Отчеты по MPI

Задание 1. Поиск минимального элемента массива.

Описание задачи

Задача состояла в распараллеливании алгоритма поиска минимального элемента массива, состоящего из 100, 100000 и 100000000 элементов. Программа тестировалась на 2, 4, 8 и 16 потоках. Код решения данной задачи находится по ссылке https://github.com/SkyWaet/mpi-tasks/blob/main/1.vectorMiniMax/main.cpp.

Результаты

Ниже приведены таблицы с результатами, полученными для каждого из рассмотренных размеров массивов.

Размер массива / Количество потоков	100	100000	100000000
1	0.0000011465	0.000502361	0.42585016525
2	0.000125818	0.000467804000000	0.277929767
4	0.000181858	0.000496470000000	0.248396548
8	0.000441484	0.000782762000000	0.244020261
16	0.006531869	0.001345538000000	0.273547118

Как мы можем заметить, для массивов из 100 элементов любая из многопоточных версий работает существенно медленнее однопоточной версий.

Для массивов из 100000 элементов незначительное ускорение показали версии на 2 и 4 потоках, версии с большим количеством потоков работают значительно медленнее однопоточной версии.

Для массивов из 100000000 элементов, напротив, любая из многопоточных версий оказалась быстрее однопоточного алгоритма примерно в 1.5 раза.

Столь малое ускорение (особенно по сравнению с результатами, полученными при использовании OpenMP) можно объяснить тем, что сама задача имеет крайне низкую вычислительную сложность, которая намного меньше затрат на обмен данными между процессами.

Задание 2. Скалярное произведение векторов.

Описание задачи

Задача состояла в распараллеливании алгоритма нахождения скалярного произведения векторов, состоящего из 100, 100000 и 100000000 элементов. Программа тестировалась на 2, 4, 8 и 16 потоках. Код решения данной задачи находится по ссылке https://github.com/SkyWaet/mpi-tasks/blob/tasks123/2.dotProduct/main.cpp.

Результаты

Ниже приведены таблицы с результатами, полученными для каждого из рассмотренных размеров векторов.

Размер вектора / Количество потоков	100	100000	100000000
1	0.0000012465	0.00067152525	0.6407557155
2	0.000146685	0.000702709	0.477656312
4	0.000175152	0.000734716	0.455096188
8	0.00039686	0.00093255	0.460696181
16	0.003129552	0.001488621	0.560043132

Как мы можем заметить, для векторов из 100 элементов любая из многопоточных версий работает существенно медленнее однопоточной версий.

Для векторов из 100000 все многопоточные версии все так же работают медленнее однопоточной, но различие хотя бы является не таким сильным, как в первом случае

Для векторов из 100000000 элементов, аналогично задаче 1, все многопоточные версии выдают незначительно ускорение по сравнению с однопоточной версией.

Столь малое ускорение (особенно по сравнению с результатами, полученными при использовании OpenMP) можно объяснить тем, что сама задача имеет крайне низкую вычислительную сложность, которая намного меньше затрат на обмен данными между процессами.

Задание 3. Измерение времени обмена сообщениями между потоками

Постановка задачи

Необходимо измерить затраты на обмен сообщениями между процессами. В нашем случае процессы обмениваются строками по n байт, где n от 1 до 10000000, $n_i = 10\ n_{i-1}$. Для каждого n по 30 измерениям берется среднее значение. При этом были произведены замеры как для случая, когда оба процесса находятся на одном узле, так и при нахождении процессов на разных узлах.

Код примера находится по ссылке https://github.com/SkyWaet/mpi-tasks/blob/tasks123/3.messageExchange/main.cpp.

Результаты

Размер сообщения	Один узел	Разные узлы
1	0.000002733633333333332102806253	0.000491560666666666467965185472
10	0.000000233800000000003986145070	0.000007237333333333835844116029
100	0.000002589499999999999093491408	0.000006471900000000474156651963
1000	0.000002637033333333325692385317	0.000007265366666666060046922762
10000	0.000004616099999999913333506921	0.000014146066666665772093439482
100000	0.000017811233333333773687758364	0.000080647133333327603119805582
1000000	0.000218747466666674444816034772	0.000963661100000012815769423469
10000000	0.003110425800000093943253887119	0.062136372999999939237270041303

Как и ожидалось, с увеличением размера сообщения растет среднее время его отправки. При этом время отправки сообщения при размещении процессов на разных узлах как минимум в 4 раза больше, чем время отправки сообщения процессу, находящемуся на одном узле с отправителем.

