УДК 004.21 doi 10.26089/NumMet.v16r451

МНОГОУРОВНЕВЫЙ ПОДХОД К РАЗРАБОТКЕ АЛГОРИТМИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭКЗАФЛОПСНЫХ СУПЕРЭВМ

Б. М. Глинский[1], И. М. Куликов[2], А. В. Снытников[3], И. Г. Черных[4], Д. В. Винс[5]

Предлагается методология разработки алгоритмического и программного обеспечения для суперкомпьютеров экзафлопсного уровня, содержащая три связанных этапа: первый этап определяется со-дизайном, под которым мы понимаем адаптацию вычислительного алгоритма и математического метода под архитектуру суперкомпьютера на всех этапах разработки программы; на втором предполагается создание упреждающего алгоритмического и программного обеспечения для наиболее перспективных экзафлопсных суперкомпьютеров на основе имитационного моделирования с целью адаптации алгоритмов под заданную архитектуру суперкомпьютера; третий этап связан с оценкой энергоэффективности алгоритма при различных реализациях на данной архитектуре либо на различных архитектурах. Данный подход иллюстрируется примерами решения двух задач из области астрофизики и физики плазмы.

Ключевые слова: экзафлопсные вычисления, со-дизайн, энергоэффективность, агентное моделирование.

1. Введение. Основная сложность применения перспективных экзафлопсных суперкомпьютеров будет определяться экстремальной степенью параллелизма: сотни миллионов и миллиарды одновременно работающих вычислительных ядер. Тем самым уровень развития супервычислений для вычислительных систем этого класса связан как с разработкой структур самих суперкомпьютеров, так и с совершенствованием их алгоритмического обеспечения, что позволяет отнести задачу исследования масштабируемости параллельных алгоритмов для оценки эффективности их реализации на будущих экзафлопсных суперкомпьютерах к актуальным задачам этого научного направления. Следует отметить, что вычислительные алгоритмы, как правило, являются более консервативными по сравнению с развитием средств вычислительной техники, поэтому, пока не будет разработана специальная математика, учитывающая особенности экзафлопсных суперкомпьютеров, будут применяться алгоритмы, реализованные на современных суперкомпьютерах. Следовательно, важно уже сейчас оценить перспективы их реализации на будущих суперкомпьютерах, исследовать параллелизм, масштабируемость и энергопотребление.

В работе [1] вводится понятие со-дизайна, подразумевающее необходимость анализа эффективности параллельной реализации на всех этапах разработки алгоритма. В данном случае со-дизайн начинается еще на уровне физической постановки задачи. Последний элемент со-дизайна — доработка параллельного вычислительного алгоритма с учетом архитектуры суперкомпьютера.

Задача моделирования масштабируемых алгоритмов не является новой, ею занимаются многие группы исследователей во всем мире [2]. Хорошим примером ранних работ является программный пакет PARSIT [3], с помощью которого можно моделировать поведение параллельной программы с учетом архитектуры ЭВМ. Кроме того, заслуживают внимания работы [4, 5], в которых предложены программы для моделирования алгоритмов, однако они достаточно просты и не подходят для моделирования алгоритмов, связанных с высокопроизводительными вычислениями. Следует упомянуть о проекте BIGSIM [6]. Проект направлен на создание имитационного окружения, позволяющего проводить разработку, тестирование и настройку посредством моделирования ЭВМ будущих поколений, одновременно позволяя разработчикам ЭВМ улучшать их проектные решения с учетом специального набора приложений.

[1] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; зав. лабораторией, профессор, e-mail: gbm@sscc.ru
[2] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; мл. науч. сотр., e-mail: kulikov@ssd.sscc.ru
[3] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; науч. сотр., e-mail: snytav@gmail.com
[4] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; ст. науч. сотр., e-mail: chernykh@parbz.sscc.ru
[5] Институт вычислительной математики и математической геофизики СО РАН, просп. Лаврентьева, д. 6, 630090, Новосибирск; мл. науч. сотр., e-mail: wns.dmitry@gmail.com

В Институте системного программирования РАН (г. Москва) разработана модель параллельной программы, которая может эффективно интерпретироваться на инструментальном компьютере, обеспечивая возможность достаточно точной оценки времени реального выполнения параллельной программы на заданном параллельном вычислительном комплексе. Модель разработана для параллельных программ с явным обменом сообщениями, написанных на языке Java с обращениями к библиотеке MPI, и включена в состав среды ParJava [7, 8]. Предсказание времени счета отдельных участков параллельной программы производится с учетом затрат, связанных с управлением MPI, т.е. производится корректировка модельных часов с учетом средней доли процессорного времени, которую занимает нить RTS (RunTimeSystem). Таким образом, проект ParJava, с одной стороны, позволяет решать широкий круг задач по оценке эффективности исполнения параллельных программ на перспективных вычислительных системах, но, с другой стороны, привязан к конкретному языку программирования, что существенно сужает его возможности.

В работах [8, 9] развивается подход, основанный на мультиагентном моделировании, который органично подходит для задачи имитации вычислений. В качестве атомарной, независимой частицы в модели вычислений выбран вычислительный узел и исполняемый на нем код алгоритма. Каждый функциональный агент эмулирует поведение вычислительного узла кластера и программу вычислений, работающую на этом узле. Вычисления представляются в виде набора примитивных операций (вычисление на ядре; запись/чтение данных в память; парный обмен данными; синхронизация данных между вычислителями) и временны́х характеристик каждой операции. Таким образом, оценить поведение алгоритмов, разработать модифицированные схемы вычислений можно уже сейчас путем реализации их на имитационной модели, отображающей миллионы и десятки миллионов вычислительных ядер. Имитационная модель позволяет выявить узкие места в алгоритмах, понять, как нужно модифицировать алгоритм, какие параметры необходимо настраивать при его масштабировании на большое количество ядер.

Так как вопрос архитектуры будущего экзафлопсного суперкомпьютера остается открытым, важным фактором становится энергоэффективность алгоритмов. Энергоэффективность алгоритмов можно разделить на три основных составляющих [10–14]: эффективность работы с процессором и оперативной памятью, эффективность работы с сетевыми устройствами и эффективность работы с периферическими системами, например с системами ввода-вывода данных. Для увеличения энергоэффективности работы с процессором и памятью рекомендуется максимально эффективно использовать аппаратные возможности платформы, под которую пишется программа, при этом рекомендуется максимально упрощать код, облегчая его оптимизацию компилятору. Крайне необходимо следить за использованием памяти. Снижение потерей ресурсов памяти в результате многократного размещения данных без последующего ее освобождения (reduce memory leaks) позволяет значительно сократить обращение операционной системы к файлу подкачки. Использование высокопроизводительных библиотек (например, Intel IPP или Intel MKL) позволяет значительно улучшить качество программы и ее энергоэффективность. При разработке программы рекомендуется максимально сократить объем передаваемых данных между процессами и минимизировать количество операций передачи данных. Значительно улучшить энергоэффективность можно за счет балансировки загрузки процессов. В случае необходимости простоя некоторых процессов, например с целью сбора данных со всех вычислительных узлов, рекомендуется принудительно понижать частоту ядер/процессоров, обслуживающих эти операции. Энергопотребление различных периферических систем вычислительных узлов порой может достигать и превосходить энергопотребление процессоров и ускорителей вычислений. Необходимо максимально сократить обращения к файловой системе, особенно в случае использования сетевой параллельной файловой системы. Финальная версия разрабатываемого кода не должна содержать отладочной информации, должны быть максимально сокращены все процедуры ввода/вывода данных. Кроме того, значительно повышает энергоэффективность алгоритмов использование последних версий компиляторов с максимальной оптимизацией кода под архитектуру целевой платформы. Например, последние версии компиляторов Intel позволяют получить рекомендации по оптимизации кода, что может увеличить производительность алгоритма на 30% на последних версиях процессоров и ускорителей вычислений Intel.

2. Со-дизайн методов решения вычислительно сложных задач. Со-дизайн параллельных методов решения больших задач — достаточно сложный и неформализуемый процесс. Это связано прежде всего с тем, что каждая задача, даже из одной области знаний, имеет свои особенности, учет которых позволяет достаточно серьезно изменить не только параллельный вычислительный метод, но и математическую модель, а также программную реализацию и ее эффективность.

