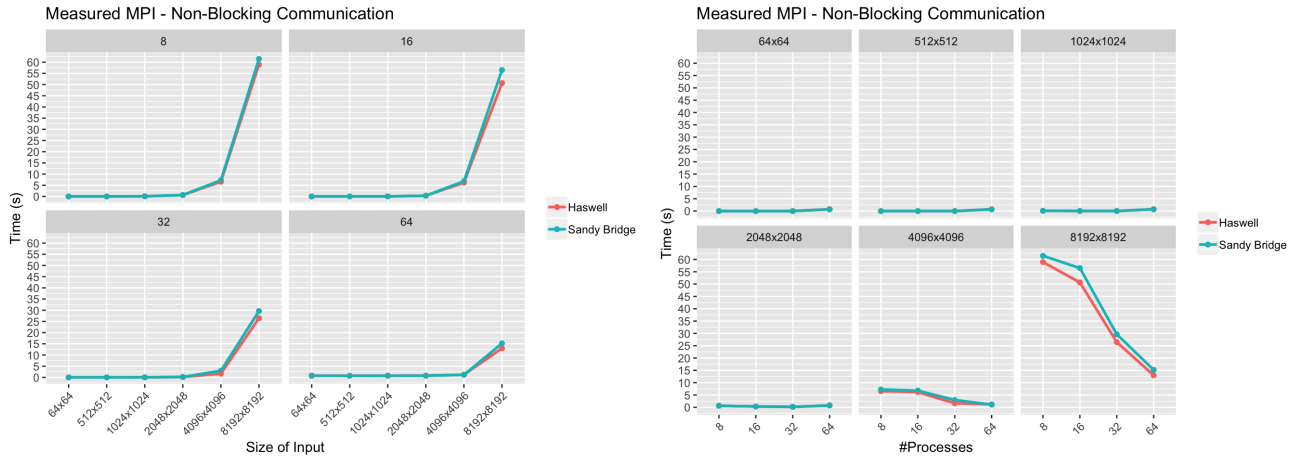
2. **The new performance plots and the description in the report.**

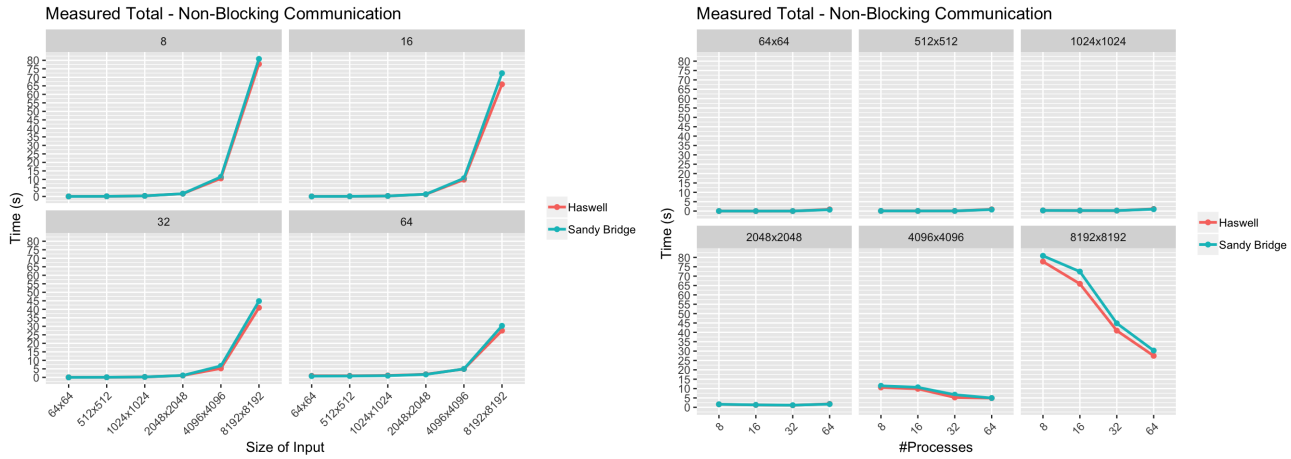
   The various performance plots for MPI Point-to-Point communication are:




Measured Setup - Non-Blocking Communication
8
16
32
64
Time (s)
Size of Input
64x64
512x512
1024x1024
2048x2048
4096x4096
8192x8192
#Processes
Haswell
Sandy Bridge


Measured Compute - Non-Blocking Communication
8
16
32
64
Time (s)
Size of Input
64x64
512x512
1024x1024
2048x2048
4096x4096
8192x8192
#Processes
Haswell
Sandy Bridge


Measured MPI - Non-Blocking Communication
8
16
32
64
Time (s)
Size of Input
64x64
512x512
1024x1024
2048x2048
4096x4096
8192x8192
#Processes
Haswell
Sandy Bridge


Measured Total - Non-Blocking Communication
8
16
32
64
Time (s)
Size of Input
64x64
512x512
1024x1024
2048x2048
4096x4096
8192x8192
#Processes
Haswell
Sandy Bridge

4.2 Questions

1. **Which non-blocking operations were used? Justify your choice.**

   The non-blocking operations selected were `MPI_Isend` and `MPI_Irecv`. They seemed to provide the most amount of overlap, though its less than expected. `MPI_Isend` was used to initiate the computation independent of whether data is received or not and thus reduce the waiting time. `MPI_Irecv` was used to try to anticipate the communication.

