УДК 004.421.2; 519.612.2

ОБ ОСОБЕННОСТЯХ РЕШЕНИЯ БОЛЬШИХ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ НА МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ РАЗЛИЧНОЙ АРХИТЕКТУРЫ

Б. И. Краснопольский[1]

Представлены результаты тестирования набора итерационных методов подпространств Крылова (CGS, BiCGStab) с алгебраическим многосеточным предобусловливателем для решения больших сильно разреженных систем линейных алгебраических уравнений. Получены характеристики масштабируемости для MPI- и гибридной реализаций этих методов на трех вычислительных системах. Обсуждаются особенности использования реализованного набора методов на многопроцессорных вычислительных системах, построенных на базе процессоров Intel Harpertown, Intel Nehalem и AMD Magny-Cours. Статья рекомендована к публикации Программным комитетом Международной научной конференции "Параллельные вычислительные технологии" (ПАВТ–2011; http://agora.guru.ru/pavt2011).

Ключевые слова: итерационные методы, системы линейных алгебраических уравнений, масштабируемость.

1. Введение. Решение больших систем линейных алгебраических уравнений $Ax = b$ является одной из традиционных и широко распространенных задач для многопроцессорных вычислительных систем. Например, такие матрицы возникают при разностной аппроксимации дифференциальных уравнений в частных производных на больших расчетных сетках. Зачастую шаблон аппроксимации, а как следствие, и количество ненулевых элементов в строке матрицы, не превышает нескольких десятков элементов, тогда как порядок матрицы может достигать 10^8 и более неизвестных. Это приводит к необходимости использования специализированных форматов хранения данных, таких как CSR (Compressed Sparce Row) или DMSR (Distributed Modified Sparce Row), и выбора соответствующих численных методов, которые могут быть реализованы для такого рода форматов данных. В общем случае наиболее распространенными численными методами для таких задач являются итерационные методы подпространств Крылова с неполным LU [1] или многосеточным [2] предобусловливателями, которые могут успешно применяться на вычислительных системах с распределенной памятью.

Настоящая статья является продолжением работ [3, 4] по созданию библиотеки эффективных методов решения больших систем линейных алгебраических уравнений. В работе [4] основное внимание было уделено исследованию результатов масштабируемости MPI- и MPI+ShM-реализаций переупорядоченного метода BiCGStab [3] с алгебраическим многосеточным предобусловливателем на вычислительных платформах СКИФ МГУ "Чебышёв" и "Ломоносов". Здесь основной акцент ставится на обсуждении особенностей использования таких систем (библиотеки MPI, схемы привязки вычислительных процессов к физическим ядрам и др.) на трех вычислительных системах, построенных на базе процессоров Intel Harpertown, Intel Nehalem и AMD Magny-Cours.

2. Тестовые задачи. В качестве тестовой матрицы был использован конечно-разностный аналог уравнения Пуассона на равномерной расчетной сетке в кубической области (7-точечный шаблон аппроксимации, размер сетки 8 миллионов ячеек). Для решения этой системы уравнений использовался итерационный метод BiCGStab с алгебраическим многосеточным предобусловливателем. Оптимальное разбиение между вычислительными процессами и переупорядочивание строк матрицы строилось с помощью свободно распространяемой библиотеки PT-Scotch [5].

3. Тестовые платформы. Основные характеристики использованных в работе аппаратных платформ, реализации библиотек MPI и версии компиляторов приведены в табл. 1 и 2.