Задание 4. Различные функции для обмена сообщениями

Постановка задачи

Необходимо измерить затраты на обмен сообщениями между процессами для различных функций передачи сообщений:

- MPI_Ssend
- MPI_Bsend
- MPI_Rsend

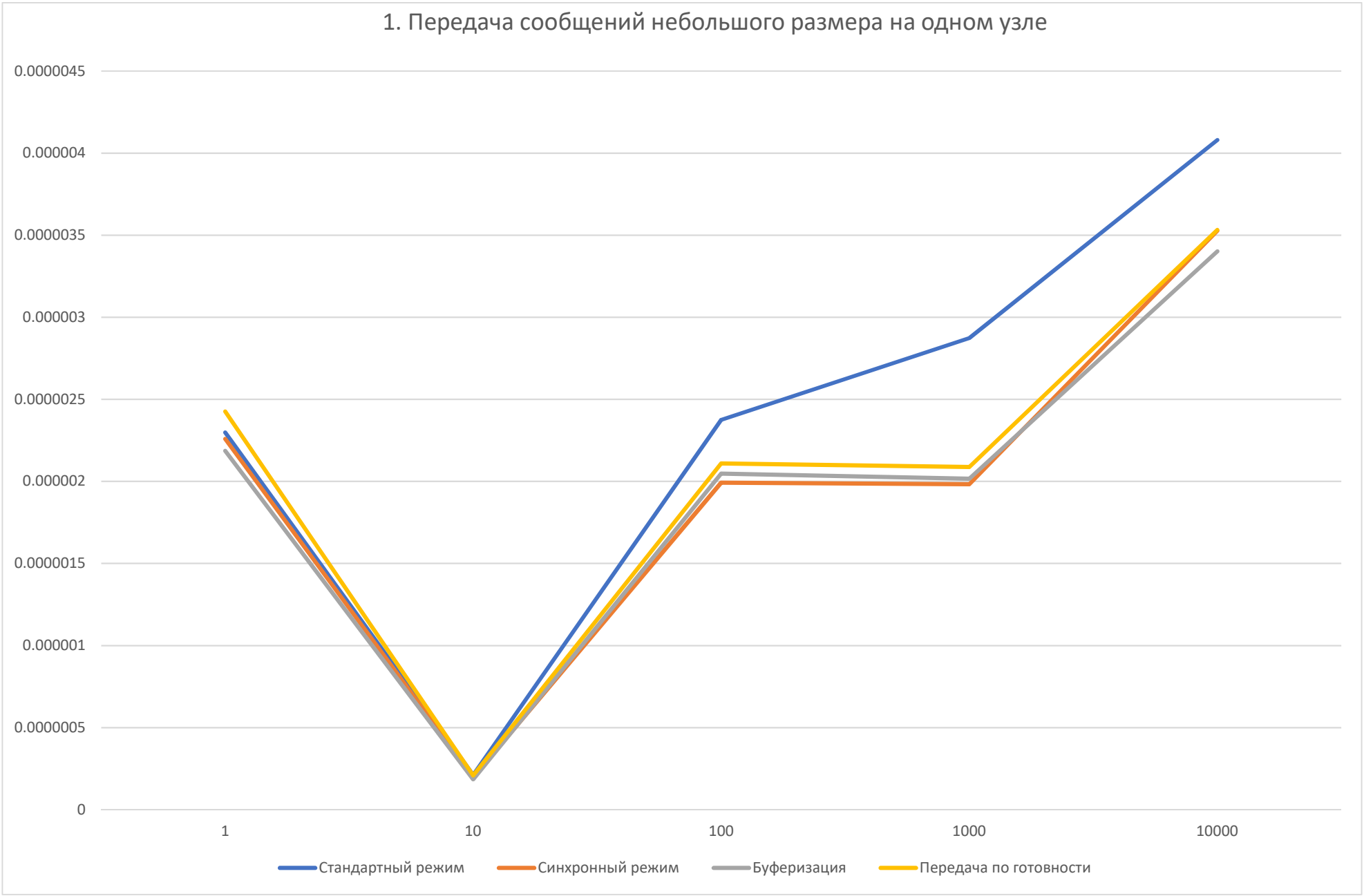
В нашем случае процессы обмениваются строками по n байт, где n от 10 до 10000000, $n_i = 10\ n_{i-1}$. Для каждого n по 30 измерениям берется среднее значение. При этом были произведены замеры как для случая, когда оба процесса находятся на одном узле, так и при нахождении процессов на разных узлах.

