Development project

Aim

The goal of this project is to create a script that can measure the latency between workers in a MPJ cluster. This is done by sending 'work' from the master worker to all other workers, which they then immediately send back. The work itself is empty, as the aim is to measure the trip time, rather than the processing speed of the worker. To determine the trip time, the master worker stores the nano-time just before sending to the worker, and then immediately stores the nano-time again when the worker sends the message back. The difference can then be found and halved.

Implementation

The implementation of this program takes 3 parameters. The first is the number of workers, which is required by MPJ, and can be changed to match the number of PC's set up for use in the cluster. The second parameter is a *payload* value. This represents the amount of data that is to be sent to the workers. The third parameter is the number of tests per worker. That is to say, how many messages will be sent to each worker. If the program is run with 5 workers, a payload of 4KB and 100 tests per worker, then 500 round trip messages will take place in total. Each message would contain 4KB of data.

When the program is run, the round-trip times are outputted to a log file (in nanoseconds). By running the program with various parameters, and analysing the results, it is possible to calculate the latency and bandwidth.

To get some results to analyse, the program was run a number of times with different parameters. All of the tests were run with 5 total workers, due to a limit in access to computers. However, the number of tests per worker parameter was ran with values 2, 10, 20, 50, 100, and 1000. The reason for having a range of tests per worker is to see if the accuracy increases with a larger sample size, and to check if a larger number of tests effects the results. The program will also be run with varying payload sizes, from 0.04KB to 4MB per message. Theoretically, the larger payloads should have a higher latency.

Results

Figure 1 – Average latency (nanoseconds)

	Payload Size						
Tests per worker	**10 (0.04 KB)**	**100 (0.4 KB)**	**250 (1 KB)**	**1000 (4 KB)**	**10000 (40 KB)**	**100000 (400 KB)**	**1000000 (4000 KB)**
2	1025610	1805171	899398	916758	1617177	11938787	118254533
10	802913	731294	1003938	619017	1801746	11770273	113045051
20	642596	789812	841741	734046	1608662	11805424	115100848
50	524974	533786	516558	654198	1313971	11573497	111285107
100	431860	489851	541072	552396	1858712	11699452	119339991
1000	287483	298945	339811	387992	1686591	10998696	110784230

Figure 1 contains the results of running the program with various parameters. The results represent the average latency for the group of tests, measured in nanoseconds. The results are divided by 2 to represent the latency of a single message from the master worker to another worker.

Figure 2 – Bandwidth speeds

	Bandwidth speeds						
	0.04 KB	0.4 KB	1 KB	4 KB	40 KB	400 KB	4000 KB
KB/ns	0.00	0.00	0.00	0.00	0.00	0.00	0.00
KB/s	139.14	1338.04	2942.81	10309.48	23716.48	36367.95	36106.22
MB/s	0.14	1.31	2.87	10.07	23.16	35.52	35.26
Mb/s	1.09	10.45	22.99	80.54	185.28	284.12	282.08
		Speeds found using 1000 tests per worker metric for highest accuracy					

Figure 2 shows the calculated bandwidth. The bandwidth has been calculated based on the results highlighted in yellow on figure 1 (1000 tests per worker) as this is likely to be the most accurate. The next section analyses these results.

