УДК 004.383.4

Г. С. Речистов, м. н. с., инженер по программному обеспечению

Лаборатория суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур, факультет радиотехники и кибернетики, Московский физико-технический институт

ЗАО «Интел А/О»

grigory.rechistov@phystech.edu

ИСПОЛЬЗОВАНИЕ ПОЛНОПЛАТФОРМЕННОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СУПЕРКОМПЬЮТЕРНОЙ СИСТЕМЫ ДЛЯ ОПРЕДЕЛЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СЧЁТНЫХ ПРИЛОЖЕНИЙ

В работе предлагается расширение двух существующих моделей описания производительности многопроцессорных систем с целью их применения к кластерным суперкомпьютерам, состоящим из нескольких многопроцессорных узлов. На их основе предлагается способ предсказания производительности и поведения научных приложений на будущих аппаратных вычислительных платформах, использующий компьютерную симуляцию для получения исходных данных.

Ключевые слова: моделирование, суперкомпьютеры, Simics, анализ CPI, анализ узких мест.

Rechistov G. S.

FULL PLATFORM SIMULATION OF SUPERCOMPUTER SYSTEMS AIMED AT PERFORMANCE PREDICTION OF SCIENTIFIC APPLICATIONS

Abstract

Extensions for two existing analytical models of performance of multicore computers into area of multi-host computing clusters are proposed. For both extended models full platform simulation is used to obtain input data in order to predict speed and behavior of scientific applications on future hardware systems.

Key words: simulation, supercomputers, Simics, CPI analysis, bottleneck analysis.

Введение

Одним из основных принципов наращивания доступной производительности вычислений для научных и коммерческих задач является их параллельное исполнение. Он проявляется на всех уровнях организации ЭВМ, начиная от микроархитектуры в виде суперскалярных процессоров, многоядерных узлов и заканчивая объединением в сеть отдельных компьютеров для организации кластера. К сожалению, простое увеличение количества элементов (АЛУ, ядер, машин) практически никогда не приводит к пропорциональному росту полной вычислительной производительности. Причин этому много: необходимость синхронизации удалённых процессов, обращение к ограниченным общим ресурсам и пр. Предсказание поведения приложений на будущих, ещё не построенных системах, представляется крайне важным, так как позволяет заранее обнаружить слабые места проекта и перераспределить средства для получения оптимальной системы. Для этого используется компьютерное моделирование (полноплатформенная симуляция) больших систем на доступных кластерах меньшего масштаба.

В работе описывается расширение двух существующих методик описания производительности, изначально разработанных для многопроцессорных систем, на системы типа «кластер» (группа идентичных ЭВМ, объединённых в сеть) для научных приложений, использующих библиотеку MPI [1]. Затем описывается решение задачи моделирования с помощью Simics [2], позволяющего балансировать точность и скорость исследования. Затем приводятся результаты практического применения методики для анализа существующего проекта серии суперкомпьютеров, развиваемых в Московском физико-техническом институте.

Модель

Описание характеристик моделируемого кластера дано в таблице 1. Для построения и исследования модели был задействован программный симулятор Wind River Simics. Это приложение предоставляет ряд уникальных инструментов, являвшихся жизненно необходимыми для нашего исследования: параллельное исполнение модели на всех процессорах системы, распределение модели на все узлы кластера, скорость исполнения моделируемого приложения, близкая к его скорости на реальной системе, изучение влияния немодифицированной операционной системы.

К сожалению, включение всех деталей работы ядер микропроцессора, устройств оперативной памяти, сетевых маршрутизаторов в модель привело бы к существенному замедлению ее работы и невозможности проведения сколько-нибудь существенного объёма исследований в разумное время. Поэтому нами был использован комбинированный подход, в котором моделирование используется для получения значений ключевых параметров, характеризующих работу приложения внутри симулируемой системы. Эти данные затем используются в аналитических формулах описания производительности многопроцессорных ЭВМ, адаптированных для кластерных систем.

