На правах рукописи

ЕЖОВА Надежда Александровна

МОДЕЛЬ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ОЦЕНКИ МАСШТАБИРУЕМОСТИ ИТЕРАЦИОННЫХ АЛГОРИТМОВ НА КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Челябинск — 2019

Работа выполнена на кафедре системного программирования
ФГАОУ ВО «Южно-Уральский государственный университет
(национальный исследовательский университет)»

Научный руководитель: СОКОЛИНСКИЙ Леонид Борисович,
доктор физ.-мат. наук, профессор,
проректор по информатизации ФГАОУ ВО
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
(г. Челябинск)

Официальные оппоненты: КУЛИКОВ Игорь Михайлович,
доктор физ.-мат. наук, старший научный сотрудник
Лаборатории суперкомпьютерного моделирования
ФГБУН «Институт вычислительной математики
и математической геофизики Сибирского отделения
Российской академии наук» (г. Новосибирск)

БАХТИН Владимир Александрович,
кандидат физ.-мат. наук,
ведущий научный сотрудник отдела № 17 «Отдел
программного обеспечения высокопроизводительных
вычислительных систем и сетей», сектор № 2
«Автоматизация параллельного программирования»
ФГУ «Федеральный исследовательский центр
Институт прикладной математики им. М.В. Келдыша
Российской академии наук» (г. Москва)

Ведущая организация: ФГБОУ ВО «Московский государственный
университет имени М.В. Ломоносова» (г. Москва)

Защита состоится 12 февраля 2020 г. в 11:00 часов на заседании диссертационного совета Д 212.298.18 при Южно-Уральском государственном университете по адресу: 454080, г. Челябинск, пр. Ленина, 76, ауд. 1001.

С диссертацией можно ознакомиться в библиотеке Южно-Уральского государственного университета и на сайте:
https://www.susu.ru/ru/dissertation/d-21229818/ezhova-nadezhda-aleksandrovna.

Автореферат разослан «___» декабря 2019 г.

Ученый секретарь
диссертационного совета М.Л. Цымблер

Общая характеристика работы

Актуальность темы основывается на следующих основных факторах:

1) появление в ближайшие 2-3 года суперкомпьютеров с производительностью более одного экзафлопса (10^{18} операций с плавающей точкой в секунду);
2) доминирование кластерной архитектуры в области высокопроизводительных вычислений;
3) необходимость разработки и исследования новых классов сверхмасштабируемых параллельных алгоритмов, ориентированных на вычислительные системы экзафлопсного уровня производительности;
4) потребность в адекватных и простых в использовании моделей параллельных вычислений, способных на ранних этапах проектирования предсказывать границу масштабируемости для определенных классов алгоритмов, ориентированных на высокопроизводительные вычислительные системы с кластерной архитектурой.

Проанализируем эти факторы более подробно.

Проекты построения экзамасштабных вычислительных систем сегодня реализуются в нескольких странах. В Соединенных Штатах в соответствии с проектом ECP (Exascale Computing Project), являющимся частью федеральной программы NSCI (National Strategic Computing Initiative), в лабораториях Министерства энергетики США к 2023 году должно быть установлено не менее трех экзамасштабных суперкомпьютерных систем, которые на тесте HPL (High-Performance Linpack) будут способны показать производительность более одного экзафлопса. В Китае в 2018-2019 годах были созданы по крайней мере три компьютерные системы с различными архитектурами, демонстрирующие производительность в сотни петафлопс. Базируясь на этих результатах, Китай планирует создать не менее двух экзамасштабных суперкомпьютеров с производительностью более одного экзафлопса на тесте HPL. Европейской Комиссией в 2018 году запущен масштабный суперкомпьютерный проект EuroHPC в котором принимают участие 25 европейских стран. В соответствии с этим проектом в Европе к 2023 году должны быть запущены две экзамасштабные вычислительные системы, по крайней мере одна из которых должна базироваться на европейских технологиях. В Японии Министерством образования, культуры, спорта, науки и технологий в 2014 году на базе Центра вычислительных наук института Riken совместно с компанией Fujitsu запущен проект по созданию экзамасштабной вычислительной системы Post-K. В отличие от американского и европейского проекта компьютер Post-K будет включать в себя только 48-ядерные центральные процессоры с векторизацией, разрабатываемые компанией Fujitsu на базе техно-

логии ARM. Предполагается, что в компьютере Post-K будет более 100 000 процессорных узлов с такими процессорами. Запуск системы намечен на середину 2020 года. В России также выполняются проекты по созданию суперкомпьютеров экзафлопсной производительности, среди которых можно отметить проект «Торнадо» Группы компаний РСК и проект «Эльбрус-16СВ», выполняемый совместно Институтом электронных управляющих машин имени И. С. Брука и компанией МЦСТ. В результате выполнения этих проектов к 2023 году в России должны появиться технологии, обеспечивающие возможность создания вычислительных систем экзафлопсного уровня производительности. В соответствии с этим возникает необходимость разработки моделей параллельных вычислений для систем такого масштаба.

Еще одной важной тенденцией в области высокопроизводительной вычислительной техники является возрастающее преобладание кластерной вычислительной архитектуры. В рейтинге TOP500 самых мощных суперкомпьютеров мира (редакция от 18.06.2019)) семь из первых десяти мест, включая первое и второе, занимают кластеры. В целом суперкомпьютеры с кластерной архитектурой занимают более 90% списка TOP500. Это связано с тем, что кластеры показывают наилучшее соотношение производительность/стоимость. Таким образом наиболее важной является разработка моделей параллельных вычислений для кластерной архитектуры.

Многие параллельные численные алгоритмы, используемые до сих пор, были разработаны для многопроцессорных систем с общей памятью. Дизайн таких алгоритмов ориентирован на рост производительности за счет увеличения тактовой частоты процессоров. Подобные алгоритмы, как правило, невозможно адаптировать для эффективного использования экзамасштабных систем с кластерной архитектурой. В соответствии с этим необходимо разрабатывать принципиально новые параллельные алгоритмы для достижения экзамасштабного уровня параллелизма на системах с миллионами процессорных ядер. Одним из ключевых требований при проектировании сверхмасштабируемых вычислительных алгоритмов для экзамасштабных систем с распределенной памятью является минимизация межпроцессорных коммуникаций. Граница масштабируемости параллельного алгоритма при этом становится ключевой характеристикой. Поэтому актуальной является задача разработки простых и адекватных моделей параллельных вычислений, позволяющих уже на ранних этапах проектирования алгоритма предсказать верхнюю границу его масштабируемости.

К настоящему моменту разработаны многие десятки моделей параллельных вычислений и продолжают создаваться новые модели. Обзор различных моделей параллельных вычислений можно найти в работах. По мере развития многопроцессорных систем происходило совершенствование существующих и создание новых моделей параллельных вычислений с целью учесть особенности современных параллельных архитектур. Адекватность моделей достигалась ценой повышения сложности их использования, что препятствовало широкому

применению моделей параллельных вычислений в практике параллельного программирования. Для уменьшения сложности модели разработчики стали ограничивать класс параллельных архитектур, для которого эта модель применима. Так появились модели параллельных вычислений для общей, памяти, для распределенной памяти, для иерархической памяти, для графических ускорителей и др. Такой подход в некоторой мере позволил уменьшить сложность моделей, но достичь приемлемого компромисса между простотой, универсальностью и адекватностью не удалось. Дальнейшее упрощение моделей параллельных вычислений без ущерба для универсальности и адекватности возможно путем ограничения класса алгоритмов, к которому применима модель. Таким образом актуальной является задача разработки моделей параллельных вычислений для отдельных классов алгоритмов.

