Oscar Teeninga

Komunikacja PP

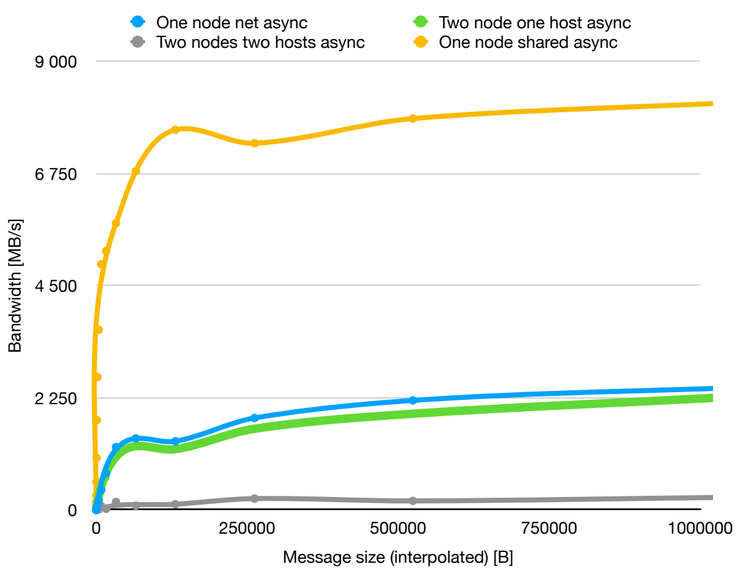
Message size [B] to bandwidth [MB/s]	One node shared		One node net		Two node one host		Two nodes two hosts	
	async	buffered	async	buffered	async	buffered	async	buffered
1	3,47481	0,64221	0,07133	0,06202	0,07097	0,06266	0,00915	0,00541
2	6,90315	1,29606	0,14337	0,12424	0,14223	0,12570	0,01963	0,01109
4	13,46313	2,59432	0,28662	0,24796	0,28296	0,25054	0,03666	0,02216
8	25,64529	5,18443	0,57349	0,49700	0,56383	0,50288	0,07305	0,04449
16	45,44716	10,33314	1,14697	0,99501	1,12770	1,00692	0,15642	0,08856
32	92,78568	20,65463	2,27395	1,98572	2,24234	2,00643	0,29166	0,17726
64	167,45139	40,56795	4,53317	3,95442	4,47823	4,00001	0,58070	0,35558
128	294,75000	78,75945	9,03572	7,89586	8,90513	7,96771	1,16143	0,67162
256	567,32791	151,17897	17,95204	15,56531	17,71103	15,76567	2,32327	1,34383
512	1045,54861	291,22667	35,45699	30,86791	35,27349	31,10639	4,63407	2,68167
1024	1801,89013	547,22231	50,51533	45,62475	49,91699	45,62363	7,45644	4,65173
2048	2665,82706	948,88776	98,08761	87,75868	96,76388	86,52687	9,56583	6,18443
4096	3614,77507	1506,58845	194,15132	173,10610	193,70914	173,25645	11,63861	10,73454
8192	4930,72230	2241,48597	399,94807	354,58234	356,69153	319,96329	78,50694	65,05093
16384	5195,00127	2615,99135	733,41455	633,31837	644,40922	569,14546	24,62368	25,20374
32768	5752,99094	3075,69465	1262,26274	1065,19942	1101,78571	945,45860	161,56307	251,06628
65536	6798,77585	3383,75822	1435,19891	1229,55420	1279,59103	1090,53868	85,27131	75,20011
131072	7625,06590	3688,16354	1379,06629	1229,11436	1219,83628	1085,26749	116,12759	127,98618
262144	7357,02877	3954,59414	1845,43434	1592,74260	1629,50360	1409,28610	228,57130	230,96754
524288	7850,87864	4183,91778	2198,11104	1745,35836	1930,75626	1639,83959	182,29377	184,08587
1048576	8168,90113	4101,24744	2450,03415	2130,44322	2264,67972	1873,68183	254,31356	258,83691
Latency [µs]	0,0059719	0,0246500	0,2205962	0,2528816	0,2233025	0,2499994	1,7220596	2,8123067