2. **Was communication and computation overlap achieved? Use Vampir.**

   Communication and computation overlap was achieved, but not as much wanted. This can be observed from the below vampir plots. Some overlap done thanks to Isend but no Irecv as communication was performed for following loop iteration while computing the current one.

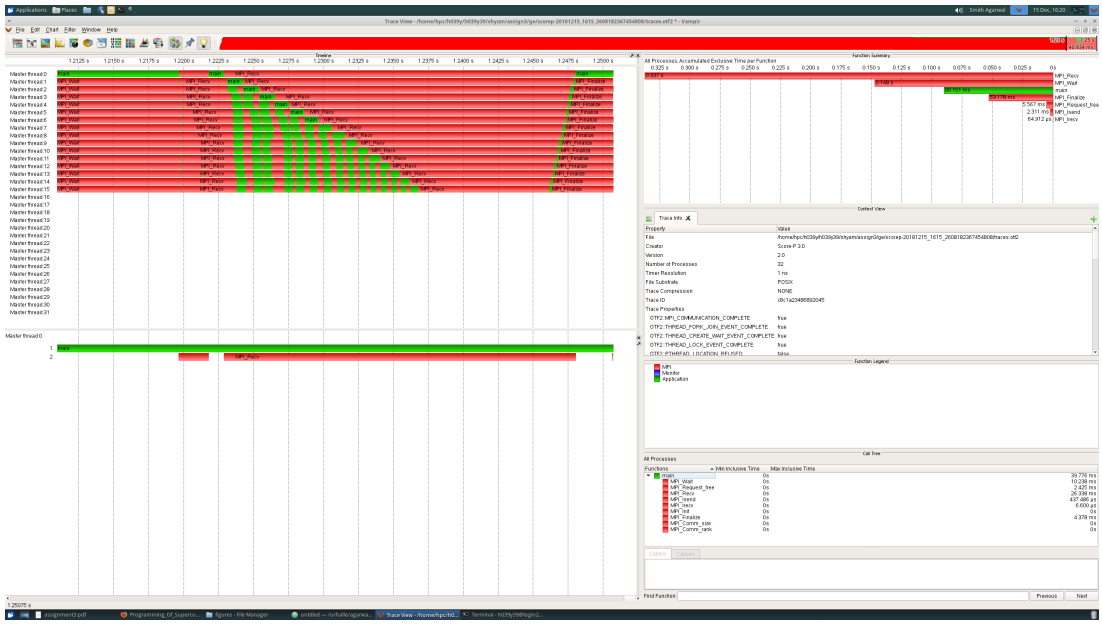


Figure : Vampir output for Point-to-point Haswell

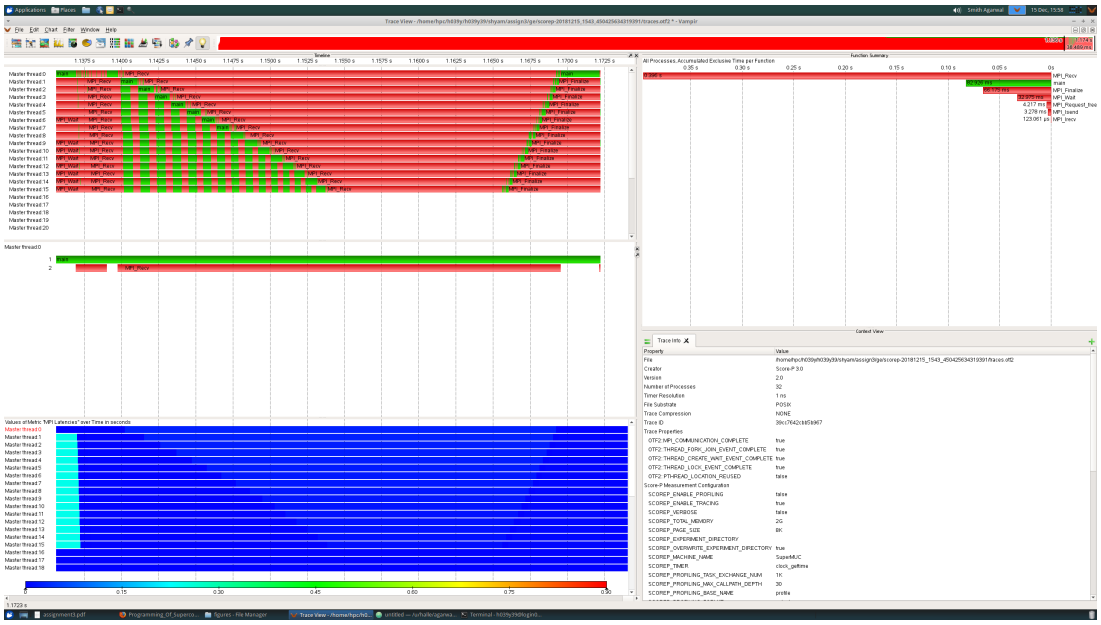
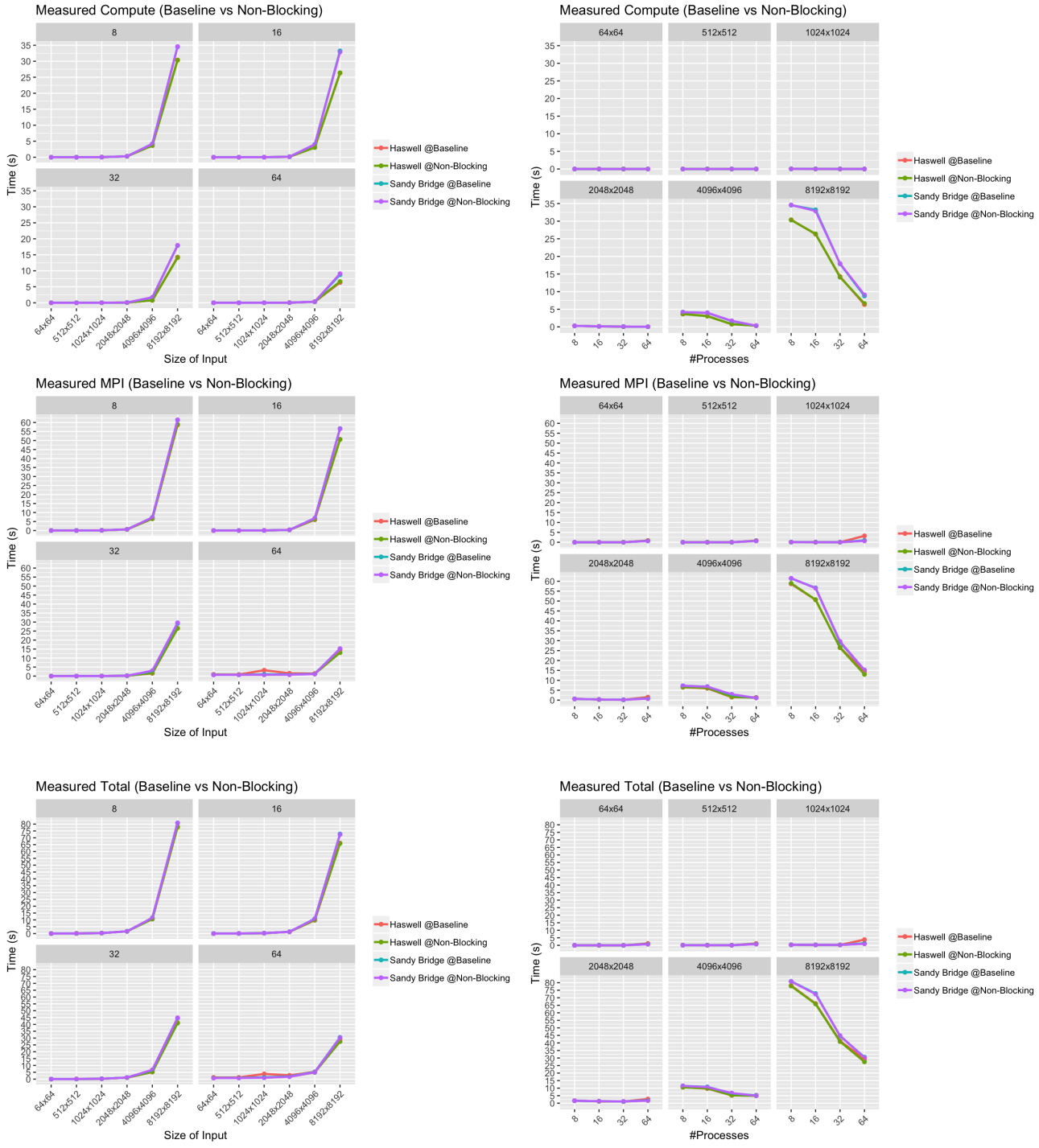


Figure : Vampir output for Point-to-point Sandy Bridge

3. **Was a speedup observed versus the baseline for the Sandy Bridge and Haswell nodes?**



Speedup is defined as `time(baseline)` / `time(non-blocking)` . From the plots, we observe than the non-blocking point-to-point communication performed slightly better than the baseline, however speedup was little due to lack of overlap of communication and computation. As we proceed with computation, more processes remain idle because each process requires the data computed by previous loop.