Одним из важных классов алгоритмов являются итерационные алгоритмы с высокой вычислительной сложностью. Такие алгоритмы эффективно применяются для решения систем линейных и нелинейных уравнений, линейных и нелинейных неравенств, декомпозиции многомерных матриц, эллиптических дифференциальных уравнений в частных производных, для решения задач математического программирования и оптимизации, выпуклой разрешимости и многих других. В силу этого разработка модели параллельных вычислений, ориентированной на итерационные алгоритмы с высокой вычислительной сложностью, является значимой.

Цель и задачи исследования. *Цель* диссертационной работы состояла в разработке и исследовании новой модели параллельных вычислений для итерационных алгоритмов применительно к многопроцессорным системам с распределенной памятью, позволяющей предсказывать границу масштабируемости алгоритма на ранних стадиях его проектирования, и построении на ее основе параллельного каркаса для кластерных вычислительных систем экзафлопсного уровня производительности. Для достижения этой цели необходимо было решить следующие *задачи*.

1. Разработать модель параллельных вычислений.
2. Создать на языке C++ параллельный каркас, основанный на разработанной модели.
3. Выполнить проектирование и реализацию визуального конструктора программ на языке C++ в соответствии с разработанными моделью и параллельным каркасом.
4. Провести вычислительные эксперименты для верификации предложенной модели и разработанного программного обеспечения.

Методология и методы исследования. Методологической основой диссертационного исследования являются теория множеств, формализм Бёрда-Мир-

тенса, математический анализ. Для описания алгоритмов использовались функции высшего порядка и операции над списками. При разработке параллельного каркаса и конструктора приложений применялись методы объектно-ориентированного проектирования и язык UML. Для организации параллельных вычислений использовались схема программирования SPMD, фреймворк «мастер-рабочие», библиотеки MPI и OpenMP.

Научная новизна работы заключается в том, что впервые разработана адекватная и простая в использовании модель параллельных вычислений для многопроцессорных систем с распределенной памятью, позволяющая предсказать границу масштабируемости для итерационных вычислительно сложных алгоритмов на ранней стадии их проектирования.

Теоретическая ценность работы состоит в том, что разработанная модель параллельных вычислений BSF включает в себя стоимостную метрику, позволяющую аналитически с достаточной точностью оценить границу масштабируемости вычислительно сложного итерационного алгоритма, ориентированного на экзамасштабные вычислительные системы.

Практическая ценность работы состоит в разработке компилируемого программного шаблона и визуального конструктора программ, позволяющих для BSF-алгоритма быстро создать правильно работающую параллельную реализацию на языке C++ с использованием библиотек MPI и OpenMP.

Степень достоверности результатов. Утверждение, связанное с оценкой границы масштабируемости алгоритма, сформулировано в виде теоремы, снабженной строгим доказательством. Теоретические построения подтверждены имитационным моделированием на кластерной вычислительной системе. Верификация модели BSF и соответствующего параллельного каркаса были выполнены на известных итерационных численных алгоритмах из различных областей вычислительной математики и физики.

Апробация работы. Основные положения диссертационной работы, разработанные модели, методы, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных научных конференциях и семинарах:

- ПаВТ'2019: XIII международная конференция «Параллельные вычислительные технологии» (2-4 апреля 2019 г., Калининград);
- GloSIC'2018: Global Smart Industry Conference (13-15 ноября 2018 г., Челябинск);
- ПаВТ'2018: XII международная конференция «Параллельные вычислительные технологии» (1-5 апреля 2018 г., Ростов-на-Дону);
- Ural-PDC'2017: 3rd Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists (19 октября 2017 г., Екатеринбург).

Публикации. По теме диссертации опубликована 21 печатная работа. Работы [1–5] опубликованы в журналах, включенных ВАК в перечень изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора и кандидата наук. Работы [6,7] опубликованы в изданиях, индексируемых в Scopus и Web of Science.

Личный вклад. В работах [1–6] научному руководителю Л.Б. Соколинскому принадлежит постановка задачи, Н.А. Ежовой – все полученные результаты.

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения и библиографии. Объем диссертации составляет 137 страниц, объем библиографии – 195 наименований.

Содержание работы

Во введении приводится обоснование актуальности темы и степень ее разработанности; формулируются цели и задачи исследования; раскрываются новизна, теоретическая и практическая значимость полученных результатов; формулируется методологическая основа диссертационного исследования; дается обзор содержания диссертации.

В первой главе, «Модели параллельных вычислений», рассматривается понятие модели параллельных вычислений, дается классификация различных моделей и формулируются требования к модели параллельных вычислений. Особое внимание уделяется формализму Бёрда-Миртенса, являющемуся теоретическим базисом для построения параллельных каркасов. Дается обзор известных моделей параллельных вычислений и параллельных каркасов.

Во второй главе, «Модель параллельных вычислений BSF», описывается новая модель параллельных вычислений *BSF* (*Bulk Synchronous Farm*) – *блочно-синхронная ферма*, ориентированная на многопроцессорные системы с кластерной архитектурой и предназначенная для разработки итерационных численных алгоритмов с высокой вычислительной сложностью. Приводятся базисные концепции, лежащие в основе модели BSF. Описываются архитектура BSF-компьютера и структура BSF-алгоритма. Вводится стоимостная метрика, формулируется и доказывается теорема о границе масштабируемости BSF-алгоритма.

BSF-компьютер представляет собой множество однородных процессорных узлов с приватной памятью, соединенных сетью, позволяющей передавать данные от одного процессорного узла другому. Среди процессорных узлов выделяется один, называемый *узлом-мастером* (или кратко *мастером*). Остальные K узлов называются *узлами-рабочими* (или просто *рабочими*). В *BSF*-компьютере должен быть по крайней мере один узел мастер и один рабочий ($K \geq 1$). Схематично архитектура *BSF*-компьютера изображена на рис. 1.

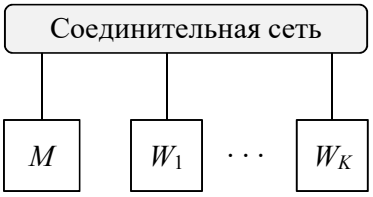

Рис. 1. *BSF*-компьютер. М –мастер; W_1, …, W_K – рабочие

BSF-алгоритм должен быть представлен в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определяемым в формализме Бёрда-Миртенса. *Общий шаблон итерационного алгоритма* в модели BSF представлен на рис. 2. Переменная i обозначает номер итерации; $x^{(0)}$ – начальное приближение, $x^{(i)}$ – i-тое приближение (в качестве приближения может фигурировать число, вектор или другая произвольная структура данных); A – список элементов некоторого множества $\mathbb{A}$, представляющий собой исходные данные задачи; $F_x : \mathbb{A} \to \mathbb{B}$ – параметризованная функция (в качестве параметра фигурирует приближение x), отображающая множество $\mathbb{A}$ в некоторое множество $\mathbb{B}$; B – список элементов множества $\mathbb{B}$, получающийся путем применения функции F_x к каждому элементу списка A; $\oplus$ – некоторая бинарная ассоциативная операция над множеством $\mathbb{B}$. На шаге 1 вводятся исходные данные задачи (список A) и начальное приближение $x^{(0)}$. На шаге 2 номер итерации устанавливается в значение 0. На шаге 3 вычисляется список B как результат выполнения функции высшего порядка $Map(F_{x^{(i)}}, A)$. На шаге 4 вычисляется промежуточная переменная $s \in \mathbb{B}$, которая получается в результате выполнения функции высшего порядка $Reduce(\oplus, B)$. На шаге 5 выполняется пользовательская функция *Compute*, которая на основе $x^{(i)}$ и s вычисляет следующее приближение $x^{(i+1)}$. На шаге 6 номер итерации i увеличивается на единицу. На шаге 7 выполняется пользовательская булева функция *StopCond* с аргументами $x^{(i)}$ и $x^{(i-1)}$, проверяющая условие завершения работы итерационного алгоритма. Если эта функция возвращает значение «истина», то происходит переход на шаг 9. На шаге 8 осуществляется безусловный переход на шаг 3 для выполнения очередной итерации. На шаге 9 в качестве результата выводится последнее полученное приближение. Шаг 10 останавливает работу алгоритма.