Результаты

Один узел

Размер сообщения	Стандартный режим	Синхронный режим	Буферизация	Передача по готовности
10	0.000000211733333333334117563044	0.000000197400000000005486617039	0.000000184766666666664780821805	0.000000208299999999997305797226
100	0.000002375466666666665703104349	0.000001992000000000007778794319	0.000002047400000000003470607135	0.000002109366666666662863715339
1000	0.000002873500000000003912962836	0.000001982333333333352950203455	0.000002016633333333332868994147	0.000002088000000000010332489638
10000	0.000004079866666666633240095809	0.000003527299999999922165661807	0.000003401599999999941732834732	0.000003532633333333216794994993
100000	0.000018540166666666225248972025	0.000017963200000000018020061832	0.000017461866666667116843036922	0.000018180066666667028075577220
1000000	0.000215005066666671768244381568	0.000216139900000001755576636842	0.000217325666666678957291911067	0.000219096300000010135462097383
10000000	0.003084577333333323422015492810	0.003097345366666651948067690014	0.003176892199999910900570787575	0.003064304233333326084320891169

Как мы можем заметить из диаграммы 1, для сообщений небольшого размера все варианты превосходят стандартный метод по скорости работы. Наиболее быстрыми способами передачи данных оказываются синхронный режим и передача с буферизацией. Для сообщений большого размера все методы показали примерно одинаковое время работы, что отражает диаграмма 2.