5 MPI One-Sided Communication

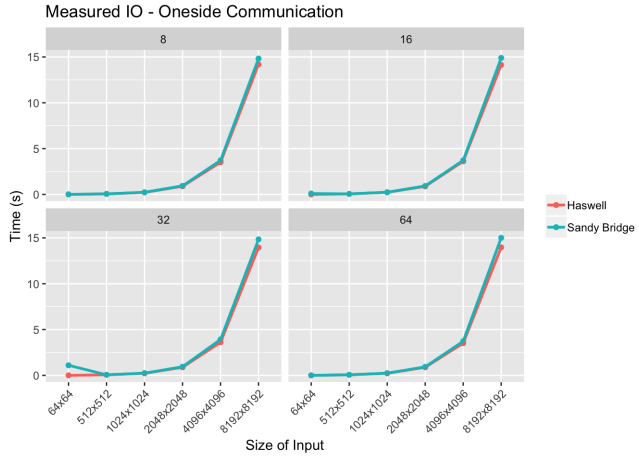
5.1 Required submission files

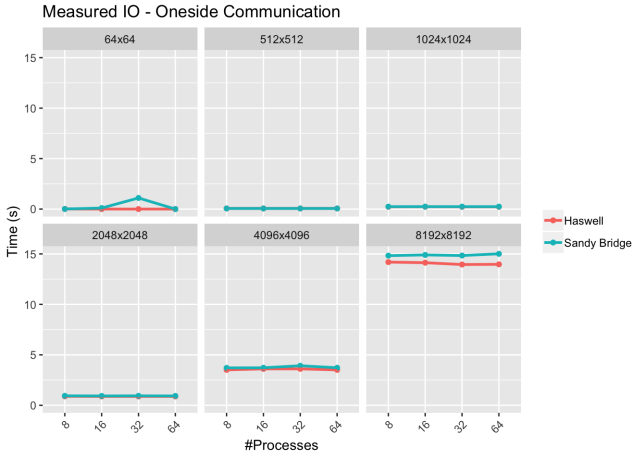
1. **The updated gauss.c file.**

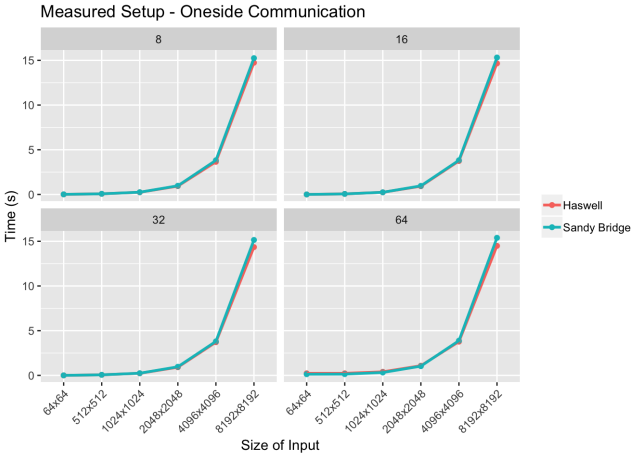
   Refer gauss_oneside.c file from code scripts of Task 5.1

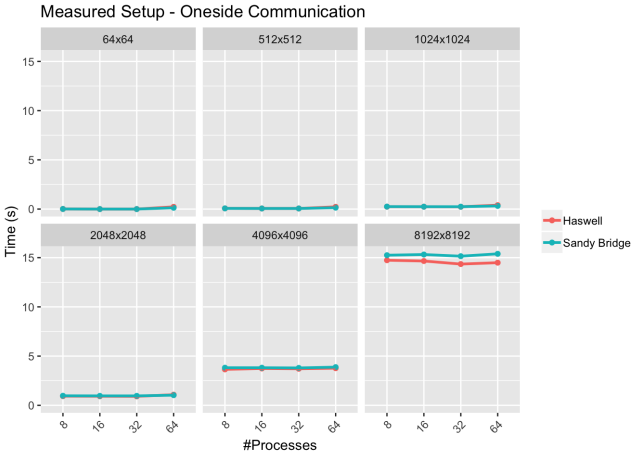
2. **The new performance plots and the description in the report.**

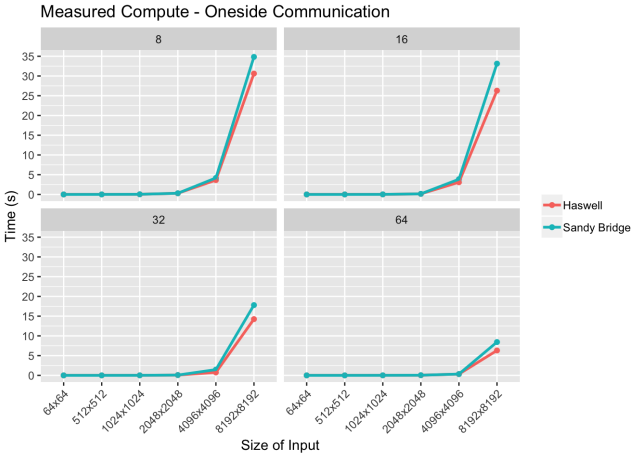
The various performance plots for MPI One-Sided communication are:

