УДК 004.032.24

Исследования скорости выполнения операций обмена данными при использовании Intel MPI

А.В. Леус
Институт кибернетики имени В. М. Глушкова НАН Украины
Московский физико-технический институт (государственный университет)

Введение и связанные работы

В последнее время в связи с интенсивным развитием вычислительной техники ведется все больше и больше работ, связанных с параллельным обменом и обработкой данных. Один из способов повышения эффективности таких работ, связан с поиском оптимальных алгоритмов обмена и обработки данных. Целью данной работы являлось сравнение времени выполнения различных алгоритмов обмена информацией при различном объёме данных, количества потоков и разных версий MPI.

В данной работе были проведены 3 серии экспериментов по разным видам обмена данных для разных кластеров. Для этого были рассмотрены базовые функции MPI используемые для данных задач, функции, написанные из базовых примитивов MPI, алгоритмы для обмена информации предложенный *Байдиным Г.В* [1] и алгоритм, предложенный *Rabenseifner, R* [2].

Эксперименты проводились на базе вычислительного кластера Института вычислительной математика РАН. Ниже приведена сводная таблица использованных кластеров.

Очередь	CPU	Количество узлов	Количество ядер на узел	Оперативная память	Версии Intel MPI
4 x4core	Intel Xeon E5462 (12M Cache, 2.80 GHz)	6	8	8 Гб	3.2.2, 4.0.1, 5.0.3
8 x8core	Intel Xeon E5-2665 (20M Cache, 2.40 GHz)	4	16	64 Гб	4.0.1, 4.1.3, 5.0.3
10 x10core	Intel Xeon E5-2670v2 (25M Cache, 2.50 GHz)	4	20	64 Гб	4.0.1, 4.1.3, 5.0.3
12 x12core	Intel Xeon E5-2670v3 (30M Cache, 2.30 GHz)	8	24	64 Гб	4.0.1, 4.1.3, 5.0.3

Подобные эксперименты проводиться довольно часто для оценки параметров системы, результаты подобных тестов можно увидеть в работах [1], [3]. Вопросы об эффективности обмена данных можно обнаружить в работах [2], [4], [5]. Данные работы довольно сильно пересекаются, но последняя более практическая. В них рассматривают коллективные операции на различных системах, в поиске оптимальных решений, как и в данной работе.

Основное отличие данной работы, состоит в том, что рассматривается различные кластера (различная конфигурация) и различные версии Intel MPI (реализация MPI).

Методика проведения экспериментов

Эксперименты проводились следующим образом: для каждого кластера, для каждой версии Intel MPI, проводились 20 запусков приложения для замера времени. Данная программа исполнялась для различного объема данных, в ней происходил сбор среднего времени исполнения предложенных алгоритмов для решения поставленной задачи на выбранной конфигурации. По результатам всех запусков получались усредненные значения времени исполнения, именно их мы и будем сравнивать в данной работе. Тип данных используемый для данного исследования double.

Исследования передачи данных от одного процесса другому:

Использованные алгоритмы:

- Использование блокирующих операций MPI_Send() и MPI_Recv().
- Использование неблокирующих операций MPI_Isend() и MPI_Irecv().
- Использование MPI_Bcast() для группы из двух потоков.

Полученные результаты:

Таблица 1. Время передачи данных в зависимости от версий MPI, способа передачи и объема данных для кластера x12core, когда потоки находятся на одном узле.