Два способа описания производительности приложения

Существует большое количество работ, посвящённых аналитическому описанию производительности микропроцессоров и многоядерных систем. В них используются два подхода: приведение производительности к усреднённому времени исполнения одной инструкции [3-5], и анализ узких мест в системах передачи данных [6]. Ниже мы приводим адаптацию обоих подходов для кластерных систем.

Анализ среднего времени выполнения одной инструкции

Ключевой метрикой производительности является величина CPI (*cycles per instruction*), показывающая, сколько в среднем тактов процессора необходимо для исполнения одной инструкции. При анализе CPI разбивается на сумму факторов, независимо влияющих на производительность [3, 7]:

$$CPI = CPI_{core} + CPI_{caches} + CPI_{memory} + CPI_{MPI}$$

Здесь CPI_{core} – количество тактов, затрачиваемых на исполнение непосредственно в ядре процессора, CPI_{caches} – задержки при обращениях в кэш-память, CPI_{memory} – задержки при доступе в оперативную память, CPI_{MPI} – коммуникации с помощью интерфейса MPI. Все члены, кроме CPI_{core}, равны математическому ожиданию некоторого класса событий, прерывающих вычисления в ядре. Так, для CPI_{caches}

$$CPI_{caches} = \Sigma P\{i\}L_i$$

Здесь $P\{i\}$ – вероятность i-го типа промаха кэшей (при чтении, записи, для различных уровней кэша), L_i – задержка в тактах для данного события. По сравнению с оригинальной моделью, разработанной для одноядерной системы, описанный подход учитывает эффекты от взаимодействия нескольких ядер через совместное использование памяти и кэшей. Кроме того, новый член CPI_{MPI} учитывает взаимодействие, специфичное для распределённых приложений, использующих библиотеку MPI.

Следующий шаг методики – это учёт того факта, что CPI характеризует быстроту исполнения инструкций безотносительно их природы: в эту величину включаются операции загрузки данных, условных переходов и т.п., непосредственно не связанные с вычислительным алгоритмом приложения. В конечном счёте, нас интересует скорость исполнения операций с числами с плавающей запятой – FLOPS (*floating operations per second*). Не все исполненные инструкции производят собственно вычисления, кроме того, современные векторные инструкции содержат операции над двумя (в случае набора SSE) или четырьмя (для набора AVX) парами операндов. Для получения FLOPS необходимо знать ещё одну характеристику приложения – коэффициент α, показывающий, сколько операций с плавающей точкой в среднем приходится на одну инструкцию. Значение FLOPS получается из следующего соотношения:

$$FLOPS^{-1} = CPI_{core}/\alpha + CPI_{caches} + CPI_{memory} + CPI_{MPI}$$

Анализ узких мест

Эта методика исходит из наблюдения, что производительность систем определяется «узким местом» – одной из подсистем передачи данных с наибольшей утилизацией при исполнении конкретной задачи. Для различных задач и конфигураций аппаратуры лимитирующими факторами будут различные подсистемы. Ниже изложена адаптация методики нахождения пиковой производительности вычислений с плавающей запятой (FLOPS) [5].

Мы выделяем и анализируем три возможные точки образования «узкого места» – кэш-память, оперативную память и сеть передачи сообщений между узлами. Для каждой из них введём характеристику c – удельную интенсивность операций, равную среднему количеству операций над числами с плавающей запятой, отнесённому к одному переданному байту. Эта величина будет различна для кэша, памяти и сети; например, в ОЗУ попадают только те запросы данных, которые не были удовлетворены системой кэшей. Вторая характеристика – это пропускная способность каждой подсистемы X, измеряемая в байт/сек. Последняя величина – это пиковая производительность

вычислительного ядра T в случае, если ни одна из подсистем её не ограничивает. Производительность приложения при условии, что узким местом является подсистема i, определяется из соотношения

$FLOPS_i = \min(cX, T)$

Если в системе несколько потенциальных узких мест, то необходимо выбрать минимальное из них (рис. 1)