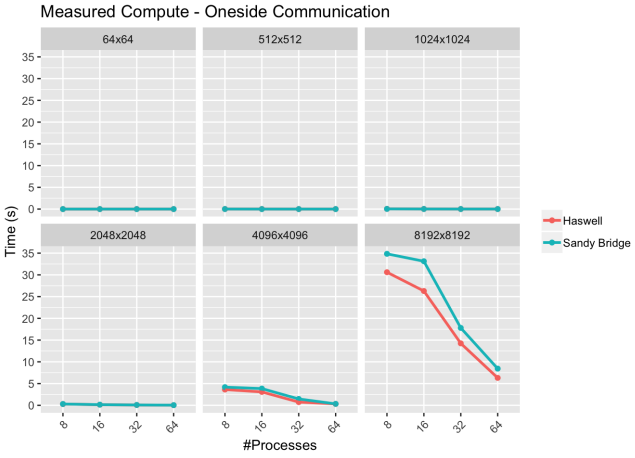


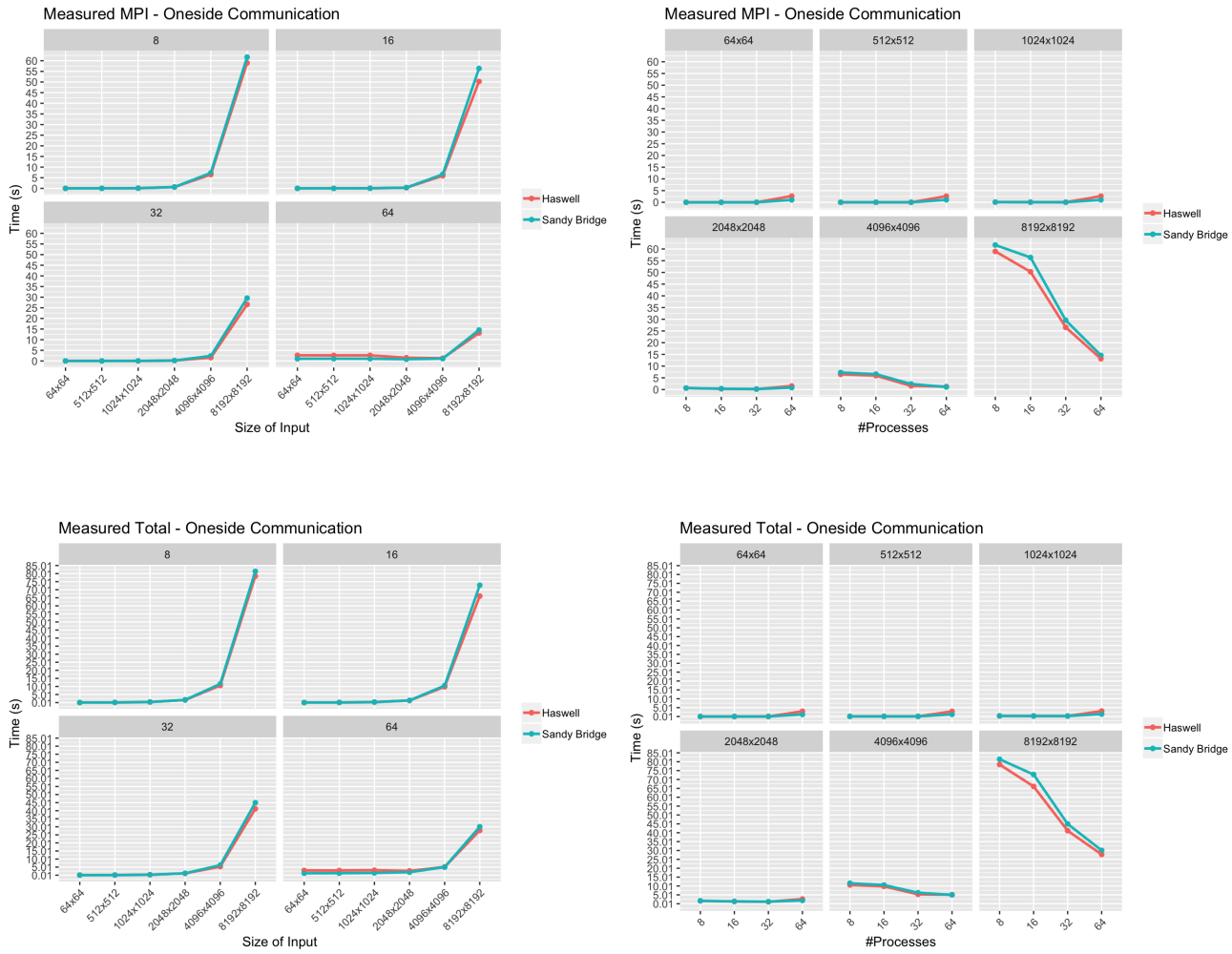












5.2 Questions

1. **Which one-sided operations were used? Justify your choice.**

   MPI two-sided communication might sometimes cause a delay in sending data, since the sender has to wait for the receiver to be ready to receive the data before it can send it.

   To overcome this drawback, the MPI 2 standard introduced Remote Memory Access RMA, also called 'One-Sided communication' because it requires only one process to transfer data. Basically, the idea of one-sided communication models is to decouple data movement with process synchronization.

   We have used Post-Start-Wait-Complete methodology and following one-sided operations were used :-

   (a) **MPI_Win_create :** it is called by all processes to create a window of shared memory, the window specifies all process memory which is available for remote operations. Each process then initializes its portion of the memory window allowing remote processes read/write access to the pre-defined memory.

   (b) **MPI_Win_allocate :** MPI allocates local memory and returns a pointer to it

   (c) **MPI_Win_start :** Starts an RMA access epoch for MPI

   (d) **MPI_Put :** writes data from local memory into a memory window on a remote process

   (e) **MPI_Get :** reads data from a memory window on a remote process into local memory

   (f) **MPI_Win_fence :** helps in collective synchronization of all processes.

   (g) **MPI_Win_free :** Terminates the memory window and deallocates window object.

   (h) **MPI_Finalize :** Cleans up all MPI states and MPI environment. The parallel code ends here.