Metoda testowa

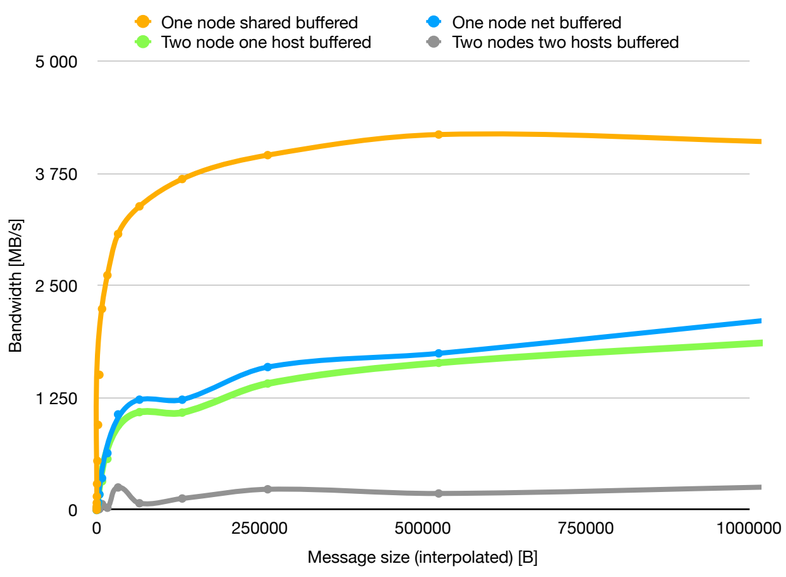
Stworzyłem jeden program mogący działać w dwóch trybach - sendera (nadawcę) oraz receivera (odbiorcę). Jest to skrót myślowy, ponieważ komunikacja zachodzi w obie strony: sender wysyła wiadomość do receivera, który odbiera wiadomość i sam wysyła odpowiedź zwrotną (coś na styl ping pong). Ze względu na to, że metoda odbierająca MPI_Recv jest blokująca, cała procedura pozwala oszacować czas dwukrotnego wysłania i odebrania wiadomości. Wybrałem dwie funkcje wysyłające: *MPI_Send* oraz *MPI_Bsend*. Całość została napisana w języku C. Do obliczeń czasu korzystałem z funkcji *MPI_Wtime*. Aby wyniki były bardziej wiarygodne liczba wysłanych wiadomości podczas jednego pomiaru wynosiła 100000. Względem polecenia nastąpiła jedna modyfikacja w postaci przedstawienia przepustowości w jednostce M**B**/s zamiast Mb/s (1 MB = 1 000 000 B). Przepustowość była czasem wysłania jednego bajta obliczana na podstawie przepustowości. Dla pomiarów powyżej 4096B liczba przesłanych wiadomości to 1000.