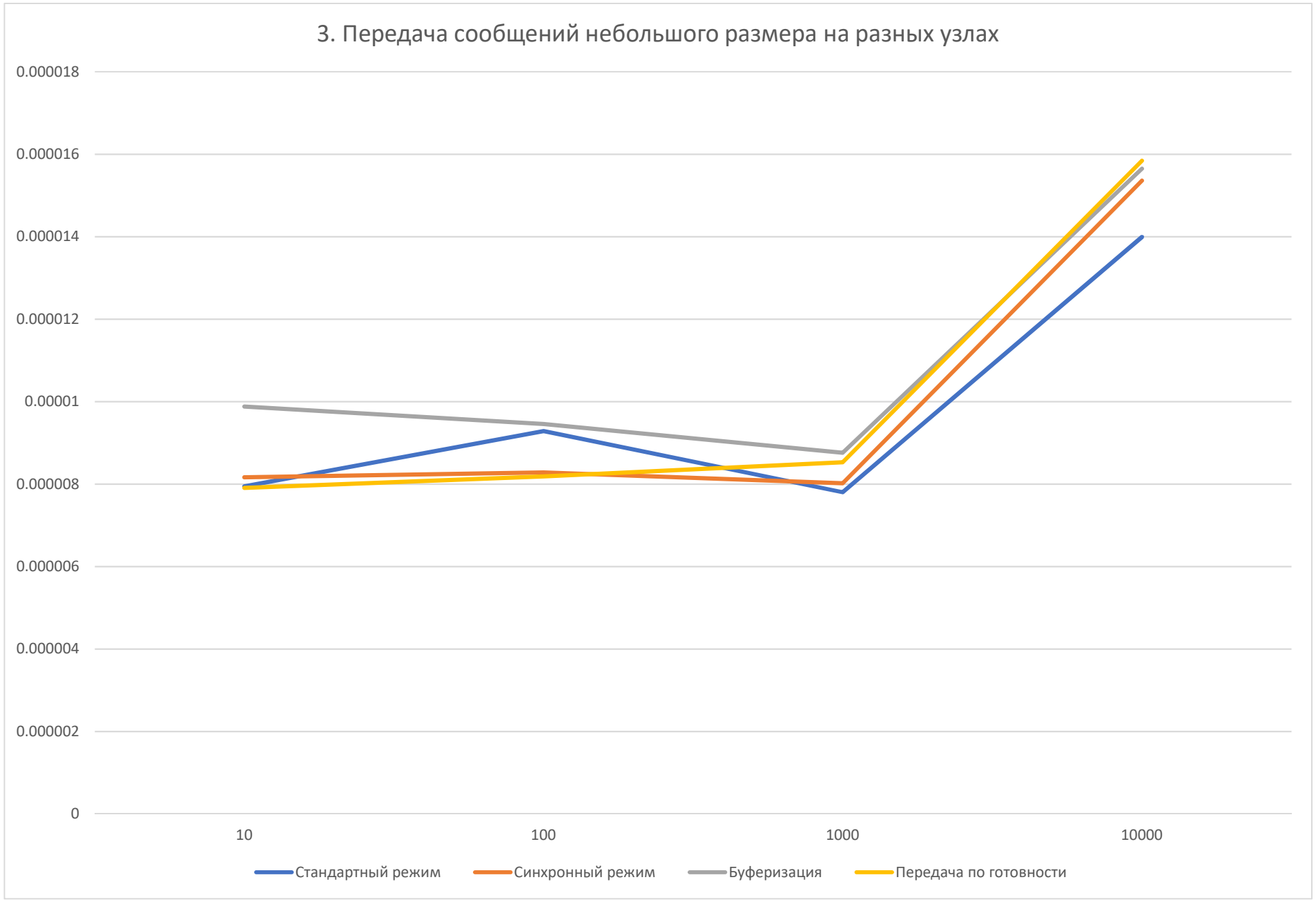

1. Передача сообщений небольшого размера на одном узле
0.0000045
0.000004
0.0000035
0.000003
0.0000025
0.000002
0.0000015
0.000001
0.0000005
0
1
10
100
1000
10000
Стандартный режим
Синхронный режим
Буферизация
Передача по готовности

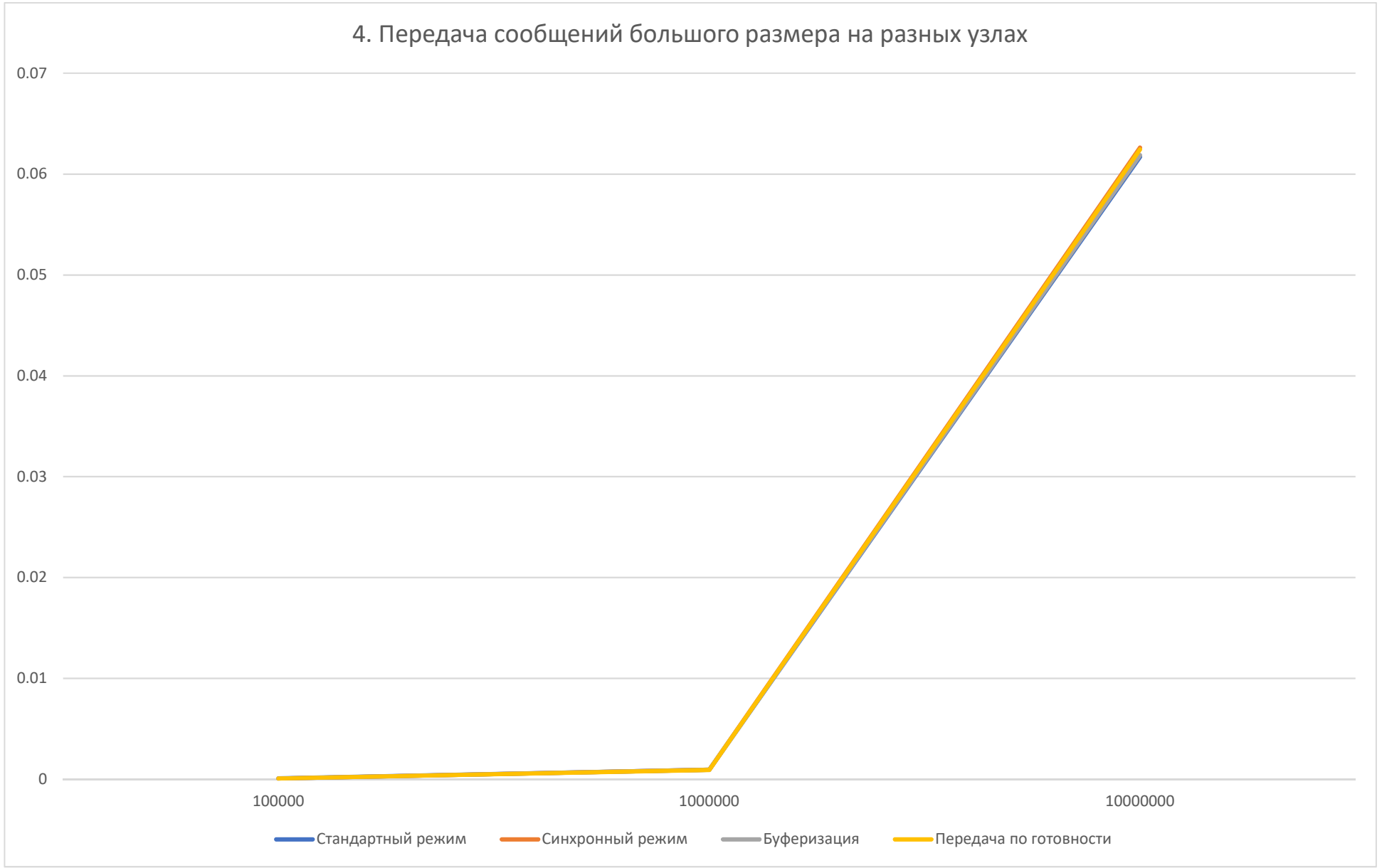
	100000	1000000	10000000
Стандартный режим	1.85E-05	0.000215005	0.003084577
Синхронный режим	1.80E-05	0.00021614	0.003097345
Буферизация	1.75E-05	0.000217326	0.003176892
Передача по готовности	1.82E-05	0.000219096	0.003064304

