Отображения параллельных алгоритмов на суперкомпьютеры экзафлопсной производительности на основе имитационного моделирования*

Б. М. Глинский[1,2]***, М. А. Марченко***[1,2]***, А. С. Родионов***[1,2]***, Д. А. Караваев***[1]***, Д. И. Подкорытов***[1]

gbm@sscc.ru

[1]ФБГУН Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, Россия; [2]Новосибирский государственный университет, Новосибирск, Россия

Целью работы является исследование возможности отображения параллельных алгоритмов на архитектуру суперЭВМ экзафлопсной производительности с использованием метода имитационного моделирования. Авторами предложена система AGent NEtwork Simulator (AGNES) для исследования масштабируемости алгоритмов и программного обеспечения на предполагаемых архитектурах экзафлопсных суперкомпьютеров. Приведены результаты моделирования алгоритмов различного класса: алгоритмы прямого статистического моделирования, сеточные методы.

Ключевые слова: *агентно-ориентированная система; имитационное моделирование; масштабируемые параллельные алгоритмы*

Mappings of parallel algorithms on supercomputers with exaflops performance on the basis of simulation*

B. M. Glinsky[1,2]***, M. A. Marchenko***[1,2]***, A. S. Rodionov***[1,2]***, D. A. Karavaev***[1]***, and D. I. Podkorytov***[1]

[1]Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia; [2]Novosibirsk State University, Novosibirsk, Russia

The main objective of this research is a possibility of representation of parallel algorithms on different architectures of exaflops supercomputers based on simulation. The authors have proposed AGent NEtwork Simulator(AGNES) for investigating the scalability of algorithms and program software on admissible architectures of exaflops supercomputers. In this paper, the results of the simulation of different class algorithms are presented. These are algorithms of the forward statistical modeling and grid method. The problem of investigating the properties of scalability of parallel algorithms for implementing them on supercomputers of the future with exaflops performance goes beyond the scope of technological problems. In this paper, the authors show that it is possible to estimate the behavior of algorithms and to develop a modified computation scheme by implementing them on a simulation model. The imitating model allows one to identify the bottlenecks in algorithms and to find out how to modify an algorithm and what parameters need to be configured to scale this algorithm to a greater amount of cores. In the ICMMG, the simulation system AGNES has been developed, which was used for studying the scalability of distributed statistical modeling and for solving the problem of numerical three-dimensional modeling of seismic wave propagation. Real calculations have shown that Monte-Carlo method is linearly parallelized up to 1,000 computing cores. The

*Работа выполнена при финансовой поддержке РФФИ, проекты №№ 12-01-00727, 13-07-00589, 14-07-00832 и 14-05-00867, МИП 130 СО РАН, МИП 39 СО РАН, Программы РАН 4.9.

behavior of the method proposed was investigated with simulation up to $5 \cdot 10^5$ cores. The dependence of parallelization on the number of collector cores was shown and the modified computing scheme for a great number of cores was proposed. For the other problem, a good compliance between experimental and model results up to 32768 cores is shown. The results were obtained on the simulation model for 1,124,864 cores. The calculations were performed on clusters of The Siberian Supercomputer Center.

Keywords: *agent oriented system; scalable parallel algorithms; exaflops supercomputers*

