На правах рукописи

Речистов Григорий Сергеевич

Разработка методов моделирования для оценки производительности суперкомпьютерных систем для параллельных приложений с однородным характером поведения

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

Автореферат диссертации на соискание учёной степени кандидата технических наук

Москва 2013

Работа выполнена в лаборатории суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур Московского физико-технического института (государственного университета).

Научные руководители:	доктор технических наук Пентковский Владимир Мстиславович доктор технических наук, профессор Плоткин Арнольд Леонидович
Официальные оппоненты:	Топорков Виктор Васильевич, доктор технических наук, профессор, ФГБОУ ВПО «НИУ «МЭИ», кафедра вычислительной техники, заведующий кафдрой Добров Андрей Дмитриевич, кандидат технических наук, ЗАО «Интел А/О», менеджер программных проектов
Ведущая организация:	Институт точной механики и вычислительной техники имени С.А. Лебедева РАН

Защита состоится «29» мая 2013 года в $16^{\underline{30}}$ часов на заседании диссертационного совета Д 409.009.01 при ОАО «Институт электронных управляющих машин им. И.С. Брука» по адресу 119334, г. Москва, ул. Вавилова, д. 24, ОАО «ИНЭУМ им. И.С. Брука».

С диссертацией можно ознакомиться в библиотеке ОАО «ИНЭУМ им. И.С. Брука».

Автореферат разослан «__» _____ 2013 г.

Учёный секретарь диссертационного совета
кандидат технических наук, профессор Красовский Виктор Евгеньевич

Общая характеристика работы

Актуальность работы

Проектирование вычислительных комплексов является сложной задачей, при решении которой необходимо учитывать особенности приложений, которые планируется на них использовать. По этой причине широко используется подход, когда разработка новых устройств сопровождается созданием их компьютерных моделей, способных с некоторой точностью проявлять себя так, как работают реальные системы. Построенные согласно этим принципам программные модели различаются между собой по назначению, точности моделирования, числу конфигурируемых параметров, скорости и принципам организации. Объединяет их, как правило, общее условие изоляции (виртуализации) моделей друг от друга и от внешней среды: программы, исполняющиеся внутри, не могут повлиять на окружение или нарушить работу самой модели. Более того, их исполнение обязано с достаточной точностью повторять их поведение, наблюдаемое на реальной аппаратуре.

При росте сложности проектируемой системы, измеряемой числом входящих в неё вычислительных ядер, узлов и соединений между ними, существующие средства моделирования рискуют оказаться неэффективными или даже бесполезными из-за резкого снижения скорости работы и/или недостаточной точности моделирования. Поэтому актуальным является нахождение сбалансированных решений, позволяющих получить результаты за приемлемое время.

Цель исследования

Целью диссертационной работы является разработка теоретических оснований, практическая реализация, апробация и оценка симуляционного[1] подхода для изучения производительности параллельных и распределённых приложений. Класс изучаемых приложений был ограничен, характеризовался однородностью проявляемого ими параллелизма и рассматривался на примере двух прикладных задач молекулярной динамики.

Научная новизна работы

Решение поставленных в диссертационной работе задач определяет научную новизну исследования, содержащуюся в создании моделей подсистем компьютерного кластера и интеграции их в единый комплекс, ис-

[1] То есть основанного на имитационном компьютерном моделировании.

пользуемый для симуляции большого числа узлов и исследования поведения и производительности параллельных приложений.

Результаты, выносимые на защиту

1) Практическая реализация программного комплекса для распределённой симуляции больших вычислительных систем, содержащих тысячи моделируемых ядер. При этом достигнутое на практике отношение суммарного числа моделируемых ядер к числу используемых физических ядер хозяйской системы достигло десяти. Замедление симуляции относительно реальности составляло от 100 до 1000.
2) Комплексный метод изучения поведения и предсказания производительности приложений, состоящий из: функциональной симуляции процессоров и периферийного оборудования, модели иерархии кэшей, инструментированной библиотеки MPI и механизма учёта сетевых коммуникаций, методики использования показаний микроархитектурных счётчиков.
3) Результаты исследования масштабирования двух приложений молекулярной динамики, полученные с использованием разработанного метода.

Практическая ценность

Практическая ценность результатов работы заключена в разработке вышеозначенного комплекса для симуляции приложений и использование его для исследования поведения и производительности программ и вычислительных систем, используемых на практике рабочими группами лаборатории суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур факультета радиотехники и кибернетики Московского физико-технического института.

Личный вклад автора

Представленные в диссертационной работе модели кэшей и неоднородной памяти, процесс распределения симуляции на большое число узлов, принципы и постановка эксперимента по сбору данных аппаратных счётчиков, а также общая схема цикла исследования разработаны и реализованы лично автором. Программа для сбора трассы MPI-вызовов и анализатор собранных трасс созданы под руководством автора.

Работы были выполнены в составе группы симуляции архитектур лаборатории суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур факультета радиотехники и киберне-

тики Московского физико-технического института, на аппаратуре вычислительного кластера лаборатории.