Общий шаблон параллельной реализации итерационного алгоритма модели BSF представлен на рис. 3. Параллельная реализация включает в себя $K+1$ потоков управления. Поток управления с номером 0 выполняется на узле-мастере. Потоки $1,\ldots,K$ выполняются на узлах-рабочих, имеют идентичный код, однако обрабатывают различные части $A^{[1]},\ldots,A^{[K]}$ списка A, формируя различные части $B^{[1]},\ldots,B^{[K]}$ списка B. Деление списка A на подсписки $A^{[1]},\ldots,A^{[K]}$ осуществляется таким образом, чтобы все они, возможно кроме последнего, имели равную длину. На шаге 3 мастер посылает, а рабочие получают текущее приближение $x^{(i)}$. После этого j-тый рабочий выполняет следующие действия: на шаге 4

Алгоритм 1. Шаблон итерационного алгоритма в модели BSF

1: **input** $A, x^{(0)}$
2: $i := 0$
3: $B := Map\left(F_{x^{(i)}}, A\right)$
4: $s := Reduce(\oplus, B)$
5: $x^{(i+1)} := Compute\left(x^{(i)}, s\right)$
6: $i := i + 1$
7: **if** $StopCond(x^{(i)}, x^{(i-1)})$ **goto** 9
8: **goto** 2
9: **output** $x^{(i)}$
10: **stop**

Рис. 2. Шаблон итерационного алгоритма в модели BSF

вычисляет подсписок $B^{[j]}$, применяя функцию $F_{x^{(i)}}$ к каждому элементу подсписка $A^{[j]}$; на шаге 5 считает частичную «сумму» $s^{(j)}$, «складывая» элементы подсписка $B^{[j]}$ с помощью операции $\oplus$; на шаге 6 посылает вычисленное значение $s^{(j)}$ мастеру. Отметим, что указанные шаги зависят только от локальных данных и выполняются всеми рабочими параллельно. При этом время, затрачиваемое рабочими с номерами $1,\ldots,K-1$ на выполнение шагов 4 и 5, будет одинаково, так как они обрабатывают списки равной длины. K -тый рабочий будет работать меньшее время, если длина исходного списка не кратна K. Мастер на шаге 6 получает значения $s^{(j)}$ от всех рабочих и выполняет оставшиеся шаги итерационного алгоритма. Отметим, что для корректной работы параллельного алгоритма 2 необходимо и достаточно, чтобы операция $\oplus$ была ассоциативной. В дальнейшем список A, являющийся аргументом функции *Map*, будем называть *списком* «*Map*», а список B, являющийся аргументом функции *Reduce*, будем называть *списком* «*Reduce*».

Стоимостная метрика модели BSF исходит из предположения, что временные затраты на инициализацию и завершение *BSF*-программы пренебрежимо малы по сравнению с затратами на выполнение итерационного процесса, и строится следующим образом. Стоимость итерационного процесса получается как сумма стоимостей отдельных итераций. Поэтому для оценки времени выполнения *BSF*-программы нам достаточно получить оценку временной стоимости одной итерации. Модель BSF включает в себя следующие основные стоимостные параметры в рамках одной итерации:

K – количество узлов-рабочих;
t_s – время, затрачиваемое мастером на передачу задания одному рабочему (без учета латентности);
t_{Map} – время выполнения одним рабочим функции *Map* для *всего* списка исходных данных A;

Алгоритм 2. Шаблон параллельной реализации итерационного алгоритма в модели BSF

Мастер	j-ый рабочий (j=1,…,K)
1: **input** $x^{(0)}$	1: **input** $A^{[j]}$
2: $i := 0$	2:
3: $SendToAllWorkers(x^{(i)})$	3: $RecvFromMaster(x^{(i)})$
4:	4: $B^{[j]} := Map(F_{x^{(i)}}, A^{[j]})$
5:	5: $s^{(j)} := Reduce(\oplus, B^{[j]})$
6: $RecvFromWorkers([s^{(1)}, \dots, s^{(K)}])$	6: $SendToMaster(s^{(j)})$
7: $s := Reduce(\oplus, [s^{(1)}, \dots, s^{(K)}])$	7:
8: $x^{(i+1)} := Compute(x^{(i)}, s)$	8:
9: $i := i + 1$	9:
10: **if** $StopCond(x^{(i)}, x^{(i-1)})$ **goto** 12	10:
11: **goto** 3	11: **goto** 3
12: **output** x_i	12:
13: **stop**	13:

Рис. 3. Шаблон параллельной реализации итерационного алгоритма в модели BSF

t_p – время, затрачиваемое мастером на обработку полученных результатов и проверку условия завершения;

t_r – время, необходимое для передачи мастеру результата, полученного одним рабочим (без учета латентности);

t_a – время, необходимое для выполнения одной операции $\oplus$, являющейся параметром функции *Reduce*;

l – длина списка исходных данных A (совпадает с длиной списка B);

L – латентность (время пересылки сообщения длиной в 1 байт).

Обозначим через T_1 время выполнения одной итерации алгоритма 2 (рис. 3) системой из одного мастера и одного рабочего. Используя введенные стоимостные параметры, можно получить следующую оценку времени T_1:

$$T_1 = 2L + t_s + t_r + t_p + t_{Map} + lt_a.$$

Обозначим через T_K время выполнения одной итерации алгоритма 2 системой из одного мастера и K рабочих. Для времени T_K можно получить следующую оценку:

$$T_K = K\left(2L + t_s + t_r + t_a\right) + \frac{t_{Map} + lt_a}{K} - t_a + t_p.$$

Ускорение a как функция от K в модели BSF вычисляется по формуле:

$$a(K) = \frac{T_1}{T_K} = \frac{2L + t_s + t_r + t_p + t_{Map} + lt_a}{K\left(2L + t_s + t_r + t_a\right) + \frac{t_{Map} + lt_a}{K} - t_a + t_p}. \quad (1)$$

При положительных значениях всех параметров и при $K \geq 1$ функция, определяемая данной формулой, обладает следующими свойствами:

1) $a(1) = 1$;
2) $a(K) > 0$;
3) $\lim a(K) = \frac{1}{K}$.

С содержательной точки зрения свойство 1) соответствует реальности: ускорение на одном процессорном узле должно быть равно 1. Свойство 2) также подтверждает адекватность формулы (1), так как ускорение – всегда величина положительная. Свойство 3) говорит, что при очень малых значениях параметров t_{Map}, t_a и t_p, определяющих суммарный объем вычислений, формула (1) на интервале $[1;+\infty)$ вырождается в монотонно убывающую функцию $a(K) = 1/K$, не имеющую максимума внутри указанного интервала. Это означает, что модель BSF не применима для алгоритмов, в которых время, затрачиваемое на пересылку данных, несоизмеримо больше времени, затрачиваемого на вычисления. Главное свойство формулы (1) сформулируем в виде следующей *теоремы о границе масштабируемости BSF-алгоритма*.

Теорема 2. Пусть все параметры в формуле (1) положительны. Тогда функция $a(K)$, определяемая формулой (1), имеет единственный экстремум на интервале $(1;+\infty)$, являющийся максимумом.