Wnioski

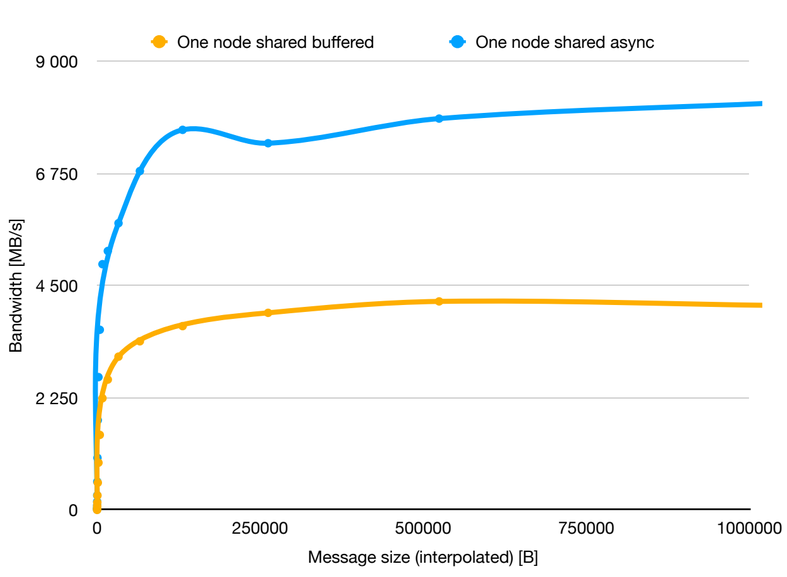
Zgodnie z oczekiwaniami - korzystanie z pamięci współdzielonej daje najlepszą przepustowość. Wersja z wysyłaniem z buforem w tym przypadku sprawdza się wyraźnie (dwukrotnie) gorzej niż asynchroniczny odpowiednik. W każdej innej konfiguracji wersja buforowana działa podobnie lub nieznacznie gorzej od wersji standardowej. Spowodowane jest to faktem, że wówczas ograniczeniem nie jest szybkość działania aplikacji, a medium sieciowe.
Interpolacja pokazuje, że przepustowość rośnie względem rozmiaru wiadomości w sposób pierwiastkowy, aby ostatecznie osiągnąć maksymalną wartość i oscylować w jej granicy z niewielkim wzrostem. Widzimy maksimum lokalne w przypadku komunikacji sieciowej dla wiadomości rozmiaru 65kB. W związku z maksymalną wielkością ramki Ethernet można sądzić, że jest to ze sobą skorelowane.
Najgorszą przepustowość osiąga połączenie pomiędzy dwoma fizycznymi węzłami, osiągając ~10-krotnie gorsze wyniki od sieciowej komunikacji na jednym node\host.
Komunikacja sieciowa na jednym node i dwóch node'ach na jednym fizycznym hoście osiągają porównywalną przepustowość (z niewielką przewagą wersji dla jednego węzła). Oznacza to, że logiczna implementacja wielu węzłów na jednej jednostce jest bardzo wydaja.
Najmniejsze opóźnienie osiągnąłem korzystając z pamięci współdzielonej na jednym node, korzystając z asynchronicznej funkcji wysyłającej, gdzie osiągnąłem prawie rząd wielkości lepszy wynik niż dla funkcji buforowanej. Opóźnienie dla komunikacji sieciowej na jednym fizycznym hoście było o rząd większe od pamięci współdzielonej, czego należało oczekiwać. O kolejny rząd wielkości zwiększyło się opóźnienie dla komunikacji między dwoma fizycznymi węzłami. W każdym przypadku funkcja buforowana była gorsza względem asynchronicznej.