Конечно же, невозможно собрать “сборник рецептов” для эффективного решения всех крупномасштабных задач. Однако некоторые общие подходы, на наш взгляд, сформулировать вполне возможно. В любом случае, на сегодняшний день концепцию со-дизайна можно сформулировать в виде шести взаимо-

связанных пунктов:

1) формулировка “физической” постановки задачи;

2) математическая формулировка “физической” задачи;

3) создание численного метода для реализации математической формулировки;

4) выбор структур данных и параллельного алгоритма;

5) учет особенностей архитектуры суперЭВМ;

6) использование инструментальных средств разработки.

На этапе “физической” формулировки задачи мы уже можем определить процесс, который будет основным и относительно которого мы будем строить все остальные этапы. Так, например, моделирование газодинамических течений в зависимости от постановки задачи может быть сведено в случае расчета динамики сопла к решению гиперболических уравнений — уравнений газовой динамики [15], а в случае движения летательного аппарата в разреженной атмосфере уже кинетическими уравнениями, которые могут быть решены методами частиц [16] или методами Монте-Карло [17]. Все эти методы имеют разную природу и, следовательно, требуют различных подходов к параллельной реализации.

Тем не менее, уже на этапе “физической” постановки задачи мы начинаем использовать некоторый аппарат формализации и можем определить тип параллелизма, который будет использоваться в решении задачи. На наш взгляд, наиболее естественным является разделение типов параллелизма на распределенные задачи и задачи с зависимым параллелизмом. Не сужая общность, все распределенные задачи можно условно отнести к задачам интегрирования в том смысле, что задача разделяется на множество независимых подзадач, результаты работы которых аккумулируются в некотором разделяемом объекте. В случае задач численного интегрирования или суммирования в задачах квантовой химии [18] и методов Монте-Карло [17] результатом является набор некоторых скалярных характеристик. В задачах спектрального анализа основной проблемой является получение матрицы динамического процесса или двумерного графика спектра матриц [19, 20]. В обоих случаях результаты аккумулируются в некотором разделяемом объекте. То же самое касается задач обработки изображений [21]. Однако и задача интегрирования в случае использования суперЭВМ экзафлопсного класса требует доработки параллельного алгоритма решения [8].

Задачи с зависимым параллелизмом, на наш взгляд, можно условно разбить на три типа: задачи явного обращения оператора; задачи неявного (скрытого) обращения оператора, решаемые сеточными методами; задачи неявного (скрытого) обращения оператора, решаемые бессеточными методами. Если говорить о задачах, сводящихся к обращению операторов, — это прежде всего решение эллиптических уравнений [23, 24] либо решение задач, сводимых к параболическим и гиперболическим [25] уравнениям. В этом случае основной задачей является задача обращения разреженной матрицы итерационными [26] или прямыми методами [27], в том числе и методами, основанными на сведении матриц к более простой структуре: к двухдиагональной [19] или трехдиагональной [28] форме. Это же относится и к методам решения систем обыкновенных дифференциальных уравнений (ОДУ) [29].

Неявное обращение оператора с помощью сеточных методов — это решение гиперболических уравнений на компактном шаблоне большинства моделей сплошной среды: газовая динамика [30], магнитная газовая динамика [31], первых моментов уравнения Больцмана для описания звездного компонента галактик [32, 33], уравнений релятивистской газовой динамики [34], уравнений релятивистской газовой динамики с электрическим полем [35, 36], уравнений теории упругости [37], многофазной гидродинамики [38], гравитационной газовой динамики [39–41] и др. В этом случае основной стратегией эффективности параллельной реализации является использование локального вычислительного шаблона, а следовательно, и локальных (или линейных [42]) вычислений. Неявное обращение оператора с помощью бессеточных методов — это решение в том числе и гиперболических уравнений с помощью представления сплошной среды газа [43] или твердого тела [44] в виде частиц с ограниченным радиусом взаимодействия или разреженной среды.

2.1. Со-дизайн параллельных численных методов для моделирования задач астрофизики. За несколько лет авторами статьи на основе метода течения жидкостей в клетках *крупных частиц* (FLIC: FLuid In Cell) и метода Годунова был разработан эйлерово–лагранжев подход к решению астрофизических задач. Использование такой комбинации позволяет получить галилеево инвариантное решение, а использование метода Годунова на эйлеровом этапе позволяет также правильно моделировать разрывы.

Архитектура GPU представляет собой набор параллельно выполняющихся нитей, объединенных многоуровневой 2D-топологией. Максимальную производительность при использовании данной архитектуры можно достичь при отсутствии синхронизации между нитями. Для такой архитектуры со-дизайн численного метода состоит в использовании топологии вычислительных сеток, которые непосредственно проецируются на GPU-архитектуру (например, регулярные сетки), и независимых вычислений в каждой ячейке расчетной области значений для следующего шага. Задача Римана на эйлеровом этапе численного метода вполне удовлетворяет данным характеристикам, но на лагранжевом этапе требовались изменения. Внесенные изменения позволяют независимо выполнить адвективный перенос для каждой ячейки расчетной области.

Традиционно, бесстолкновительная компонента описывается с помощью модели N тел. Тем не менее, эта модель имеет недостатки, такие как ложное введение энтропии, большое число лишних обменов и плохая балансировка нагрузки. Поэтому во время со-дизайна алгоритмов для описания бесстолкновительной компоненты в контексте моделирования столкновения галактик был выбран подход, именуемый "бесстолкновительная звездная гидродинамика". Этот подход основан на уравнениях для первых моментов бесстолкновительных уравнений Больцмана. Выбранный подход адекватно воспроизводит бесстолкновительные свойства кластера частиц. Кроме того, для описания "Collisionless Stellar Hydrodynamics" моделей могут быть использованы единые численные методы и параллельные алгоритмы, применяемые для решения гидродинамических уравнений. Следовательно, они могут быть реализованы на суперкомпьютерах, использующих графические ускорители.

2.2. Со-дизайн параллельных численных методов для решения задач физики плазмы. В данном случае со-дизайн начинается с этапа физического представления задачи. Из эксперимента известно, что модуляция плотности плазмы не превышает 30%. Это означает, что количество модельных частиц в отдельной ячейке не может увеличиваться неограниченно. Таким образом, необходимость в динамической балансировке загрузки отсутствует, что повышает надежность и эффективность параллельной реализации.

На уровне математической модели предпочтение было отдано расчету поля на основе законов Фарадея и Ампера, без явного решения уравнения Пуассона. Использование этих уравнений и схемы Лэнгдона–Лазинского дает возможность получить метод расчета поля с почти неограниченным масштабированием при увеличении числа процессорных ядер.

На этапе выбора суперкомпьютерной архитектуры учитываются особенности метода частиц в ячейках (PIC: Particle-In-Cell). Чтобы получить новые значения местоположения и импульсов частиц, необходимо знать характеристики электронного и магнитных полей частицы в текущей позиции. Каждое из трех значений электронного и магнитных полей хранится в отдельном 3D-массиве. Таким образом, на каждом временно́м шаге хранятся шесть 3D-массивов для каждой частицы. Так как частицы в области располагаются случайным образом, доступ к этим массивам так же не упорядочен. Это означает, что использование кэш-памяти не поможет уменьшить время вычислений. Если в процессе движения частицы часть массива поля была перенесена в кэш-память, то становится невозможным использовать ее для вычисления другой частицы, так как она (наиболее вероятно) располагается в совершенно другой части подобласти. Так как кэш-память не может вместить все 6 массивов для полей, необходимо использование оперативной памяти (RAM) для расчета движения частицы. А так как производительность процессора, как правило, ограничивается пропускной способностью памяти, именно она влияет на скорость вычислений параметров частиц и производительность программы в целом (время вычисление параметров частицы занимает от 60% до 90% всего времени выполнения). Это ограничение определяет необходимость перехода к суперкомпьютерам, использующим графические процессоры (GPU), так как технология CUDA предоставляет обширный набор инструментов для ускорения работы с памятью при решении задач метода частиц в ячейках.