Введение

Проблема исследования свойств масштабируемости параллельных алгоритмов при их реализации на будущих суперЭВМ экзафлопсной производительности выходит за уровень технологических задач и требует научно-исследовательского подхода к ее решению. Вычислительные алгоритмы, как правило, являются более консервативными по сравнению с развитием средств вычислительной техники. Оценить поведение алгоритмов, разработать модифицированные схемы вычислений можно уже сейчас путем реализации их на имитационной модели, отображающей тысячи и миллионы 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. Задача моделирования алгоритмов для исследования их масштабируемости не является новой, ею занимаются многие группы исследователей во всем мире, ктически с начала «эры параллельного программирования», один из ранних обзоров приведен в [1]. Хорошим примером ранних проектов по моделированию исполнения параллельных программ в изменяемом вычислительном окружении является проект PARSIT [2]. Идеи этого проекта актуальны и сейчас, но он, естественно, не был ориентирован на крупномасштабные вычисления, количество ядер ограничивалось несколькими тысячами. Имеется много исследований, ориентированных на моделирование исполнения конкретного алгоритма или узкого класса алгоритмов (например, матричной алгебры) [3,4]. Построение и исследование подобных моделей, очевидно, существенно проще, чем построение моделей универсальных. Наиболее известным из подобных проектов является BigSim (http://charm.cs.uiuc.edu/research/bigsim), проект проводимый в США (Университет Урбана-Шампань, Иллинойс), руководитель проекта Kale Laxmikant). Проект направлен на создание имитационного окружения, позволяющего разработку, тестирование и настройку посредством моделирования ЭВМ будущих поколений, одновременно позволяя разработчикам ЭВМ улучшать их проектные решения с учетом специального набора приложений [5]. Однако этот проект для целей исследования масштабирования слишком глобален, он требует детального описания вычислительной архитектуры и профилирования исполнения программы на низком уровне, что зачастую излишне для простой сравнительной проверки решений, когда интересны лишь относительные, а не абсолютные значения времен. В Институте системного программирования РАН (г. Москва) под руководством академика В. П. Иванникова разработана модель параллельной программы, которая может эффективно интерпретироваться на инструментальном компьютере, обеспечивая возможность достаточно точного предсказания 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 [6, 7]. Модель получается преобразованием дерева управления программы, которое для Java-

программ может быть построено путем модификации абстрактного синтаксического дерева. Для моделирования коммуникационных функций используется модель LogGP, что позволяет учитывать специфику распределенной вычислительной системы. 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 (Run Time System). 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. Стоит отметить, что ParJava и BIGSIM не учитывают, по крайней мере явно, вопросы отказоустойчивости при исполнении больших программ, в то время как использование в вычислениях одновременно десятков и сотен тысяч, а для отдельных задач и миллионов вычислительных ядер не может их не поставить. В ИВМиМГ СО РАН развивается мультиагентный подход, 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 временных характеристик каждой операции [8].

Система моделирования AGNES

Первоначально разработанная для моделирования телекоммуникационных и информационных сетей [8, 9], система AGNES показала свою эффективность и для моделирования исполнения высокопроизводительных параллельных программ [10]. Пакет AGNES базируется на Java Agent Development Framework (JADE) [4]. JADE - это мощный инструмент для создание мультиагентных систем, и он состоит из трех частей: среда исполнения агентов; библиотека базовых классов, необходимых для разработки агентной системы; набор утилит, позволяющих наблюдать и администрировать MAC. JADE приложение обладает рядом важных свойств для агентных систем. Распределенность, JADE позволяет создавать приложения, запускаемые на локальной сети. Универсальность, поддержка стандарта FIPA обеспечивает легкость взаимодействия агентов JADE с другими программными, аппаратными или комплексными агентами, поддерживающими этот стандарт. Благодаря этому инструменту, разработчик имеет ряд готовых средств, для управления агентами: создание, удаление, регистрация и миграция между вычислителями агентов; регистрация функций агентов в единой базе; взаимодействие между агентами; отслеживание сообщений внутри MAC; графическая поддержка отладки во время разработки агентов. Основные преимущества, использования платформы JADE:

1. FIPA-совместимая агентная платформа, которая основана на рекомендациях FIPA и включает в себя три обязательных типа системных агентов: сервис управления агентами (AMS), канал связи агентов (ACC) и «Желтые страницы» (DF).
2. Распределенная агентная платформа, которая может использовать один или несколько компьютеров (узлов сети), на каждом из которых должна работать только одна виртуальная JAVA машина.
3. Наличие многопоточной среды исполнения с двухуровневым распределением.

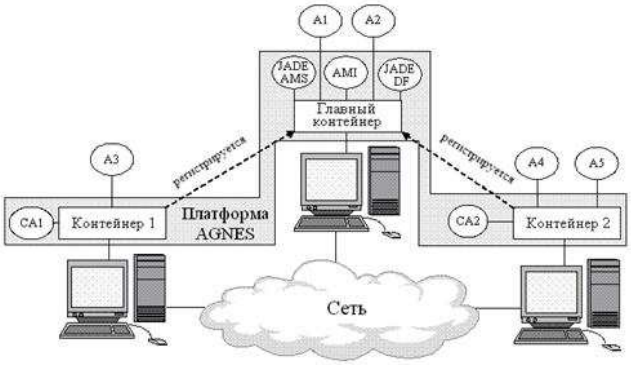

Рис. 1. Платформа AGNES

4. Наличие библиотеки со стандартными протоколами взаимодействия fipa-request и fipa-contract-net.
5. Наличие графических утилит для администрирования.