На разных узлах

Размер сообщения	Стандартный режим	Синхронный режим	Буферизация	Передача по готовности
10	0.000007946233333333089715664992	0.000008165366666666729056345275	0.000009880933333333957374936506	0.000007908833333332651954424887
100	0.000009284466666665871020726504	0.000008283733333333311371561754	0.000009460099999999740239117405	0.000008186699999999241805781480
1000	0.000007801666666667344418205007	0.000008021300000000027708010539	0.000008761233333332701594758050	0.000008527500000000039903766055
10000	0.000013993166666666454961485792	0.000015360100000000784893150821	0.000015650933333332997499619255	0.000015843299999998561311542744
100000	0.000089091366666665947125595959	0.000079805366666667784648035722	0.000080423933333333144581230034	0.000079618800000000783433432017
1000000	0.000958756866666651808538135437	0.000945225999999983268820558102	0.000959202099999995083136339336	0.000923729266666687841465244180
10000000	0.061712974566666529974146016002	0.062626432200000095584790926750	0.061876137899999968106090619813	0.062442425833333287110704645784

Согласно диаграмме 3, при обмене сообщениями небольшого размера между разными узлами наиболее эффективным оказывается стандартный метод Send. Однако для сообщений большого размера эффективнее оказывается использование метода с буферизацией.


3. Передача сообщений небольшого размера на разных узлах
0.000018
0.000016
0.000014
0.000012
0.00001
0.000008
0.000006
0.000004
0.000002
0
10
100
1000
10000
Стандартный режим
Синхронный режим
Буферизация
Передача по готовности


4. Передача сообщений большого размера на разных узлах
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
100000
1000000
10000000
Стандартный режим
Синхронный режим
Буферизация
Передача по готовности

Задание 6. Одновременный прием и передача сообщений

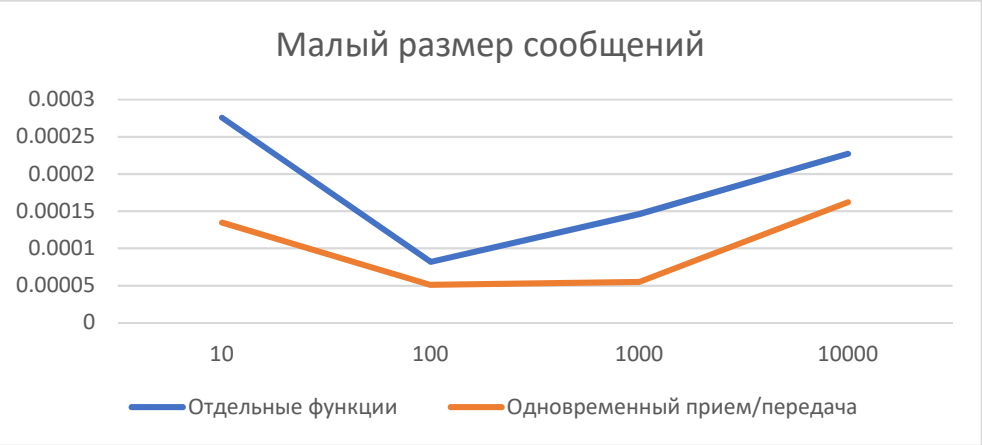
Постановка задачи

Необходимо доработать задание 3, используя одновременный прием и передачу сообщений