	Блокирующие операции (мс)			Не блокирующие операции (мс)			MPI_Bcast() (мс)		
Версия Intel MPI	5.0.3	4.1.3	4.0.1	5.0.3	4.1.3	4.0.1	5.0.3	4.1.3	4.0.1
1 элемент	0,003	0,003	0,002	0,002	0,003	0,002	0,003	0,004	0,002
10 элементов	0,002	0,002	0,002	0,001	0,001	0,001	0,002	0,002	0,001
10^2 элементов	0,002	0,001	0,002	0,001	0,001	0,001	0,001	0,001	0,001
10^3 элементов	0,004	0,004	0,004	0,004	0,004	0,003	0,004	0,004	0,004
10^4 элементов	0,027	0,028	0,027	0,023	0,023	0,021	0,023	0,024	0,022
10^5 элементов	0,175	0,165	0,17	0,156	0,158	0,154	0,158	0,158	0,154
10^6 элементов	1,238	1,279	1,309	1,195	1,192	1,218	1,208	1,195	1,212
10^7 элементов	12,214	12,556	12,639	11,743	11,946	12,036	11,896	11,975	12,024
10^8 элементов	123,813	125,601	126,78	119,25	121,733	122,832	121,035	121,928	122,742

При рассмотрении этой таблицы можно увидеть, что оптимальное соотношения затрачиваемого времени на пересылку к количеству элементов достигается при длине сообщения порядка 10^7 элементов, это значения имеет порядок кэша процессора. Также видно, что асинхронная передача данных выгоднее, чем при использовании блокирующих операций. Это верно и для тестов на других кластерах.

Таблица 2. Время передачи данных в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x4core, когда потоки находятся на разных узлах.

	Блокирующие операции (мс)			Не блокирующие операции (мс)			MPI_Bcast() (мс)		
Версия Intel MPI	5.0.3	4.0.1	3.2.2	5.0.3	4.0.1	3.2.2	5.0.3	4.0.1	3.2.2
1 элемент	0,005	0,004	0,004	0,004	0,004	0,005	0,005	0,003	0,004
10 элементов	0,002	0,002	0,002	0,002	0,001	0,002	0,002	0,002	0,003
10^2 элементов	0,003	0,002	0,003	0,003	0,002	0,003	0,003	0,002	0,003
10^3 элементов	0,007	0,005	0,008	0,006	0,005	0,008	0,007	0,006	0,007
10^4 элементов	0,049	0,048	0,05	0,04	0,039	0,042	0,042	0,03	0,045
10^5 элементов	0,335	0,338	0,336	0,319	0,321	0,317	0,319	0,321	0,318
10^6 элементов	3,48	3,485	3,454	3,341	3,472	3,318	3,343	3,373	3,325
10^7 элементов	33,66	33,969	33,424	32,329	33,65	32,112	32,35	32,867	32,197
10^8 элементов	335,474	336,442	333,897	322,103	326,311	320,803	322,248	326,477	320,734

Аналогично прошлой таблице, асинхронная передача данных показывает лучшие результаты. Если сравнить быстродействие от версии Intel MPI – существенной разницы не наблюдается, значения находятся в пределах допустимой погрешности.