AGNES использует все эти возможности, а также расширяет мультиагентную систему до системы моделирования.

Среда моделирования AGNES состоит из двух типов агентов: управляющие агенты (УА), которые создают среду моделирования; функциональные агенты (ФА), которые образуют модель, работающую в среде моделирования. Приложение AGNES — это распределенная МАС, называемая платформой. Платформа AGNES состоит из системы контейнеров, распределенных в сети (рис. 1).

Обычно на каждом хосте находится по одному контейнеру (но при необходимости их может быть несколько). Агенты существуют внутри контейнеров. В системе может быть только один главный контейнер, который представляет собой точку начальной загрузки платформы. Главный контейнер создается первым, все созданные позже контейнеры должны быть зарегистрированы в главном контейнере. На главном контейнере обязательно работают управляющие агенты (УА) AMS (Agent Management System) и DF (Directory Facilitator) [5].

Управляющие агенты AGNES

Основные задачи УА: инициализация и запуск модели; сбор и хранение информации о ходе моделирования; синхронизация модельного времени; перераспределение нагрузки между вычислителями, участвующими в моделировании; взаимодействие с пользователем (вывод отчетов и возможность влияния на ход моделирования); обеспечение отказоустойчивости, восстановление модели. При запуске модели все ФА разделяются на виртуальные кластеры, и над каждым таким кластером назначается контролирующий агент (КА), который является разновидностью УА. Основные функции КА: идентификация отказа агентов в среде моделирования; пересылка всех управляющих команд функциональным агентам; хранение информации для восстановления агентов; восстановление модели при сбое. Инициализация модели. Во время запуска AGNES, первой стартует платформа JADE. Затем запускается агент AMI (AGNES Model Initializer) — агент, инициализирующий среду моделирования и запускающий в этой среде подготовленную модель. AMI действует по следующему алгоритму. Инициализация AGNES, запуск необходимых УА: ARM (AGNES resource manager) — агент, наблюдающий за состоянием платформы (отключением, подключением контейнеров). При изменении состояния он оповещает об этом

балансировщиков нагрузки. ALB (AGNES load balancer) — агент, наблюдающий за состоянием группы контейнеров, характеризующимися количеством ФА каждого типа, интенсивностью обмена сообщениями, средней скоростью доставки сообщения. В масштабных моделях, запущенных на больших количествах вычислителей (контейнерах), одновременно работают несколько ALB агентов, отвечающих за разные контейнеры. Обмениваясь информацией о состояниях всех контейнеров в платформе, агенты инициируют процессы миграции ФА между контейнерами, для выравнивания характеристик загруженности контейнеров. ADS (AGNES data storage) — агенты, собирающие всю информацию о характеристиках модели — модельных параметрах ФА. Каждый ФА агент через сервис JADE DF находит ADS агентов и в течение всего времени работы оповещает их о своем модельном состоянии. Инициализация модели. AMI получает конфигурационный файл модели с описанием типов, количеством и параметрами ФА, необходимых для моделирования. При запуске ФА разбиваются на «кластеры» и создается УА — ACA (AGNES controlling agent), отвечающий за этот «кластер ФА». Для уменьшения трафика управляющих команд все команды ФА получает только от своего ACA, а управляющие агенты обмениваются сообщениями между собой напрямую. После инициализации ФА, объединенные в кластеры, «расползаются» по доступным контейнерам и начинается моделирование, по команде от AMI. При необходимости дополнительно могут быть запущены агенты-утилиты с графическим интерфейсом, для взаимодействия AGNES с пользователем. В настоящее время пользователь может: контролировать состояние платформы; видеть количество контейнеров, список агентов на каждом из контейнеров, создавать или удалять агенты, подключать или отключать контейнеры; обмениваться сообщениями с агентами; пользователь может отправлять любые сообщения любому агенту, зная его AID, а так же он может получать ответные сообщения от агента как реакцию на свой запрос; наблюдать структуру сети, если модель можно представить в виде графа, узлами которого являются агенты, а ребрами информационные связи между ним; имеет механизмы глобального управления моделированием — приостановка, возобновление, преждевременное прекращение моделирования, получение промежуточных результатов.