Апробация

Результаты работы докладывались на следующих конференциях:

1) International Conference on Computational Science 2012, г. Омаха, шт. Небраска, США [5].
2) Международная конференция «Научный сервис в сети Интернет» в 2011 и 2012 гг., г. Новороссийск [1, 8].
3) «Разработка ПО 2011» CEE-SECR 2011 г., г. Москва [2]
4) Научная конференция МФТИ в 2010, 2011, 2012 гг., г. Москва, г. Долгопрудный [3, 4, 6, 9, 12].

Публикации

По теме диссертации опубликовано пять работ, из них три — в журналах, входящих в список ВАК.

Структура и объём работы

Диссертация состоит из четырёх глав и списка литературы. Общий объём работы 162 страницы.

Содержание работы

Глава 1 определяет цель, научную новизну, преследуемые результаты и производит постановку задачи диссертационной работы. В ней приводится информация о публикациях и апробации, даются определения основных понятий и обзор литературы. Определения основных терминов, используемых в работе, приведены ниже.

Симулятор (*англ.* simulator) — программа, моделирующая некоторую физическую систему через предоставление корректных интерфейсов входящих в неё подсистем и обеспечивающая правильное их функционирование, но не гарантирующая того, что их внутреннее устройство будет похоже на устройство аналогичных подсистем реальной ЭВМ. В частности, длительности отдельных операций, задержки доставки данных и т.п. в модели и реальной системе не обязаны совпадать.

Хозяин (*англ.* host) — физическая система, на которой исполняется программа-эмулятор.

Гость (*англ.* guest) — система, поведение которой должен демонстрировать симулятор и внутри которой исполняются гостевые приложения. Синонимичным является понятие «целевая система» (*англ.* target system).

Полноплатформенный симулятор (*англ.* full platform simulator) — модель, включающая в себя компоненты, достаточные для изучения некоторой ЭВМ в целом, состоящая из следующих основных компонентов: процессор, память, дисковые устройства, сетевые устройства, клавиатуру, мышь, монитор и др. Внутри такого симулятора возможно запустить операционную систему, и она будет работать так же, как работала бы на реальной аппаратуре.

Функциональная модель (*англ.* functional model) — система, точность которой ограничена корректной функциональностью модели без обеспечения правильных значений длительностей её операций. Например, в такой модели доступ к оперативной памяти возвращает правильное значение, но за один моделируемый такт времени, тогда как в реальности он занял бы от 3 до 200 тактов, в зависимости от состояния системы кэшей. Подобные модели недостаточно точны для предсказания производительности, но, как правило, достаточны для корректной работы большинства ПО, включая операционные системы.

Потактовая модель (*англ.* cycle precise model, performance model) — симулятор, корректно высчитывающий ход времени внутри моделируемой системы. Для этого он моделирует её внутреннее устройство более детально, чем это необходимо для функциональных моделей. Потактовые модели почти всегда во много раз медленнее функциональных.

Проведённый в работе обзор литературы показывает, что в настоящее время компьютерная симуляция широко используется для различных типов вычислительных систем, программ и алгоритмов, в том числе: отдельных микропроцессорных блоков, полных платформ, сетей передачи данных; пользовательских приложений как под управлением операционных систем, так и в предположении изоляции. Тем не менее, из обзора видно, что не существует известных успешных попыток промоделировать целиком сверхбольшие системы, содержащие в себе одновременно многопроцессорные вычислительные блоки и сеть передачи данных, исполняющие операционную систему и прикладные приложения, при этом спроектированные на основе неспециализиированного, коммерчески до-

ступного оборудования.

Завершается глава формулировкой основной задачи диссертационной работы: построение новых и адаптация существующих инструментов для исследования производительности параллельных приложений на больших аппаратных комплексах заранее, до момента фактического построения последних.

В **главе 2** даются теоретические обоснования возможности симуляции и практические аспекты применимости метода для исследуемой системы и приложений. Делаются выводы о практической целесообразности и ограничениях масштабируемости симуляции для различных классов прикладных задач.

Теоретическим основанием для самой возможности моделирования ЭВМ является тезис Чёрча–Тьюринга: любая интуитивно вычислимая функция является частично вычислимой, или, иными словами, может быть вычислена некоторой машиной Тьюринга.

Использование симуляторов для изучения параллельных приложений является важной альтернативой аналитическим методам, так как позволяет значительно глубже заглянуть в структуру поведения изучаемого приложения и аппаратного обеспечения. Кроме того, понятие «виртуализация» и технологии, лежащие в её основе, в недавнее время давшие новый толчок для развития средств консолидации серверов, повышения уровня изолированности и безопасности исполнения, в основе своей имеют принципы, идентичные используемым при построении полноплатформенных симуляторов.

В главе вводятся понятия служебных, привилегированных и безвредных инструкций и операций и показывается, что достаточное условие возможности построения эффективного симулятора выражается в следующем утверждении: **множество служебных операций является подмножеством привилегированных операций**.

В главе анализируется применимость разрабатываемых методов для следующих классов задач: приложения с большим объёмом выходных данных; приложения, использующую общую память; приложения с большим объёмом промежуточных данных; приложения с большим объёмом перекачиваемых данных; приложения, использующие сопроцессорные устройства. Показывается, что не все из них могут быть эффективно или целесообразным образом запущены внутри существующих симуляторов.