Рассмотрим время, затрачиваемое на передачу одного элемента

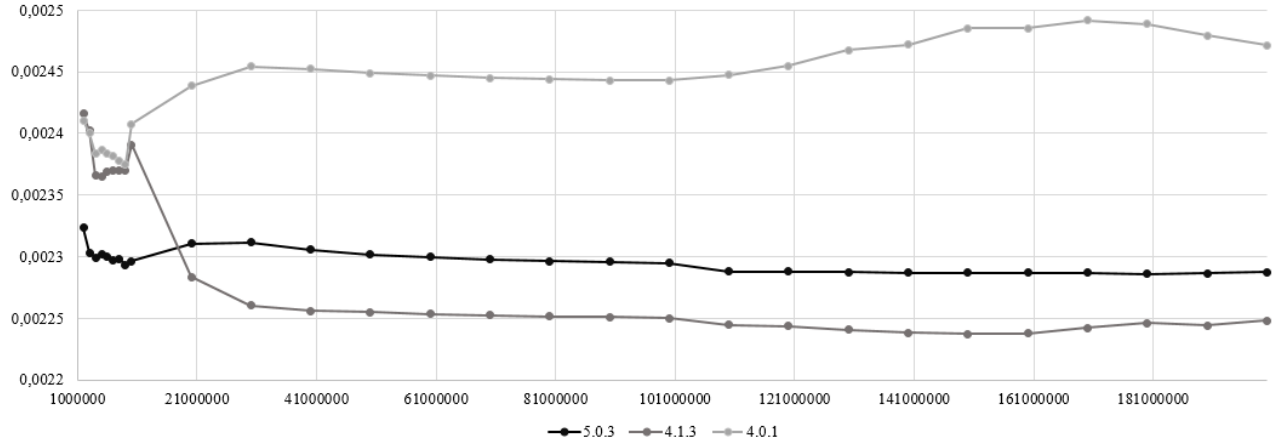

Рис1. Зависимость времени (ось Y, (мкс)) на передачу одного элемента в сообщении определенной длины (ось X) для разных версий Intel MPI, для кластера x8core.

Данный график показывает, что при возрастании длины сообщения, время на передачу одного элемента изменяется не значительно, особенно это актуально для последних версий. Следствие из этого: нет необходимости разбивать данные на несколько частей для передачи одного от одного потока другому.

Исследования широковещательного обмена данными:

Использованные алгоритмы:

- Стандартная функция MPI_Bcast() и ее модификации, в которой данные разбиваются на несколько частей
- Алгоритмы передачи данных основанные на бинарном дереве
- Алгоритм, предложенный *Байдиным Г.В.* [1]

Полученные результаты:

Таблица 3. Время выполнение коллективной рассылки данных (MPI_Bcast()) в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x12core.

MPI_Bcast()	Количество потоков = 2, время мс			Количество потоков = 4, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,005	0,006	0,004	0,012	0,013	0,01
10^3 элементов	0,006	0,006	0,003	0,014	0,014	0,013
10^4 элементов	0,037	0,038	0,029	0,52	0,504	0,578
10^5 элементов	0,196	0,197	0,175	0,792	0,445	0,611
10^6 элементов	2,086	1,955	1,352	5,333	4,331	5,203
10^7 элементов	18,002	19,071	13,187	39,599	46,274	39,756
10^8 элементов	140,034	139,599	132,805	337,633	348,711	314,128
MPI_Bcast()	Количество потоков = 8, время мс			Количество потоков = 16, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,022	0,018	0,029	0,04	0,035	0,042
10^3 элементов	0,029	0,032	0,037	0,058	0,067	0,06
10^4 элементов	0,114	0,089	0,14	0,15	0,181	0,151
10^5 элементов	0,645	0,672	0,809	1,211	1,017	0,978
10^6 элементов	4,401	5,541	5,716	9,07	9,188	5,895
10^7 элементов	81,559	88,687	74,948	84,264	83,424	55,393

10^8 элементов	1945,91	996,401	1713,482	900,787	696,991	639,091

Таблица 4. Время выполнение коллективной рассылки данных (разделение данных на две группы MPI_Bcast()) в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x12core.

MPI_Bcast() x2	Количество потоков = 2, время мс			Количество потоков = 4, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,006	0,006	0,003	0,01	0,009	0,007
10^3 элементов	0,007	0,007	0,003	0,013	0,01	0,011
10^4 элементов	0,043	0,034	0,021	0,078	0,055	0,058
10^5 элементов	0,166	0,168	0,164	0,379	0,347	0,393
10^6 элементов	1,752	1,628	1,173	3,937	3,899	3,863
10^7 элементов	15,931	17,067	11,653	35,475	42,469	35,932
10^8 элементов	124,646	125,388	119,541	305,674	330,587	295,131
MPI_Bcast() x2	Количество потоков = 8, время мс			Количество потоков = 16, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,017	0,022	0,021	0,033	0,047	0,032
10^3 элементов	0,026	0,037	0,031	0,054	0,074	0,054
10^4 элементов	0,086	0,106	0,099	0,167	0,195	0,159
10^5 элементов	0,435	0,508	0,567	0,806	0,917	0,657
10^6 элементов	3,992	4,623	4,863	8,306	8,892	5,405
10^7 элементов	70,39	75,007	64,02	80,295	83	54,085
10^8 элементов	1239,466	892,129	1445,27	675,169	689,1	600,66

