УДК 519.6; 519.688 doi 10.26089/NumMet.v16r111

ОТКРЫТАЯ ЭНЦИКЛОПЕДИЯ СВОЙСТВ АЛГОРИТМОВ ALGOWIKI: ОТ МОБИЛЬНЫХ ПЛАТФОРМ ДО ЭКЗАФЛОПСНЫХ СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

Вл. В. Воеводин[1]

Фундаментальная проблема высокопроизводительных вычислений — это необходимость аккуратного согласования структуры алгоритмов и программ с особенностями архитектуры компьютеров. Возможности современных компьютеров велики, но если хотя бы на одном из этапов процесса решения задачи согласования не будет, то и эффективность работы компьютера будет близка к нулю. Основная идея данного проекта состоит в том, что свойства самих алгоритмов никак не зависят от вычислительных систем, существующих сейчас или будущих. Иными словами, детальное описание машинно-независимых свойств алгоритма нужно сделать лишь один раз, после чего оно может быть многократно использовано при реализации данного алгоритма в различных программно-аппаратных средах. Не менее важна и вторая, машинно-зависимая часть данного исследования, которая посвящена описанию особенностей программной реализации алгоритмов с учетом конкретных программно-аппаратных компьютерных платформ. Результатом проекта, которому посвящена данная статья, является открытая энциклопедия AlgoWiki по свойствам алгоритмов и особенностям их реализации для различных компьютерных систем. Умение эффективно работать со свойствами алгоритмов (выделять, описывать, анализировать, интерпретировать) станет широко востребованным уже через несколько лет, что будет верно как для экзафлопсных суперкомпьютерных систем высшего диапазона производительности, так и для всех других компьютерных платформ: от серверных до мобильных.

Ключевые слова: параллельные вычисления, структура алгоритмов, последовательные свойства алгоритмов, параллельные свойства алгоритмов, суперкомпьютеры, компьютерные платформы, эффективная реализация алгоритмов, масштабируемость, локальность данных, энциклопедия алгоритмов.

1. Введение. Компьютеры быстро меняются. За последние сорок лет сменилось по крайней мере шесть поколений архитектур, вызвавших необходимость кардинальных изменений в программном обеспечении: векторные компьютеры, векторно-параллельные, массивно-параллельные, компьютеры с общей памятью, кластеры из машин с общей памятью, компьютеры с ускорителями и многие другие. Вычислительное сообщество через это прошло, но каждый раз все программное обеспечение нужно было, по сути, переписывать заново: очередное поколение машин требовало выделения новых свойств в алгоритмах, что отражалось и в программах.

Увы, но нет никаких оснований надеяться, что ситуация в будущем изменится к лучшему. Уже сейчас рассматриваются перспективные варианты архитектур, в которых будут использованы легкие и/или тяжелые вычислительные ядра, ускорители разного типа, идеи SIMD- и data-flow обработки. В такой ситуации, для полного использования потенциала будущих компьютеров нужно будет опять переписывать код. Бесконечный процесс, который оптимизма у разработчиков программного обеспечения не вызывает.

Однако ситуация, все же, не столь безнадежна. Да, появление новых вычислительных систем вызывает необходимость пересмотра алгоритмического багажа. Однако сами алгоритмы при этом, как правило, остаются неизменными: меняются лишь требования, которые новые компьютеры предъявляют к структуре алгоритмов и программ. Для поддержки векторизации необходим параллелизм по данным в самых внутренних циклах программы. Для векторно-параллельных машин дополнительно к этому необходим небольшой ресурс параллелизма на верхнем уровне циклического профиля для использования нескольких независимых процессоров. Для эффективного использования массивно-параллельных компьютеров предмет основной заботы — это поиск такого представления алгоритма, при котором большое число вычислительных узлов могло бы долго работать независимо друг от друга, сведя обмен данными к минимуму. И так для каждого поколения параллельных вычислительных систем: архитектура компьютеров заставляет

[1] Научно-исследовательский вычислительный центр, Московский государственный университет имени М. В. Ломоносова, Ленинские горы, д. 1, стр. 4, 119234, Москва; зам. директора, чл.-корр. РАН, e-mail: voevodin@parallel.ru

по-новому взглянуть на свойства существующих алгоритмов, чтобы найти эффективный способ реализации.

В такой ситуации для нас важны два факта: сами алгоритмы меняются мало, а их свойства никак не зависят от вычислительных систем. Это значит, что описав свойства алгоритмов один раз, этими сведениями можно будет пользоваться многократно для любых компьютерных платформ, существующих сейчас или будущих. Каждый раз, когда будет возникать задача эффективной реализации алгоритма на некотором компьютере, ответ на вопрос о потенциальных возможностях самого алгоритма можно будет найти в этом описании.

Идея глубокого априорного анализа свойств алгоритмов и их реализаций легла в основу проекта AlgoWiki, начатого в НИВЦ МГУ. Основная задача проекта — составить такое описание фундаментальных свойств алгоритмов, которое даст полное понимание как их теоретического потенциала, так и особенностей их реализации на различных классах параллельных вычислительных систем.

Описание всех алгоритмов в AlgoWiki строится из двух частей. В первой части описываются собственно алгоритмы и их свойства. Вторая часть посвящена описанию особенностей их программной реализации с учетом конкретных программно-аппаратных платформ. Такое деление на две части сделано для того, чтобы машинно-независимые свойства алгоритмов, определяющие их потенциал и качество реализации на параллельных вычислительных системах, были бы выделены и описаны отдельно от множества вопросов, связанных с последующими этапами программирования алгоритмов и исполнения результирующих программ.

Цель AlgoWiki — дать исчерпывающее описание алгоритма, которое поможет оценить его потенциал применительно к конкретной параллельной вычислительной платформе. Кроме классических свойств алгоритмов, например последовательной сложности, в AlgoWiki должен быть представлен и целый набор дополнительных сведений, составляющих в совокупности полную картину об алгоритме: параллельная сложность, параллельная структура, детерминированность, оценки локальности данных, эффективности и масштабируемости, коммуникационный профиль конкретных реализаций и многие другие.

В AlgoWiki очень важны детали. Классические базовые алгоритмы должны быть дополнены важными на практике модификациями. В некоторых случаях для описания модификаций достаточно одного небольшого комментария со ссылкой на базовый вариант алгоритма, поясняющего идею его модификации и получаемый эффект. Иногда же изменения алгоритма значительны и требуют полноценного самостоятельного описания. Так, в AlgoWiki есть описание базового точечного вещественного варианта разложения Холецкого для плотной симметричной положительно определенной матрицы. Однако на практике не менее важны модификации алгоритма: блочный вариант, вариант для плотной комплексно-сопряженной матрицы (как точечный, так и блочный), варианты алгоритма для разреженных матриц, важны соображения относительно использования разложения Холецкого в итерационных методах и т.д.