Из теоремы 2 следует, что *граница масштабируемости BSF-алгоритма* может быть оценена по следующей формуле:

$$K_{MAX} = \sqrt{\frac{t_{Map} + lt_a}{2L + t_s + t_r + t_a}}. \qquad (2)$$

В третьей главе, «Программная поддержка модели BSF», описываются структура параллельного BSF-каркаса на языке C++ и визуальный конструктор BSF-Studio, которые можно использовать для быстрого создания BSF-программ.

Параллельный BSF-каркас спроектирован таким образом, что он допускает поэтапное заполнение проблемно-зависимых частей, и при этом компилируется на всех этапах разработки. Исходные тексты BSF-каркаса свободно доступны в сети Интернет по адресу https://github.com/nadezhda-ezhova/BSF-MR. Заполненный BSF-каркас компилируется и компонуется вместе с библиотеками MPI и OpenMP в один исполняемый файл, который запускается как MPI-процесс на указанном количестве узлов в указанном количестве экземпляров.

BSF-каркас состоит из двух групп файлов:

1) файлы с префиксом «BSF» содержат проблемно-независимый код и не подлежат изменениям со стороны пользователя;
2) файлы с префиксом «Problem» предназначены для заполнения пользователем проблемно-зависимых частей программы.

Табл. 1. Файлы исходного кода BSF-каркаса

Файл	Описание
Проблемно-независимый код	
BSF-Code.cpp	Реализации головной функции main и всех проблемно-независимых функций
BSF-Data.h	Проблемно-независимые переменные и структуры данных
BSF-Forwards.h	Предописания проблемно-независимых функций
BSF-Include.h	Включение проблемно-независимых библиотек (iostream, mpi.h, omp.h)
BSF-ProblemFunctions.h	Предописания предопределенных функций с проблемно-зависимой реализацией
BSF-Types.h	Определения проблемно-независимых типов
Проблемно-зависимый код	
Problem-bsfCode.cpp	Реализация предопределенных проблемно-зависимых функций
Problem-bsfParameters.h	Предопределенные проблемно-зависимые параметры
Problem-bsfTypes.h	Предопределенные проблемно-зависимые типы
Problem-Data.h	Проблемно-зависимые переменные и структуры данных
Problem-Forwards.h	Предописания проблемно-зависимых функций
Problem-Include.h	Включение проблемно-зависимых библиотек
Problem-Parameters.h	Проблемно-зависимые параметры
Problem-Types.h	Проблемно-зависимые типы

Описания всех файлов исходного кода приведены в табл. 1. Граф зависимостей файлов BSF-каркаса по включению с помощью директивы #include приведен на рис. 4. Серым закрашены файлы, в которых пользовательские изменения не допускаются; узорной штриховкой обозначены файлы с предопределенными заголовками определений, тело которых должен заполнить пользователь; белый фон соответствует файлам, полностью заполняемым пользователем. Порядок заполнения BSF-каркаса приведен в файле ReadMe.pdf, доступном по адресу https://github.com/nadezhda-ezhova/BSF-MR.

Программная система BSF-Studio предназначена для быстрого создания корректных BSF-программ на языке C++ с использованием BSF-каркаса и представляет собой веб-приложение с клиент-серверной архитектурой. Клиентская часть представляет собой одностраничное приложение (Single Page Application), серверная часть – Docker-контейнер, в котором запускается приложение, выполняющее по запросу компиляцию и запуск программы, и возвращающее результат. Связь между клиентом и севером осуществляется посредством REST API.

Интерфейс клиента BSF-Studio представляет собой мастер (wizard) для заполнения BSF-каркаса. Процесс работы мастера разбивается на следующие *шаги*:

1) определение константных параметров модели BSF и параметров задачи (размерность, число уравнений и др.);

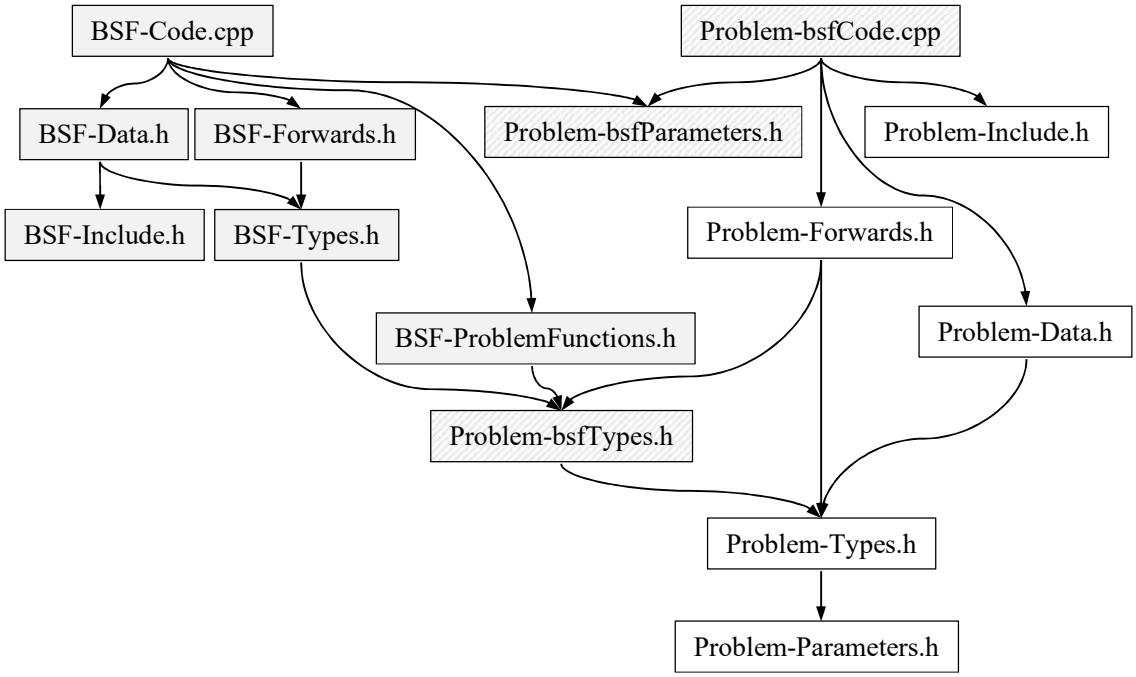

Рис. 4. Граф зависимостей файлов BSF-каркаса по включению #include

2) определение типов данных текущего приближения и элементов списков «Map» и «Reduce»;

3) определение пользовательских переменных и структур данных;

4) реализация пользовательских функций.

Указанные шаги реализуются путем последовательного заполнения HTML-форм, содержащих текстовые поля для заполнения соответствующих фрагментов программного кода. Реализация интерфейса выполнена с помощью JavaScript-библиотеки React с открытым исходным кодом. React позволяет создавать интерактивные пользовательские интерфейсы на основе компонентного подхода: фрагменты веб-приложения (компоненты) инкапсулируются и используются независимо друг от друга. А за счет большого количества библиотек с открытым исходным кодом, предоставляющих готовые React-компоненты, разработка приложений сводится к подбору, подключению и использованию сторонних модулей. Для разработки серверной части приложения BSF-Studio использована технология Docker, которая обеспечивает возможность развертывания приложения на базе контейнерной виртуализированной среды. Это позволило создать виртуальное окружение, максимально приближенное к окружению кластерной вычислительной системы. Данный подход позволяет развернуть серверную часть приложения BSF-Studio в рамках любой операционной системы и аппаратного обеспечения, а также упростить процесс развертывания. Исходные тексты BSF-Studio свободно доступны в сети Интернет:

– серверная часть: https://github.com/ezhova-nadezhda/BSF-Studio-API;
– клиентская часть: https://github.com/ezhova-nadezhda/BSF-Studio.