4. Результаты. В настоящей статье не обсуждаются алгоритмические особенности скорости сходимости рассматриваемых методов, но только вопросы программной реализации. В зависимости от количества

[1] Научно-исследовательский институт механики Московского государственного университета им. М. В. Ломоносова, Мичуринский просп., д. 1, 119192, Москва; науч. сотр., e-mail: krasnopolsky@imec.msu.ru

Таблица 1

Аппаратные характеристики тестовых платформ

Название	“Чебышёв”	“Ломоносов”	“Зилант”
Тип процессора	Intel Harpertown	Intel Nehalem	AMD Magny-Cours
Модель процессора	E5472	X5570	Opteron 6174
Количество процессоров	2 (8 ядер/узел)	2 (8 ядер/узел)	2 (24 ядра/узел)
Тактовая частота	3.0 ГГц	2.93 ГГц	2.2 ГГц
Размер кэш-памяти	L2 12 МБ	L3 8 МБ	L3 12 МБ
Оперативная память	8 ГБ	12 ГБ	64 ГБ
Частота и тип памяти	DDR2 1333 МГц	DDR3 1333 МГц	DDR3 1333 МГц
Пропускная способность	12.8 ГБ/сек	32 ГБ/сек на проц.	42.7 ГБ/сек на проц.
Внутриузловой интерконнект	—	Quick Path	HyperTransport 3
Межузловой интерконнект	IB DDR	IB QDR	IB QDR
Принадлежность	Суперкомпьютерный комплекс МГУ		ЗАО “Т-Сервисы”

Таблица 2

Библиотеки MPI и компиляторы

Вычислительная система	Библиотека MPI	Компилятор
“Чебышёв”	MVAPICH 1.1	Intel Compiler 11.1
	Open MPI 1.4	GNU Compiler Collection 4.1.2
“Ломоносов”	Open MPI 1.4.3	Intel Compiler 12.0
“Зилант”	MVAPICH 1.2	Intel Compiler 12.0
	Open MPI 1.4.1	Intel Compiler 12.0
	Platform MPI 8.0	Intel Compiler 12.0

используемых вычислительных процессов может несколько варьироваться необходимое количество итераций метода до достижения критерия сходимости итерационного процесса. Ввиду этого, чтобы упростить анализ полученных результатов и избежать разброса данных, вызванного флуктуациями количества выполненных итераций, в ходе тестирования проводился расчет фиксированного количества итераций.

4.1. Масштабируемость внутри одного узла.

4.1.1. MPI-модель. Для MPI-модели реализации программы были исследованы два случая распределения процессов по физическим ядрам процессоров: когда вычислительные процессы располагаются на ядрах только одного процессора и когда равномерно распределяются между обоими процессорами в узле. В этом случае на всех вычислительных системах в тестах использовалась библиотека Open MPI.

Масштабируемость внутри одного процессора на вычислительной системе “Чебышёв” оказывается достаточно слабой (рис. 1а): переход от одного к четырем ядрам сопровождается незначительным монотонным ростом, а ускорение в итоге не превышает 1.4 раза. Использование двух ядер с двумя процессорами дает ускорение в 1.8 раза, однако дальнейшее наращивание количества вычислительных процессов на узле сопровождается столь же слабым приростом ускорения: для 8 ядер оно составляет всего 2.5 раза. В целом, столь низкие результаты масштабируемости для этой платформы представляются ожидаемыми ввиду наличия общего контроллера памяти для обоих процессоров и низкой скорости доступа к памяти. Выбранное же тестовое приложение реализует существенно неравномерный доступ к памяти (используемые форматы хранения разреженных матриц порождают двойную индексацию при обращении к данным, причем эти данные могут обладать достаточно слабой временно́й и пространственной локальностью), что приводит к большим задержкам при обращении к памяти.