Точно так же важны и детали, касающиеся особенностей реализации алгоритмов на конкретных параллельных вычислительных платформах: многоядерных процессорах, SMP, кластерах, разного рода ускорителях, с использованием векторной обработки. Во многих случаях нужно спускаться еще на уровень ниже, описывая особенности реализации алгоритма не просто для ускорителя, а выделяя отдельно важные частные случаи, например GPU и Xeon Phi. Полезными являются и данные вычислительных экспериментов на похожих платформах, различающихся, например, разными объемами кэш-памяти, деталями реализации векторной обработки или латентностью коммуникационной сети. При этом, приводя данные о времени выполнения, эффективности и масштабируемости, мы не только даем некоторые ориентиры возможного качества реализации алгоритма на конкретном компьютере, но и формируем базу для сравнительного анализа различных компьютерных платформ на алгоритмах, представленных в AlgoWiki.

Результатом проекта AlgoWiki является открытая энциклопедия по свойствам алгоритмов и особенностям их реализации на различных компьютерах. В реализации сразу была сделана ориентация на использование wiki-технологий с возможностью коллективной работы над энциклопедией всего вычислительного сообщества. На первом этапе проекта основное внимание уделялось отработке именно структуры описания алгоритмов. В настоящее время энциклопедия активно расширяется за счет описаний алгоритмов, составленных внешними экспертами, а также создается многоязычная версия, которая затем и станет основной.

Прототип энциклопедии доступен по адресу http://AlgoWiki-Project.org.

2. Предыстория и смежные исследования. Вопросы эффективности и поддержки параллелизма по всему спектру программного обеспечения суперкомпьютеров сегодня являются центральными на всех мировых суперкомпьютерных форумах. Этот же круг вопросов обсуждается и в рамках главных международных проектов, формулирующих ключевые проблемы и составляющих прогноз развития про-

граммного обеспечения суперкомпьютерной отрасли на ближайшие десятилетия: International Exascale Software Project, Big Data and Extreme Computing, European Exascale Software Initiative I и II [1–3].

В некоторых случаях исследование процесса отображения алгоритмов на архитектуру параллельных вычислительных систем сводится к исследованию так называемых типовых алгоритмических структур — структур, на основе которых построено значительное число задач из различных предметных областей. По такому пути идет группа исследователей из Беркли, которая использует понятие, называемое motifs, или dwarfs, для обозначения алгоритмических подходов, использующих похожие шаблоны вычислений и коммуникаций [4].

Исследования свойств параллельных алгоритмов ведутся давно, начиная с фундаментального труда [5], первое издание которого вышло в свет в 1968 г. Достаточно большой спектр алгоритмов рассмотрен в известной работе [6]. Часто авторы ограничиваются конкретной предметной областью: например, для параллельных алгоритмов линейной алгебры можно отметить работы [7, 8] и, конечно же, классическую книгу [9].

В данном проекте ключевым моментом в определении и использовании свойств алгоритмов является понятие графа алгоритма (в литературе граф алгоритма иногда называют информационным графом или графом зависимостей), введенного и детально исследованного в работах [10–12].

Реализуемая в рамках данного проекта Открытая энциклопедия свойств алгоритмов AlgoWiki не имеет прямых аналогов. Существует ряд проектов по классификации, описанию свойств и написанию реализаций алгоритмов [13–16], но ни один из них не описывает всех необходимых свойств алгоритмов и их реализаций для различных целевых архитектур по единой заранее определенной схеме.

3. Описание свойств и структуры алгоритмов. Основная задача, которую нужно решить для создания энциклопедии свойств алгоритмов, — это формализация процесса отображения алгоритмов на архитектуру параллельных вычислительных систем. Многие исследователи занимались отдельными сторонами данной задачи, однако в рамках AlgoWiki все свойства и характеристики, которые определяют потенциал алгоритма и влияют на качество его реализации, должны быть собрано воедино. Исследование свойств алгоритмов и программ выполняется на основе понятия графа алгоритма, что гарантирует главное — полноту определения ресурса параллелизма.

В процессе описания алгоритма нужно проанализировать три основных этапа процесса отображения: исследование свойств алгоритмов, исследование свойств программ и сопоставление найденных свойств с особенностями архитектуры компьютеров. Через все три этапа должны быть проанализированы два принципиально важных свойства, определяющих качество реализации алгоритма: ресурс параллелизма и эффективность работы с данными. Дополнительная сложность заключается в том, что эти два свойства должны быть согласованы как по всем трем этапам, так и между собой на каждом этапе. Для каждого этапа нужно выделить инварианты в описании алгоритмов, которые не меняются при переходе с платформы на платформу и которые как раз и станут основой для упрощения процесса программирования параллельных компьютеров. Выделив и описав в AlgoWiki свойство алгоритма, эту работу не нужно будет выполнять вновь при переходе с одного компьютера на другой.

И крайне важный вопрос — это полнота описания процесса отображения, определяющая контроль над эффективностью как при разработке новых параллельных программ, так и при их переносе с одной компьютерной платформы на другую. Иными словами, все принципиально важные особенности, влияющие на эффективность выполнения результирующих программ, должны найти отражение в описании свойств и структуры алгоритмов.

На основе подобных соображений была разработана структура описания алгоритмов, которая и легла в основу энциклопедии AlgoWiki. Для отдельных разделов в энциклопедии предлагаются элементы стандартизации и рекомендации по их формированию, чтобы описания разных алгоритмов можно было бы легко сравнивать между собой. Для правильной интерпретации представленных данных вычислительных экспериментов (время, масштабируемость, производительность и т.п.) должна быть точно описана платформа, на которой эти данные были получены.

Ниже в разделах 4 и 5 описываются две части, составляющие описание каждого алгоритма в рамках проекта AlgoWiki.

4. Энциклопедия AlgoWiki: “Часть I. Свойства и структура алгоритмов”. Свойства алгоритмов никак не зависят от вычислительных систем, и с этой точки зрения данная часть AlgoWiki имеет безусловную самостоятельную ценность. Описание алгоритма делается один раз, после чего многократно используется для его реализации в различных программно-аппаратных средах. Несмотря на то что в данной части мы рассматриваем лишь машинно-независимые свойства алгоритмов, соображения, важные на этапе реализации, или же ссылки на соответствующие пункты части II AlgoWiki здесь тоже вполне

уместны.

4.1. Общее описание алгоритма. В данном разделе представляется самый первый вариант описания тех задач (или классов задач), для решения которых предназначен алгоритм. В описании поясняются особенности как алгоритма, так и объектов, с которыми он работает. Если описание соответствует целому классу схожих по структуре алгоритмов либо же посвящено описанию отдельного представителя некоторого класса, то это здесь указывается явно. Описываются базовые математические свойства и структура входных данных. При необходимости в описании могут присутствовать формулы, а также даваться ссылки на описания других используемых алгоритмов.

4.2. Математическое описание алгоритма. Приводится математическое описание решаемой задачи в виде совокупности формул и соотношений, как это принято в книгах и учебниках. По возможности, используются общепринятые обозначения и способы записи. Должны быть явно определены все использованные обозначения и описаны свойства входных данных. Представленное описание должно быть достаточным для однозначного понимания постановки решаемой задачи человеком, знающим математику.