$FLOPS = \min(FLOPS_1, FLOPS_2, FLOPS_3)$

Описание эксперимента

В таблице 2 приведены характеристики кластера, использовавшегося для проведения исследований. Как видно из сравнения с таблицей 1, мы имели примерно в 9 раз меньше физических вычислительных ядер по сравнению с количеством моделируемых. Тем не менее, скорость моделирования оказалась достаточной для проведения серии экспериментов (наблюдаемое замедление скорости загрузки операционной системы было в пределах от 10 до 40 раз). Были исследованы конфигурации, включающие в себя от 16 до 112 узлов модели. В качестве изучаемых приложений были использованы пакеты молекулярной динамики Gromacs [8] и Amber [9], компилируемые с помощью GCC 4.4.5. В качестве библиотеки MPI использовалась MPICH2-1.4.p1.

Источники численных значений, входящих в формулы

Для получения модельных значений CPI_{caches}, CPI_{memory}, CPI_{MPI} в симуляторе были использованы следующие механизмы: подключаемая трёхуровневая иерархия кэшей, модель неоднородной (*non uniform memory access, NUMA*) памяти и инструментация приложения для обнаружения исполнения и учёта задержек всех функций MPI. Значение CPI_{core} затруднительно получить без использования медленной потактовой симуляции, поэтому оно бралось из экспериментов на реальной аппаратуре [10]. В этом эксперименте также определялось значение коэффициента α.

Величина T («потолок» производительности) находится из результатов запуска программы High Performance Linpack [11]. Значения пропускных способностей X берутся из тестов: для ОЗУ – STREAM [12], для кэшей – LMBench [13], для сетевой подсистемы – Netperfmeter [14]. Значения c находятся из экспериментов на модели путём подсчёта количеств соответствующих событий передачи данных.

Результаты

Из-за ограничений объёма ниже приведены только результаты экспериментов по определению характеристик системы памяти и анализа хода исполнения изучаемых приложений.

Часть результатов была получена и описана в предыдущих работах. Порядок нахождения и результаты измерения *CPIcore* описаны в [7]. Результаты измерения производительности сети и пиковой вычислительной производительности High Performance Linpack, полученные для изучаемой модели и для используемой физической системы, даны в [10].

На рис. 2 показаны измерения пропускной способности иерархии кэшей и оперативной памяти, измеренные на реальной аппаратуре и используемые затем в методе анализа узких мест.

На рис. 3 приведён результат одного из экспериментов по определению средней задержки передачи MPI сообщения L_i, используемый в подходе, связанном с анализом длительности средней инструкции. На рис. 4 показано снятое на модели распределение

относительных частот (вероятностей $P\{i\}$) для отдельных вызовов MPI для программы **mdrun** из пакета Gromacs. Видно, что в своей активной фазе это приложение активно использует лишь небольшое количество типов коммуникаций. На рис. 5 показаны средние расстояния между двумя MPI-вызовами для одного из запусков **mdrun.**

Выводы и направления последующей работы

Использование компьютерного моделирования для получения сведений о характере работы приложения на ещё не построенной вычислительной системе предоставляет исследователю возможность поиска оптимальной конфигураций аппаратуры и наилучшего режима работы приложения. При этом решающим фактором является возможность баланса усилий и времени, потраченных на конфигурирование модели, на проведение непосредственно экспериментов и измерений и на точность получаемых из неё данных. Из-за масштаба задачи сам процесс моделирования не является тривиальным, однако современные программные решения позволяют выполнить его на доступном оборудовании.

Дальнейшая работа включает в себя следующие направления:

- Сбор данных для большего числа прикладных приложений на симуляции в целях предсказания их производительности.
- Уточнение существующих методик путём введения поправок, учитывающих при работе программ второстепенные процессы. Одним из подобных эффектов является неоднородность скорости исполнения программы на разных потоках/узлах. Этот факт был обнаружен при анализе распределения, показанного на рис. 5, из которого видно, что все процессы делятся на три группы по характерной средней длине интервала между двумя MPI-вызовами.
- Верификация результатов, полученных на моделях, с помощью измерений, проведённых на реальной системе, определение достигнутой точности и причин расхождений.