Существенно лучшими оказываются результаты масштабируемости на системе “Ломоносов” (рис. 1б; тесты были проведены на вычислительном узле с оперативной памятью размером 24 ГБ). Ускорение внутри одного процессора составляет 3.3 раза для 4 ядер, что в сравнении с процессором E5472 демонстрирует почти трехкратный прирост. Перераспределение процессов на вычислительных ядрах, расположенных

физически на разных процессорах, также оказывается более эффективным, однако если для процессоров на системе “Чебышёв” такое перераспределение давало ощутимый прирост производительности, то здесь выигрыш оказывается в пределах 1–2%. Монотонность роста, хотя и с некоторым замедлением, наблюдается вплоть до 8 ядер и достигает 4.8 раза, что почти вдвое лучше результатов, полученных на системе “Чебышёв”. На вычислительных узлах системы “Ломоносов” с 12 ГБ оперативной памяти время работы последовательной программы оказывалось в среднем на 15–20% больше. Это обусловлено тем, что в узле с 12 ГБ памяти тестовая задача полностью умещается в памяти одного NUMA-узла, в то время как в этом случае задействуется и вторая область памяти, доступ к которой реализуется через QPI и контроллер памяти второго процессора.

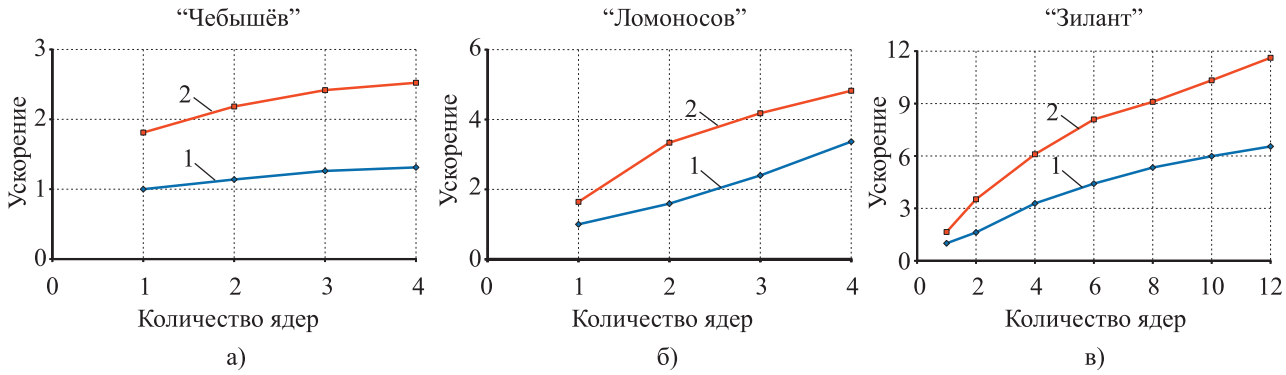

Рис. 1. Графики зависимости масштабируемости MPI-версии приложения от количества используемых ядер с каждого из процессоров внутри одного вычислительного узла: а) СКИФ МГУ “Чебышёв”, б) “Ломоносов”, в) “Зилант”; 1) вычислительные процессы выделяются на одном процессоре, 2) вычислительные процессы распределяются между двумя процессорами

Схожая с системой “Ломоносов” картина масштабируемости наблюдается и на одном узле системы “Зилант” (рис. 1в). Ускорение на одном процессоре для 12 ядер составляет 6.5 раз, а для 24 ядер и двух процессоров — 11.6 раза.

Если говорить об абсолютных величинах времени расчета тестовой задачи, то эти показатели на одном ядре системы “Чебышёв” и на одном ядре системы “Зилант” совпали с точностью до 1–2% и составили 156 секунд несмотря на разницу тактовой частоты процессоров в 800 МГц. Что интересно, полученные на системе “Чебышёв” времена решения задачи получились практически одинаковыми как для компилятора Intel, так и для GNU-компилятора. Время расчета на одном ядре системы “Ломоносов” оказалось заметно лучше и составило 77 секунд.

С точки зрения абсолютных времен, при использовании одного вычислительного узла наиболее эффективным на данном тесте оказывается узел на базе процессоров AMD Magny-Cours, для которого время расчета задачи составляет 14.1 секунды. Близкие результаты наблюдаются и для системы “Ломоносов”: 16.1 секунды. В свою очередь, вычислительный узел на базе процессоров E5472 демонстрирует куда худшие результаты: время расчета задачи оказывается почти в 4 раза больше — 62 секунды.