4.3. Вычислительное ядро алгоритма. В описываемом алгоритме выделяется и описывается вычислительное ядро, т.е. та часть алгоритма, на которую приходится основное время работы алгоритма. Если в алгоритме несколько вычислительных ядер, то отдельно описывается каждое ядро. Описание может быть сделано в достаточно произвольной форме: словесной или с использованием языка математических формул.

4.4. Макроструктура алгоритма. Если алгоритм использует в качестве составных частей другие алгоритмы, то это указывается в данном разделе. Если в дальнейшем имеет смысл описывать алгоритм не в максимально детализированном виде (на уровне арифметических операций), а давать только его макроструктуру, то здесь описывается структура и состав макроопераций. Если в других разделах описания рассматриваемого алгоритма в рамках AlgoWiki используются введенные макрооперации, то здесь даются пояснения, необходимые для однозначной интерпретации материала. Типичные варианты макроопераций, часто встречающиеся на практике: нахождение суммы элементов вектора, скалярное произведение векторов, умножение матрицы на вектор, решение системы линейных уравнений малого порядка, сортировка, вычисление значения функции в некоторой точке, поиск минимального значения в массиве, транспонирование матрицы, вычисление обратной матрицы и многие другие.

Описание макроструктуры очень полезно на практике. Параллельная структура алгоритмов может быть хорошо видна именно на макроуровне, в то время как максимально детальное отображение всех операций может значительно усложнить картину. Аналогичные аргументы касаются и многих вопросов реализации, и если для алгоритма эффективнее и/или технологичнее оставаться на макроуровне, оформив макровершину, например в виде отдельной процедуры, то это и нужно отразить в данном разделе.

Выбор макроопераций не однозначен, причем на этапе выделения макроопераций можно делать акценты на различных свойствах алгоритмов. С этой точки зрения, в описании одного алгоритма может быть представлено несколько вариантов его макроструктуры, дающих дополнительную информацию о его структуре. На практике подобные альтернативные формы представления макроструктуры алгоритма могут оказаться исключительно полезными для его эффективной реализации на различных вычислительных платформах.

4.5. Описание схемы реализации последовательного алгоритма. Здесь описываются все шаги, которые нужно выполнить при последовательной реализации данного алгоритма. В некотором смысле данный раздел является избыточным, поскольку математическое описание уже содержит всю необходимую информацию, однако он, несомненно, полезен: схема реализации алгоритма выписывается явно, помогая однозначной интерпретации приводимых далее оценок и свойств.

Описание может быть выполнено в виде блок-схемы, последовательности математических формул, обращений к описанию других алгоритмов, фрагмента кода на Фортране, Си или другом языке программирования, фрагмента кода на псевдокоде и т.п. Главное — это сделать схему реализации последовательного алгоритма полностью понятной. Совершенно не обязательно все шаги детализировать до элементарных операций, отдельные шаги могут соответствовать макрооперациям, отвечающим другим алгоритмам.

Описание схемы реализации вполне может содержать и словесные пояснения, отражающие какие-либо тонкие нюансы самого алгоритма или его реализации. Уже в данном разделе можно сказать про возможный компромисс между объемом требуемой оперативной памяти и временем работы алгоритма, между используемыми структурами данных и степенью доступного параллелизма. В частности, нередко возникает ситуация, когда можно ввести дополнительные вре́менные массивы или же отказаться от использования специальных компактных схем хранения данных, увеличивая степень доступного парал-

лелизма.

4.6. Последовательная сложность алгоритма. В данном разделе описания свойств алгоритма приводится оценка его последовательной сложности, т.е. числа операций, которые нужно выполнить при последовательном исполнении алгоритма (в соответствии с п. 4.5). Для разных алгоритмов понятие операции, в терминах которой оценивается его сложность, может существенно различаться. Это могут быть операции для работы с вещественными числами, целыми числами, поразрядные операции, обращения в память, обновления элементов массива, элементарные функции, макрооперации и другие. В LU-разложении матриц преобладают арифметические операции над вещественными числами, а для транспонирования матриц важны лишь обращения к памяти: это и должно найти отражение в описании.

Если выбор конкретного типа операций для оценки сложности алгоритма не очевиден, то нужно привести обоснование возможных вариантов. В некоторых случаях можно приводить оценку не всего алгоритма, а лишь его вычислительного ядра: в таком случае это нужно отметить, сославшись на п. 4.3.

Например, сложность алгоритма суммирования элементов вектора сдваиванием равна $n-1$. Сложность быстрого преобразования Фурье (базовый алгоритм Кули–Тьюки) для векторов с длиной, равной степени двойки, составляет $n\log_2(n)$ операций комплексного сложения и $n\log_2(n)/2$ операций комплексного умножения. Сложность базового алгоритма разложения Холецкого (точечный вариант для плотной симметричной и положительно определенной матрицы) — это n вычислений квадратного корня, $n(n-1)/2$ операций деления и по $\big(n^3-n\big)/6$ операций умножения и сложения (вычитания).

4.7. Информационный граф. Это очень важный раздел описания. Именно здесь можно показать (увидеть): как устроена параллельная структура алгоритма, для чего приводится описание и изображение его информационного графа (графа алгоритма [12]). Для рисунков с изображением графа будут составлены рекомендации по их формированию, чтобы все информационные графы, внесенные в энциклопедию, можно было бы воспринимать и интерпретировать одинаково. Дополнительно можно привести полное параметрическое описание графа в терминах покрывающих функций [12].

Интересных вариантов для отражения информационной структуры алгоритмов много. Для каких-то алгоритмов нужно показать максимально подробную структуру, а иногда важнее макроструктура. Много информации несут разного рода проекции информационного графа, выделяя его регулярные составляющие и одновременно скрывая несущественные детали. Иногда оказывается полезным показать последовательность в изменении графа при изменении значений внешних переменных (например, размеров матриц): мы часто ожидаем “подобное” изменение информационного графа, но это изменение не всегда очевидно на практике.

В целом, задача изображения графа алгоритма весьма нетривиальна. Начнем с того, что это потенциально бесконечный граф, число вершин и дуг которого определяется значениями внешних переменных, а они могут быть весьма и весьма велики. В такой ситуации, как правило, спасают упомянутые выше соображения подобия, делающие графы для разных значений внешних переменных “похожими”: почти всегда достаточно привести лишь один граф небольшого размера, добавив, что графы для остальных значений будут устроены “точно так же”. На практике, увы, не всегда все так просто, и здесь нужно быть аккуратным.

Далее, граф алгоритма — это потенциально многомерный объект. Наиболее естественная система координат для размещения вершин и дуг информационного графа опирается на структуру вложенности циклов в реализации алгоритма. Если глубина вложенности циклов не превышает трех, то и граф размещается в привычном трехмерном пространстве, однако для более сложных циклических конструкций с глубиной вложенности 4 и больше необходимы специальные методы представления и изображения графов.