Список литературы

1. **MPI: A Message-Passing Interface Standard**. Version 2.2. [Электронный ресурс] // <http://www.mpi-forum.org/docs/docs.html> (16.04.2012).
2. **Magnusson P. S., Christensson M. et al.** Simics: A Full System Simulation Platform // Computer. 2002. Т. 35-2. Pp. 50–58.
3. **Simonson L.J., He L**. Micro-architecture Performance Estimation by Formula // SAMOS'05. 2005. Pp. 192–201.
4. **Eyerman S., Eeckhout L., Karkhanis T., Smith J. E**. A mechanistic performance model for superscalar out-of-order processors // ACM Transactions on Computer Systems. 2009. Т. 27(2). Pp. 3:1–3:37.
5. **Sorin D. J., Lemon J. L., Eager D. L., Vernon M. K.** Analytic Evaluation of Shared-Memory Architectures // IEEE Transactions on Parallel and Distributed Systems. 2003. Т. 14. P. 180.
6. **Williams S. W., Waterman A., Patterson D. A.** Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures // Tech. rep. UCB/EECS-2008-134. EECS Department, University of California, Berkeley. 2008.<http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-134.html> (16.04.2012)
7. **Речистов Г. С., Иванов А. А., Шишпор П. Л., Пентковский В. М.** Симуляционный подход для нахождения производительности параллельных MPI-приложений на вычислительном кластере // Труды 54 научной конференции МФТИ «Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе». 2011. С. 82–83.
8. **Van Der Spoel D. et al.** GROMACS: Fast, Flexible, and Free // Journal of Computational Chemistry. 2005. Т. 26. № 16. Pp. 1701–1718.
9. **Case D. A. et al.** Amber 11 Users' Manual // University of California. 2010.
10. **Речистов Г. С., Иванов А. А., Шишпор П. Л., Пентковский В. М.** Моделирование компьютерного кластера на распределённом симуляторе. Верификация моделей вычислительных узлов и сети кластера. [Электронный ресурс] // Конференция «Разработка ПО 2011» CEE-SECR <http://2011.secr.ru/lang/ru-ru/talks/modeling-of-a-computer-cluster-on-a-distributed-simulator> (18.04.2012)
11. **Dongarra J. J.** Performance of Various Computers Using Standard Linear Equations Software [Электронный ресурс] // <ftp://netlib2.cs.utk.edu/benchmark/performance.pdf> (18.04.2012)
12. **McCalpin John D**. STREAM: Sustainable Memory Bandwidth in High Performance Computers [Электронный ресурс] <http://www.cs.virginia.edu/stream> (18.04.2012)
13. **McVoy L., Staelin C.** Lmbench: portable tools for performance analysis // Proceedings of the 1996 annual conference on USENIX Annual Technical Conference. San Diego, CA: USENIX Association, 1996. С.23–28.
14. **Dreibholz T.** Netperfmeter: A TCP/UDP/SCTP/DCCP Network Performance Meter Tool [Электронный ресурс] <http://www.iem.uni-due.de/~dreibh/netperfmeter> (18.04.2012)

Таблица 1. Моделируемая система

Процессор	Intel Xeon E5 (Sandy Bridge) 2.8 ГГц
Ядер в процессоре	8
Количество процессоров в узле	2
Количество узлов	112
ОЗУ на узел	48 Гбайт
Сеть	Infiniband QDR 10 Гбит/с
Общее количество ядер	**1792**
Общий объём ОЗУ, Гбайт	**5376**

Таблица 2. Использованная реальная система

Процессор	Intel Xeon E5680 (Westmere) 3,3 ГГц
Ядер в процессоре	6
Количество процессоров в узле	2
Количество узлов	16
ОЗУ на узел	24 Гбайт
Сеть	Infiniband QDR 10 Гбит/с
Общее количество ядер	**192**
Общий объём ОЗУ, Гбайт	**384**