Сравнивая значения из этих двух таблиц, можно заметить, что в некоторых случаях можно существенно уменьшить время работы, разбив данные на две группы. Эта особенность заметна на большом количестве элементов, также разбиение на две группы не единственный способ разделить данные, подобное разделение необходимо смотреть на конкретной вычислительной системе при конкретных размерах, передаваемых данных.

В этом разделе публикуются данные только по кластеру x12core, на других кластерах мы имеем похожие значения, но иногда разделение на несколько групп данных, дает незначительное уменьшение времени.

Таблица 5. Время выполнение коллективной рассылки данных (по бинарному дереву с использованием асинхронной передачей данных) в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x12core.

Binary tree, async	Количество потоков = 2, время мс			Количество потоков = 4, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,004	0,004	0,003	0,009	0,01	0,008
10^3 элементов	0,007	0,004	0,003	0,022	0,021	0,022
10^4 элементов	0,023	0,024	0,022	0,056	0,058	0,053
10^5 элементов	0,157	0,157	0,153	0,389	0,392	0,37
10^6 элементов	1,747	1,63	1,181	3,93	4,183	3,6

10^7 элементов	16,061	17,178	11,808	36,211	46,25	33,903
10^8 элементов	124,46	125,197	121,376	301,976	317,841	253,91
Binary tree, async	Количество потоков = 8, время мс			Количество потоков = 16, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,018	0,017	0,016	0,019	0,026	0,021
10^3 элементов	0,018	0,019	0,02	0,028	0,031	0,029
10^4 элементов	0,082	0,084	0,084	0,139	0,125	0,134
10^5 элементов	0,593	0,595	0,606	1,1	0,888	1,019
10^6 элементов	5,28	5,675	5,962	12,952	9,929	10,762
10^7 элементов	66,256	58,426	48,565	121,289	98,614	117,259
10^8 элементов	549,539	586,373	465,8	1103,678	887,523	1131,853

При рассмотрении данных результатов, видно, что данный метод может показать меньшее время, чем базовый алгоритм и его модификация, при количестве потоков до 8 включительно. Особенно это заметно при количестве данных не более 10^4 элементов.

Таблица 6. Время выполнение коллективной рассылки данных (алгоритм, предложенный *Байдиным Г.В.* [1]) в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x12core.

Байдиным Г.В.	Количество потоков = 2, время мс			Количество потоков = 4, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,004	0,004	0,003	0,009	0,011	0,011
10^3 элементов	0,007	0,005	0,003	0,009	0,011	0,01
10^4 элементов	0,024	0,024	0,021	0,071	0,065	0,072
10^5 элементов	0,157	0,157	0,154	0,481	0,529	0,534
10^6 элементов	1,749	1,626	1,173	4,286	4,505	4,645
10^7 элементов	16,073	17,192	11,715	38,872	44,616	42,013
10^8 элементов	124,488	125,208	119,657	355,814	374,522	352,674
Байдиным Г.В.	Количество потоков = 8, время мс			Количество потоков = 16, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,019	0,021	0,02	0,036	0,041	0,036
10^3 элементов	0,016	0,021	0,018	0,033	0,042	0,032
10^4 элементов	0,06	0,072	0,071	0,096	0,119	0,08
10^5 элементов	0,69	0,731	0,752	0,813	0,921	0,719
10^6 элементов	5,874	6,267	6,177	7,877	8,978	8,057
10^7 элементов	62,809	57,921	52,854	85,736	96,201	83,281
10^8 элементов	520,873	568,093	494,704	812,028	862,625	798,003

При изучении данной таблицы, можно сделать вывод, что результаты прошлой таблицы для 8 потоков могут быть улучшены, данным алгоритмом. Это можно объяснить тем, что у предыдущего алгоритма временная сложность логарифмически зависима от количества потоков, а у данного алгоритма зависимость линейная, но есть квадратичная от количества потоков накладка по времени

вызываемая передачей сообщений. Именно это делает не дает данному алгоритму показать результаты лучше, чем базовый на количестве потоков от 16.