2. **Was communication and computation overlap achieved? Use Vampir.**

Yes, communication and computation overlap was achieved and this can be observed from the below vampir plots.

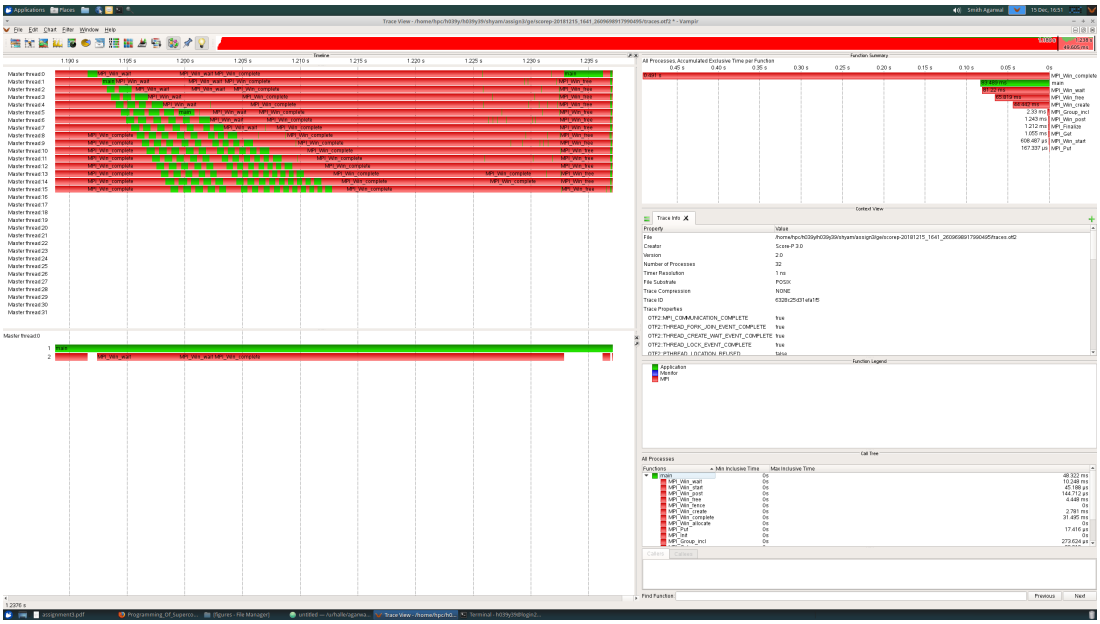


Figure : Vampir output for One-sided Haswell

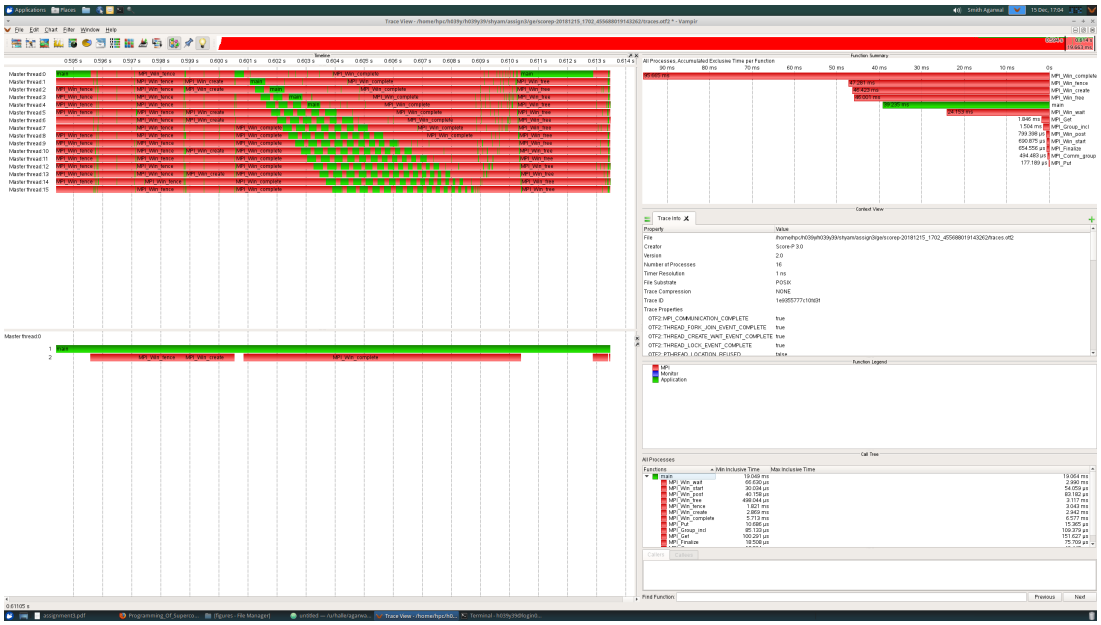
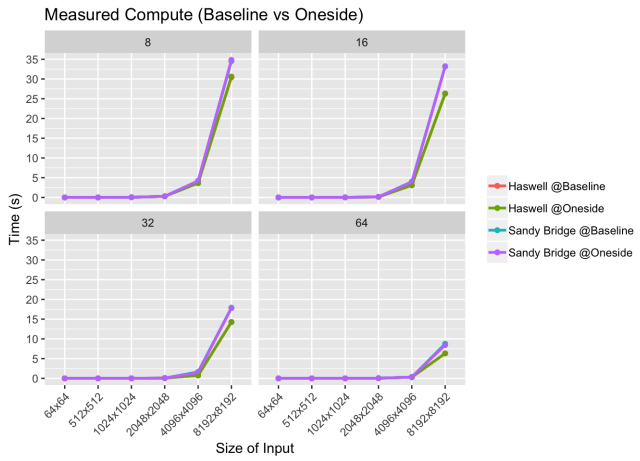
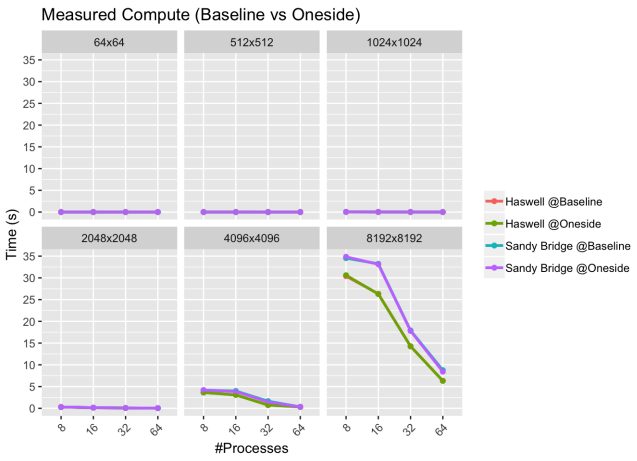