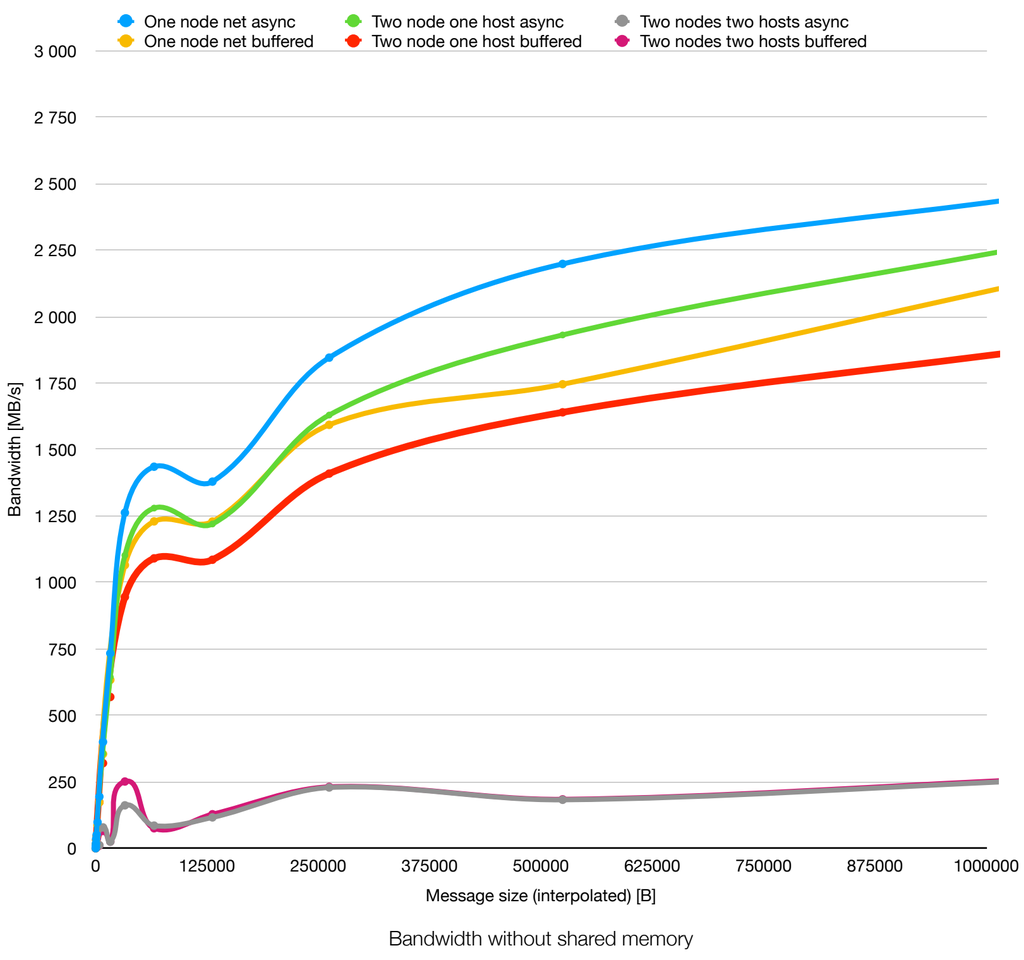
Asynchronous bandwidth



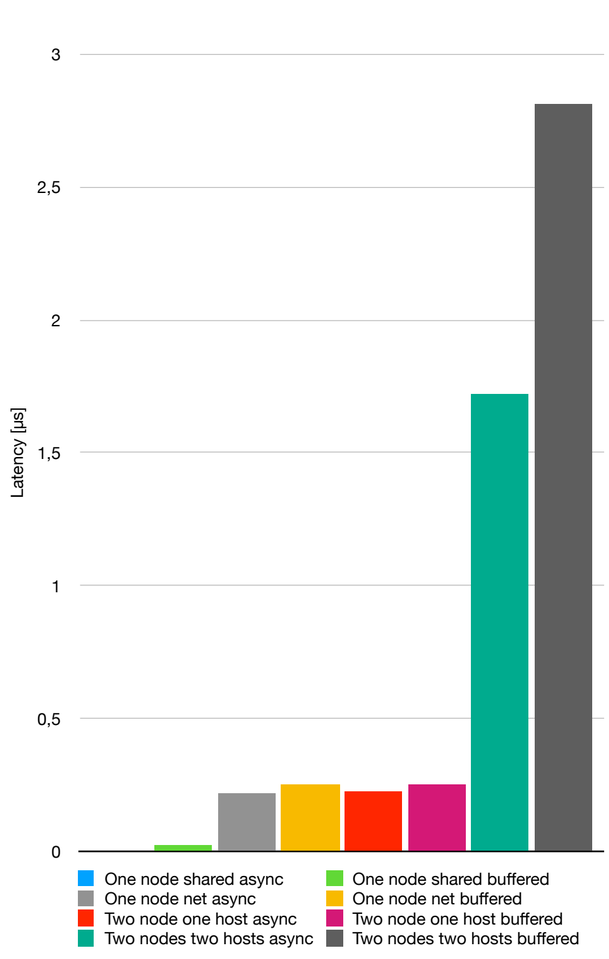
Buffered bandwidth



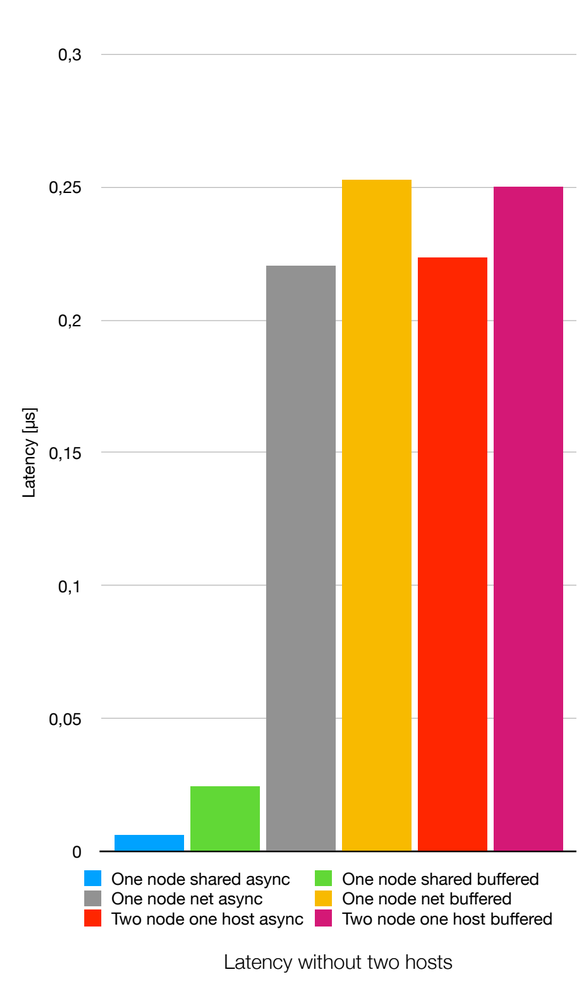