Автором делается вывод, что не все типы приложений могут быть эффективно или целесообразным образом запущены внутри существующих симуляторов, так как особенности работы программ на реальных системах вызовут резкое падение скорости внутри модели или будут иметь

невыполнимые требования на ресурсы. Однако, для прикладных задач, использованных в данной работе и описанных далее, были обнаружены следующие особенности, значительно облегчающие их симуляцию.

- Фиксированный размер обрабатываемых данных после выхода на рабочий режим.
- Однородный характер поведения — все потоки приложения при выходе на рабочий режим исполняют один и тот же код, при этом частоты событий, таких как промахи кэшей, пересылка сообщений по сети и т.п. с высокой точностью одинаковы.
- Приложения исполняются только на центральных процессорах, для которых в наличии имеются высокопроизводительные программные модели, и не используют ускорители и сопроцессорные платы.

В таких условиях наблюдаемое замедление гостевых приложений будет в основном определяться отношением числа моделируемых ядер к числу доступных хозяйских, что открывает путь для их изучения с помощью симуляции, детали реализации которой описываются в следующей главе.

В **главе 3** даётся описание основных компонент полной модели: кэшей, сети, а также определяется метод компенсации недостаточной точности симуляции с помощью прямых измерений. Затем приводится используемая формула вычисления производительности параллельных приложений на многомашинных многоядерных системах. В конце главы рассматривается проблема организации распределения самого процесса симуляции на множество узлов хозяйской системы.

Модель кэша В начале главы даётся описание созданной модели иерархии памяти, подключаемой к Simics. Оригинальный кэш Simics, представленный классом `g-cache`, не подходил для нужд этой работы по ряду причин:

1) `g-cache` предназначен для исследования систем с одним или двумя уровнями кэшей, тогда как существующие системы процессоров Intel имеют три уровня;
2) моделируются только SMP-системы с постоянной задержкой для всех доступов в память, реальные серверные и кластерные узлы имеют неоднородную организацию памяти (*англ.* NUMA);
3) обеспечивается поддержание когерентности памяти только с помощью общей шины по протоколу MESI, тогда как в современных системах отдельные процессоры соединены с помощью линий QPI, сообщение по которым организовано иначе — используется протокол MESIF.

Для обхода этих недостатков на основе исходного кода `g-cache` автором были созданы новые классы моделей: `h-cache` для моделирования кэшей и `qpi-staller` для моделирования неоднородности памяти. Все уровни кэшей (L1–L3) моделировались как устройства класса `h-cache`. Модель кэшей поддерживала протокол MESIF, учитывающий возможность наличия данных в L3-кэше другого сокета, конфигурируемые длительности различных событий (промахи чтения, промахи записи, промахи с получением данных из кэша другого сокета и т.п.), политику вытеснения LRU. Модель неоднородной памяти `qpi-staller` позволяла задать диапазон физических адресов общей памяти, локальный для данного сокета; для него она возвращала одно значение задержки доступа, для всех остальных адресов, считавшихся дальними — второе, большее значение. Кроме того, она перенаправляла запросы о наличии данных в кэше при промахах в иерархию соседнего сокета.

Так как учёт работы иерархии памяти существенно замедляет симуляцию, эти модели подключались только в момент запуска исследуемых приложений и «прогревались» в течение 200 симулируемых секунд. Параметры конфигурации каждого уровня иерархии даны в табл. 1, организация соединений между ними соответствовала спецификациями физической системы изображена на рис. 1.

Таблица 1

Параметры системы кэшей модели

Величина	**Значение, тактов**
Задержка промах L1 (чтение)	3
Задержка промах L1 (запись)	3
Задержка промах L2 (чтение)	12
Задержка промах L2 (запись)	10
Задержка промах L3 (чт./зап.)	35
Задержка DDR3 (ближняя)	100
Задержка DDR3 (дальняя)	220

Сбор трассы сетевых коммуникаций Инструмент `mpi-tracer` был создан под руководством автора для исследования влияния коммуникаций, осуществляемых по стандарту MPI, на производительность программ. Собираемая с его помощью история (трасса) всех вызовов MPI процедур, связанных с коммуникациями, позволяет определить характерные для конкретного приложения типы взаимодействия (наример, парные или коллективные) с их длительностями, увидеть характерные фазы в поведении

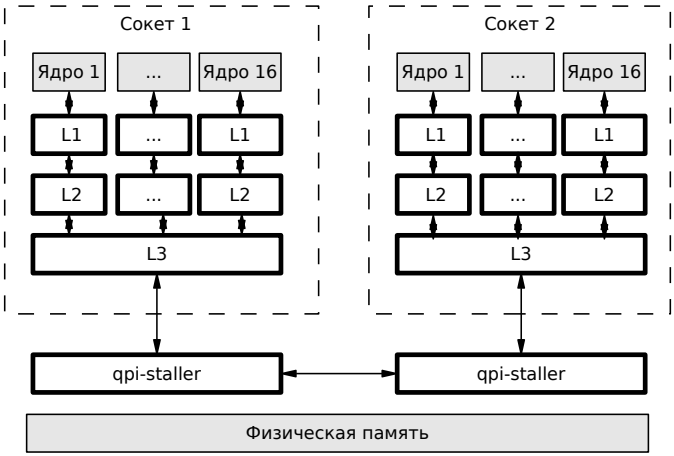

Рис. 1 Иерархия памяти в моделируемой системе. Белые блоки соответствуют созданным моделям, вычисляющим задержки операций, серые блоки — исходным функциональным моделям, к которым они подключаются

приложения, и в результате проанализировать влияние параметров сети (пропускная способность отдельных соединений, топология) на скорость вычислений.