4.1.2. Гибридная модель. На следующем этапе работы было проведено исследование эффективности гибридной версии программы, когда обмены между процессами внутри одного узла реализуются через общую память вычислительного узла. В этом случае на узле использовались все доступные физические ядра. В тестах было рассмотрено несколько случаев, когда варьировалось количество процессов, которые объединялись через общую память.

Как и можно было ожидать, для одного узла системы “Чебышёв” увеличение количества процессов, взаимодействующих через общую память, приводит к незначительному, но улучшению масштабируемости (табл. 3). Полностью противоположная ситуация наблюдается для одного узла системы “Ломоносов”: оптимальным оказывается использование 8 независимых MPI-процессов. В свою очередь, на системе “Зилант” минимальные времена достигаются для блоков размером 3–6 процессов при условии привязки этих процессов к физическим ядрам одного процессора и, как следствие, к одному NUMA-узлу, а при дальнейшем увеличении размера блоков наблюдается резкое ухудшение результатов.

Для пояснения полученных результатов масштабируемости необходимо сказать несколько слов об особенностях реализации гибридной модели. С целью экономии оперативной памяти на узлах в этой версии программы имеются достаточно большие фрагменты данных, расположенные в общей для всех процессов памяти (локальные фрагменты матриц на вычислительных узлах). Изначально протестированный

на системе “Чебышёв”, такой подход на UMA-архитектуре не выявил существенных недостатков с точки зрения скорости доступа к памяти, при этом потребовав относительно небольших затрат на реализацию. Однако, как показали проведенные тесты, для NUMA-архитектуры такой подход неудачен. Это вызвано тем, что несколько процессов с разных процессоров вынуждены обращаться в общую память, которая распределена между областями памяти обоих NUMA-узлов. Это порождает большое количество обращений к памяти “чужого” NUMA-узла через межпроцессорный интерконнект (QPI или HyperTransport), что и приводит к значительному падению производительности. Наиболее наглядно этот эффект наблюдается на вычислительной системе “Зилант” для блоков по 24 процесса.

Таблица 3

Время расчета задачи при использовании гибридной версии кода

Вычислительная система	Версия MPI	Размер блоков							
		1	2	3	4	6	8	12	24
“Чебышёв”	Open MPI	62	61.74	—	61.25	—	60.43	—	—
“Ломоносов”	Open MPI	16.13	17.04	—	22.6	—	21.1	—	—
“Зилант”	Open MPI	14.13	12.92	13.06	—	13.89	—	23.88	49.19
	MVAPICH	14.53	—	12.87	—	12.77	—	24.3	48.86
	Platform MPI	14.06	—	12.75	—	13.52	—	24.74	48.97

В целом достаточно хорошие результаты масштабируемости для MPI-версии кода в сравнении с гибридной моделью можно объяснить еще двумя соображениями. Во-первых, увеличение количества MPI-процессов приводит к разбиению матриц на блоки, что несколько улучшает локализацию данных и, как следствие, эффективность доступа к памяти. Во-вторых, большинство реализаций библиотеки MPI на системном уровне также реализует обмен сообщениями внутри узла через общую память, что в данном случае нивелирует выигрыш от применения гибридной модели.

4.2. Межузловая масштабируемость.

4.2.1. MPI модель. На следующем этапе работ была исследована масштабируемость MPI-версии кода в зависимости от количества вычислительных узлов и задействованных ядер.

Графики зависимости ускорения от количества вычислительных узлов на системе “Чебышёв” приведены на рис. 2а. В данном случае использование всех 8 ядер на узле оказывается бесполезным даже для малого количества узлов. Крайне незначительный выигрыш по отношению к 4 ядрам, сравнимый с погрешностью измерений, имеется только для 1–4 узлов. Пиковые значения ускорения в 74 раза достигаются для 1 и 2 ядер на 128 узлах. При этом для двух вычислительных процессов прирост ускорения к этому моменту существенно замедляется и кривая уже близка к насыщению, тогда как график для одного вычислительного процесса на узел демонстрирует почти линейный рост. Минимальное время расчета задачи в этом случае составляет 2.12 секунды.