Список подрисуночных подписей

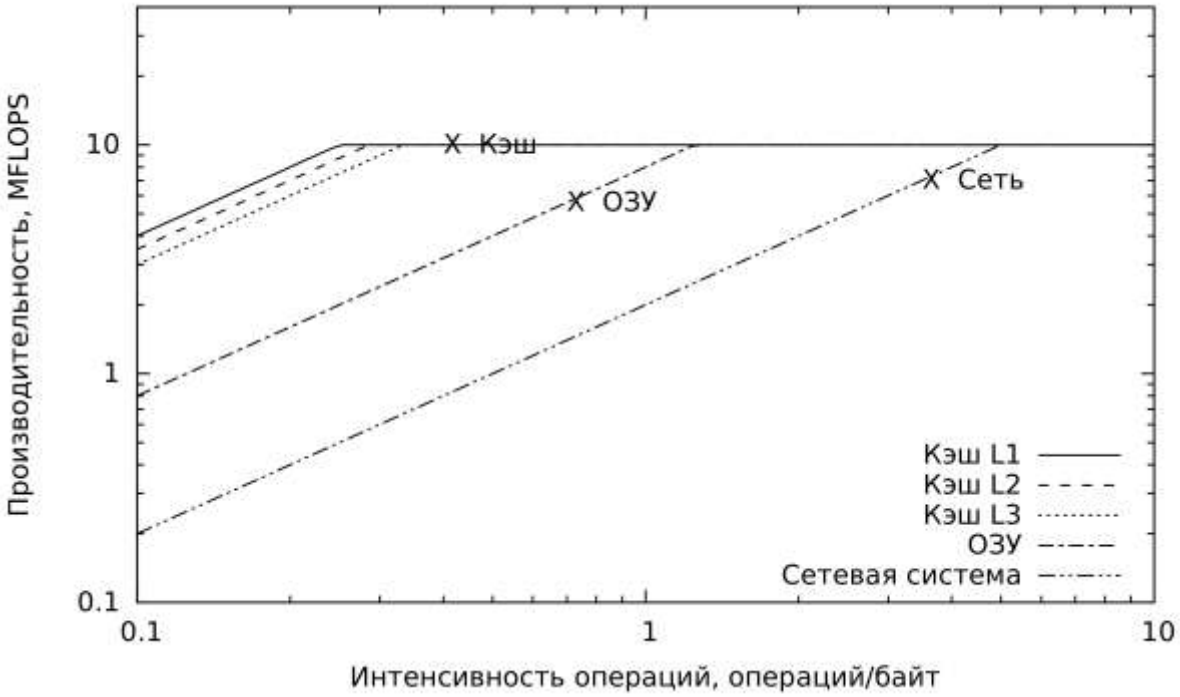
Рис. 1. Зависимости производительности приложения от интенсивности использования для кэш-памяти, оперативной памяти и сети. Наклон графиков в их левой части определяется пропускной способностью соответствующего канала; общая «крыша» графика определена максимальной производительностью ядра. Итоговая скорость системы характеризуется наименьшей из трёх величин (в данном примере – ОЗУ).