Analysis

Figure 3 – Payload vs transfer speed

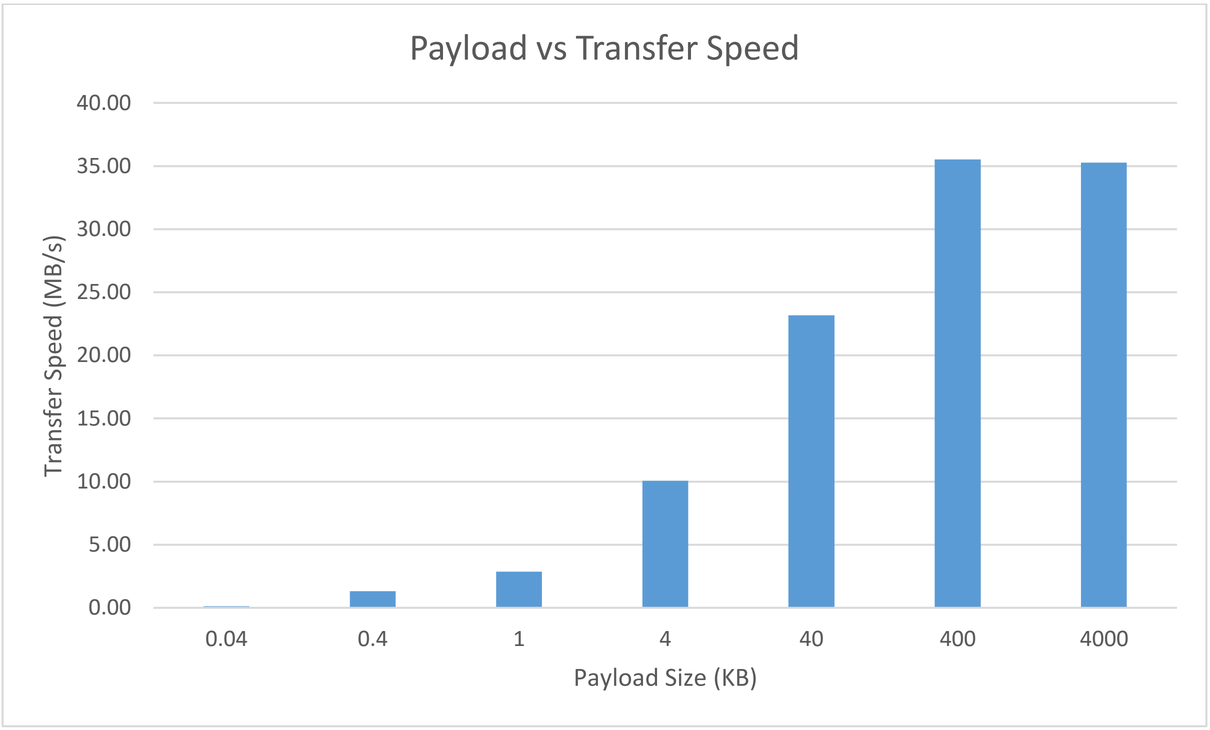

Figure 3 shows the transfer speeds (bandwidth) against the payload size, for the 1000 tests per worker results. The transfer speed increases with payload size, up to around 35 MB/s, where it seems to stop increasing. Unfortunately, no tests were run with a higher payload size to confirm this. It seems likely that the reason for this 'increase' in transfer speeds is due to the setup time required to send a message to a worker. For a low payload message, this process takes up a much larger amount of time relative to the time it takes to actually transfer the data. When sending larger amounts of data, this process is only a small fraction of the time required, and so the transfer speed is higher. It is important to note that just because it has a lower transfer speed, doesn't mean that it is slower to send the entire payload. It is just *less efficient* at it.

Unfortunately, due to a bug in the code, the payload is only included when being sent to the worker, and is not included when being sent back to the master worker. The tests were taken with this bug included, and due to time constraints could not be tested again once the bug was fixed. This means that the 35 MB/s transfer speed somewhat inaccurate, but can still server as a good estimate.

Figure 4 – Number of tests vs average time taken

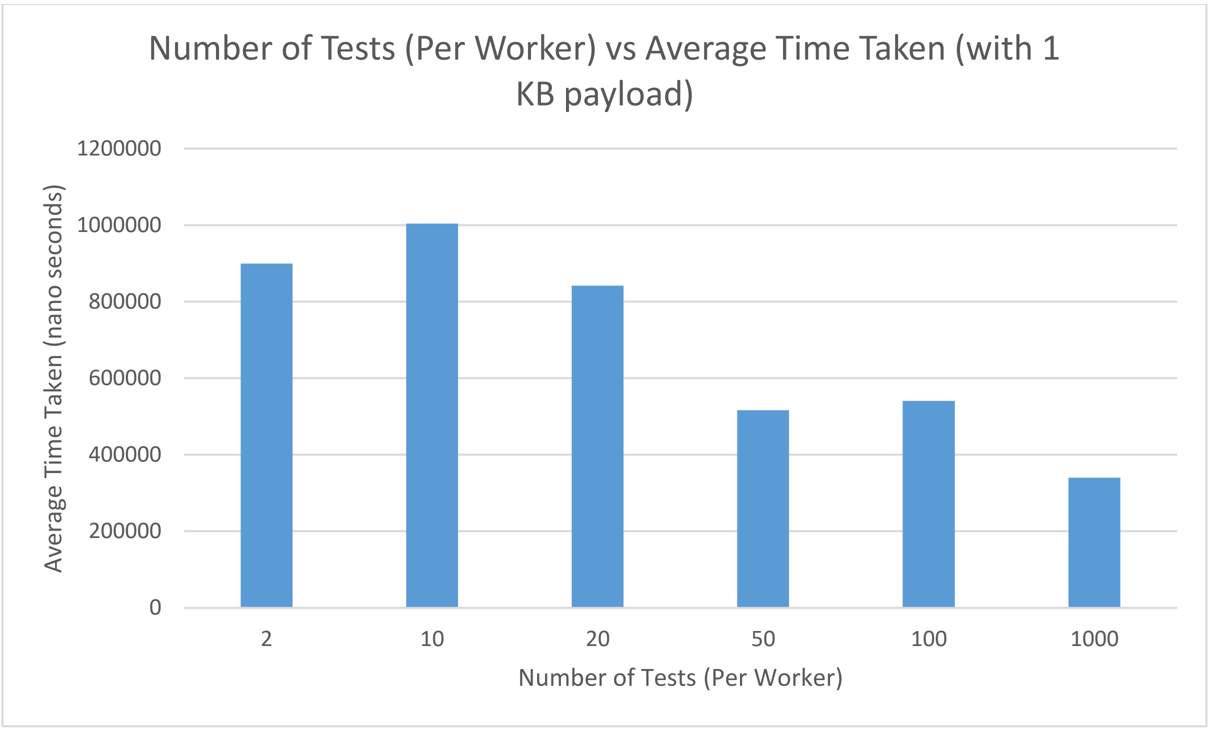

Figure 4 shows the average latency (for a 1 KB payload) compared to the number of tests per worker. When a small number of tests are conducted in a run, the latency seems to be higher. I am unsure of the reason for this, but it can likely be attributed to the fact that a low number of tests requires a very short amount of time to run, and so the program does not 'kick into high gear', in a sense.

Conclusion

In conclusion, the results seem to indicate that the message bandwidth for the Library PC's is 35 MB/s. This value may be higher with larger payloads, but unfortunately this was not tested. Future testing would involve a larger sample of parameters, mainly greater payload sizes to test the upper bound for the bandwidth. It is also important that any future tests are run with the payload fix, meaning that both trips to and from the worker have a payload. However, despite these issues, the result of 35 MB/s can be used as a good estimate for the bandwidth.