`mpi-tracer` состоит из профилирующей библиотеки, подключаемой к исследуемой программе, и модуля для Simics. Библиотека переопределяет процедуры MPI для перехвата обращения к ним. Для этого используется «магическая» инструкция моделируемого процессора[1], не влияющая на исполнение программы, но вызывающая в симуляторе исполнение определённой пользователем функции (рис. 2). Она позволяет передавать ограниченный объём информации в симулятор. Инструмент `mpi-tracer` сохраняет порядковый номер, код возврата и аргументы функции, а также номер процесса (MPI rank). Эти данные позволяют определить, какие процессы взаимодействовали и какой алгоритм они использовали, сколько данных было переслано каждым процессом и сколько времени для этого понадобилось.

Далее созданная трасса используется для вычисления среднего времени, проводимого каждым потоком приложения внутри обработки вызова MPI.

[1]В архитектуре Intel для этого выбрана `CPUID`.

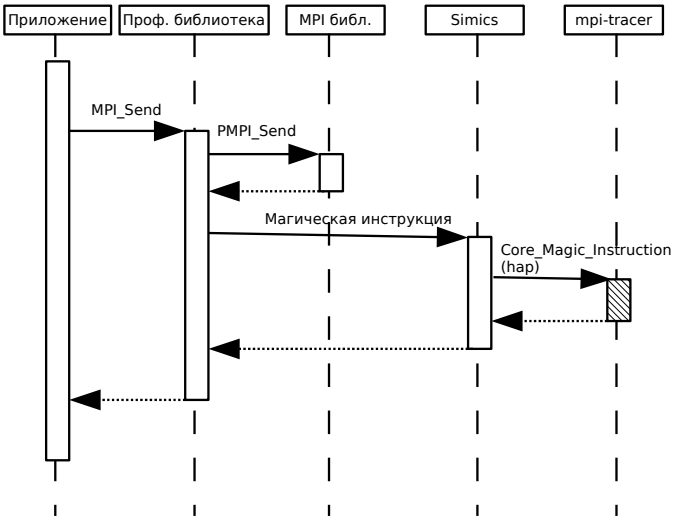

Рис. 2 Последовательность вызовов при исполнении магической инструкции

Получение метрик производительности приложений Далее в главе приводится описание разработанной методики для определения производительности. Определения двух изучаемых метрик приложений:

- число завершившихся машинных инструкций за один такт процессора — IPC (*англ.* instructions per cycle); обратная ей величина — CPI (*англ.* cycles per instruction);
- число совершённых операций над числами с плавающей точкой за одну секунду — FLOPS (*англ.* floating point operations per second).

Основой метода является анализ причин, влияющих на среднюю длительностью выполнения одной инструкции, то есть на CPI. Численное значение CPI разбивается на сумму факторов, *независимо* влияющих на производительность:

$$CPI = CPI_{core} + CPI_{caches} + CPI_{memory} + CPI_{MPI}, \quad (1)$$

где

$$CPI_{caches} = CPI_{L1} + CPI_{L2} + CPI_{L3}. \quad (2)$$

В (1) CPI_{core} — количество тактов, затрачиваемых на исполнение непосредственно в ядре процессора, CPI_{caches} — суммарные задержки при обращениях в кэш-память, CPI_{memory} — задержки при доступе в оперативную память, CPI_{MPI} — коммуникации с помощью интерфейса MPI.

Все члены, кроме CPI_{core}, могут быть раскрыты как математическое ожидание соответствующего набора событий для каждой из моделей:

$$CPI_{Lk} = \sum P\{cacheLk_i\} \cdot L_i^{cacheLk}, \text{где } k \in \{1, 2, 3\}, \tag{3}$$

$$CPI_{memory} = \sum P\{mem_i\} \cdot L_i^{mem}, \tag{4}$$

$$CPI_{MPI} = \sum P\{MPI_i\} \cdot L_i^{MPI}. \tag{5}$$

Здесь $P\{cacheLk_i\}$ — вероятность i-го типа события для кэша уровня k, таких как попадания и промахи при чтении и записи; $L_i^{cacheLk}$ — длительности обработки событий в кэшах; $P\{mem_i\}$ — вероятность i-го типа обращения к памяти (дальней или локальной), L_i^{mem} — соответствующая им задержка; $P\{MPI_i\}$ — вероятность вызова i-ой функции библиотеки MPI, L_i^{MPI} — длительность передачи данных соответствующей функцией.

Нахождение CPI_{core} методами симуляции затруднено, так как требует наличия потактовой модели, учитывающей все особенности конвейерной обработки инструкций, знаний о деталях микроархитектуры исследуемого ЦПУ и т.п. Кроме того, такие модели обладают крайне низкой скоростью работы, и их использование усугубило бы проблему с большим временем симуляции больших систем. По этой причине эта величина вычислялась из значений архитектурных счётчиков, встроенных в процессоры Intel, с помощью Intel VTune.