В данном разделе AlgoWiki могут использоваться многие интересные возможности, которые еще подлежат обсуждению: возможность повернуть граф при его отображении на экране компьютера для выбора наиболее удобного угла обзора, разметка вершин по типу соответствующим им операций, отражение ярусно-параллельной формы графа и др. Однако в любом случае нужно не забывать главную задачу данного раздела — показать информационную структуру алгоритма так, чтобы стали понятны все его ключевые особенности, особенности параллельной структуры, особенности множеств дуг, участки регулярности и, напротив, участки с недетерминированной структурой, зависящей от входных данных.

На рис. 1 показана информационная структура алгоритма умножения матриц, на рис. 2 — информационная структура одного из вариантов алгоритма решения систем линейных алгебраических уравнений с блочно-двухдиагональной матрицей.

4.8. Описание ресурса параллелизма алгоритма. Здесь приводится оценка параллельной сложности алгоритма: числа шагов, за которое можно выполнить данный алгоритм в предположении доступ-

ности неограниченного числа необходимых процессоров (функциональных устройств, вычислительных узлов, ядер и т.п.). Параллельная сложность алгоритма понимается как высота канонической ярусно-параллельной формы [12]. Необходимо указать, в терминах каких операций дается оценка. Необходимо описать сбалансированность параллельных шагов по числу и типу операций, что определяется шириной ярусов канонической ярусно-параллельной формы и составом операций на ярусах.

Параллелизм в алгоритме часто имеет естественную иерархическую структуру. Этот факт очень полезен на практике, и его необходимо отразить в описании. Как правило, подобная иерархическая структура параллелизма хорошо отражается в последовательной реализации алгоритма через циклический профиль результирующей программы (конечно же, с учетом графа вызовов), поэтому циклический профиль (п. 4.5) вполне может быть использован и для отражения ресурса параллелизма.

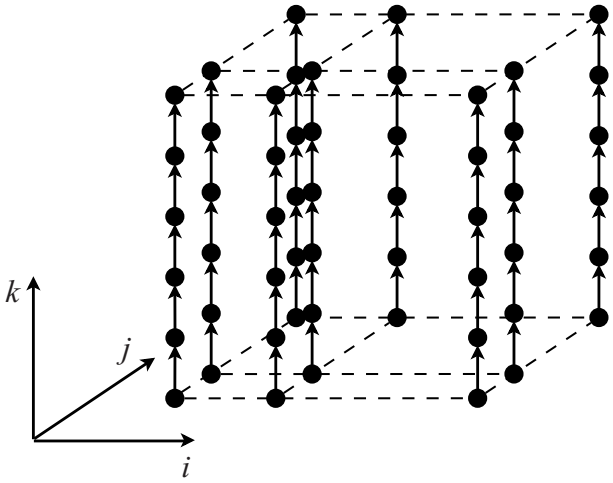

Рис. 1. Информационная структура алгоритма умножения матриц

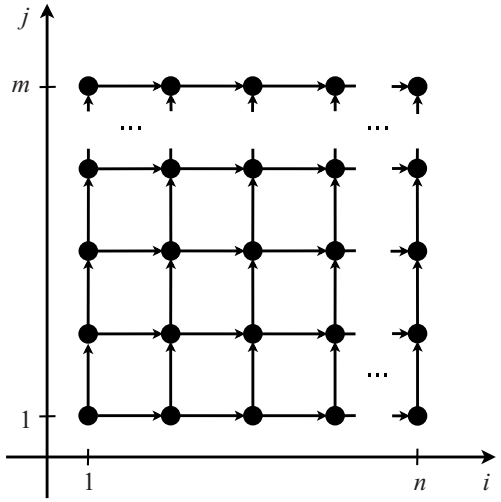

Рис. 2. Информационная структура одного из вариантов алгоритма решения систем линейных алгебраических уравнений с блочно-двухдиагональной матрицей

Для описания ресурса параллелизма алгоритма (ресурса параллелизма информационного графа) необходимо указать ключевые параллельные ветви в терминах конечного и массового параллелизма. Далеко не всегда ресурс параллелизма выражается просто, например, через координатный параллелизм или, что то же самое, через независимость итераций некоторых циклов. В данном случае координатный параллелизм означает, что информационно независимые вершины лежат на гиперплоскостях, перпендикулярных одной из координатных осей. С этой точки зрения не менее важен и ресурс скошенного параллелизма. В отличие от координатного параллелизма, скошенный параллелизм намного сложнее использовать на практике, но знать о нем необходимо, поскольку иногда других вариантов и не остается: нужно оценить потенциал алгоритма и лишь после этого, взвесив все альтернативы, принимать решение о конкретной параллельной реализации. Хорошей иллюстрацией может служить алгоритм, структура которого показана на рис. 2: координатного параллелизма нет, но есть параллелизм скошенный, использование которого снижает сложность алгоритма с $n*m$ в последовательном случае до $(n+m-1)$ в параллельном варианте.

Рассмотрим алгоритмы, последовательная сложность которых уже оценивалась в п. 4.6. Параллельная сложность алгоритма суммирования элементов вектора сдваиванием равна $\log_2(n)$, причем число операций на каждом ярусе убывает с $n/2$ до 1. Параллельная сложность быстрого преобразования Фурье (базовый алгоритм Кули–Тьюки) для векторов с длиной, равной степени двойки, — $\log_2(n)$. Параллельная сложность базового алгоритма разложения Холецкого (точечный вариант для плотной симметричной и положительно определенной матрицы) — это n шагов для вычислений квадратного корня, $(n-1)$ шагов для операций деления и $(n-1)$ шагов для операций умножения и сложения.

4.9. Описание входных и выходных данных. В данном разделе необходимо описать объем, структуру, особенности и свойства входных и выходных данных алгоритма: векторы, матрицы, скаляры, множества, плотные или разреженные структуры данных, их объем. Полезны предположения относительно диапазона значений или структуры, например: диагональное преобладание в структуре входных матриц, соотношение между размером матриц по отдельным размерностям, большое число матриц очень малой размерности, близость значений каких-то элементов матрицы к машинному нулю, характер разре-

женности матриц и др.

4.10. Свойства алгоритма. Описываются прочие свойства алгоритма, на которые имеет смысл обратить внимание на этапе реализации. Как и ранее, никакой привязки к конкретной программно-аппаратной платформе не предполагается, однако вопросы реализации в проекте AlgoWiki всегда превалируют, и необходимость обсуждения каких-либо свойств алгоритмов определяется именно этим.

Весьма полезным является *соотношение последовательной и параллельной сложности* алгоритма. Оба понятия мы рассматривали ранее, но здесь делается акцент на том выигрыше, который теоретически может дать параллельная реализация алгоритма. Не менее важно описать и те сложности, которые могут возникнуть в процессе получения параллельной версии алгоритма.