В четвертой главе, «Верификация модели BSF», была выполнена верификация модели параллельных вычислений BSF тремя различными способами. Первый способ заключался в разработке эмулятора BSF-программ на языке C++ с использованием библиотеки MPI. Исходный код эмулятора свободно доступен на по адресу https://github.com/nadezhda-ezhova/BSF-simulator.

Эмулятор BSF-программ использует схему программирования SPMD (Single Program – Multiple Data) и включает в себя два вида секций: *секции мастера*, и *секции рабочего*. Секции мастера выполняются, если функция *MPI_Comm_rank* возвращает ноль. *Секция рабочего* выполняется, если функция *MPI_Comm_rank* возвращает значение больше нуля. Задание представляет собой символьную строку фиксированной длины, передаваемую от мастера к рабочим командой *MPI_Isend*. Рабочий считывает данные командой *MPI_Irecv*. Обработка задачи моделируется путем вызова функции *usleep*(t_v), которая приводит к приостановке рабочего MPI-процесса до выполнения указанного количества микросекунд в реальном времени. После обработки мастер и все рабочие выполняют глобальную барьерную синхронизацию с помощью команды *MPI_Barrier*. Затем рабочие посылают мастеру символьную строку определенной длины с использованием команды *MPI_Send*. Мастер считывает результаты командой *MPI_Recv*. Синхронизация выполняется командой *MPI_Waitall*. Далее мастер моделирует обработку результатов, вызывая функцию *usleep*(t_p), которая приводит к приостановке мастер-MPI-процесса до выполнения указанного количества микросекунд в реальном времени. Для верификации формулы вычисления ускорения был введен параметр $v = \lg(t_w / t_s)$, связывающий время t_w, необходимое для выполнения задания бригадой из одного рабочего, и время t_s, необходимое для посылки задания рабочему (без учета латентности). Исследовались следующие значения параметра v: 4, 4.5 и 6. Кривые ускорения, полученные аналитически путем использования формулы ускорения модели BSF, сравнивались с кривыми, полученными в результате численных экспериментов, выполненных с помощью эмулятора BSF-программ. Используемые значения параметров стоимостной метрики модели BSF приведены в табл. 2.

Табл. 2. Значения стоимостных параметров модели BSF

Параметр	**Семантика**	**Значение (сек.)**
t_r	Время, необходимое для передачи результата мастеру от рабочего (без учета латентности)	0.01
t_p	Время обработки мастером результатов, полученных от рабочих	4.99
t_w	Время выполнения задания бригадой из одного рабочего	500
L	Затраты на инициализацию операции передачи сообщения (латентность)	$2 \cdot 10^{-5}$

Табл. 3. Параметры экспериментов

v	t_s **(сек.)**	**Объем данных**
4	0.0206978	60 MB
4.5	0.0158160	6 MB
6	0.00048	200 KB

Значение латентности L было получено экспериментально как время передачи сообщения в один байт с помощью функций *MPI_Send* и *MPI_Recv*. Значение t_r соответствует пересылке сообщения длиной 14 Мб. Значения параметра t_s зависят от v и t_w. Эта зависимость определяется формулой $t_s=10^{-v}t_w$. Соответствующие значения параметра t_s были получены варьированием длины задания (см. табл. 3). С помощью эмулятора верифицировалась следующая обобщенная формула ускорения модели BSF:

$$a(K)=\frac{2L+10^{-v}t_w+t_r+t_p+t_w}{K(2L+10^{-v}t_w)+t_r+t_p+t_w/K}.$$

Результаты вычислительных экспериментов, представленные на рис. 5, показывают, что стоимостная метрика модели BSF достаточно хорошо предсказывает экспериментальные результаты, полученные с помощью эмулятора BSF-программ. При этом точность аналитических оценок ускорения возрастает с увеличением значения параметра v.

Второй способ верификации модели BSF заключался в разработке и реализации BSF-алгоритма на основе итерационного метода Якоби для решения СЛАУ. Пусть в евклидовом пространстве $\mathbb{R}^n$ задана совместная система линейных неравенств в матричном виде:

$$Ax=b,$$

где

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}; \quad x=(x_1,\ldots,x_n); \quad b=(b_1,\ldots,b_n).$$

Предполагается, что $a_{ii}\neq 0$, для всех $i=1,\ldots,n$. Определим матрицу

$$C=\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix}$$

следующим образом:

$$c_{ij}=\begin{cases} -\dfrac{a_{ij}}{a_{ii}}, \forall j\neq i; \\ 0, \forall j=i. \end{cases}$$

Определим вектор $d=(d_1,\ldots,d_n)$: $d_i=b_i/a_{ii}$.

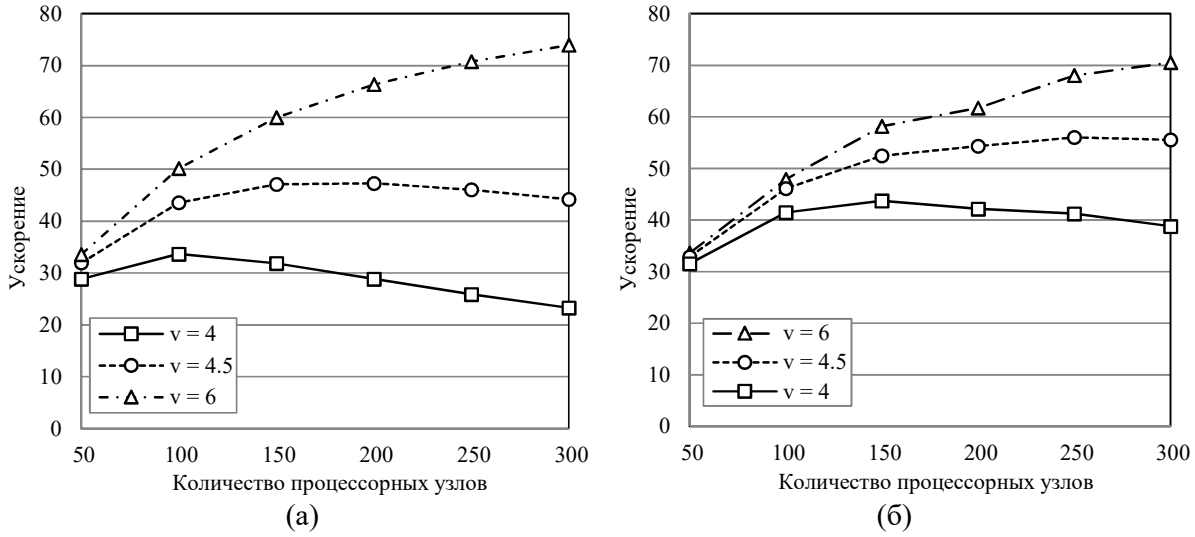

Рис. 5. Верификация формулы для вычисления ускорения:
(а) графики, полученные аналитически; (б) графики, полученные экспериментально

Метод Якоби нахождения приближенного решения системы линейных алгебраических уравнений состоит из следующих шагов:

Шаг 1. $k := 0$; $x^{(0)} := d$.
Шаг 2. $x^{(k+1)} := Cx^{(k)} + d$.
Шаг 3. Если $\left\| x^{(k+1)} - x^{(k)} \right\|^2 < \varepsilon$, перейти на шаг 5.
Шаг 4. $k := k + 1$; перейти на шаг 2.
Шаг 5. Стоп.

В соответствии с требованиями модели BSF представим алгоритм Якоби в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce* и назовем такой алгоритм *Jacobi-BSF*. Определим параметризованную функцию $F_x : \{1,\ldots,n\} \to \mathbb{R}^n$ для оператора Map следующим образом:

$$F_x(j) = (c_{1j}x_j,\ldots,c_{nj}x_j) = x_j \begin{pmatrix} c_{1j} \\ \vdots \\ c_{nj} \end{pmatrix}^T . \qquad (3)$$

С неформальной точки зрения функция $F_x(j)$ умножает j-тый столбец матрицы C на j-тую координату вектора x.