Для получения значения FLOPS из CPI необходимо учесть, что не все инструкции выполняют собственно вычисления над числами с плавающей запятой, а также что современные векторные инструкции способны выполнять операции над двумя (в случае SSE) или четырьмя (для AVX) парами операндов. Вводится коэффициент α, показывающий, сколько операций с плавающей точкой приходится на одну инструкцию, значение FLOPS получается из следующего соотношения:

$$FLOPS^{-1} = CPI_{core}/\alpha + CPI_{caches} + CPI_{memory} + CPI_{MPI}. \tag{6}$$

Проведённый в работе анализ данных, полученных с помощью VTune, позволил обнаружить следующий факт: для исследуемых приложений для измерения CPI_{core} с удовлетворительной точностью достаточно рассматривать только самый «горячий» их участок, при этом в нём содержится наибольшее количество инструкций над значениями с плавающей точкой (векторных команд SSE/AVX), необходимых для вычисления α.

Значение α получается из рассмотрения статистики, собранной с помощью VTune, для счётчиков исполненных приложением векторных команд.

Организация распределённой симуляции Модели исследуемых систем содержали в себе большое число ядер — до тысяч — и требовали для работы сотни гигабайт памяти. Современные одиночные компьютеры не в состоянии удовлетворить этим условиям, поэтому необходимым являлось рассредоточение частей модели по многим хозяйским системам. Оригинальный Simics поддерживает распределение симуляции на несколько хозяйских систем, однако с увеличением их количества задача координирования и управления их работой сильно усложняется; кроме того, часть узлов хозяйского кластера обычно занята другими пользователями, непрерывно занимающими и освобождающими ресурсы. По этой причине в работе автором реализована интеграция запуска Simics с системой управления ресурсами кластерных систем SLURM (Simple Linux Resource Manager). Она реализована в виде скрипта `simics-slurm`, принимающего все обычные опции Simics, но дополнительно выполняющего следующие шаги.

1) Запросить у SLURM число узлов, достаточное для старта симуляции. Если они недоступны немедленно, то встать в очередь ожидания вместе с задачами других пользователей.
2) При успешном получении ресурсов от SLURM в переменной окружения `SLURM_NODELIST` хранится список выданных в данном сеансе узлов. На первом их них с помощью команды SSH запускается первая копия Simics, содержащая в себе модель master-узла. Этот шаг необходимо выполнять, чтобы освободить головной узел физического кластера, т.к. он используется многими пользователями и не рассчитан на вычислительно интенсивные задачи.
3) Дополнительные процессы Simics стартуют на оставшихся выданных узлах с помощью команды SSH. Каждый из них открывает новое соединение, используемое для нужд координации симуляции, с центральным процессом (рис. 3) и создаёт подгруппу гостевых slave-систем, которые будут исполняться параллельно.
4) После сигнализации готовности всех процессов Simics выдаётся глобальная команда начать симуляцию.
5) Последующие шаги определяются переданным `simics-slurm` файлом сценария, написанном на Python или языке скриптов Simics.

Использование такой схемы проведения отдельных экспериментов позволило автоматизировать процесс сбора данных для различных конфи-

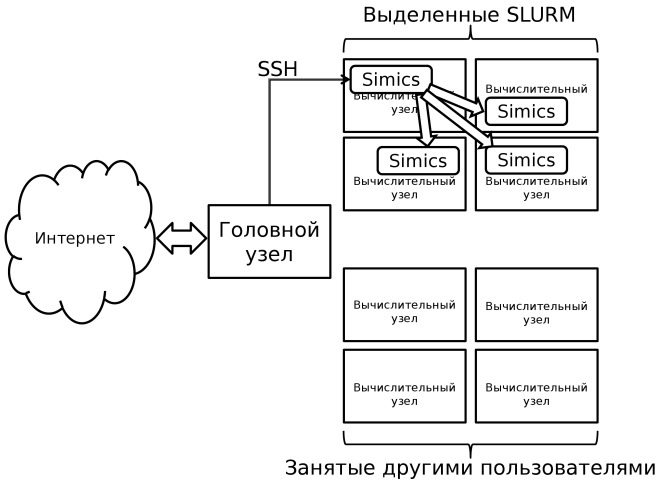

Рис. 3 Процесс распределения частей модели при использовании SLURM

гураций симулятора и при этом эффективно утилизировать возможности Simics и ресурсы хозяйской системы.

Общая последовательность проведения исследования изображена на рис. 4. Для получения части компонент задействованы модели кэшей, памяти и сети, построенные на основе фреймворка симулятора Wind River Simics. Для измерения CPI_{core}, точное моделирование которого в условиях рассматриваемой задачи затруднено, используется Intel VTune. Смысл входящих в схему обозначений раскрываются в следующей главе.

В **главе 4** приводится описание систем, на которых проводилось внедрение и модели которых были построены, а также прикладных приложений, изучение поведения которых на моделируемой системе представляло интерес. Затем даются результаты масштабируемости и производительности, полученные для созданой модели и изучаемых приложений.

Кластеры Gen1 и Gen2 Исследования велись на двух кластерах, различавшихся конфигурациями, пиковыми вычислительными мощностями и датами введения в строй. Их модели создавались таким образом, чтобы на уже существующем оборудовании иметь возможность изучать будущую систему. В таблице 2 приведены параметры этих систем, далее именуемых по номерам «поколений»: Gen1 и Gen2.