Вычислительная мощность алгоритма равна отношению числа операций к суммарному объему входных и выходных данных. Она показывает, сколько операций приходится на единицу переданных данных. Несмотря на простоту данного понятия, это значение исключительно полезно на практике: чем выше вычислительная мощность, тем меньше накладных расходов вызывает перемещение данных для их обработки, например на сопроцессоре, ускорителе или другом узле кластера. Заметим, что вычислительная мощность скалярного произведения двух векторов равна всего лишь 1, а вычислительная мощность алгоритма умножения двух квадратных матриц равна $2n/3$.

Вопрос первостепенной важности на последующем этапе реализации — это *устойчивость алгоритма*. Все, что касается различных сторон этого понятия, в частности оценки устойчивости, должно быть описано в данном разделе.

Сбалансированность вычислительного процесса можно рассматривать с разных сторон. Здесь и сбалансированность типов операций, в частности арифметических операций между собой (сложение, умножение, деление) или же арифметических операций по отношению к операциям обращения к памяти (чтение/запись). Здесь и сбалансированность операций между параллельными ветвями алгоритма. С одной стороны, балансировка нагрузки является необходимым условием эффективной реализации алгоритма. Вместе с тем, это очень непростая задача, и в описании должно быть отмечено явно, насколько алгоритм обладает этой особенностью. Если обеспечение сбалансированности не очевидно, то желательно описать возможные пути решения этой задачи.

На практике важна *детерминированность алгоритмов*, под которой будем понимать постоянство структуры вычислительного процесса. С этой точки зрения, классическое умножение плотных матриц является детерминированным алгоритмом, поскольку его структура при фиксированном размере матриц никак не зависит от элементов входных матриц. Умножение разреженных матриц, когда матрицы хранятся в одном из специальных форматов, свойством детерминированности уже не обладает: его свойства, например степень локальности данных, зависят от структуры разреженности входных матриц. Итерационный алгоритм с выходом по точности тоже не является детерминированным: число итераций, а значит и число операций, меняется в зависимости от входных данных. В этом же ряду стоит использование датчиков случайных чисел, меняющих вычислительный процесс для различных запусков программы.

Причина выделения свойства детерминированности понятна: работать с детерминированным алгоритмом проще, поскольку один раз найденная структура и будет определять качество его реализации. Если детерминированность нарушается, то это должно быть здесь описано вместе с описанием того, как недетерминированность влияет на структуру вычислительного процесса.

Серьезной причиной недетерминированности работы параллельных программ является изменение порядка выполнения ассоциативных операций. Типичный пример — это использование глобальных MPI-операций на множестве параллельных процессов, например суммирование элементов распределенного массива. Система времени исполнения, встроенная в MPI, сама выбирает порядок выполнения операций, предполагая выполнение свойства ассоциативности, из-за чего ошибки округления меняются от запуска программы к запуску, внося изменения в конечный результат ее работы. Это очень серьезная проблема, которая сегодня часто встречается на системах с массовым параллелизмом и определяет отсутствие повторяемости результатов работы параллельных программ. Данная особенность характерна для второй части AlgoWiki, посвященной реализации алгоритмов, но вопрос очень важный и соответствующие соображения, по возможности, должны быть отмечены и здесь.

Заметим, что, в некоторых случаях, недетерминированность в структуре алгоритмов можно "убрать" введением соответствующих макроопераций, после чего структура становится не только детерминированной, но и более понятной для восприятия. Подобное действие тоже следует отразить в данном разделе.

Степень исхода вершины информационного графа показывает, в скольких операциях ее результат будет использоваться в качестве аргумента. Если степень исхода вершины велика, то на этапе реализации алгоритма нужно позаботиться об эффективном доступе к результату ее работы. В этом смысле, особый

интерес представляют рассылки данных, когда результат выполнения одной операции используется во многих других вершинах графа, причем число таких вершин растет с увеличением значения внешних переменных.

"Длинные" дуги в информационном графе [12] говорят о потенциальных сложностях с размещением данных в иерархии памяти компьютера на этапе выполнения программы. С одной стороны, длина дуги зависит от выбора конкретной системы координат, в которой расположены вершины графа, а потому в другой системе координат они попросту могут исчезнуть (но не появятся ли другие длинные дуги?). А с другой стороны, вне зависимости от системы координат их присутствие может быть сигналом о необходимости длительного хранения данных на определенном уровне иерархии, что накладывает дополнительные ограничения на эффективность реализации алгоритма. Одной из причин возникновения длинных дуг являются рассылки скалярных величин по всем итерациям какого-либо цикла: в таком виде длинные дуги не вызывают на практике каких-либо серьезных проблем.

Для проектирования специализированных процессоров или реализации алгоритма на ПЛИС представляют интерес компактные укладки информационного графа [12], которые тоже имеет смысл привести в данном разделе.

5. Энциклопедия AlgoWiki: "Часть II. Программная реализация алгоритмов". Вторая часть описания алгоритмов в рамках AlgoWiki рассматривает все составные части процесса их реализации. Обсуждается как последовательная реализация алгоритма, так и параллельная. Описывается взаимосвязь свойств программ, реализующих алгоритм, и особенности архитектуры компьютера, на которой они выполняются. Исследуется работа с памятью, локальность данных и вычислений, описывается масштабируемость и эффективность параллельных программ, производительность компьютеров, достигаемая на данной программе. Обсуждаются особенности реализации для разных классов архитектур компьютеров, приводятся ссылки на реализации в существующих библиотеках.

5.1. Особенности реализации последовательного алгоритма. Здесь описываются особенности и варианты реализации алгоритма в виде последовательной программы, которые влияют на эффективность ее выполнения. В частности, в данном разделе имеет смысл сказать *о существовании блочных вариантов реализации алгоритма*, дополнительно описав потенциальные преимущества или недостатки, сопровождающие такую реализацию. Важный вопрос — это *возможные варианты организации работы с данными*, варианты структур данных, наборов временных массивов и другие подобные вопросы. Для различных вариантов реализации следует оценить доступный ресурс параллелизма и объем требуемой памяти.

Важным нюансом является *описание необходимой разрядности выполнения операций алгоритма* (точности). На практике часто нет никакой необходимости выполнять все арифметические операции над вещественными числами с двойной точностью, так как это не влияет ни на устойчивость алгоритма, ни на точность получаемого результата. В таком случае, если значительную часть операций можно выполнять над типом float и лишь в некоторых фрагментах необходим переход к типу double, это обязательно нужно отметить. Это прямое указание не только на правильную реализацию с точки зрения устойчивости по отношению к ошибкам округления, но и на более эффективную по быстродействию.

Опираясь на информацию из п. 4.8 (описание ресурса параллелизма алгоритма), при описании последовательной версии стоит сказать про *возможности эквивалентного преобразования программ*, реализующих данных алгоритм. В дальнейшем это даст возможность простого использования доступного параллелизма или же просто покажет, как использовать присущий алгоритму параллелизм на практике. Например, параллелизм на уровне итераций самого внутреннего цикла обычно используется для векторизации. Однако в некоторых случаях этот параллелизм можно поднять "вверх" по структуре вложенности объемлющих циклов, что делает возможной и эффективную реализацию данного алгоритма на многоядерных SMP-компьютерах.