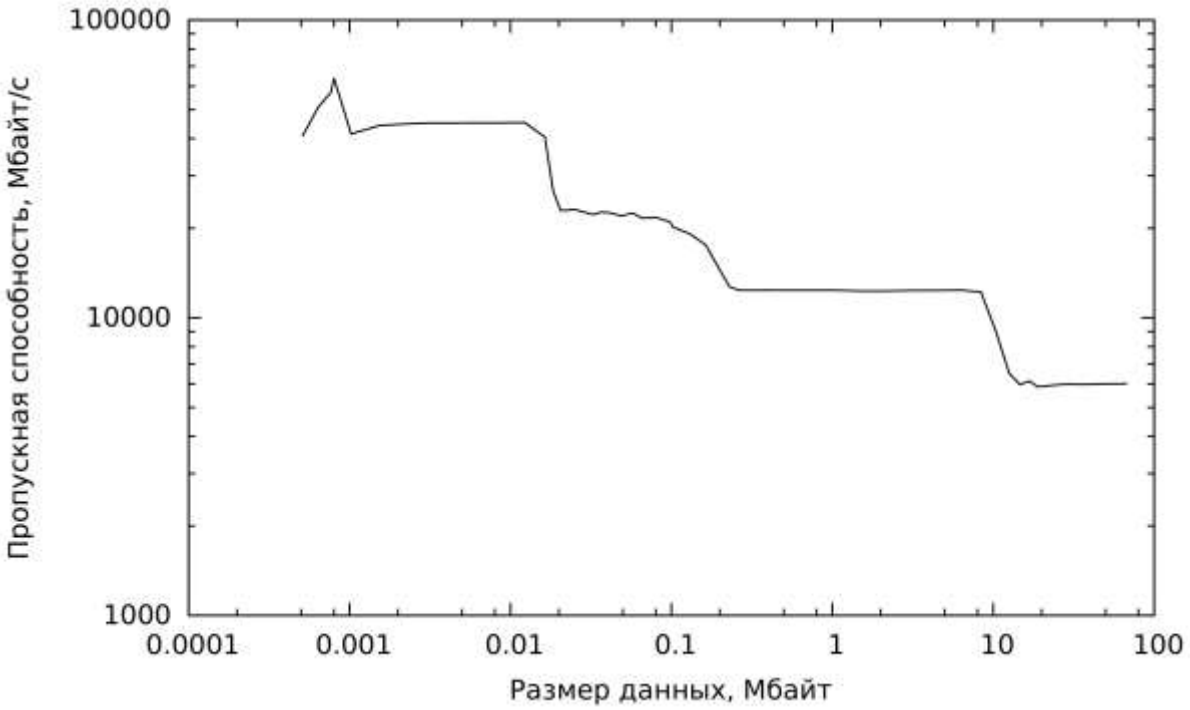
Рис. 2. Результаты измерения пропускной способности компонент подсистемы памяти (кэшей L1, L2, L3, и ОЗУ) для реальной аппаратуры, полученные с помощью бенчмарка **Lmbench** на реальной аппаратуре. Каждый горизонтальный участок зависимости соответствует своему уровню иерархии памяти.

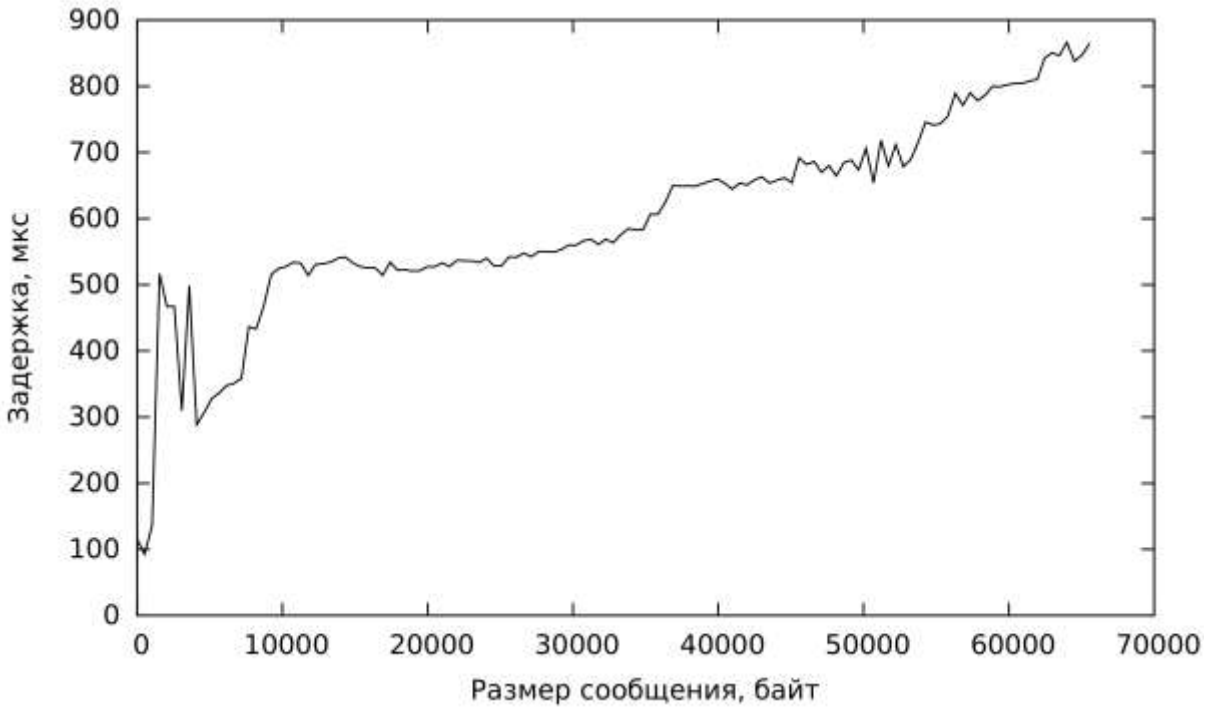
Рис. 3. Зависимость задержки сетевых коммуникаций MPI типа «точка-точка» от размера пакета для изучаемого кластера.