Два приложения, моделирование которых проводилось в работе:

- Gromacs — пакет молекулярной динамики для моделирования ле-

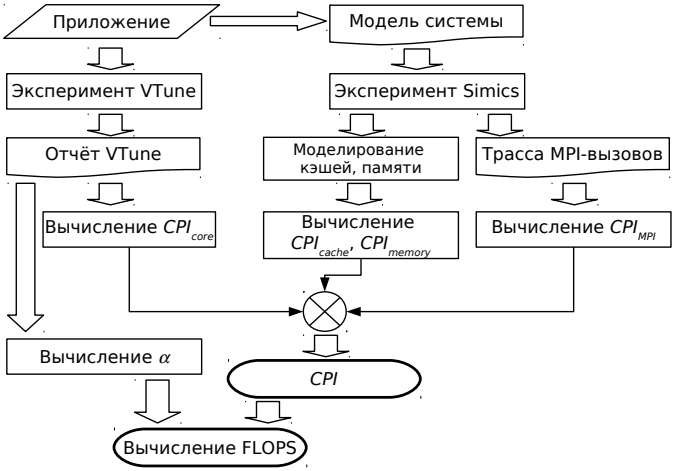

Рис. 4 Общая последовательность исследования

Таблица 2

Конфигурации Gen1 и Gen2

Параметр	**Gen1**	**Gen2**
Год ввода в эксплуатацию	2011	2012
Число выч. узлов	16	112
Число процессоров в одном выч. узле	2	2
Число ядер в одном процессоре	12	16
Тип процессора	Intel Xeon X5680 (Westmere)	Intel Xeon CPU E5-2690 (Sandy Bridge)
Частота процессоров	3,33 ГГц	2,9 ГГц
ОЗУ одного узла	24 Гбайт	48 Гбайт
Сеть	Infiniband QDR 10 Гбит/с	Infiniband QDR 10 Гбит/с
Полное число ядер в системе	384	3584

карственных средств, использовалась версия 4.5.4, собранная с поддержкой MPI и двойной точностью;
- Amber — ещё один набор приложений для молекулярной динамики, использовалась версия Amber 11, собранная с поддержкой MPI.

Масштабируемость и скорость симуляции Главным фактором, жёстко ограничивающим максимальное число моделируемых машин, которое было возможно запустить на одном хозяйском узле, являлось потребление памяти гостевыми системами. Каждой копии ОС для устойчивой работы при симуляции требовалось около 1,5 Гбайт, что при объёме хозяйского ОЗУ 24 Гбайт создавало ограничение в 16 систем на узел. Однако, на практике их число было ограничено 12 по числу хозяйских ядер Gen1 для того, чтобы все потоки симулятора имели возможность исполняться параллельно.

Simcis имеет встроенный механизм `system-perfmeter` для измерения скорости симуляции, что позволяет оценить замедление приложений по сравнению с их прямым исполнением на реальной аппаратуре. В таблице 3 приведены результаты измерения скорости симуляции Gromacs для экспериментов с 64 моделируемыми узлами и аналогичные данные для Amber, исполнявшегося на 16 моделируемых узлах, для одного процесса Simics в различных фазах экспериментов.

Таблица 3

Скорость симуляции для различных фаз экспериментов Gromacs и Amber. Данные усреднены по интервалу каждой фазы

Фаза	**Длительность, сим. с**	**Скорость Gromacs, MIPS**	**Скорость Amber, MIPS**	**Комментарий**
Загрузка ОС	≈ 200	4900	5500	Аппаратно ускоренная симуляция
Моделирование кэшей	200+200	52	61	Подключены иерархии кэшей и памяти с предварительным «прогревом» модели
Сбор трасс MPI	200	154	142	Подключен `mpi-tracer`

Применение `mpi-tracer` На рис. 5 приведены зависимости относительных частот вызовов для различных функций MPI, связанных с точечными и коллективными взаимодействиями, от числа моделируемых узлов кластера, для приложения Gromacs, собранные с помощью `mpi-tracer`.

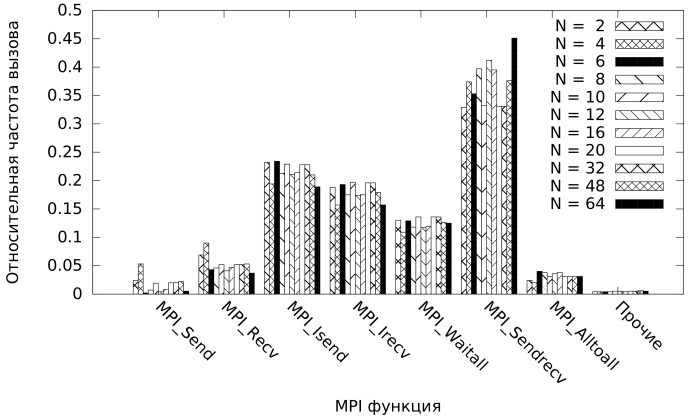

Рис. 5 Профиль относительной частоты вызовов MPI для Gromacs в различных конфигурациях

Независимость CPI_{core}, CPI_{caches} от числа потоков На рис. 6 приводится сравнение точности метода нахождения CPI_{core} с помощью VTune — зависимость величины от числа процессов MPI для Gromacs для двух задач моделирования веществ. Сравниваются значения, полученные двумя способами: с помощью непосредственных замеров на аппаратуре и с помощью моделирования на Simics.