Shared memory bandwidth



Bandwidth without shared memory

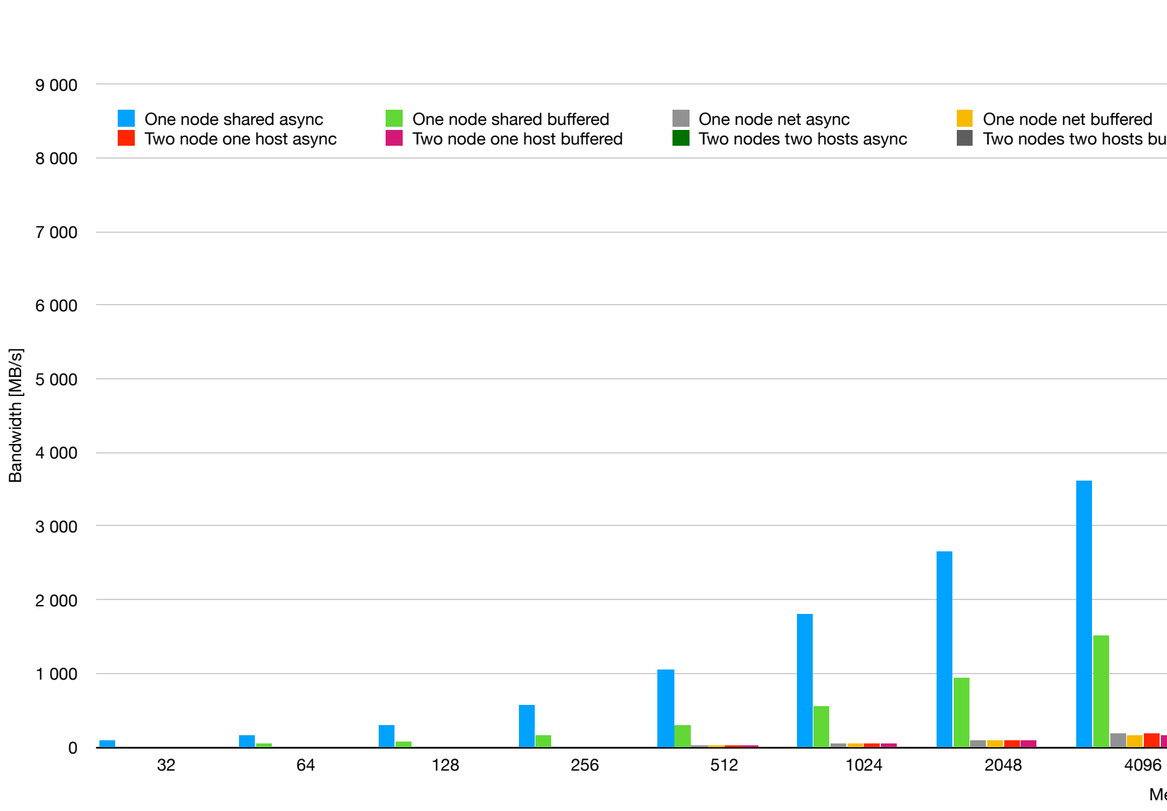


Latency

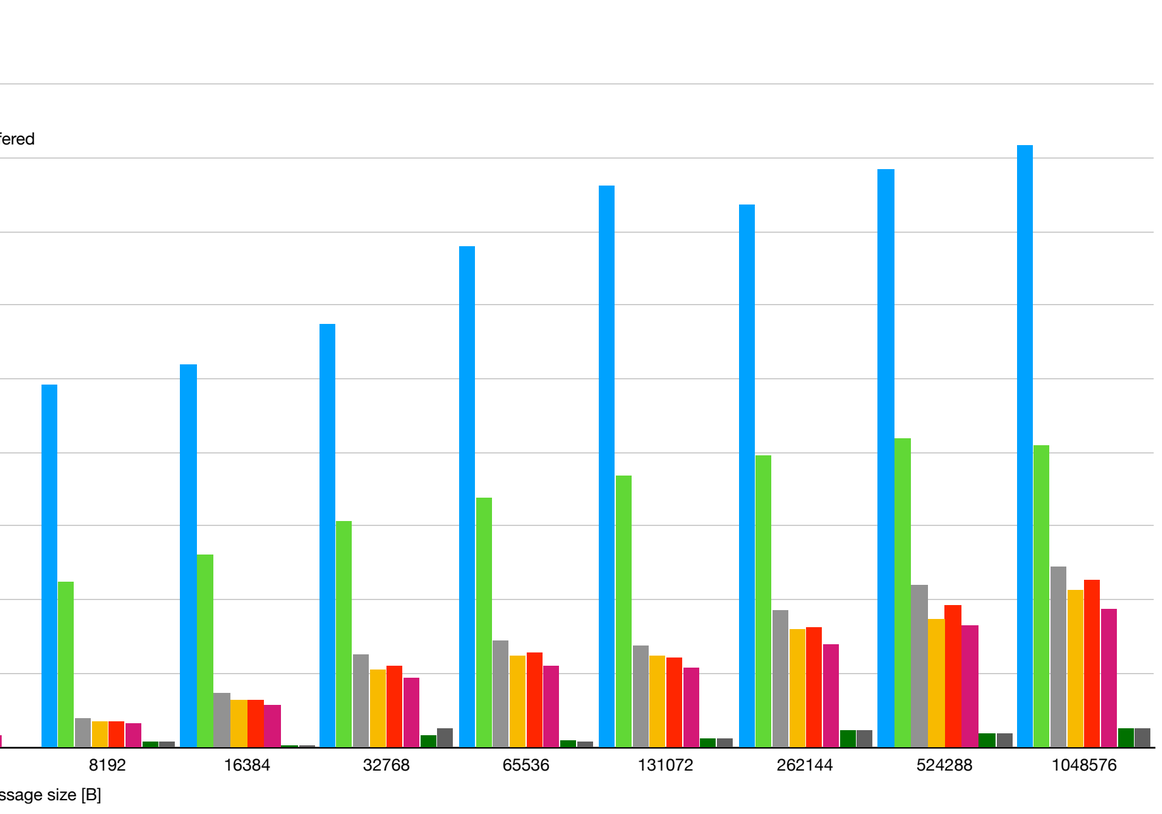


Latency without two hosts

Bandwidth for message size



Bandwidth for message size without shared memory



Bandwidth for message size without shared memory