Алгоритм Jacobi-BSF строится на основе шаблона итерационного алгоритма 1 (рис. 2) и приведен на рис. 6. Параллельный алгоритм строится автоматически на основе шаблона параллельной реализации итерационного алгоритма 2. Выполним аналитическое исследование алгоритма Jacobi-BSF с помощью стоимостной метрики модели BSF. В рамках одной итерации введем следующие обозначения для анализа масштабируемости алгоритма Jacobi-BSF:

c_s – количество вещественных чисел, передаваемых от мастера рабочему;

Алгоритм 3. Последовательный Jacobi-BSF

1: **input** C, d; $x^{(0)} := d$
2: $k := 0$
3: $[g^1, \dots, g^n] := Map\left(F_{x^{(k)}}, [1, \dots, n]\right)$
4: $g := Reduce(+, [g^1, \dots, g^n])$
5: $x^{(k+1)} := g + d$
6: $k := k + 1$
7: **if** $\left\| x^{(k)} - x^{(k-1)} \right\|^2 < \varepsilon^2$ **goto** 9
8: **goto** 2
9: **output** $x^{(i)}$
10: **stop**

Рис. 6. Последовательный алгоритм Jacobi-BSF

c_{Map} – количество арифметических операций, выполняемых на шаге Map (шаг 3 алгоритма);

c_a – количество арифметических операций, необходимое для сложения двух векторов;

c_r – количество вещественных чисел, передаваемых от рабочего мастеру;

c_p – количество арифметических операций, выполняемых мастером на шагах 5 и 7 алгоритма.

Обозначим с помощью τ_{op} время, затрачиваемое рабочим на выполнение одной арифметической операции, и τ_{tr} – время, затрачиваемое на пересылку по сети одного вещественного числа без учета латентности. Тогда, используя метрику модели BSF, можно получить следующие значения для стоимостных параметров алгоритма Jacobi-BSF:

$$t_s = c_s \tau_{tr} = n\tau_{tr}; \tag{4}$$

$$t_{Map} = c_{Map} \tau_{op} = n^2 \tau_{op}; \tag{5}$$

$$t_a = c_a \tau_{op} = n\tau_{op}; \tag{6}$$

$$t_r = \tau_{tr} n; \tag{7}$$

$$t_p = c_p \tau_{op} = 4n\tau_{op}. \tag{8}$$

Подставляя в формулу (1) значения правых частей из формул (4) – (8) получаем следующую формулу для оценки ускорения алгоритма Jacobi-BSF:

$$a_{Jacobi-BSF}(K) = \frac{2(L + n\tau_{tr}) + 2n\tau_{op}\left(n + 2\right)}{K\left(2(L + n\tau_{tr}) + n\tau_{op}\right) + n\tau_{op}\left(2n / K + 3\right)}. \tag{9}$$

Граница масштабируемости алгоритма Jacobi-BSF получается путем подстановки значений правых частей из формул (4) – (8) в формулу (2):

$$P_{Jacobi-BSF} = \sqrt{\frac{2n^2 \tau_{op}}{2(L + \tau_{tr} n) + n\tau_{op}}}.$$

При $n \to \infty$ имеем

$$P_{Jacobi-BSF} \approx \sqrt{O(n)}.$$

Таким образом, граница масштабируемости алгоритма Jacobi-BSF растет пропорционально корню квадратному из размерности задачи n.

Для верификации результатов, полученных аналитическим путем, была выполнена параллельная реализация алгоритма Jacobi-BSF на языке C++ с использованием параллельного BSF-каркаса. Данная реализация свободно доступна в сети Интернет по адресу https://github.com/nadezhda-ezhova/Jacobi-BSF. Для проведения экспериментов была использована масштабируемая система линейных уравнений, имеющая следующие матрицу коэффициентов A и столбец свободных членов b:

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & n \end{pmatrix}, \quad b = \begin{pmatrix} n \\ n+1 \\ \vdots \\ 2n-1 \end{pmatrix}.$$

Исследовано ускорение параллельной реализации алгоритма Jacobi-BSF на суперкомпьютере «Торнадо ЮУрГУ». Тестирование проводилось для размерностей 1500, 5000, 10000 и 16000. Одновременно для этих же размерностей построены графики ускорения с использованием формулы (9). Для этого экспериментальным путем были определены величины в секундах: $L = 1.5 \cdot 10^{-5}$, $\tau_{op} = 2.9 \cdot 10^{-8}$ и $\tau_{tr} = 1.9 \cdot 10^{-7}$. Результаты приведены на рис. 7 – рис. 10. Во всех случаях аналитические оценки оказались очень близки к экспериментальным. Кроме того, границы масштабируемости алгоритма Jacobi-BSF, полученные аналитически, оказались очень близки к границам масштабируемости, полученным в результате вычислительных экспериментов. Это подтверждает адекватность модели параллельных вычислений BSF.

Третий способ верификации модели BSF выполнялся с помощью гравитационной задачи, моделирующей движение тела малой массы среди *n* неподвижных тел большой массы. Указанная задача является упрощенным вариантом известной задачи *n* тел. Пусть в $\mathbb{R}^3$ задано конечное множество точек $\mathbb{Y}$, представляющих собой неподвижные тела большой массы. Будем обозначать эти тела следующим образом: $\mathbb{Y} = \{Y_1, \ldots, Y_n\} \subset \mathbb{R}^3$, а их массы соответственно – $\{m_1, \ldots, m_n\}$, где $n = |\mathbb{Y}|$. Пусть также задано некоторое движущееся тело x малой массы m_x. Предполагается, что на x не действуют никакие силы, кроме гравитационных. Для тела x задано его начальное положение $X^{(t_0)} \in \mathbb{R}^3$ и вектор скорости $V^{(t_0)} \in \mathbb{R}^3$ в начальный момент времени t_0. Необходимо рассчитать траекторию движения тела x среди тел $\mathbb{Y}$. Для этого будем последовательно считать новые положения тела x через равные промежутки времени Δt:

$$X^{(t_0)}, X^{(t_0+\Delta t)}, X^{(t_0+2\Delta t)}, X^{(t_0+3\Delta t)}, \ldots \qquad (10)$$