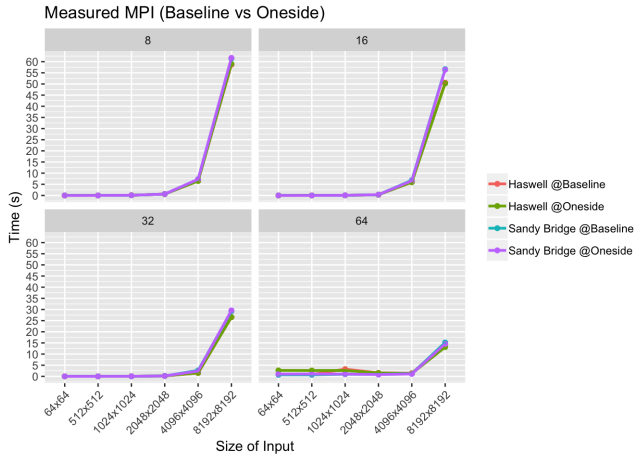
Figure : Vampir output for One-sided Sandy Bridge

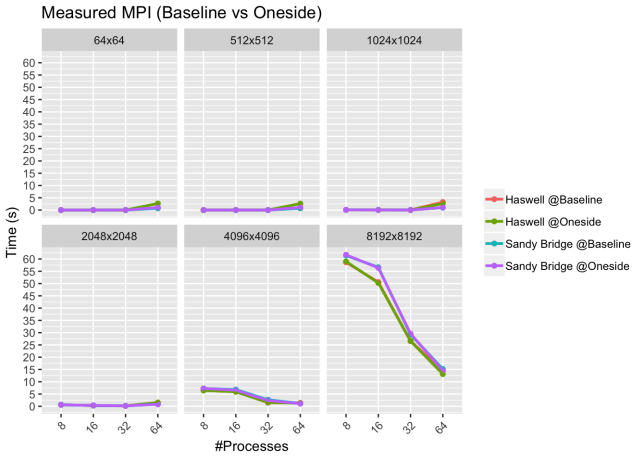
3. **Was a speedup observed versus the baseline for the Sandy Bridge and Haswell nodes?**

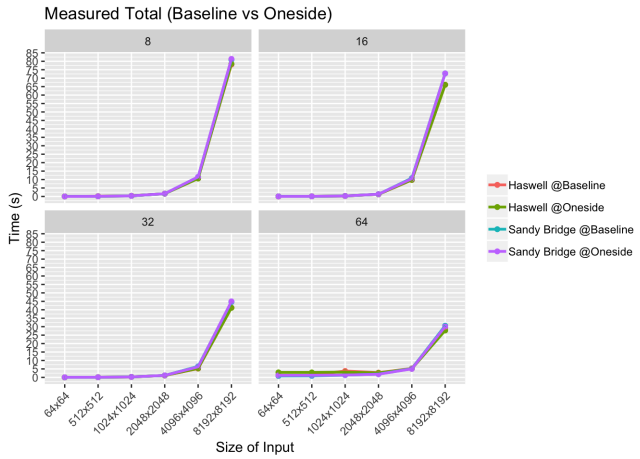
Speedup here is defined as `time(baseline)` / `time(one-sided)` . Both Sandy Bridge and Haswell nodes performed almost the same for One-sided communication with respect to baseline. From the plots, we observe that good overlap was achieved, but there was significant performance slow down.