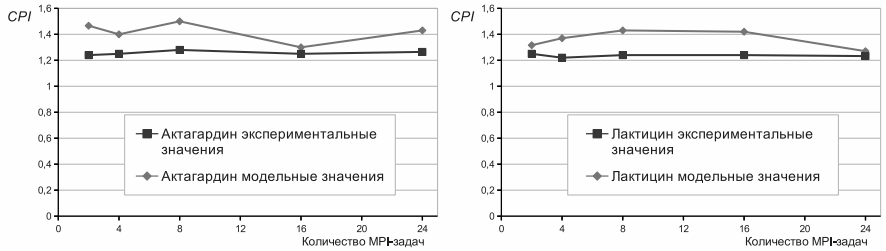

Рис. 6 Сравнение модельного и измеренного CPI_{core} для Gromacs

Для первой симуляции (вещество актагардин) средняя погрешность мо-

дельного значения составила 8%, для второй (вещество лактицин) — 6%. Модельные результаты всех измерений оказались завышенными по сравнению с реальными. Скорее всего, это вызвано излишне пессимистичной оценкой производительности подсистемы кэшей, заложенной в модель.

Сравнение вычисленной и экспериментальной производительностей приложений На рис. 7 показаны зависимости величины FLOPS от числа узлов для обоих приложений (каждый узел содержал 16 процессов), а также значения их экспериментально измеренной производительности для аналогичных конфигураций для тех точек, для которых удалось провести измерения на реальной аппаратуре.

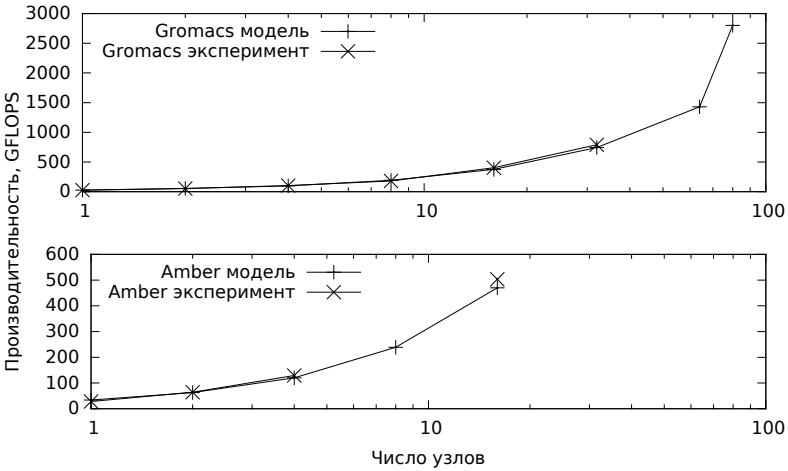

Рис. 7 Зависимости предсказанной и измеренной производительности от числа узлов

Следует отметить, что в процессе выполнения работы было обнаружено, что Amber имеет ограничение на максимальное количество потоков, равное 256, что отражено в графике рис. 7, тогда как Gromacs лишён подобного недостатка.

Выводы по результатам диссертации

В диссертационной работе рассмотрены вопросы возможности реализации и описана практическое воплощение методов программной симуляции приложений, проявляющих однородный параллелизм, для исследова-

ния вопросов их функциональности и производительности на аппаратных системах заранее, до их физической доступности.

Следует отметить, что достигнутая в работе скорость симуляции сравнима с опубликованными для подобных исследований результатами. Однако представленное в работе решение ориентировано на системы архитектуры Intel, тогда как описанные в литературе параллельные симуляторы в основном нацелены на менее распространённые системы или не ставят перед собой задач исследования производительности.

Использование компьютерной симуляции для получения сведений о характере работы приложения на ещё не построенной вычислительной системе предоставляет исследователю выгодные компромиссы между временем создания модельной конфигурации, скоростью измерений и количеством/точностью получаемых данных. Из-за масштаба задачи сам процесс моделирования не является тривиальным, однако современные программные решения позволяют выполнить его на доступном оборудовании.

При выполнении диссертации были получены следующие основные результаты:

- Получены теоретические обоснования применимости разрабатываемого метода к различным классам прикладных приложений и выбран класс задач, на котором будет сфокусирована работа.
- Разработана подключаемая и конфигурируемая модель неоднородных иерархий памяти, позволяющая получить данные о влиянии задержек при обращении к памяти на скорость исполнения программ.
- Разработан инструмент `mpi-tracer`, предназначенный для исследования влияния коммуникаций MPI на производительность параллельных программ.
- Выработана теоретическая база и создана методика для определения производительности приложений через анализ встроенных в микропроцессоры Intel счётчиков производительности и инструмента Intel VTune Analyzer.
- Отдельные созданные компоненты объединены в единый комплекс на основе Wind River Simics, который был использован для симуляции приложений молекулярной динамики.

Проведенный в работе анализ показывает, что для большого класса практических задач симуляция позволяет получить базовые оценки параметров работы приложений без привлечения натурного эксперимента, зачастую невозможного из-за недоступности оборудования. Для программ с однородным параллелизмом показано, что такой анализ является эффективным.