Таким образом, при количестве потоков до ~8 оптимальный алгоритм — это пересылка по бинарному дереву с асинхронной передачей данных, при ~8 оптимальный алгоритм – предложенный *Байдиным Г.В.* [1], в остальных случаях – следует использовать базовый алгоритм широковещательной рассылки или его модификацию (более эффективна при больших объёмах данных).

В сравнении с оригинальной работой [1], базовая функция MPI_Bcast(), стала ощутимо эффективней и алгоритм предложенный в данной работе, уже не является столь эффективным как был ранее.

Если говорить о времени исполнения в зависимости от версии Intel MPI, тесты однозначно не показали этой информации. В целом время исполнения не имеет сильных отличий, но в целом лучше ориентировать на последние версии Intel MPI.

Исследования коллективных операций на примере векторного сложения:

Использованные алгоритмы:

- Стандартная функция MPI_Allreduce()
- Алгоритм, предложенный *Rabenseifner, R* [2]
- Алгоритмы, основанные на бинарном дереве
- Модификация алгоритма, предложенного *Байдиным Г.В.* [1]

Полученные результаты:

Таблица 7. Время выполнение коллективной операции векторного сложения данных (Allreduce()) в зависимости от версий Intel MPI, способа передачи и объема данных для кластера x12core.

Allreduce	Количество потоков = 2, время мс			Количество потоков = 4, время мс			Количество потоков = 8, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,007	0,008	0,009	0,012	0,026	0,016	0,022	0,038	0,027
10^3 элементов	0,012	0,017	0,015	0,012	0,02	0,016	0,046	0,05	0,057
10^4 элементов	0,061	0,089	0,066	0,062	0,094	0,069	0,149	0,137	0,19
10^5 элементов	0,429	0,416	0,426	0,933	1,14	1,043	1,185	0,942	1,54
10^6 элементов	4,183	4,273	4,862	9,512	7,595	12,976	13,964	10,089	18,69
10^7 элементов	44,629	46,135	65,889	85,471	71,939	143,742	122,814	114,841	180,151

Allreduce	Количество потоков = 16, время мс			Количество потоков = 32, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,033	0,043	0,03	0,243	0,199	0,217
10^3 элементов	0,053	0,044	0,037	0,210	0,207	0,240
10^4 элементов	0,223	0,18	0,229	0,441	0,36	0,387
10^5 элементов	2,102	2,54	2,402	5,327	5,463	7,077
10^6 элементов	26,955	15,665	28,195	23,387	20,71	25,302
10^7 элементов	283,608	144,266	226,945	192,633	175,017	204,769

Таблица 8. Время выполнение коллективной операции векторного сложения данных (алгоритм, предложенный *Rabenseifner, R* [2]) в зависимости от версий MPI, способа передачи и объема данных для кластера x12core.

Rabenseifner, R	Количество потоков = 2, время мс			Количество потоков = 4, время мс			Количество потоков = 8, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,005	0,005	0,008	0,006	0,009	0,008	0,014	0,023	0,019
10^3 элементов	0,009	0,009	0,012	0,008	0,014	0,011	0,039	0,043	0,05
10^4 элементов	0,045	0,039	0,051	0,041	0,072	0,057	0,128	0,124	0,174
10^5 элементов	0,335	0,331	0,364	0,797	0,642	0,873	1,012	0,865	1,412
10^6 элементов	3,823	3,868	4,308	9,132	7,24	12,616	13,466	9,715	18,62
10^7 элементов	42,298	43,362	63,946	84,057	70,301	142,10	120,64	112,722	179,127