Расчеты на системе “Ломоносов” показали, что максимальные ускорения порядка 65 раз достигаются при использовании 192 узлов и только одного вычислительного процесса на узле (рис. 2б). Для MPI-версии кода это представляется вполне ожидаемым, поскольку в таком режиме каждый процесс “монопольно” использует коммуникационную сеть узла. Однако при ограниченном количестве ресурсов ситуация оказывается не столь однозначной: в зависимости от количества доступных узлов наиболее эффективным может оказаться использование и большего количества ядер. Так, для 1–4 узлов оптимально использовать 8 ядер, для 8–32 узлов — 4 ядра, для 48–112 узлов — 2 ядра и только для 128 и более узлов эффективным оказывается использование лишь одного ядра. Минимальное время расчета задачи в этом случае составляет 1.18 секунды.

Относительные результаты масштабируемости на системе “Ломоносов”, несмотря на более быструю коммуникационную сеть, оказываются даже несколько хуже, чем на системе “Чебышёв”. Однако этот факт объясняется тем, что непосредственно сами вычисления на процессорах X5570 проходят почти вдвое быстрее (если судить по времени расчета последовательных запусков), тем самым оставляя существенно меньше времени “маскировки” MPI-коммуникаций между узлами.

Графики масштабируемости тестовой задачи на вычислительной системе “Зилант” для библиотек MPI MVAPICH и Platform MPI приведены на рис. 2в и 2г соответственно. Для библиотеки MVAPICH лучшие результаты достигаются при использовании 16 ядер на узлах. Пиковое ускорение в этом случае достигает 70 раз. Близкие значения также получены и для четырех MPI-процессов, приходящихся на один узел. При

этом ускорение в 67 раз достигается для максимального количества доступных узлов, а до этого момента зависимость ускорения от количества узлов оказывается близкой к линейной. Это позволяет предполагать, что при дальнейшем увеличении узлов текущее пиковое значение может быть существенно превышено. В целом, согласно приведенным данным, наблюдается достаточно типичная картина поведения графиков масштабируемости. Существенно выбиваются из этой картины лишь две кривые для 8 и 12 процессов. Начиная с 20 узлов графики ускорения демонстрируют значительные колебания, причем эти колебания носят регулярный характер.