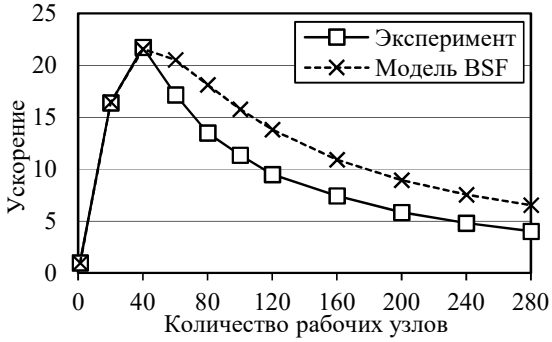

Рис. 7. Jacobi-BSF для n = 1500

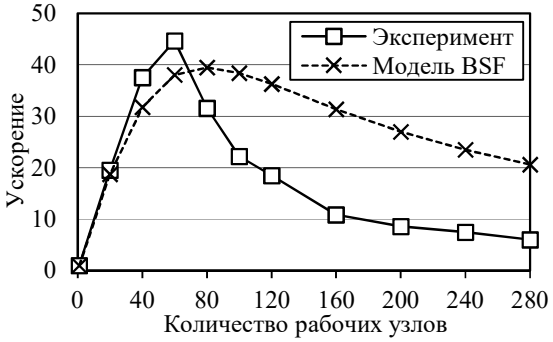

Рис. 8. Jacobi-BSF для n = 5000

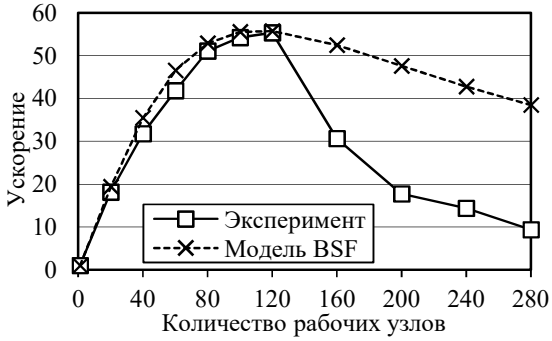

Рис. 9. Jacobi-BSF для n = 10000

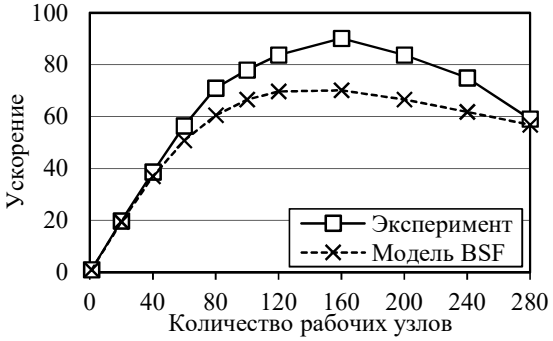

Рис. 10. Jacobi-BSF для n = 16000

По закону всемирного тяготения сила притяжения F_i тела x к телу Y_i $(i=1,\ldots,n)$ вычисляется по формуле

$$F_i = G\frac{m_i m_x}{\|Y_i - X\|^3}(Y_i - X), \tag{11}$$

где $X \in \mathbb{R}^3$ задает текущие координаты точки x. По второму закону Ньютона ускорение α_i тела x под действием силы F_i вычисляется по формуле

$$\alpha_i = \frac{F_i}{m_x}.$$

Ускорение под действием всех сил $F_1,\ldots,F_n$ вычисляется по формуле

$$\alpha = \sum_{i=1}^{n}\alpha_i.$$

В соответствии с этим вектора скорости для последовательности (10) могут быть посчитаны по следующей итерационной формуле

$$V^{(t+\Delta t)} = V^{(t)} + \alpha^{(t+\Delta t)}\Delta t, \tag{12}$$

где

$$\alpha^{(t+\Delta t)} = \sum_{i=1}^{n} G\frac{m_i}{\|Y_i - X^{(t)}\|^3}(Y_i - X^{(t)}).$$

Используя (12) получаем

$$X^{(t+\Delta t)} = X^{(t)} + V^{(t+\Delta t)}\Delta t.$$

В контексте гравитационного алгоритма определим список исходных данных A следующим образом:

$$A = [(Y_1, m_1), \ldots, (Y_n, m_n)],$$

то есть A – список пар вида (Y_i, m_i), задающих координаты и массу i-того неподвижного тела большой массы. Для произвольной точки $X \in \mathbb{R}^3$ определим функцию $f_X : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$:

$$f_X(Y_i, m_i) = G \frac{m_i}{\|Y_i - X\|^3}(Y_i - X) \tag{13}$$

для всех $i \in \{1, \ldots, n\}$. Иначе говоря, функция $f_X(Y_i, m_i)$ с использованием формулы (11) вычисляет значение F_i / m_x, где F_i – гравитационная сила, с которой тело Y_i действует на материальную точку с координатами X и массой m_x. Для произвольного $X \in \mathbb{R}^3$ определим список $B \subset \mathbb{R}^3$ следующим образом:

$$B^{(X)} = [f_X(Y_1, m_1), \ldots, f_X(Y_n, m_n)],$$

то есть список B получается из списка A путем применения к нему функции высшего порядка *Map*, использующей в качестве параметра функцию f_X:

$$B = Map(f_X, A).$$

Тогда суммарное ускорение α, получаемое материальной точкой с массой m_x и координатами X под воздействием тел $\mathbb{Y}$, может быть вычислено путем применения к списку B функции высшего порядка *Reduce*, использующей в качестве параметра операцию сложения векторов +:

$$\alpha = Reduce(+, B).$$

На основе шаблона итерационного алгоритма, представленного на рис. 2, мы можем записать гравитационный алгоритм Gravitation-BSF в виде операций над списками (см. рис. 11). Здесь t_0 обозначает начальный момент времени, T – конечный момент времени, Δt – шаг по времени. Параллельный алгоритм строится автоматически на основе шаблона параллельной реализации итерационного алгоритма 2.

Выполним аналитическое исследование алгоритма Gravitation-BSF. Для простоты мы будем везде далее предполагать, что количество тел большой массы n кратно количеству рабочих K, то есть $n = qK$ при некотором $q \in \mathbb{N}$. В рамках одной итерации ведем следующие обозначения для анализа масштабируемости алгоритма Gravitation-BSF:

- c_s – количество вещественных чисел, передаваемых от мастера рабочему;
- c_{Map} – количество арифметических операций, выполняемых на шаге *Map* при обработке всего списка исходных данных A (шаг 3 алгоритма 3);
- c_a – количество арифметических операций, необходимое для сложения двух векторов в трехмерном пространстве;
- c_r – количество вещественных чисел, передаваемых от рабочего мастеру;
- c_p – количество арифметических операций, выполняемых мастером на шагах 10 – 12 алгоритма 4.

Алгоритм 4. Алгоритм Gravitation-BSF

1: **input** $A, X^{(t_0)}, V^{(t_0)}, \Delta t, t_0, T$
2: $t := t_0$
3: $B := Map\left(f_{X^{(t)}}, A\right)$
4: $\alpha := Reduce(+, B)$
5: $V^{(t+\Delta t)} := V^{(t)} + \alpha \cdot \Delta t;\ \ X^{(t+\Delta t)} := X^{(t)} + V^{(t+\Delta t)}\Delta t$
6: $t := t + \Delta t$
7: **if** $t \geq T$ **goto** 10
8: **goto** 3
9: **output** $X^{(t)}$
10: **stop**

Рис. 11. Алгоритм Gravitation-BSF.

Пусть τ_{op} обозначает время, затрачиваемое рабочим на выполнение одной арифметической операции (или операции сравнения), а τ_{tr} – время, затрачиваемое на пересылку по сети одного вещественного числа без учета латентности. Тогда, используя метрику модели BSF, можно получить следующие значения для стоимостных параметров алгоритма Gravitation-BSF:

$$t_s = c_s\tau_{tr} = 3\tau_{tr}; \tag{14}$$

$$t_{Map} = c_{Map}\tau_{op} = 20n\tau_{op}; \tag{15}$$

$$t_r = c_r\tau_{tr} = 3\tau_{tr}; \tag{16}$$

$$t_a = c_a\tau_{op} = 3\tau_{op}; \tag{17}$$

$$t_p = c_p\tau_{op} = 14\tau_{op}. \tag{18}$$

Подставляя в формулу (1) значения правых частей из формул (14) – (18) получаем следующую формулу для оценки ускорения алгоритма Gravitation-BSF:

$$a_{Gravitation-BSF}(K) = \frac{2L + 6\tau_{tr} + 14\tau_{op} + 20n\tau_{op} + 3l\tau_{op}}{K\left(2L + 6\tau_{tr} + 14\tau_{op}\right) + \dfrac{20n\tau_{op} + 3l\tau_{op}}{K} + 11\tau_{op}}. \tag{19}$$

Граница масштабируемости алгоритма Gravitation-BSF получается путем подстановки значений правых частей из формул (14) – (18) в формулу (2):

$$K_{Gravitation-BSF} = \sqrt{\frac{20n\tau_{op} + 3l\tau_{op}}{2L + 6\tau_{tr} + 3\tau_{op}}}.$$

При $n \to \infty$ имеем

$$K_{Gravitation-BSF} \approx O\left(\sqrt{n}\right).$$

Таким образом, граница масштабируемости алгоритма Gravitation-BSF растет пропорционально корню квадратному из числа, задающего количество неподвижных тел большой массы.