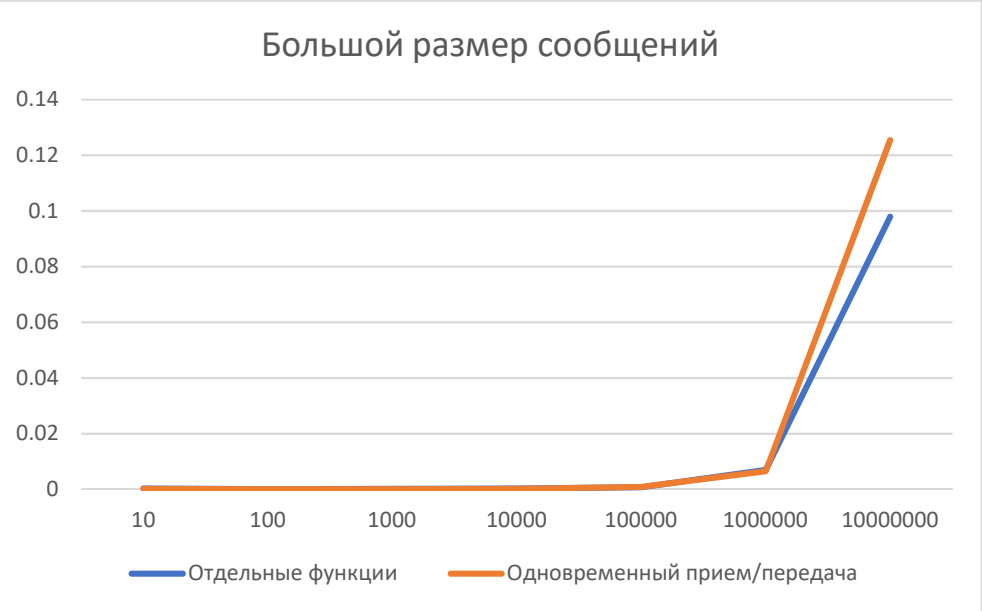
Результаты

Один узел

Размер сообщения	Отдельные функции	Одновременный прием/передача
10	0.00027584816666666666665992890817	0.00013451613333333333358281652670
100	0.00008198739999999999932662282653	0.00005106386666666666738543169796
1000	0.00014637133333333333224040184728	0.00005495403333333331735134012760
10000	0.00022730623333333332373533139381	0.00016222923333333333890901967189
100000	0.00073753006666666545316985104110	0.00087062299999999629768497877440
1000000	0.00696992179999999530173682593670	0.00650385176666666569722044854980
10000000	0.09797273323333334504869895909000	0.12546660509999868484824669226145



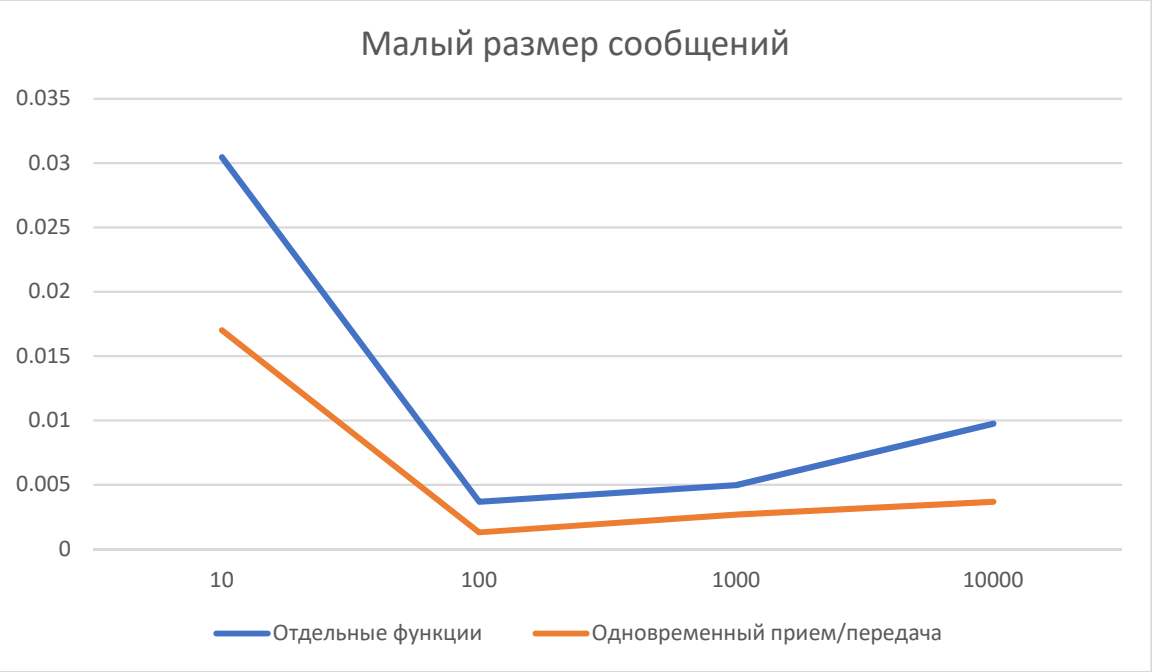
При обмене сообщениями малого размера наиболее эффективным оказывается использование одновременного приема/передачи сообщений.