Этап выбора инструментов программирования заключается в следующем. Использование технологии CUDA для метода частиц в ячейках с очень большим количеством независимо обрабатывающихся элементов (моделируемые частицы) является наиболее эффективным. Кроме того, можно использовать другие параллельные технологии, такие как OpenCL, OpenMP, OpenACC, но именно CUDA дает возможность запустить наибольшее количество параллельных процессов и получить наилучшую производительность.

Последний этап со-дизайна — доработка параллельного вычислительного алгоритма с учетом архитектуры графических ускорителей. Традиционная реализация метода частиц в ячейках, когда все частицы хранятся в одном очень большом массиве, неприемлема для графических ускорителей, так как он не позволяет использовать преимущества GPU-памяти. В этой связи частицы разделяются по ячейкам, а процесс вычислений происходит по такой схеме: один блок нитей на одну ячейку.

3. Имитационное моделирование масштабируемости алгоритмов. Для исследования поведения алгоритмов была использована система мультиагентного моделирования AGNES (AGent NEtwork Simulator) [45]. В данной системе для каждого из исследуемых алгоритмов были созданы агенты, имитирующие поведение вычислительных узлов при выполнении соответствующих алгоритмов (AstroGrid и PlasmaGrid). Эти агенты, имитируя поведение вычислительных узлов, моделируют вычисления для каждой из задач и отправку данных соседям.

Агент AstroGrid отправляет сообщения двум соседям и ожидает поступления сообщений от обоих соседей. Только после этого продолжается цикл вычислений.

Агенты PlasmaGrid суммируют данные всех агентов перед следующей итерацией цикла.

Собираются и анализируются задержки при передаче данных на соответствие с данными реальных запусков кодов астрофизики и физики плазмы.

3.1. Моделирование астрофизического кода. Модель взаимодействующих процессов представляет собой множество параллельно выполняемых на индивидуальном вычислительном узле нитей, которые взаимодействуют между собой посредством обмена значениями, например средствами MPI. Главной характеристикой нити является время выполнения и время передачи данных другому вычислительному узлу. Особенностью параллельных сеточных методов является возможность геометрической декомпозиции расчетной области и последующего обмена граничными значениями только между соседними узлами. Эти методы являются основным инструментом решения гиперболических уравнений не только для задач астрофизики. При этом, если в подобласти содержится N^3 ячеек, то число передаваемых элементов соседнему вычислительному узлу составляет N^2 элементов. Исходя из этих соображений, представим схемы взаимодействий между нитями для задач астрофизики и физики плазмы, приведенной на рис. 1.

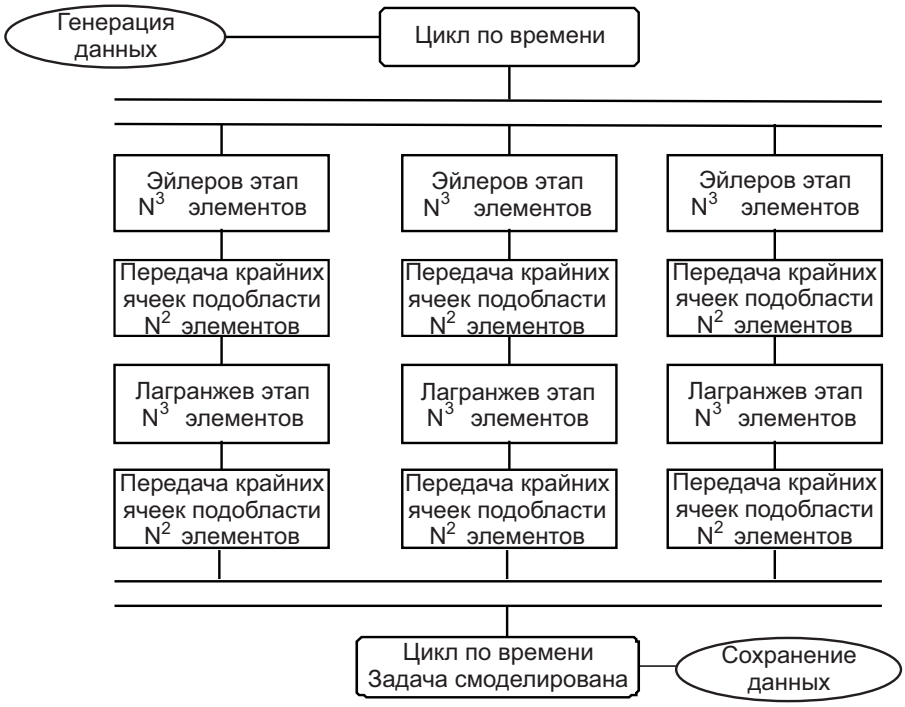

Рис. 1. Схема взаимодействия между вычислительными узлами

Для исследования масштабируемости сеточных методов на центральном процессоре, графических ускорителях и ускорителях Intel Xeon Phi будем исходить из следующих предположений.

1. Для нахождения времени выполнения вычислений на эйлеровом и лагранжевом этапе будем использовать следующие формулы:

$$T^{\mathrm{Euler}} = T_E \times N^3, \quad T^{\mathrm{Lagrange}} = T_L \times N^3,$$

т.е. для каждого типа вычислителя мы знаем оптимальное время вычислений одной ячейки на обоих этапах численного метода. Общее время вычислений подобласти вычисляется пропорционально числу ячеек.

2. Мы не будем моделировать масштабируемость и ускорение расчета одной ячейки в рамках одного ускорителя или центрального процессора. Это связано с тем, что в настоящее время количество ядер вычислительных элементов фиксировано и намного меньше, чем потенциальное число вычислительных узлов пета- и экзафлопсных суперкомпьютеров. Таким образом, время выполнения этапов $T_{E,L}$ является оптимальным для:

   - 12 ядер процессора Intel Xeon E5-2697;
   - 1024 ядер графического ускорителя Nvidia Kepler;
   - 60 ядер ускорителя Intel Xeon Phi 7110 в режиме выполнения offload;
   - 240 ядер ускорителя Intel Xeon Phi 7120 в режиме выполнения native.

   В случае центрального процессора имело место 12-кратное ускорение, в случае графического ускорителя 55-кратное ускорение, в случае ускорителя Intel Xeon Phi в режиме offload 27-кратное ускорение, а в режиме native 54-кратное ускорение. Это ускорение связано с архитектурой диспетчера задач вычислительного элемента и введено в модель с помощью проведенных вычислительных экспериментов средствами кодов GPUPEGAS и AstroPhi.

3. Время выполнения коммуникаций будем считать линейной функцией от числа передаваемых элементов с учетом латентности:
$$T^{\mathrm{Comm}} = \Lambda + T_C \times N^2,$$
где Λ — латентность при передаче данных, T_C — среднее время передачи одного элемента данных.

4. Особенностью построения численного метода является тот факт, что количество передаваемых элементов, а следовательно, и время передачи, после эйлерового и лагранжевого этапов одинаково.

5. Так как вычислительные эксперименты были проведены на разном оборудовании и на различных архитектурах, а в некоторых случаях это был один узел или вообще один графический ускоритель, то вычисление и тем более сравнение времени выполнения коммуникаций невозможно. Поэтому мы будем предполагать, что для исследования масштабируемости системы используется сетевая инфраструктура ССКЦ, что является вполне оправданным предположением.

6. В общем случае при решении трехмерных задач на большом числе узлов наиболее эффективный способ геометрической декомпозиции — это многомерная (в том числе трехмерная) декомпозиция. Скорее всего, именно многомерная декартова топология будет использована и в экзафлопсных архитектурах и активно используется уже и сейчас. На рис. 1 изображена линейная топология, что соответствует реальной архитектуре ССКЦ. Использование такой топологии взаимодействия между узлами позволяет значительно упростить имитационное моделирование выполнения нитей и в то же время не ограничивает использование многомерных топологий. Это связано с тем, что обмен между нитями будет происходить пропорционально числу измерений, т.е. время коммуникаций при K-мерной декомпозиции связано с временем коммуникаций по одному направлению по формуле $T^{\mathrm{Comm}}_{KD} = K \times T^{\mathrm{Comm}}_{1D} = K \times T^{\mathrm{Comm}}$, что не нарушает общности рассмотрения представленной на рис. 1 модели.