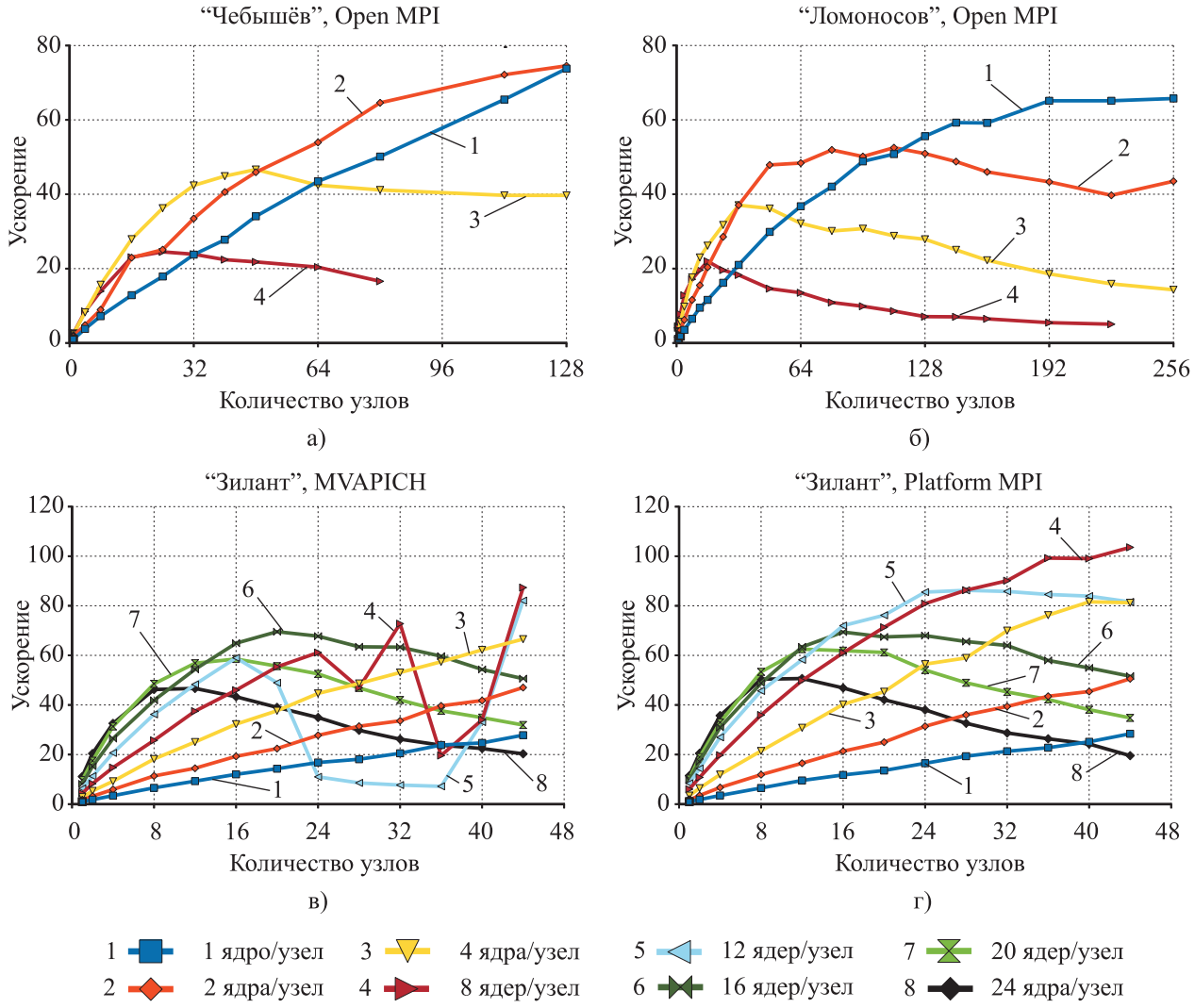

Рис. 2. Графики зависимости масштабируемости MPI-версии приложения от количества вычислительных узлов: а) СКИФ МГУ “Чебышёв”, б) “Ломоносов”, в) “Зилант”: MVAPICH, г) “Зилант”: Platform MPI

Наличие таких осцилляций времени расчета задачи для библиотеки MVAPICH подтолкнуло к исследованию других реализаций библиотеки обмена сообщениями. Графики зависимости ускорения от количества узлов, полученные при использовании библиотеки Platform MPI, приведены на рис. 2г. В этом случае графики масштабируемости оказываются более предсказуемыми, а результаты — лучше. Так, пиковое значение ускорения было получено для восьми MPI-процессов на узле и составило 103 раза. Соответствующее время расчета задачи в этом случае оказалось равным 1.55 секунды.