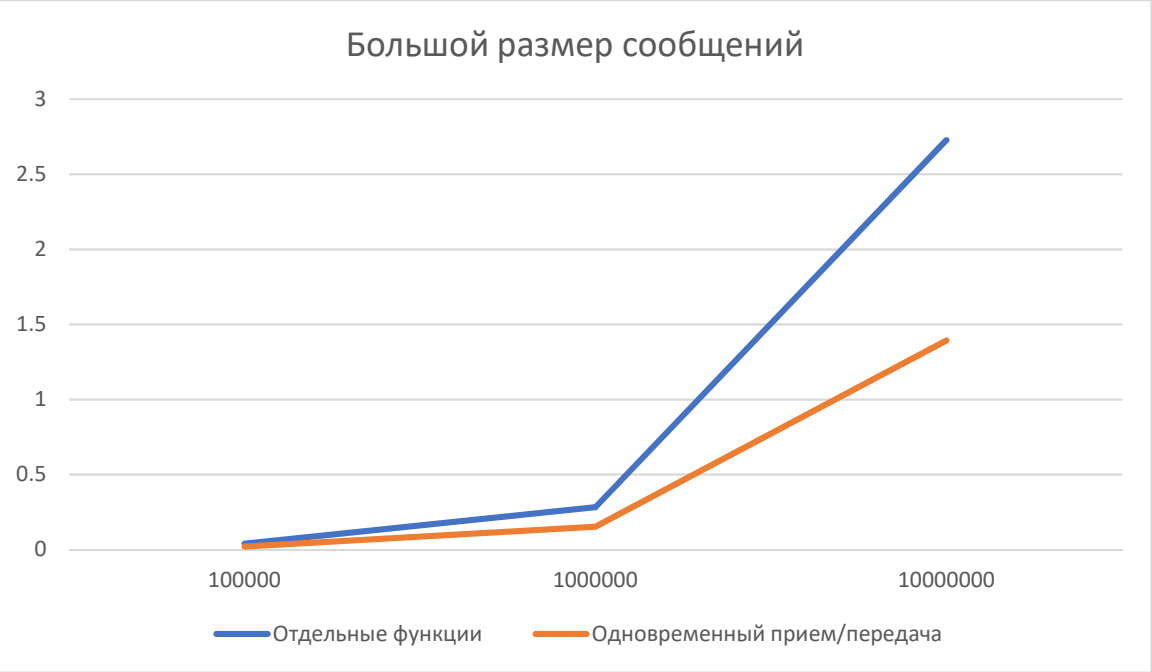


Однако при обмене сообщениями большого размера использование отдельных методов оказывается немного эффективнее.

Два узла

Размер сообщения	Отдельные функции	Одновременный прием/передача
10	0.030457309499999998120278732472	0.017011576133333334431840810907
100	0.003677273633333334127565317218	0.001301332866666665612903619476
1000	0.004965895499999998553841695781	0.002681490466666667463885298517
10000	0.009753945566666651079135164082	0.003692018399999991932342036094
100000	0.039721452733333356688305570970	0.020725167066666667914365262959
1000000	0.283195273600000019253997152191	0.151860061166666647680624180338
10000000	2.727361570599998064068358871737	1.393398906300000739122424420202





При обмене сообщениями между разными узлами независимо от размера сообщения одновременный прием/передача оказываются эффективнее отдельных функций.