С этой же точки зрения в данном разделе весьма полезны соображения по реализации алгоритма на различных параллельных вычислительных платформах. Высокопроизводительные кластеры, многоядерные узлы, возможности для векторизации или использования ускорителей — особенности этих архитектур не только опираются на разные свойства алгоритмов, но и по-разному должны быть выражены в программах, что тоже желательно описать в данном разделе.

5.2. Описание локальности данных и вычислений. Вопросы локальности данных и вычислений не часто изучаются на практике, но именно локальность определяет эффективность выполнения программ на современных вычислительных платформах [17, 18]. В этом разделе приводятся оценки степени локальности данных и вычислений в программе, причем рассматривается как временна́я, так и пространственная локальность. Отмечаются позитивные и негативные факты, связанные с локальностью: какие

ситуации и при каких условиях могут возникать. Исследуется, как меняется локальность при переходе от последовательной реализации к параллельной. Выделяются ключевые шаблоны взаимодействия программы, реализующей описываемый алгоритм, с памятью. Отмечается возможная взаимосвязь между используемыми конструкциями языков программирования и степенью локальности, которыми обладают результирующие программы.

Отдельно приводятся профили взаимодействия с памятью для вычислительных ядер и ключевых фрагментов. Если из-за большого числа обращений по общему профилю сложно понять реальную специфику взаимодействия программ с памятью, то проводится последовательная детализация и приводится серия профилей более мелкого масштаба.

На рис. 3 показаны профили обращения в память для программ, реализующих разложение Холецкого и быстрое преобразование Фурье, по которым хорошо видно различие свойств локальности у этих алгоритмов.

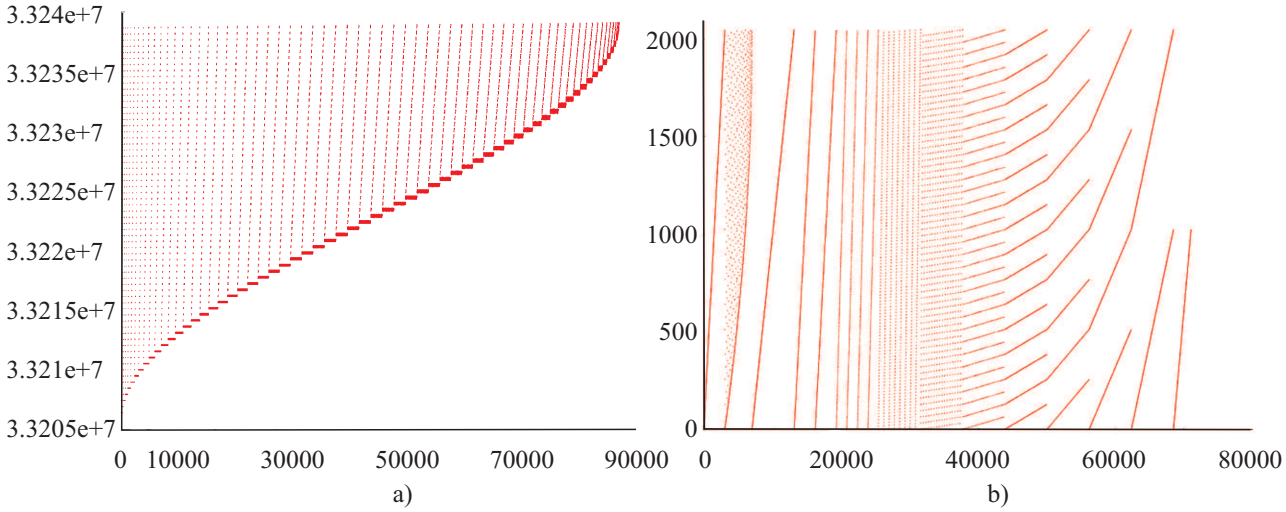

Рис. 3. Профили обращения в память для программ, реализующих разложение Холецкого (a) и быстрое преобразование Фурье (b)

5.3. Возможные способы и особенности параллельной реализации алгоритма. Этот раздел довольно обширный, в нем должны быть описаны основные факты и положения, формирующие параллельную программу. К их числу можно отнести:

— представленный иерархически ресурс параллелизма, опирающийся на структуру циклических конструкций и на граф вызовов программы;

— комбинацию (иерархию) массового параллелизма и параллелизма конечного;

— возможные способы распределения операций между процессами/нитями;

— возможные способы распределения данных;

— оценку количества операций, объема и числа пересылок данных (как общего числа, так и в пересчете на каждый параллельный процесс);

— оценки локальности данных и др.

В этом же разделе должны быть даны рекомендации или сделаны комментарии относительно реализации алгоритма с помощью различных технологий параллельного программирования: MPI, OpenMP, CUDA, использования директив векторизации и др.

5.4. Масштабируемость алгоритма и его реализации. Задача данного раздела — показать пределы масштабируемости алгоритма на различных платформах. Очень важный раздел. Нужно выделить, описать и оценить влияние точек барьерной синхронизации, глобальных операций, операций сборки/разборки данных, привести оценки или провести исследование сильной и слабой масштабируемости алгоритма и его реализаций.

Масштабируемость алгоритма определяет свойства самого алгоритма безотносительно конкретных особенностей используемого компьютера. Она показывает, насколько параллельные свойства алгоритма позволяют использовать возможности растущего числа процессорных элементов. Масштабируемость параллельных программ определяется как относительно конкретного компьютера, так и относительно используемой технологии программирования, и в этом случае она показывает, насколько может вырасти реальная производительность данного компьютера на данной программе, записанной с помощью данной

технологии программирования, при использовании бóльших вычислительных ресурсов (ядер, процессоров, вычислительных узлов).

Ключевой момент данного раздела заключается в том, чтобы показать реальные параметры масштабируемости программы для данного алгоритма на различных вычислительных платформах в зависимости от числа процессов и размера задачи [19]. При этом важно подобрать такое соотношение между числом процессов и размером задачи, чтобы отразить все характерные точки в поведении параллельной программы, в частности достижение максимальной производительности, а также тонкие эффекты, возникающие, например, из-за блочной структуры алгоритма или иерархии памяти.