Таким образом, несмотря на существенно лучшие результаты масштабируемости, полученные на AMD-платформе, абсолютное время расчета задачи на системе “Ломоносов” оказалось меньше на 30%. Вместе с тем, судя по поведению кривых масштабируемости для системы “Зилант”, можно предполагать, что на большем количестве узлов показатели масштабируемости могут быть значительно улучшены, а время расчета задачи будет сравнимым или лучшим по сравнению со временем расчета на системе “Ломоносов”. Отдельно следует отметить, что на результаты масштабируемости оказывает очень существенное влияние выбор схемы привязки вычислительных процессов к физическим ядрам процессоров: в некото-

рых случаях различия в результатах могут составлять до одного порядка. Наиболее удачной оказалась следующая схема привязки процессов:

```
CPU_MAP:0,12,6,18,3,15,9,21,1,13,7,19,4,16,10,22,2,14,8,20,5,17,11,23
```

В этом случае происходит равномерное распределение процессов как между двумя 12-ядерными процессорами, так и внутри этих процессоров между двумя “склеенными” 6-ядерными процессорами.

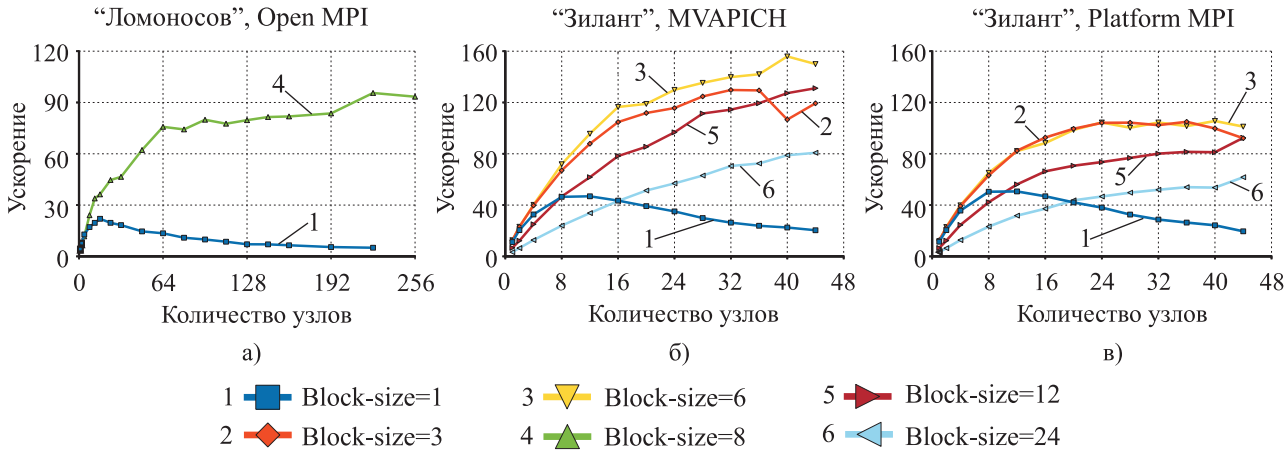

Рис. 3. Графики зависимости масштабируемости гибридной версии приложения от количества вычислительных узлов: а) “Ломоносов”, б) “Зилант”: MVAPICH, в) “Зилант”: Platform MPI. Параметр “Block-size” характеризует количество MPI-процессов, объединенных в единый блок, обмен данными внутри которого происходит через общую память и в котором используется только один внешний MPI-процесс; случай “Block-size=1” эквивалентен обычной MPI-версии приложения

4.2.2. Гибридная модель. Достаточно интересными оказались и результаты исследования масштабируемости для гибридной версии кода (рис. 3). В этих тестах в расчетах также использовались все доступные физические ядра (8 для Intel-платформ и 24 для AMD-платформы).

На вычислительной системе “Ломоносов” объединение 8 процессов в один блок позволяет заметно улучшить результаты масштабируемости. Пиковое значение достигается на 224 узлах и составляет 96 раз в сравнении с ускорением в 65 раз для MPI-версии кода. Абсолютное время расчета тестовой задачи в этом случае уменьшается в полтора раза с 1.18 до 0.81 секунды.