7. Кроме того, в нашей модели мы рассматриваем только экстенсивность вычислительной системы. Естественно, что при большем распространении петафлопсных вычислений и движении к экзафлопсным вычислениям будет расти как скорость вычислителей, так и скорость обмена данными. Вместе с этим также будет расти и вычислительная сложность, так как будет усложняться физическая модель астрофизических процессов, что даст усложнение численного метода и рост числа операций на одну ячейку. Поэтому в рамках данной модели мы исследуем масштабируемость параллельно выполняющихся процессов с учетом текущего уровня характеристик оборудования и, как следствие, сложности вычислительной модели. Так, в настоящее время характерное число ячеек на один ускоритель составляет величину $N = 256^3 \approx 16 \times 10^5$.

3.2. Моделирование кода физики плазмы. В данном разделе продемонстрирована стратегия распараллеливания, поддерживающая шаблон разработки методов частиц в ячейках, так как очевидно, что реализации для решения задач физики плазмы никогда не будут укладываться в решение с одним CPU.

Программа распараллеливается методом декомпозиции. Расчетная область делится на части по направлению, перпендикулярному направлению пучка (вдоль оси Y), пучок следует вдоль оси X. Расчетная сетка по всей области делится на равные части (подобласти) вдоль оси Y. Каждая подобласть назначается группе процессоров (в случае многоядерных систем отдельное ядро будем считать процессором, так как не используется никакого гибридного распараллеливания типа MPI+OpenMP, а только лишь MPI). Кроме того, частицы каждой подобласти распределены равномерно между процессорами группы без учета их положения, как это показано на рис. 2.

Расчетная область разделена на 4 подобласти. Частицы каждой подобласти распределены между четырьмя процессорами равномерно независимо от их местонахождения. Частицы, принадлежащие разным процессорам из одной подобласти, также обозначаются различными символами (круг, квадрат, алмаз, звезда).

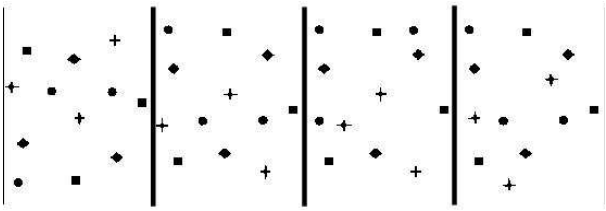

Рис. 2. Схема декомпозиция области

Каждый процессор из группы решает уравнение Максвелла по всей подобласти и обменивается граничными значениями полей с процессорами, назначенными для соседних подобластей. Затем каждый процессор решает уравнения движения частиц и вычисляет 3D-матрицу плотности тока и плотности заряда. Так как процессор располагает только частью частиц из подобласти, то для того чтобы получить полную матрицу текущей плотности в подобласти, необходимо суммировать матрицы всех процессоров группы.

Межпроцессорный обмен данными осуществляется подпрограммами MPI. Общая схема связи в целом соответствует схеме, приведенной на рис. 1 для астрофизического кода, но существует одно важное отличие — в настоящее время код физики плазмы не предусматривает связи все-со-всеми (peer-to-peer), поэтому моделирование всей области выполняется командой MPI_Allreduce.

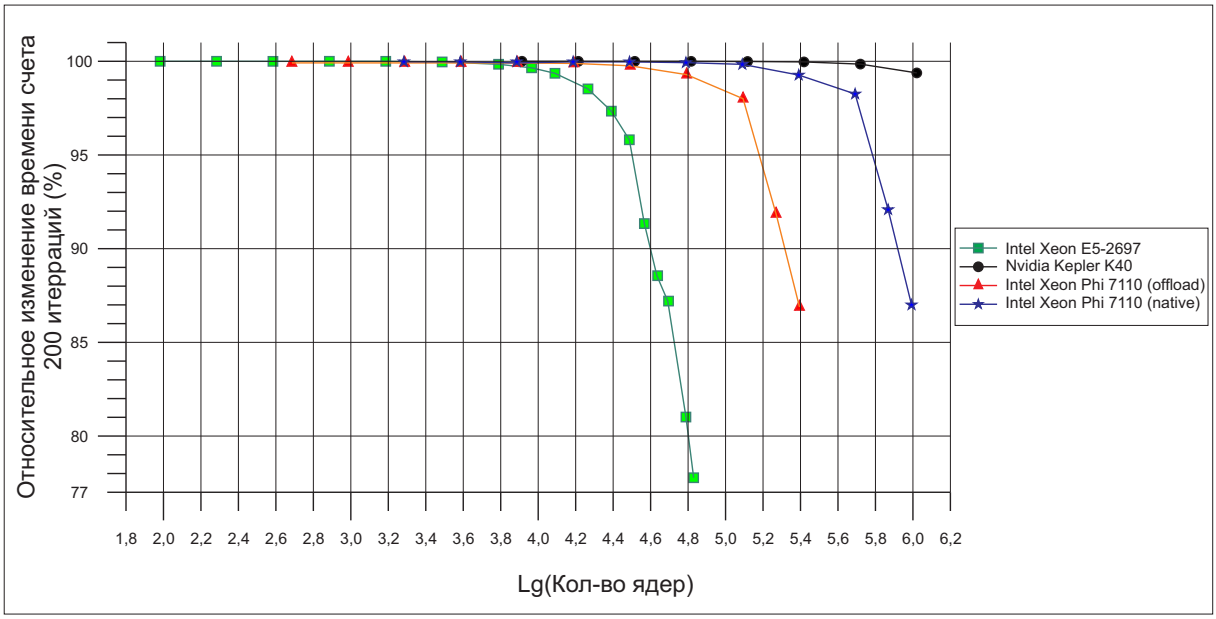

Рис. 3. Результаты моделирования с помощью AGNES масштабируемости астрофизического кода. Размер сетки увеличивается с количеством ядер

3.3. Результаты моделирования. Моделирование выполнения алгоритмов было проведено по данным их реальной работы — для астрофизического кода на узлах с Intel Xeon E5-2697 до 5632 узла (67 584 ядра), на узлах с GPU Nvidia Kepler до 1024 узлов (1 048 576 ядер), на узлах с Intel Xeon Phi 7110 до 4096 узлов (245 760–983 040 ядер); для кода физики плазмы на узлах с Intel Xeon E5-2697 до 3072 узлов (36 864 ядра), на узлах с GPU Nvidia Tesla 2090M до 1536 узлов (786 432 ядер), на узлах с Nvidia Kepler K40 до 1536 узлов (4.4 миллиона ядер).

Результаты моделирования астрофизического кода (рис. 3):

— время выполнения растет несущественно (около 20%) для 5120 вычислительных узлов;

— лучшую масштабируемость показали вычислительные узлы с ускорителями Nvidia Kepler K40 и Intel Xeon Phi (native mode).

Результаты моделирования кода физики плазмы (рис. 4):

— существенное увеличение коммуникационных взаимодействий увеличивает на 58% время выполнения кода для 3072 вычислительных узлов с Intel Xeon CPUs;

— код физики плазмы хорошо масштабируется на вычислительных узлах с ускорителями Nvidia Kepler K40 GPUs.