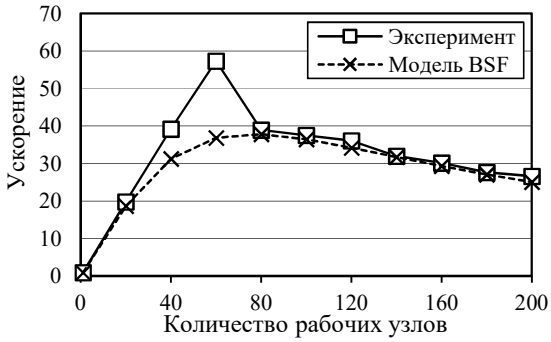

Рис. 12. Эксперимент для $n = 450$

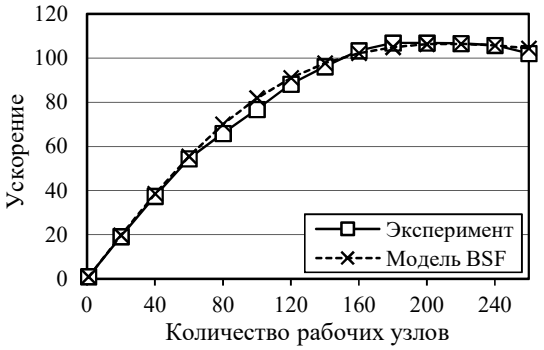

Рис. 13. Эксперимент для $n = 900$

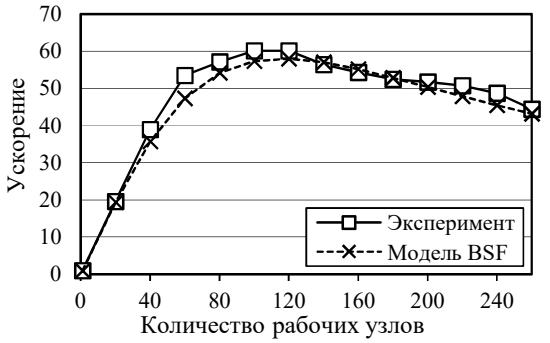

Рис. 14. Эксперимент для $n = 600$

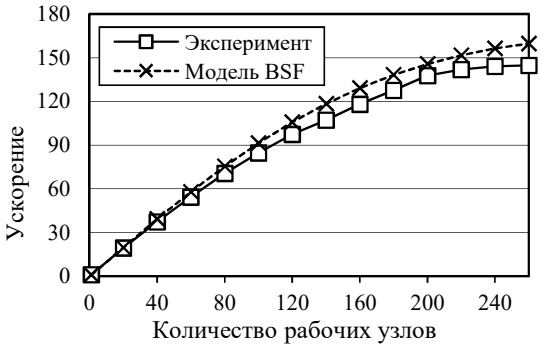

Рис. 15. Эксперимент для $n = 1200$

Для верификации результатов, полученных аналитическим путем, была выполнена реализация алгоритма Gravitation-BSF на языке C++ с использованием программного каркаса BSF и библиотеки параллельного программирования MPI. Данная реализация свободно доступна в сети Интернет по адресу https://github.com/nadezhda-ezhova/Gravitation-BSF. С помощью этой реализации исследованы ускорение алгоритма Gravitation-BSF на суперкомпьютере «Торнадо ЮУрГУ». Тестирование проводилось для 450, 600, 900 и 1200 тел. Выполнялось 10 итераций для каждого случая. Для верификации модели BSF были также построены графики ускорения алгоритма Gravitation-BSF с использованием формулы (19). Для этого экспериментальным путем были определены следующие величины в секундах: $L = 1.5 \cdot 10^{-5}$, $\tau_{op} = 2.9 \cdot 10^{-8}$ и $\tau_{tr} = 1.9 \cdot 10^{-7}$. Результаты приведены на рис. 12 – рис. 15. Во всех случаях аналитические оценки оказались очень близки к экспериментальным. Кроме того, границы масштабируемости алгоритма Gravitation-BSF, полученные аналитически, оказались очень близки к границам масштабируемости, полученным в результате экспериментов. Это подтверждает адекватность модели параллельных вычислений BSF.

В заключении в краткой форме излагаются итоги выполненного диссертационного исследования, представляются отличия диссертационной работы от ранее выполненных родственных работ других авторов, даются рекомендации по использованию полученных результатов и рассматриваются перспективы дальнейшего развития темы.

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты.

1. Разработана модель параллельных вычислений BSF, ориентированная на вычислительно сложные итерационные алгоритмы для кластерных вычислительных систем, позволяющая предсказать границу масштабируемости алгоритма на ранних стадиях его проектирования.
2. Разработан компилируемый BSF-каркас на языке C++ с использованием библиотек параллельного программирования MPI и OpenMP, инкапсулирующий все аспекты, связанные с параллелизмом, и позволяющий быстро создавать параллельные реализации алгоритмов, представленных в виде операций над списками.
3. Выполнены проектирование и реализация визуального конструктора программ BSF-Studio, позволяющего автоматизировать процесс создания BSF-программ на языке C++.
4. С использованием BSF-каркаса созданы BSF-программы для известных численных итерационных алгоритмов, на базе которых выполнена верификация BSF-модели.

Публикации по теме диссертации

Статьи в журналах из перечня ВАК

1. Ежова, Н.А. Исследование масштабируемости итерационных алгоритмов при суперкомпьютерном моделировании физических процессов / Н.А. Ежова, Л.Б. Соколинский // Вычислительные методы и программирование: новые вычислительные технологии. –2018. –Т. 19, № 4. –С. 416–430.
2. Ежова, Н.А. Модель параллельных вычислений для многопроцессорных систем с распределенной памятью / Н.А. Ежова, Л.Б. Соколинский // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. –2018. –Т. 7, № 2. –С. 32-49.
3. Ежова, Н.А. Модель параллельных вычислений BSF-MR / Н.А. Ежова, Л.Б. Соколинский // Системы управления и информационные технологии. –2019. № 3(77). –С. 15–21.
4. Ежова, Н.А. Обзор моделей параллельных вычислений / Н.А. Ежова, Л.Б. Соколинский // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. –2019. –Т. 8, № 3. –С. 58-91.
5. Ежова, Н.А. Программная поддержка модели BSF / Н.А. Ежова, Л.Б. Соколинский // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. –2019. –Т. 8, № 4. –С. 84-99.

Статьи, индексированные в SCOPUS

6. Ezhova, N.A. Scalability Evaluation of Iterative Algorithms Used for Supercomputer Simulation of Physical processes / N.A. Ezhova, L.B. Sokolinsky // Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018. –IEEE, 2018. –10 p. DOI: 10.1109/GloSIC.2018.8570131.

7. Ezhova, N. Verification of BSF Parallel Computational Model / N. Ezhova // Proceedings of the 3rd Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists. CEUR Workshop Proceedings. -2017. -Vol. 1990. P. 30–39. URL: http://ceur-ws.org/Vol-1990/paper-04.pdf.

Работа выполнена при финансовой поддержке Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства образования и науки РФ (государственное задание 2.7905.2017/8.9).

Формат 60 \ 84 1/16. Бумага офсетная. Печать офсетная. Авт. л. 1,0. Уч.-изд. л. 1,0.
Подписано в печать 7.12.2019. Тираж 100 экз.
Издательский центр ЮУрГУ. 454080, г. Челябинск, пр. Ленина, 76