Важным результатом данной работы является создание комплексной модели вычислительного кластера, позволяющей исследовать различные факторы, влияющие на производительность. Модель являлась адекватной для анализа поведения программ и вычислительных систем, используемых на практике рабочими группами лаборатории и была нацелена на применимость к системам, которые планируется построить в ближайшем будущем. Опытная эксплуатация описанных решений в работе группы симуляции лаборатории суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур МФТИ позволила сделать вывод об их пригодности к практическому использованию для решения существующих задач анализа и оптимизации существующих и будущих программно-аппаратных комплексов.

Публикации по теме диссертации

1. Моделирование и исследование архитектуры многопроцессорной системы с использованием распределённой моделирующей среды Graphite / Г.С. Речистов, А.А. Иванов, П.Л. Шишпор, В.М. Пентковский // Труды международной суперкомпьютерной конференции и конференции молодых учёных «Научный сервис в сети Интернет. Экзафлопсное будущее». — 2011. — С. 143—146. — ISBN: 978-5-211-06229-0. — URL: `http://agora.guru.ru/abrau2011`.
2. Моделирование компьютерного кластера на распределённом симуляторе. Верификация моделей вычислительных узлов и сети кластера. / Г.С. Речистов, А.А. Иванов, П.Л. Шишпор, В.М. Пентковский // Труды конференции «Разработка ПО 2011» CEE-SECR 2011. — 2011. — URL: `http://2011.secr.ru/lang/ru-ru/talks/modeling-of-a-computer-cluster-on-a-distributed-simulator`.
3. Опыт подготовки студентов в учебно-исследовательской лаборатории МФТИ–«Интел» / Речистов Г.С. [и др.] // Труды МФТИ. т. 3. — 2011. — С. 168—170.
4. Симуляционный подход для нахождения производительности параллельных MPI-приложений на вычислительном кластере / Г.С. Речистов, А.А. Иванов, П.Л. Шишпор, В.М. Пентковский // Труды 54-й научной конференции МФТИ «Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе». Т. 01. — Москва-Долгопрудный-Жуковский, 2011. — 82–83. — ISBN: 978-5-7417-0396-0.

5. Simulation and Performance Study of Large Scale Computer Cluster Configuration: Combined Multi-level Approach / G.S. Rechistov, A.A. Ivanov, P.L. Shishpor, V.M. Pentkovski // Procedia Computer Science. — 2012. — Т. 9. — С. 774—783. — ISSN: 1877-0509. — DOI: 10.1016/j.procs.2012.04.083. — URL: http://www.sciencedirect.com/science/article/pii/S1877050912002049 ; Proceedings of the International Conference on Computational Science, ICCS 2012.

6. Анализ производительности системы при выполнении программы Gromacs / Н.Н. Щелкунов, Д.А. Гаврилов, Г.С. Речистов, А.А. Абдухаликов // Труды 55-й научной конференции МФТИ «Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе». Т. 01. — Москва-Долгопрудный-Жуковский, 2012. — С. 14—15. — ISBN: 978-5-7417-0481-3.

7. Моделирование компьютерного кластера на распределённом симуляторе. Верификация моделей вычислительных узлов и сети кластера / Г.С. Речистов, А.А. Иванов, П.Л. Шишпор, В.М. Пентковский // Программная инженерия. — 2012. — № 6. — С. 24—29. — ISSN: 2220-3397. — URL: http://novtex.ru/pi.html.

8. *Речистов Г.С.* О верхних и нижних границах оценок производительности многомашинных многопроцессорных комплексов на научных приложениях // Труды международной суперкомпьютерной конференции и конференции молодых учёных «Научный сервис в сети Интернет. Поиск новых решений». — Издательство МГУ, 2012. — С. 671—675. — ISBN: 978-5-211-06394-5. — URL: http://agora.guru.ru/display.php?conf=abrau2012&page=item011 (дата обр. 26.12.2012).

9. *Юлюгин Е.А.*, *Речистов Г.С.*, *Иванов А.А.* Моделирование нагрузки на сетевое оборудование // Труды 55-й научной конференции МФТИ «Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе». Т. 01. — Москва-Долгопрудный-Жуковский, 2012. — С. 81—82. — ISBN: 978-5-7417-0481-3.

10. Реализация инструментария для исследования сетевой производительности MPI-приложений на распределённом симуляторе / Н.С. Поливанов, Г.С. Речистов, А.А. Абдухаликов, В.М. Пентковский // Информационные технологии. — 2013. — Янв. — № 1. — С. 46—50.

11. *Речистов Г.С.* Использование полноплатформенного имитационного моделирования суперкомпьютерной системы для определения производительности счётных приложений // Информационные технологии. — 2013. — Мар. — № 3. — С. 29—32.

12. *Речистов Г.С.* Бенчмарк LINPACK для архитектуры Intel IA-32: существующие варианты, сравнительный анализ возможностей и демонстрируемой производительности // Труды 54-й научной конференции МФТИ «Проблемы фундаментальных и прикладных естественных и технических наук в современном информационном обществе». Т. 01. — 71–72. — ISBN: 978-5-7417-0396-0.