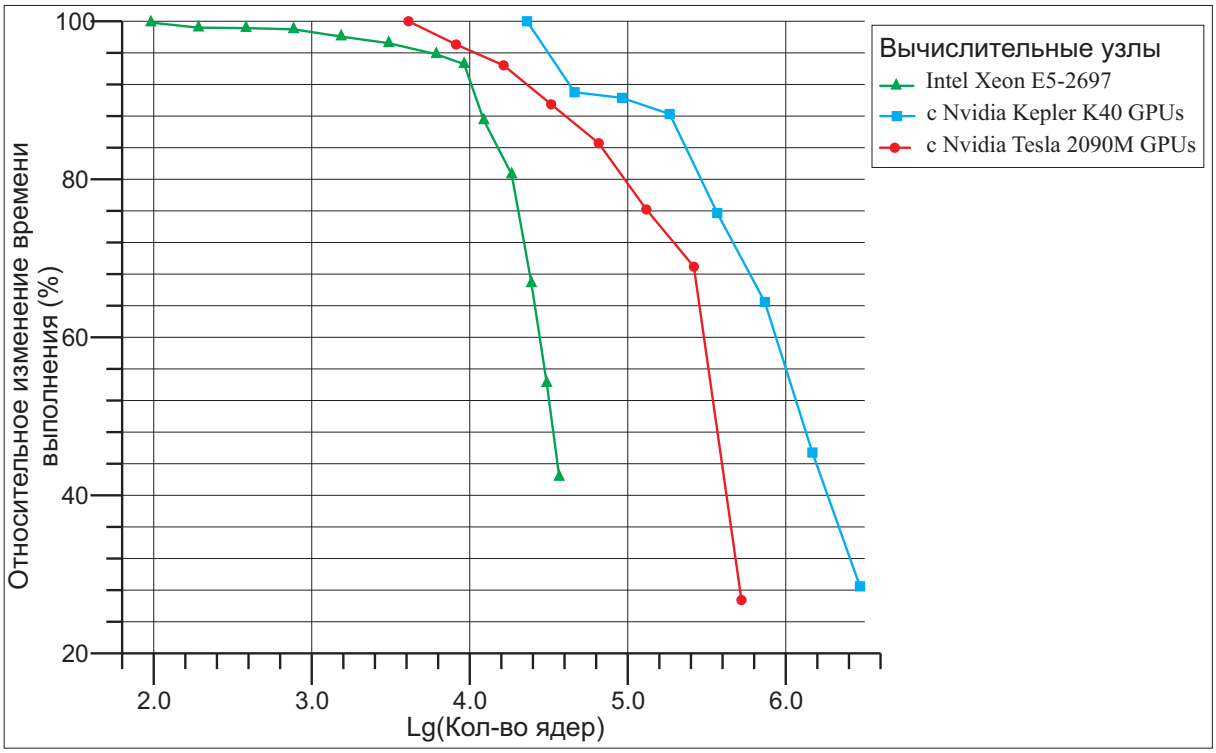

Рис. 4. Результаты моделирования с помощью AGNES масштабируемости кода физики плазмы. Размер области декомпозиции увеличивается с количеством ядер

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	78.174 s		80.0835 s	4	19.5435 s
Process 0	19.2928 s		20.021 s	1	19.2928 s
Process 1	19.8552 s		20.0281 s	1	19.8552 s
Process 2	19.9098 s		20.0216 s	1	19.9098 s
Process 3	19.1162 s		20.0128 s	1	19.1162 s
Group MPI	1.90951 s		1.90951 s	384	4.97269e-3 s

Рис. 5. Профиль исполнения астрофизического кода на CPU Intel Core-i7 3820 Sandy Bridge

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	154.396 s		157.082 s	4	38.5989 s
Process 0	38.4594 s		39.2725 s	1	38.4594 s
Process 1	38.9564 s		39.2733 s	1	38.9564 s
Process 2	38.9702 s		39.2756 s	1	38.9702 s
Process 3	38.0096 s		39.2609 s	1	38.0096 s
Group MPI	2.68675 s		2.68675 s	384	6.99675e-3 s

Рис. 6. Профиль исполнения астрофизического кода на рабочем узле SMP-G7 (Intel E7-4870) ЦКП ССКЦ СО РАН

4. Энергоэффективность алгоритмов. С целью повышения энергоэффективности алгоритмов для задач астрофизики и физики плазмы каждый программный код оптимизировался по следующим

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	44.1507e+3 s		45.0657e+3 s	16	2.75942e+3 s
Process 0	2.75706e+3 s		2.81678e+3 s	1	2.75706e+3 s
Process 1	2.7155e+3 s		2.81672e+3 s	1	2.7155e+3 s
Process 2	2.73732e+3 s		2.81674e+3 s	1	2.73732e+3 s
Process 3	2.73277e+3 s		2.81675e+3 s	1	2.73277e+3 s
Process 4	2.73852e+3 s		2.81672e+3 s	1	2.73852e+3 s
Process 5	2.70039e+3 s		2.81675e+3 s	1	2.70039e+3 s
Process 6	2.73803e+3 s		2.81674e+3 s	1	2.73803e+3 s
Process 7	2.75461e+3 s		2.81677e+3 s	1	2.75461e+3 s
Process 8	2.78787e+3 s		2.8166e+3 s	1	2.78787e+3 s
Process 9	2.78898e+3 s		2.81653e+3 s	1	2.78898e+3 s
Process 10	2.78098e+3 s		2.81635e+3 s	1	2.78098e+3 s
Process 11	2.78907e+3 s		2.81662e+3 s	1	2.78907e+3 s
Process 12	2.78067e+3 s		2.81623e+3 s	1	2.78067e+3 s
Process 13	2.79598e+3 s		2.8167e+3 s	1	2.79598e+3 s
Process 14	2.77751e+3 s		2.81614e+3 s	1	2.77751e+3 s
Process 15	2.77545e+3 s		2.81654e+3 s	1	2.77545e+3 s
Group MPI	914.946 s		914.946 s	1776	515.172e-3 s

Рис. 7. Профиль исполнения астрофизического кода на рабочем узле G8-K40 (Intel Xeon E5-2650v2) ЦКП ССКЦ СО РАН

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	128.395 s		136.26 s	10	12.8395 s
Process 0	12.8576 s		13.7542 s	1	12.8576 s
Process 1	13.0029 s		13.6415 s	1	13.0029 s
Process 2	13.1126 s		13.6327 s	1	13.1126 s
Process 3	12.6506 s		13.5972 s	1	12.6506 s
Process 4	13.1373 s		13.5917 s	1	13.1373 s
Process 5	12.703 s		13.5807 s	1	12.703 s
Process 6	12.3619 s		13.6444 s	1	12.3619 s
Process 7	12.1864 s		13.6153 s	1	12.1864 s
Process 8	13.1687 s		13.6079 s	1	13.1687 s
Process 9	13.2136 s		13.5945 s	1	13.2136 s
Group MPI	7.86551 s		7.86551 s	3200210	2.45781e-6 s

Рис. 8. Профиль исполнения кода физики плазмы на G8-K20 nodes (Intel Xeon E5-2650v2 and 3 Nvidia Tesla M2090) ЦКП ССКЦ СО РАН

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	389.82 s		422.531 s	30	12.994 s
Process 0	12.5916 s		13.9837 s	1	12.5916 s
Process 1	12.941 s		14.2177 s	1	12.941 s
Process 2	12.9323 s		14.2493 s	1	12.9323 s
Process 3	12.5263 s		14.037 s	1	12.5263 s
Process 4	13.4026 s		14.132 s	1	13.4026 s
Process 5	13.6495 s		14.1396 s	1	13.6495 s
Process 6	12.9546 s		14.0106 s	1	12.9546 s
Process 7	13.4088 s		14.1386 s	1	13.4088 s
Process 8	13.5501 s		14.1408 s	1	13.5501 s
Process 9	13.0014 s		14.0147 s	1	13.0014 s
Process 10	12.6144 s		14.1758 s	1	12.6144 s
Process 11	12.828 s		14.1234 s	1	12.828 s
Process 12	12.3164 s		14.0278 s	1	12.3164 s
Process 13	12.6281 s		14.159 s	1	12.6281 s
Process 14	13.1659 s		14.0234 s	1	13.1659 s
Process 15	13.3675 s		14.0839 s	1	13.3675 s
Process 16	13.1959 s		14.1117 s	1	13.1959 s
Process 17	13.3358 s		14.168 s	1	13.3358 s
Process 18	13.4274 s		14.1031 s	1	13.4274 s
Process 19	13.2678 s		14.1091 s	1	13.2678 s
Process 20	12.3406 s		14.0329 s	1	12.3406 s
Process 21	12.8743 s		14.08 s	1	12.8743 s
Process 22	12.3167 s		13.9653 s	1	12.3167 s
Process 23	12.314 s		14.0282 s	1	12.314 s
Process 24	13.3187 s		14.1198 s	1	13.3187 s
Process 25	13.4332 s		14.023 s	1	13.4332 s
Process 26	12.8908 s		13.992 s	1	12.8908 s
Process 27	13.1831 s		14.1562 s	1	13.1831 s
Process 28	13.09 s		14.0077 s	1	13.09 s
Process 29	12.9528 s		13.9768 s	1	12.9528 s
Group MPI	32.7115 s		32.7115 s	9600660	3.40721e-6 s