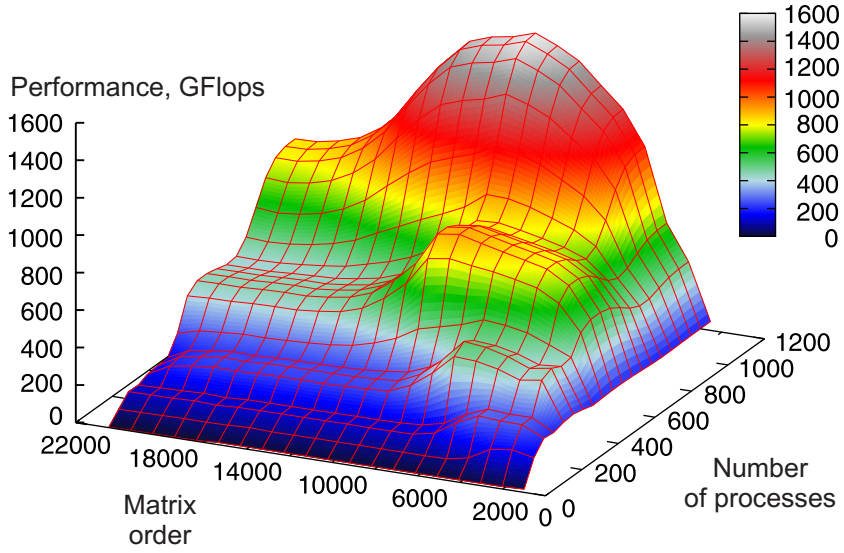

Рис. 4. Масштабируемость классического алгоритма умножения плотных матриц в зависимости от числа процессов и размера задачи

На рис. 4 показана масштабируемость классического алгоритма умножения плотных матриц в зависимости от числа процессов и размера задачи. На графике хорошо видны области с большей производительностью, отвечающие уровням кэш-памяти.

5.5. Динамические характеристики и эффективность реализации алгоритма. Это объемный раздел AlgoWiki, поскольку оценка эффективности реализации алгоритма требует комплексного подхода [20], предполагающего аккуратный анализ всех этапов от архитектуры компьютера до самого алгоритма. Основная задача данного раздела заключается в том, чтобы оценить степень эффективности параллельных программ, реализующих данный алгоритм на различных платформах, в зависимости от числа процессов и размера задачи. Эффективность в данном разделе понимается широко: это и эффективность распараллеливания программы, это и эффективность выполнения программ по отношению к пиковым показателям работы вычислительных систем.

Помимо собственно показателей эффективности, нужно описать и все основные причины, из-за которых эффективность работы параллельной программы на конкретной вычислительной платформе не удается сделать выше. Это не самая простая задача, поскольку на данный момент нет общепринятой методики и соответствующего инструментария, с помощью которых подобный анализ можно было бы провести. Требуется оценить и описать эффективность работы с памятью (особенности профиля взаимодействия программы с памятью), эффективность использования заложенного в алгоритм ресурса параллелизма, эффективность использования коммуникационной сети (особенности коммуникационного профиля), эффективность операций ввода/вывода и др. Иногда достаточно интегральных характеристик по работе программы, в некоторых случаях полезно показать данные мониторинга нижнего уровня, например по загрузке процессора, кэш-промахам, интенсивности использования сети Infiniband и т.п. Хорошее представление о работе параллельной MPI-программы дают данные трассировки, полученные, например, с помощью системы Scalasca.

5.6. Выводы для классов архитектур. В данный раздел должны быть включены рекомендации по реализации алгоритма для разных классов архитектур. Если архитектура какого-либо компьютера или платформы обладает специфическими особенностями, влияющими на эффективность реализации, то это здесь нужно отметить.

На практике сказанное сделать можно по-разному: либо все свести в один текущий раздел, либо же соответствующие факты сразу включать в предшествующие разделы, где они обсуждаются и необходимы по смыслу. В некоторых случаях имеет смысл делать отдельные варианты всей части II AlgoWiki применительно к отдельным классам архитектур, оставляя общей машинно-независимую часть I. В любом случае, важно указать и позитивные, и негативные факты по отношению к конкретным классам. Можно говорить о возможных вариантах оптимизации или даже о “трюках” в написании программ, ориентированных на целевые классы архитектур.

5.7. Существующие реализации алгоритма. Для многих пар алгоритм+компьютер уже созданы хорошие реализации, которыми можно и нужно пользоваться на практике. Данный раздел предназначен для того, чтобы дать ссылки на основные существующие последовательные и параллельные реализации алгоритма, доступные для использования уже сейчас. Указывается, является ли реализация коммерческой или свободной, под какой лицензией распространяется, приводится местоположение дистрибутива и имеющихся описаний. Если есть информация об особенностях, достоинствах и/или недостатках различных реализаций, то это также нужно здесь указать. Хорошими примерами реализации многих алгоритмов являются MKL, ScaLAPACK, PETSc, FFTW, ATLAS, Magma и другие подобные библиотеки.

6. Заключение. Энциклопедия AlgoWiki — это исключительно масштабный проект, который, несмотря на молодость, не только активно развивается, но уже сейчас имеет целый ряд перспективных направлений развития. Материалы энциклопедии могут быть эффективно использованы для сравнительного анализа компьютерных платформ на различных классах приложений. В частности, речь может идти о прямом расширении методики, используемой в списке Top500 самых мощных компьютеров мира, которая в настоящее время опирается только на тест LINPACK, за что ее многие исследователи и критикуют. Имея AlgoWiki, можно добавить в часть II для каждого алгоритма отдельный раздел для представления данных прогонов с суперкомпьютерных платформ. Для AlgoWiki такое расширение будет абсолютно естественным, а с учетом открытости энциклопедии и возможности ее коллективного пополнения мы получим огромную базу экспериментальных данных по эффективности выполнения различных алгоритмов на различных платформах, что и станет аналогом Top500, расширенным на любые алгоритмы.

По подобной схеме хотелось бы сделать и описание свойств различных прикладных пакетов, адаптированных под параллельные вычислительные платформы (например, ANSYS CFX, OpenFOAM, FlowVision, Namd, Gromacs, VASP и др.). Пользователи должны сразу видеть, на каких конфигурациях использование пакетов разумно, а когда увеличение числа процессоров не целесообразно.

Важное направление — это использование информации о структуре алгоритмов из AlgoWiki в учебном процессе. Мало знать математическое описание алгоритма, необходимо понимать структуру и особенности всех основных этапов, от формулировки алгоритма до его исполнения. Эти знания необходимы в суперкомпьютерном мире, где все должно быть сделано в полном соответствии с идеями суперкомпьютерного кодизайна, когда структура всех этапов решения задачи должна быть согласована. Однако сейчас все это нужно и в обычном компьютерном мире, где и смартфоны, и планшеты стали параллельными. Все проблемы параллельных вычислений стали актуальными — от сверхмощных суперкомпьютеров до мобильных компьютерных устройств, что и предопределило появление проекта AlgoWiki.

Исследование выполнено в Московском государственном университете им. М. В. Ломоносова за счет гранта Российского научного фонда (проект № 14–11–00190).