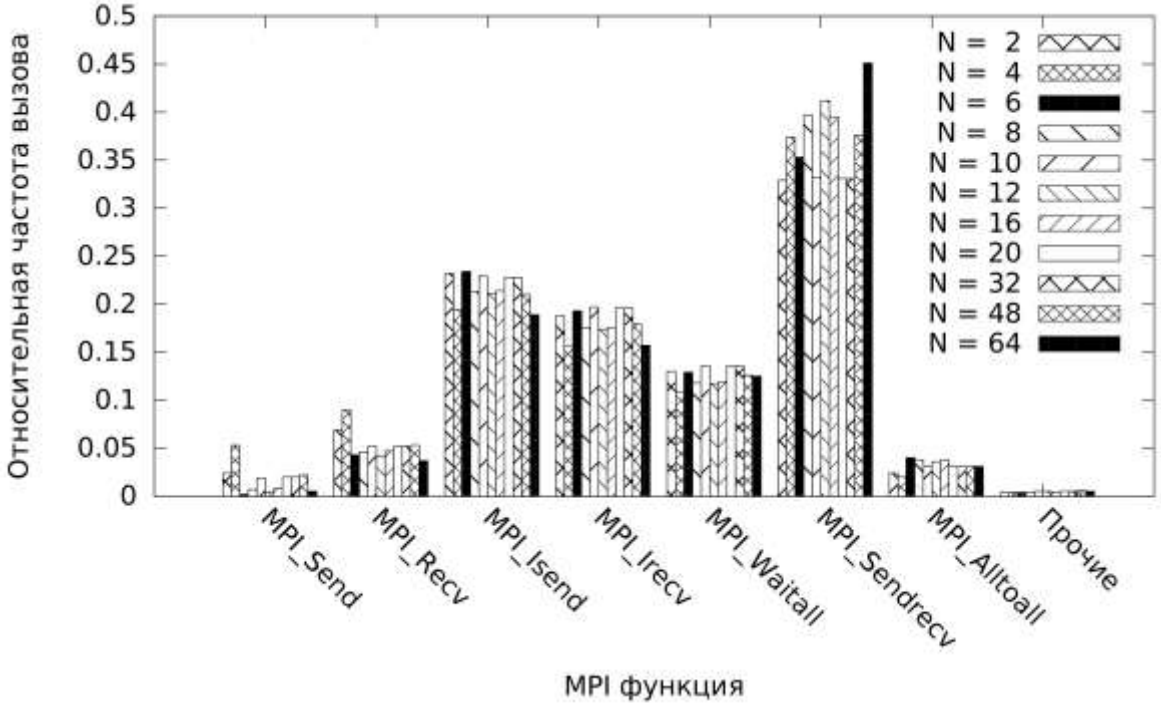
Рис. 4. Распределение частот отдельных MPI-вызовов для разного количества моделируемых узлов (N) для приложения **mdrun** из пакета Gromacs.