Рис. 9. Профиль исполнения кода физики плазмы на G8-K20 nodes (2 Intel Xeon E5-2650v2 and 3 Nvidia Tesla M2090) ЦКП ССКЦ СО РАН

трем направлениям: эффективность работы с процессором и оперативной памятью, эффективность работы с сетевыми устройствами и эффективность работы с периферийными устройствами, например с

системами ввода-вывода данных. Для эффективной работы с процессором и памятью использовался пакет Intel Parallel Studio XE 2016 Beta, включающий в себя Intel Vectorization Advisor, который анализирует код и выдает предложения для оптимизации программы с целью максимизации эффективности использования векторных инструкций. Использование этого продукта дало прирост производительности в 30% для астрофизического кода на процессорах семейства Intel Sandy Bridge. Для задач астрофизики и физики плазмы были выбраны наиболее эффективные численные методы с точки зрения минимизации сетевых обменов.

Оптимизация численных методов астрофизического кода и использование последних версий программных продуктов из пакета Intel Parallel Studio XE 2016 Beta дали отличный результат балансировки нагрузки на ядрах процессоров. Время выполнения MPI операций составляет не более 2–3% от общего времени счета. На рис. 5–7 представлены профили выполнения программ. На рис. 8–9 представлены профили кода физики плазмы. Благодаря предлагаемому подходу удалось выбрать набор численных методов таким образом, что время выполнения MPI-операций не составляет больше 10% и имеет место хорошая балансировка нагрузки.

5. Заключение. В настоящей статье предложена методика разработки алгоритмов и программного обеспечения для суперкомпьютеров экзафлопсной производительности. Данный подход имеет три взаимосвязанных этапа: со-дизайн, моделирование масштабируемости и мероприятия по повышению энергоэффективности. Идея со-дизайна состоит в том, что разработка алгоритма и программного обеспечения ведется с учетом архитектуры суперкомпьютера. Одним из наиболее важных этапов в предлагаемом подходе является моделирование масштабируемости. Для того чтобы оценить масштабируемость алгоритмов на архитектуре перспективных экзафлопсных суперкомпьютеров, был использован программный пакет AGNES. Этап повышения энергоэффективности состоит из анализа возможных модернизаций кода, таких как увеличение использования CPU или GPU, балансировка нагрузки и уменьшение числа MPI-операций. Предлагаемый подход проиллюстрирован на двух примерах — задачи астрофизики и задачи физики плазмы. В результате использования данного подхода при разработке программного продукта мы получили хорошо распараллеленные коды с подходящей балансировкой нагрузки и приемлемым временем исполнения MPI-операций.

Работа выполнена при финансовой поддержке РФФИ (гранты 15–31–20150, 15–01–00508, 13–07–00589, 14–01–31199 и 14–01–00392) и в рамках гранта Президента РФ для поддержки молодых ученых номер МК–6648.2015.9. Данный проект частично поддержан Министерством образования и науки РФ, грант 3.961.2014/К.

Статья рекомендована к публикации Программным комитетом Международной научной конференции “Суперкомпьютерные дни в России 2015” (http://russianscdays.org).

СПИСОК ЛИТЕРАТУРЫ

1. *Glinskiy B.M., Kulikov I.M., Snytnikov A., Romanenko A.A., Chernykh I.G., Vshivkov V.A.* Co-design of parallel numerical methods for plasma physics and Astrophysics // Supercomputing Frontiers and Innovations. 2014. **1**, N 3. 88–98.
2. *Sivasubramaniam A., Singla A., Ramachandran U., Venkateswaran H.* A simulation-based scalability study of parallel systems // Journal of Parallel and Distributed Computing. 1994. **22**, N 3. 411–426.
3. *Racherla G., Killian S.E., Fife L.D., Lehmann M., Parekh R.* PARSIT: a parallel algorithm reconfiguration simulation tool // Proc. of IEEE International Conference on High Performance Computing. New Delhi, 1995 (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.9802&rep=rep1&type=pdf).
4. *D'Souza R.M., Lysenko M., Marino S., Kirschner D.* Data-parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units // Proc. of Spring Simulation Multiconference (SpringSim'09). San Diego, 2009 (http://dl.acm.org/citation.cfm?id=1639831).
5. *Goodrich M.T.* Simulating parallel algorithms in the MapReduce framework with applications to parallel computational Geometry. arXiv preprint: 1004.4708v1 [cs.DS] (Cornell Univ. Library, Ithaca, 2010), available at http://arxiv.org/abs/1004.4708/.
6. *Kogge P. (Ed.)* ExaScale computing study: technology challenges in achieving exascale systems. DARPA report (http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf).
7. *Иванников В.П., Гайсарян С.С., Аветисян А.И., Падарян В.А.* Оценка динамических характеристик параллельной программы на модели // Программирование. 2006. № 4. 21–37.
8. *Glinsky B., Rodionov A., Marchenko M., Podkorytov D., Weins D.* Scaling the distributed stochastic simulation to exaflop supercomputers // Proc. 2012 IEEE 14th Int. Conf. on High Performance Computing and Communication and 2012 IEEE 9th Int. Conf. on Embedded Software and Systems. Washington: IEEE Press, 2012. 1131–1136.
9. *Глинский Б.М., Родионов А.С., Марченко М.А., Подкорытов Д.И., Винс Д.В.* Агентно-ориентированный под-