СПИСОК ЛИТЕРАТУРЫ

1. *Dongarra J., Beckman P., Moore T., et al.* The international exascale software project roadmap // International Journal of High Performance Computing Applications. 2011. **25**, N 1. 3–60.
2. http://www.exascale.org
3. http://www.eesi-project.eu
4. *Asanović K., Bodik R., Catanzaro B.C., Gebis J.J., Husbands P., Keutzer K., et al.* The landscape of parallel computing research: a view from Berkeley. Report UCB/EECS–2006–183. Berkeley: Univ. of California at Berkeley, 2006.
5. *Knuth D.E.* The art of computer programming. Volumes 1–4. 3rd Edition. Reading: Addison-Wesley, 2011.
6. *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.* Numerical recipes: the art of scientific computing. 3rd Edition. New York: Cambridge Univ. Press, 2007.
7. *Ortega J.M.* Introduction to parallel and vector solution of linear systems. New York: Plenum Press, 1988.
8. *Barrett R., Berry M., Chan T.F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C., van der Vorst H.* Templates for the solution of linear systems: building blocks for iterative methods. 2nd Edition. Philadelphia: SIAM Press, 1994.
9. *Saad Y.* Iterative methods for sparse linear systems. 2nd Edition. Society for Industrial and Philadelphia: SIAM Press, 2003.
10. *Voevodin V.V.* Mathematical foundations of parallel computing. Singapore: World Sci. Publ., 1992.
11. *Voevodin V.V.* Information structure of sequential programs // Russ. J. Numer. Anal. Math. Modelling. 1995. **10**, N 3. 279–286.
12. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
13. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (http://www.netlib.org/linalg/html_templates/Templates.html).
14. A Library of Parallel Algorithms (на языке NESL) (http://www.cs.cmu.edu/~scandal/nesl/algorithms.html).

15. Научно-образовательный Интернет-ресурс НИВЦ МГУ по численному анализу (http://num-anal.srcc.msu.ru/).
16. Wikipedia: List of Algorithms (https://en.wikipedia.org/wiki/List_of_algorithms).
17. *Воеводин В.В., Воеводин Вад.В.* Спасительная локальность суперкомпьютеров // Открытые системы. 2013. № 9. 12–15.
18. *Швец П.А., Воеводин Вад.В.* Метод покрытий для оценки локальности использования данных в программах // Вестник УГАТУ. 2014. **18**, № 1. 224–229.
19. *Антонов А.С., Теплов А.М.* О практической сложности понятия масштабируемости параллельных программ // Высокопроизводительные параллельные вычисления на кластерных системах (HPC 2014): Материалы XIV Международной конференции. Пермь: Изд-во ПНИПУ, 2014. 20–27.
20. *Никитенко Д.А.* Комплексный анализ производительности суперкомпьютерных систем, основанный на данных системного мониторинга // Вычислительные методы и программирование. 2014. **15**, вып. 1. 85–97.

Поступила в редакцию
12.01.2015

An Open AlgoWiki Encyclopedia of Algorithmic Features: From Mobile to Extreme Scale

Vl. V. Voevodin [1]

[1] *Research Computing Center, Lomonosov Moscow State University; Leninskie Gory, Moscow, 119234, Russia; Dr. Sci., Professor, Corresponding Member of Russian Academy of Sciences, Deputy Director, e-mail: voevodin@parallel.ru*

Received January 12, 2015

Abstract: One of the fundamental problems of high performance computing consists in the necessity of a careful matching between the algorithmic structure of parallel programs and the features of a particular computer architecture. The performance capabilities of modern computers are significant; however, the computer's efficiency drastically decreases if such a matching is not achieved even in one of the stages during the process of solving a problem. The AlgoWiki project is based on the fact that the features of algorithms by themselves are not dependent on computing systems. In other words, a detailed description of machine-independent properties of an algorithm should be done only once; after that, this description can be used many times when implementing this algorithm on various hardware/software environments. Also of importance of this project is its machine-dependent part devoted to the description of algorithmic implementation peculiarities with consideration of particular hardware/software platforms. The main result of this project is an open AlgoWiki encyclopedia covering the properties of algorithms and the peculiarities of their implementation on various computing systems. The knowledge of how to reveal, describe, analyze, and interpret the properties of algorithms will become of significant importance in a few years. This conclusion is valid for future exaflop supercomputers and for other computing platforms: from server to mobile devices.

Keywords: parallel computing, structure of algorithms, sequential properties of algorithms, parallel properties of algorithms, supercomputers, computing platforms, efficient implementation of algorithms, scalability, data locality, encyclopedia of algorithms.

References

1. J. Dongarra, P. Beckman, T. Moore, et al., "The International Exascale Software Project Roadmap," Int. J. High Perform. Comput. Appl. **25** (1), 3–60 (2011).

2. http://www.exascale.org. Cited February 22, 2015.

3. http://www.eesi-project.eu. Cited February 22, 2015.

4. K. Asanović, R. Bodik, B. C. Catanzaro, et al., *The Landscape of Parallel Computing Research: A View from Berkeley*, Report UCB/EECS–2006–183 (Univ. California at Berkeley, Berkeley, 2006).

5. D. E. Knuth, *The Art of Computer Programming*, 3rd ed. (Addison-Wesley, Reading, 2011), Vols. 1–4.

6. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge Univ. Press, New York, 2007).

7. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum, New York, 1988).

8. R. Barrett, M. Berry, T. F. Chan, et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd ed. (SIAM Press, Philadelphia, 1994).

9. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. (SIAM Press, Philadelphia, 2003).

10. V. V. Voevodin, *Mathematical Foundations of Parallel Computing* (World Sci. Publ., Singapore, 1992).

11. V. V. Voevodin, "Information Structure of Sequential Programs," Russ. J. Numer. Anal. Math. Modelling **10** (3), 279–286 (1995).

12. V. V. Voevodin and Vl. V. Voevodin, *Parallel Computing* (BHV-Petersburg, St. Petersburg, 2002), 608 pp.

13. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. http://www.netlib.org/linalg/html_templates/Templates.html. Cited February 22, 2015.

14. A Library of Parallel Algorithms. http://www.cs.cmu.edu/~scandal/nesl/algorithms.html. Cited February 22, 2015.

15. The Scientific Educational Internet Resource on Numerical Analysis of the Research Computing Center of Moscow State University. http://num-anal.srcc.msu.ru. Cited February 22, 2015.

16. Wikipedia: List of Algorithms. https://en.wikipedia.org/wiki/List_of_algorithms. Cited February 22, 2015.

17. Vl. V. Voevodin and Vad. V. Voevodin, "The Fortunate Locality of Supercomputers," Otkrytye Sistemy, No. 9, 12–15 (2013).

18. P. A. Shvets and Vad. V. Voevodin, "The Covering Method for Measuring Locality of Programs Memory Usage," Vestn. Ufa Aviatsion. Tekh. Univ. **18** (1), 224–229 (2014).

19. A. S. Antonov and A. M. Teplov, "On Scalability Complexity of Parallel Programs," in *Proc. 14th Int. Conf. on High Performance Computing Using Cluster Systems, Perm, Russia, November 10-12, 2014* (Perm Nat. Res. Univ., Perm, 2014), pp. 20–27.

20. D. A. Nikitenko, "Overall Supercomputer Performance Analysis Based on System Monitoring Data," Vychisl. Metody Programm. **15** (1), 85–97 (2014).