Rabenseifner, R	Количество потоков = 16, время мс			Количество потоков = 32, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,024	0,019	0,029	0,193	0,185	0,219
10^3 элементов	0,041	0,032	0,033	0,199	0,182	0,225
10^4 элементов	0,192	0,163	0,191	0,444	0,332	0,385
10^5 элементов	1,678	1,417	2,07	5,271	5,238	5,889
10^6 элементов	25,492	15,009	27,692	22,874	20,602	25,261
10^7 элементов	266,436	141,952	226,522	193,737	188,633	204,789

Таблица 9. Время выполнение коллективной операции векторного сложения данных (с использованием бинарного дерева и неблокирующие операции) в зависимости от версий MPI, способа передачи и объема данных для кластера x12core.

Binary tree, async	Количество потоков = 16, время мс			Количество потоков = 32, время мс		
Версия Intel MPI	5.0.1	4.1.3	4.0.1	5.0.1	4.1.3	4.0.1
10^2 элементов	0,049	0,034	0,036	0,083	0,073	0,091
10^3 элементов	0,06	0,048	0,052	0,105	0,09	0,094
10^4 элементов	0,32	0,206	0,221	0,321	0,284	0,368
10^5 элементов	2,756	1,755	1,951	2,419	2,252	2,523
10^6 элементов	39,626	30,431	33,187	36,907	36,937	39,412
10^7 элементов	386,364	306,341	338,363	448,992	385,941	425,734

Ввиду большого количества данных, время для других алгоритмов и кластеров, не включено в данную работу, так как показатель быстродействия там ниже. Картина времени выполнения всех алгоритмов одинакова на всех тестируемых кластерах.

Из данных таблиц следует тот факт, что алгоритм, предложенный *Rabenseifner, R* [2], показывает лучшее время в большинстве тестов. Также следует отметить важную особенность, что при количестве потоков 32 и объёмах данных до порядка 10^5 элементов, лучший результат будет показывать сложение по бинарному дереву с асинхронной передачей данных.

По результатам тестов для всех кластеров, лучше всего использовать версию Intel MPI 4.1.3, которая в данной задаче показывает лучшее время.

Заключение

Последние версии Intel MPI не имеют сильных различий во времени работы исследуемых функций, но в целом версия 4.1.3 показывает лучше результаты, на тестируемых кластерах.

Стандартные коллективные функции лучше всего показывают себя на больших объёмах данных и большом количестве потоков, но могут быть улучшены. Примером этого служит, алгоритм *Rabenseifner, R* [2].

Для малого количества данных и/или потоков использовать собственные функции для обмена и обработки информации – эффективное решение.

Литература

1. *Байдин Г. В.* О некоторых стереотипах параллельного программирования // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов 2008. № 1. С. 67-75.
2. *Rabenseifner, R., Traff, J.L.* More efficient reduction algorithms for non-powerof-two number of processors in message-passing parallel systems. // Proceedings of EuroPVM/MPI. Lecture Notes in Computer Science, Springer-Verlag (2004)
3. *Курносов М.Г.* MPIPerf: пакет оценки эффективности коммуникационных функций стандарта MPI // Труды международной научной конференции "Параллельные вычислительные технологии (ПаВТ-2012)". – Новосибирск, 2012. - С. 212-223
4. *Pjesivac-Grbovic J, Angskun T, Bosilca G, Fagg GE, Gabriel E, Dongarra J.* Performance analysis of MPI collective operations. // Cluster Computing. 2007 V. 10 N.1 P.127–143.
5. *Thakur R., Rabenseinfer R., Gropp W.*, Optimization of Collective Communication Operations in MPICH // Int'l J. High Performance Computing Applications, 2005, V. 19, N. 1, P. 49-66.