ход к имитационному моделированию суперЭВМ экзафлопсной производительности в приложении к распределенному статистическому моделированию // Вестник ЮУрГУ. 2012. № 18. 93–106.
10. *Larsson P.* Energy-efficient software guidelines. White paper for the Intel software solutions group. 2008 (https://software.intel.com/sites/default/files/2d/e2/1332).
11. *Albers S., Fujiwara H.* Energy-efficient algorithms for flow time minimization // ACM Transactions on Algorithms (TALG). 2007. **3**, N 4. doi 10.1145/1290672.1290686.
12. *Bianchini R., Rajamony R.* Power and energy management for server systems // Computer. 2004. **37**, N 11. 68–74.
13. *Weyuker E.J., Vokolos F.I.* Experience with performance testing of software systems: issues, an approach, and case study // IEEE Transactions on Software Engineering. 2000. **26**, N 12. 1147–1156.
14. *Capra E., Francalanci C., Slaughter S.A.* Is software "green"? Application development environments and energy efficiency in open source applications // Information and Software Technology. 2012. **54**, N 1. 60–71.
15. *Стулов В.П.* Лекции по газовой динамике. М.: Физматлит, 2004.
16. *Bondar Ye.A.., Shevyrin A.A., Chen Y.S., Shumakova A.N., Kashkovsky A.V., Ivanov M.S.* Direct Monte Carlo simulation of high-temperature chemical reactions in air // Thermophysics and Aeromechanics. 2013. **20**, N 5. 553–564.
17. *Марченко М.А.* Исследование параллельного алгоритма статистического моделирования для решения нелинейного уравнения коагуляции // Russian Journal of Numerical Analysis and Mathematical Modelling. 2008. **23**, № 6. 597–613.
18. *Ono S., Noguchi Y., Sahara R., Kawazoe Y., Ohno K.* TOMBO: all-electron mixed-basis approach to condensed matter physics // Computer Physics Communications. 2015. **189**. 20–30.
19. *Годунов С.К.* Лекции по современным аспектам линейной алгебры. Новосибирск: Научная книга, 2002.
20. *Trefethen L.N., Embree M.* Spectra and pseudospectra: the behavior of nonnormal matrices and operators. Princeton: Princeton Univ. Press, 2005.
21. *Gonzalez R.C.* Digital image processing. Upper Saddle River: Prentice Hall, 2008.
22. *Parker J.R.* Algorithms for image processing and computer vision. New York: Wiley, 2010.
23. *Goreinov S.A., Oseledets I.V., Savostyanov D.V., Tyrtyshnikov E.E., Zamarashkin N.L.* How to find a good submatrix // Matrix Methods: Theory, Algorithms, Applications. Hackensack: World Scientific, 2010. 247–256.
24. *Ильин В.П.* Параллельные процессы на этапах петафлопного моделирования // Вычислительные методы и программирование. 2011. **12**. 103–109.
25. *Гурьева Я.Л., Ильин В.П.* О некоторых параллельных методах и технологиях декомпозиции областей // Зап. научн. сем. ПОМИ. 2014. **428**. 89–106.
26. *Ильин В.П.* Методы бисопряженных направлений в подпространствах Крылова // Сиб. журн. индустр. матем. 2008. **11**, № 4. 47–60.
27. *Kalinkin A., Laevsky Yu.M., Gololobov S.* 2D fast Poisson solver for high-performance computing // Lecture Notes in Computer Science. Vol. 5698. Heidelberg: Springer, 2009. 112–120.
28. *Terekhov A.V.* Parallel dichotomy algorithm for solving tridiagonal system of linear equations with multiple right-hand sides // Parallel Computing. 2010. **36**, N 8. 423–438.
29. *Guglielmi N., Hairer E.* Implementing Radau IIA methods for stiff delay differential equations // Computing. 2001. **67**, N 1. 1–12.
30. *Годунов С.К., Куликов И.М.* Расчет разрывных решений уравнений гидродинамики с гарантией неубывания энтропии // Ж. вычисл. матем. и матем. физ. 2014. **54**, N 6. 1008–1021.
31. *Roe P.L., Balsara D.S.* Notes on the eigensystem of magnetohydrodynamics // SIAM Journal of Applied Mathematics. 1996. **56**, N 1. 57–67.
32. *Mitchell N.L., Vorobyov E.I., Hensler G.* Collisionless stellar hydrodynamics as an efficient alternative to N-body methods // Monthly Notices of the Royal Astronomical Society. 2013. **428**, N 3. 2674–2687.
33. *Kulikov I.* GPUPEGAS: a new GPU-accelerated hydrodynamic code for numerical simulations of interacting galaxies // The Astrophysical Journal Supplements Series. 2014. **214**, N 1. 1–12.
34. *Ibánez J.M., Cordero-Carrión I., Miralles J.A.* On numerical relativistic hydrodynamics and barotropic equations of state // Classical and Quantum Gravity. 2012. **29**, N 15. doi 10.1088/0264-9381/29/15/157001.
35. *Годунов С.К.* О включении уравнений Максвелла в системы релятивистски инвариантных уравнений // Ж. вычисл. матем. и матем. физ., 2013. **53**, № 8. 1356–1359.
36. *Годунов С.К.* Термодинамическая формализация уравнений гидродинамики заряженного диэлектрика в электромагнитном поле // Ж. вычисл. матем. и матем. физ. 2012. **52**, № 5. 916–929.
37. *Годунов С.К., Роменский Е.И.* Элементы механики сплошных сред и законы сохранения. Новосибирск: Научная книга, 1998.
38. *Романьков А.С., Роменский Е.И.* Метод Рунге–Кутты/WENO для расчета уравнений волн малой амплитуды в насыщенной упругой пористой среде // Сиб. журн. вычисл. матем. 2014. **17**, № 3. 259–271.
39. *Springel V.* E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh // Monthly Notices of the Royal Astronomical Society. 2010. **401**, N 2. 791–851.
40. *Murphy J.W., Burrows A.* BETHE-Hydro: an arbitrary Lagrangian–Eulerian multidimensional hydrodynamics code for astrophysical simulations // The Astrophysical Journal Supplement Series. 2008. **179**. 209–241.

41. *Чуев Н.П.* Построение трехмерной эволюционной дифференциальной модели динамики политропного самогравитирующего газа // Вестник Уральского гос. ун-та путей сообщения. 2011. **9**, № 1. 14–21.
42. *Kraeva M.A., Malyshkin V.E.* Assembly technology for parallel realization of numerical models on MIMD-multicomputers // Future Generation of Computer Systems. 2001. **17**, N 6. 755–765.
43. *Gingold R.A., Monaghan J.J.* Smoothed particle hydrodynamics: theory and application to non-spherical stars // Monthly Notices of the Royal Astronomical Society. 1977. **181**, N 3. 375–389.
44. *Li X.J., Mo F., Wang X.H., Wang B., Liu K.* Numerical study on mechanism of explosive welding // Science and Technology of Welding and Joining. 2012. **17**, N 1. 36–41.
45. *Podkorytov D., Rodionov A., Sokolova O., Yurgenson A.* Using agent-oriented simulation system AGNES for evaluation of sensor networks // Lecture Notes in Computer Science. Vol. 6235. 2010. Heidelberg: Springer, 247–250.

Поступила в редакцию
14.09.2015

A Multilevel Approach to Algorithm and Software Design for Exaflops Supercomputers

B. M. Glinskiy [1], I. M. Kulikov [2], A. V. Snytnikov [3], I. G. Chernykh [4], and D. V. Weins [5]

[1] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Dr. Sci., Professor, Head of Laboratory, e-mail: gbm@sscc.ru*

[2] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Ph.D., Junior Scientist, e-mail: kulikov@ssd.sscc.ru*

[3] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Ph.D., Scientist, e-mail: snytav@gmail.com*

[4] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Ph.D., Senior Scientist, e-mail: chernykh@parbz.sscc.ru*

[5] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences; prospekt Lavrentyeva 6, Novosibirsk, 630090, Russia; Junior Scientist, e-mail: wns.dmitry@gmail.com*

Received September 14, 2015

Abstract: A strategy is proposed for the development of algorithms and software for exaflops supercomputers. This strategy consists of three stages. The first stage is the co-design understood as considering the architecture of the supercomputer at all steps of the development of the code. The second stage is the forward-looking development of algorithms and software for the most promising exaflops supercomputers. The forward-looking development is based on the simulation of the algorithm behavior within a given supercomputer architecture. The third stage is the estimation of energy efficiency of the algorithm with various implementations for a particular architecture or for different supercomputer architectures. The proposed approach is illustrated by the examples of solving two problems from astrophysics and plasma physics.

Keywords: exascale computing, co-design, energy efficiency, agent simulation.

References

1. B. M. Glinskiy, I. M. Kulikov, A. Snytnikov, et al., "Co-Design of Parallel Numerical Methods for Plasma Physics and Astrophysics," Supercomput. Front. Innov. **1** (3), 88–98 (2014).

2. A. Sivasubramaniam, A. Singla, U. Ramachandran, and H. Venkateswaran, "A Simulation-Based Scalability Study of Parallel Systems," J. Parallel Distrib. Comput. **22** (3), 411–426 (1994).

3. G. Racherla, S. E. Killian, L. D. Fife, et al., "PARSIT: A Parallel Algorithm Reconfiguration Simulation Tool," in *Proc. IEEE Int. Conf. on High Performance Computing, New Delhi, India, December 27–30, 1995*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.9802&rep=rep1&type=pdf.
Cited October 25, 2015.

4. R. M. D'Souza, M. Lysenko, S. Marino, and D. Kirschner, "Data-Parallel Algorithms for Agent-Based Model Simulation of Tuberculosis on Graphics Processing Units," in *Proc. Spring Simulation Multiconference (SpringSim'09), San Diego, USA, March 22–27, 2009*, http://dl.acm.org/citation.cfm?id=1639831. Cited October 25, 2015.

5. M. T. Goodrich, *Simulating Parallel Algorithms in the MapReduce Framework with Applications to Parallel Computational Geometry*, arXiv preprint: 1004.4708v1 [cs.DS] (Cornell Univ. Library, Ithaca, 2010), available at http://arxiv.org/abs/1004.4708/.

6. P. Kogge (Ed.), *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*, DARPA Report. http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf. Cited October 25, 2015.

7. V. P. Ivannikov, S. S. Gaisaryan, A. I. Avetisyan, and V. A. Padaryan, "Estimation of Dynamical Characteristics of a Parallel Program on a Model," Programmirovanie, No. 4, 21–37 (2006) [Program. Comput. Software **32** (4), 203–214 (2006)].