Для системы “Зилант” исследование было проведено также для двух библиотек MPI: MVAPICH и Platform MPI. Если в случае MPI-приложения заметное преимущество было у библиотеки Platform MPI, то здесь наблюдаемая картина оказывается в точности противоположной. Так, для библиотеки MVAPICH при использовании не более, чем 40 узлов, наиболее удачной конфигурацией оказывается использование на каждом узле 4 блоков по 6 процессов с привязкой этих процессов к половине 12-ядерного процессора. Пиковые значения оказываются на уровне ускорения в 156 раз, что эквивалентно времени расчета тестовой задачи за 1.03 секунды. В случае порядка 40 узлов наблюдается тенденция к сближению кривых для блоков из 6 и 12 процессов, что говорит о том, что для количества узлов порядка 60 оптимальным будет использование уже блоков из 12 процессов. При использовании Platform MPI наиболее удачным оказывается использование блоков по 3 или 6 процессов с соответствующей схемой привязки процессов к ядрам. Однако даже в этом случае получаемое ускорение чуть более чем в 100 раз сравнимо с параметрами масштабируемости для MPI-приложения и заметно ниже значений для библиотеки MVAPICH.

Неудовлетворительными оказываются результаты для блоков из 24 процессов. Несмотря на то что в этом случае на узле остается всего один коммуникационный MPI-процесс, что заметно снижает нагрузку на сеть, возникает большой трафик к областям общей памяти, расположенной в обоих NUMA-узлах, что существенно ухудшает результаты масштабируемости. Полученные данные наглядно свидетельствуют о необходимости доработки гибридной версии кода с учетом особенностей NUMA-архитектуры.

5. Заключение. В работе представлены результаты исследования масштабируемости тестового приложения решения системы линейных алгебраических уравнений с сильноразреженной матрицей на трех различных аппаратных платформах. Тесты показали, что для эффективного использования многоядерных процессоров необходимо использовать привязку вычислительных процессов к физическим ядрам процессоров, причем при выборе схемы привязки следует исходить из архитектуры используемой аппаратной платформы. Выбор библиотеки MPI может существенным образом сказаться на масштабируемо-

сти приложения. Для MPI- и гибридной версий приложения наиболее удачный выбор библиотеки MPI может оказаться различным. При использовании только одного вычислительного узла MPI-приложения могут оказаться столь же эффективными, как и приложения, работающие через общую память. При реализации гибридных моделей на NUMA-архитектуре необходимо минимизировать размер общих для всех процессов областей памяти, чтобы сократить количество обращений в память другого NUMA-узла через межпроцессорный интерфейс.

Автор благодарит Вл. В. Воеводина за предоставленную возможность проведения тестирования на вычислительных системах СКИФ МГУ "Чебышёв" и "Ломоносов" суперкомпьютерного комплекса МГУ им. М. В. Ломоносова, а также компанию "Т-Сервисы" за предоставленный доступ к вычислительной системе "Зилант".

СПИСОК ЛИТЕРАТУРЫ

1. *Saad Y.* Iterative methods for sparse linear systems. Philadelphia: SIAM, 2003.
2. *Trottenberg U., Oosterlee C. W., Schuller A.* Multigrid. New York: Academic Press, 2001.
3. *Krasnopolsky B.* The reordered BiCGStab method for distributed memory computer systems // Procedia Computer Science. 2010. **1**, N 1. 213–218.
4. *Краснопольский Б.И.* Исследование эффективности переупорядоченного метода BiCGStab на многопроцессорных вычислительных системах СКИФ МГУ "Чебышёв" и "Ломоносов" // Вестник ЮУрГУ, серия "Математическое моделирование и программирование". 2011. Вып. 7, № 4(221). 56–65.
5. *Pellegrini F., Chevalier C.* SCOTCH library: software package and libraries for graph, mesh and hypergraph partitioning, static mapping, and parallel and sequential sparse matrix block ordering (http://gforge.inria.fr/projects/scotch/).

Поступила в редакцию
11.03.2011