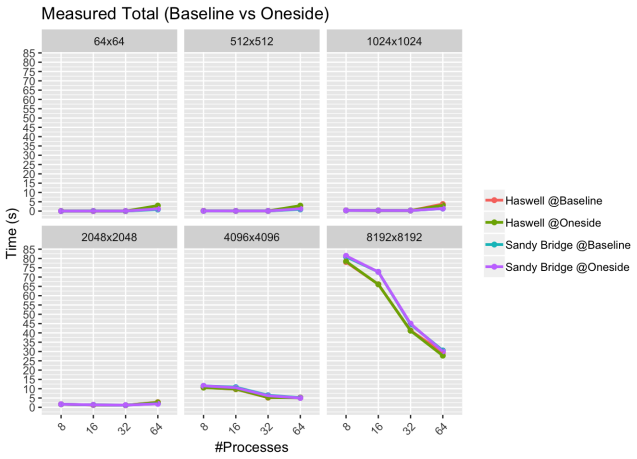






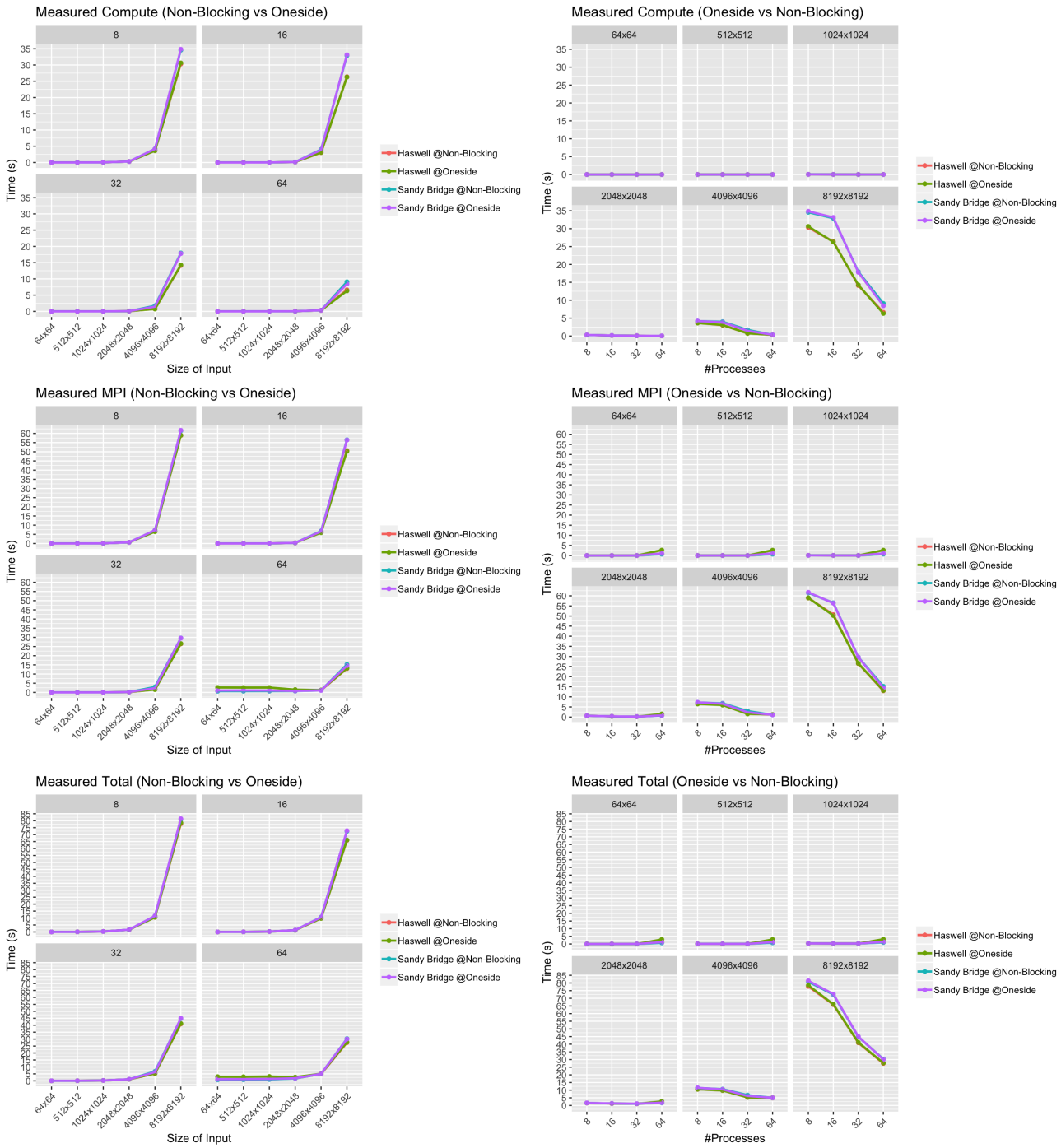






4. **Was a speedup observed versus the non-blocking version for the Sandy Bridge and Haswell nodes?**

   There was little speedup when comparing with the non-blocking version. As we can observe in the plots, the non-blocking communication performed better than the one-sided communication.



6 Contribution

1. **Smith**
   Build and run batch scripts for MPI point-to-point and one-sided communication
   Performance analysis with Vampir.

2. **Shyam**
   Understanding of the assignment and final report creation.
   Worked on MPI Point-to-Point Communication

3. **Siddhesh**
   Worked on MPI One-Sided Communication