8. B. Glinsky, A. Rodionov, M. Marchenko, et al., "Scaling the Distributed Stochastic Simulation to Exaflop Supercomputers," in *Proc. 2012 IEEE 14th Int. Conf. on High Performance Computing and Communication and 2012 IEEE 9th Int. Conf. on Embedded Software and Systems* (IEEE Press, Washington, 2012), pp. 1131–1136.

9. B. M. Glinskii, A. S. Rodionov, M. A. Marchenko, et al., "Agent-Oriented Approach to Simulate Exaflop Supercomputer with Application to Distributed Stochastic Simulation," Vestn. South Ural State Univ. Ser. Mat. Model. Programm., No 18, 93–106 (2012).

10. P. Larsson, *Energy-Efficient Software Guidelines*, White Paper for the Intel Software Solutions Group (2008). https://software.intel.com/sites/default/files/2d/e2/1332. Cited October 25, 2015.

11. S. Albers and H. Fujiwara, "Energy-Efficient Algorithms for Flow Time Minimization," ACM Trans. Algorithms **3** (4) (2007). doi 10.1145/1290672.1290686

12. R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," Computer **37** (11), 68–74 (2004).

13. E. J. Weyuker and F. I. Vokolos, "Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study" IEEE Trans. Softw. Eng. 2000. **26** (12), 1147–1156 (2000).

14. E. Capra, C. Francalanci, and S. A. Slaughter, "Is Software "Green"? Application Development Environments and Energy Efficiency in Open Source Applications," Inform. Software Tech. **54** (1), 60–71 (2012).

15. V. P. Stulov, *Lectures on Gas Dynamics* (Fizmatlit, Moscow, 2004) [in Russian].

16. Ye. A. Bondar, A. A. Shevyrin, Y. S. Chen, et al., "Direct Monte Carlo Simulation of High-Temperature Chemical Reactions in Air," Thermophys. Aeromech. 2013. **20** (5), 553–564.

17. M. A. Marchenko, "A Study of a Parallel Statistical Modelling Algorithm for Solution of the Nonlinear Coagulation Equation," Russ. J. Numer. Anal. Math. Modelling **23** (6), 597–613.

18. S. Ono, Y. Noguchi, R. Sahara, et al., "TOMBO: All-Electron Mixed-Basis Approach to Condensed Matter Physics," Comput. Phys. Commun. **189**, 20–30 (2015).

19. S. K. Godunov, *Lectures on Modern Aspects of Linear Algebra* (Nauchnaya Kniga, Novosibirsk, 2002) [in Russian].

20. L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators* (Princeton Univ. Press, Princeton, 2005).

21. R. C. Gonzalez, *Digital Image Processing* (Prentice Hall, Upper Saddle River, 2008).

22. J. R. Parker, *Algorithms for Image Processing and Computer Vision* (Wiley, New York, 2010).

23. S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, et al., "How to Find a Good Submatrix," in *Matrix Methods: Theory, Algorithms, Applications* (World Scientific, Hackensack, 2010), pp. 247–256.

24. V. P. Il'in, "Parallel Processes at the Stages of Petaflops Modeling," Vychisl. Metody Programm. **12**, 103–109 (2011).

25. Ya. L. Gurieva and V. P. Il'in, "Parallel Approaches and Technologies of Domain Decomposition Methods," Zap. Nauchn. Sem. POMI **428**, 89–106 (2014) [J. Math. Sci. **207** (5), 724–735 (2015)].

26. V. P. Il'in, "Biconjugate Direction Methods in Krylov Subspaces," Sib. J. Ind. Mat. **XI** (4), 47–60 (2008) [J. Appl. Ind. Math. **4** (1), 65–78 (2010)].

27. A. Kalinkin, Yu. M. Laevsky, and S. Gololobov, "2D Fast Poisson Solver for High-Performance Computing," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2009), Vol. 5698, pp. 112–120.

28. A. V. Terekhov, "Parallel Dichotomy Algorithm for Solving Tridiagonal System of Linear Equations with Multiple Right-Hand Sides," Parallel Comput. **36** (8), 423–438 (2010).

29. N. Guglielmi and E. Hairer, "Implementing Radau IIA Methods for Stiff Delay Differential Equations," Computing **67** (1), 1–12.

30. S. K. Godunov and I. M. Kulikov, "Computation of Discontinuous Solutions of Fluid Dynamics Equations

with Entropy Nondecrease Guarantee," Zh. Vychisl. Mat. Mat. Fiz. **54** (6), 1008–1021 (2014) [Comput. Math. Math. Phys. **54** (6), 1012–1024 (2014)].

31. P. L. Roe and D. S. Balsara, "Notes on the Eigensystem of Magnetohydrodynamics," SIAM J. Appl. Math. **56** (1), 57–67 (1996).

32. N. L. Mitchell, E. I. Vorobyov, and G. Hensler, "Collisionless Stellar Hydrodynamics as an Efficient Alternative to N-body Methods," Mon. Not. R. Astron. Soc. **428** (3), 2674–2687 (2013).

33. I. Kulikov, "GPUPEGAS: A New GPU-Accelerated Hydrodynamic Code for Numerical Simulations of Interacting Galaxies," Astrophys. J. Suppl. Ser. **214** (1), 1–12 (2014).

34. J. M. Ibánez, I. Cordero-Carrión, and J. A. Miralles, "On Numerical Relativistic Hydrodynamics and Barotropic Equations of State," Class. Quantum Grav. **29** (2012). doi 10.1088/0264-9381/29/15/157001

35. S. K. Godunov, "About Inclusion of Maxwell's Equations in Systems Relativistic of the Invariant Equations," Zh. Vychisl. Mat. Mat. Fiz. **53** (8), 1356–1359 (2013) [Comput. Math. Math. Phys. **53** (8), 1179–1182 (2013)].

36. S. K. Godunov, "Thermodynamic Formalization of the Fluid Dynamics Equations for a Charged Dielectric in an Electromagnetic Field," Zh. Vychisl. Mat. Mat. Fiz. **52** (5), 916–929 (2012) [Comput. Math. Math. Phys. **52** (5), 787–799 (2012)].

37. S. K. Godunov and E. I. Romenskii, *Elements of Continuum Mechanics and Conservation Laws* (Nauchnaya Kniga, Novosibirsk, 1998) [in Russian].

38. A. S. Roman'kov and E. I. Romenskii, "The Runge–Kutta/WENO Method for Solving Equations for Small-Amplitude Wave Propagation in a Saturated Porous Medium," Sib. Zh. Vychisl. Mat. **17** (3), 259–271 (2014) [Numer. Anal. Appl. **7** (3), 215–226 (2014)].

39. V. Springel, "E pur si muove: Galilean-Invariant Cosmological Hydrodynamical Simulations on a Moving Mesh," Mon. Not. R. Astron. Soc. **401** (2), 791–851 (2010).

40. J. W. Murphy and A. Burrows, "BETHE-Hydro: An Arbitrary Lagrangian–Eulerian Multi-Dimensional Hydrodynamics Code for Astrophysical Simulations," Astrophys. J. Suppl. Ser. **179**, 209–241 (2008).

41. N. P. Chuev, "Construction of a Three-Dimensional Evolutionary Differential Model of Polytropic Self-Gravitating Gas Dynamics," Vestn. Ural'sk. Univ. Putei Soobshcheniya **9** (1), 14–21 (2011).

42. M. A. Kraeva and V. E. Malyshkin, "Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers," Future Gener. Comput. Syst. **17** (6), 755–765 (2001).

43. R. A. Gingold and J. J. Monaghan, "Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars," Mon. Not. R. Astron. Soc. **181** (3), 375–389 (1977).

44. X. J. Li, F. Mo, X. H. Wang, et al., "Numerical Study on Mechanism of Explosive Welding," Sci. Technol. Weld. Join. **17** (1), 36–41 (2012).

45. D. Podkorytov, A. Rodionov, O. Sokolova, and A. Yurgenson, "Using Agent-Oriented Simulation System AGNES for Evaluation of Sensor Networks," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2010), Vol. 6235, pp. 247–250.