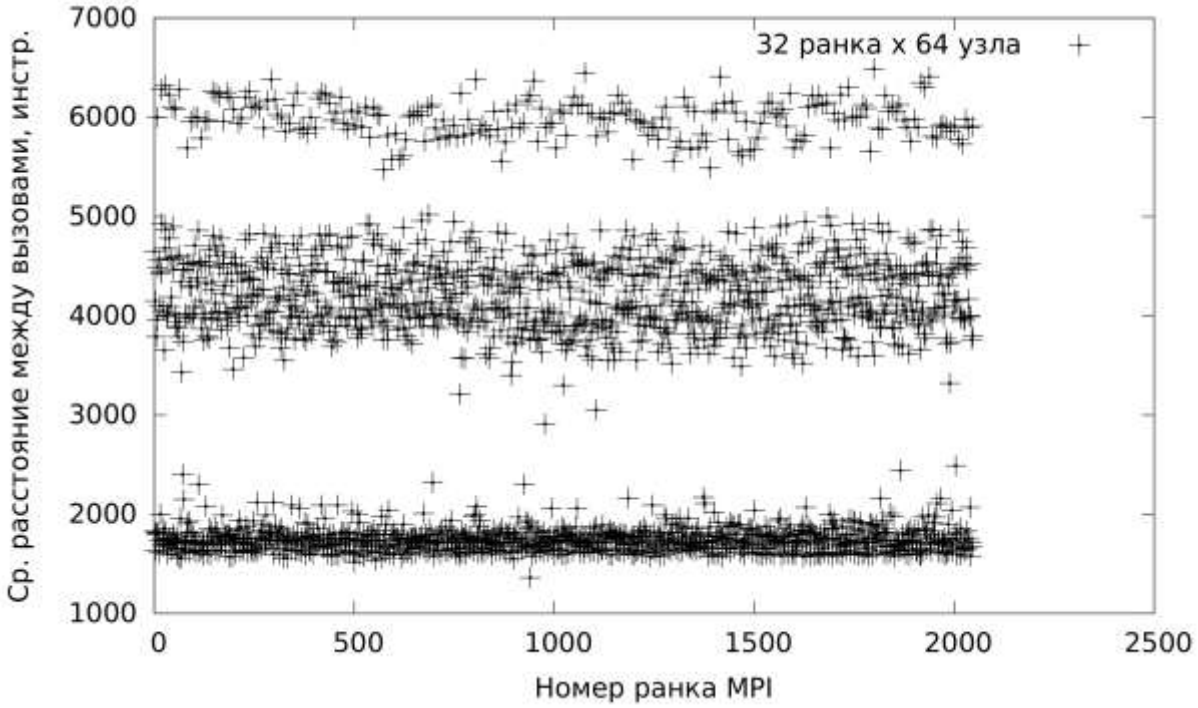
Рис. 5. Зависимость среднего расстояния между вызовами двух MPI-функций от порядкового номера подпроцесса (*англ.* rank, ранк) при симуляции исполнения **mdrun** на N=64 узлах (2048 потоков MPI).


Производительность, MFLOPS
10
1
0.1
0.1
1
10
Интенсивность операций, операций/байт
Кэш
ОЗУ
Сеть
Кэш L1
Кэш L2
Кэш L3
ОЗУ
Сетевая система


100000
10000
1000
Пропускная способность, Мбайт/с
0.0001
0.001
0.01
0.1
1
10
100
Размер данных, Мбайт


900
800
700
600
500
400
300
200
100
0
Задержка, мкс
0
10000
20000
30000
40000
50000
60000
70000
Размер сообщения, байт


Относительная частота вызова
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
N = 2
N = 4
N = 6
N = 8
N = 10
N = 12
N = 16
N = 20
N = 32
N = 48
N = 64
MPI_Send
MPI_Recv
MPI_Isend
MPI_Irecv
MPI_Waitall
MPI_Sendrecv
MPI_Alltoall
Прочие
MPI функция


32 ранка x 64 узла
Ср. расстояние между вызовами, инстр.
Номер ранка MPI
7000
6000
5000
4000
3000
2000
1000
0
500
1000
1500
2000
2500