Отказоустойчивость

Чтобы обеспечить высокий уровень отказоустойчивости, AGNES реализует несколько механизмов: отсутствие централизованного хранения данных для восстановления; хранение необходимой информации ведется подобно peer-to-peer сетям, т. е. информация располагается частями на разных агентах среды моделирования, и эта информация хранится с избытком, для гарантии ее восстановления; динамическое изменение хранилищ информации во время работы среды моделирования. То есть основные принципы улучшенной отказоустойчивости среды моделирования - это децентрализация хранилищ и избыточность информации. Рассмотрим более подробно, как происходит сохранение резервных данных и восстановление модели. В AGNES реализована служебная команда PING для проверки работоспособности агента. И все агенты должны корректно обрабатывать этот запрос. Большинство УА запускаются в нескольких экземплярах, чтобы гарантировать работы системе при отказе одного из них. КА следят за остальными УА, чтобы обнаружить отказ. Каждый агент AGNES должен реализовывать два метода SaveBackup() and RestoreBackup(), т. е. уметь сохранить свое состояние и восстановить его из сохранных данных соответственно. Все КА выбирают себе группу ФА, за которыми будет осуществляться контроль некоторое время, т. е. у КА есть список ФА, у которых он контролирует размещение backup-данных. С определенной периодичностью КА обмениваются агентами,

входящими в их кластер. Во время своей жизни ФА получают команду от КА о том, что нужно сделать backup: модельный момент времени, когда сохранить состояние, и список соседей, у кого следует разместить эти данные. В соответствующий момент агент сохраняет свое состояние, запоминает его у себя, для возможности дальнейшего отката, и отправляет его указанным агентам. Когда агент получает backup-данные соседа, он оповещает об этом своего КА. Контролирующий агент запоминает в таблицу, у кого хранятся чьи данные и за какой момент. Контролирующий агент каждый раз пытается хранить данные у разных агентов, чтобы уменьшить зависимость между агентами. Контролирующий агент регулярно опрашивает своих УА и ФА ping командами, чтобы обнаружить отказ. Помимо этого, каждый КА выбирает несколько соседних КА, состояние которых он также проверяет. При обнаружении отказа (не получит ответ на ping запрос) КА оповещает всех о необходимости прекращения моделирования, команда pause. Затем КА совместно определяют последнее стабильное состояние системы, т. е. такое состояние, в котором известна информация обо всех агентах, и местоположение данных о состоянии отказавших агентов в этот модельный момент времени. Информация об отказавших агентах запрашивается у работоспособных агентов и из нее восстанавливаются дубликаты. Затем в среде моделирования командой RestoreFrom инициализируется откат до стабильного состояния, и моделирование продолжается.

Сбор и хранение информации

Внутри AGNES циркулируют два типа сообщений: управляющие команды и информационные сообщения внутри модели. Для моделирования важно собирать и хранить информационные сообщения, т. е. все обмены данными внутри самой модели. Эту задачу выполняют агенты логгеры. Они подписываются на все сообщения определенного типа и получают их копии. В зависимости от специфики модели существует необходимость сбора сообщений определенного типа, для этого у логгера можно настроить фильтр и собирать только значимую информацию. Эта возможность реализована за счет структуры FIPA сообщений и удобных механизмов классификации сообщений. Также при моделировании важно иметь информацию о состоянии ФА. Благодаря сервису «Желтых страниц» JADE УА могут найти всех интересующих агентов модели и опрашивать их, сохраняя у себя нужную информацию.

Балансировка нагрузки

JADE приложения — это распределенные приложения, запускаемые на сети из вычислителей. JADE позволяет динамически менять среду исполнения MAC, т. е. подключаться к уже существующей программе или исключать работающие контейнеры. Для того чтобы обеспечить наилучшую производительность среды моделирования, AGNES следит за этими процессами и при обнаружении изменений в среде исполнения старается перераспределить агентов равномерно по всем доступным ресурсам, т. е. инициирует миграцию агентов из одного контейнера на другой средствами JADE. Так как агенты отличаются по своим функциям и задачам, то AGNES старается перераспределить равномерно агентов всех типов.

Взаимодействие с пользователем

Помимо доступных GUI tools среды JADE, AGNES предоставляет дополнительные средства взаимодействия с пользователем: вывод графа модели (где узлами графа являются ФА модели, а ребра каналы связи между агентами); вывод таблиц логов модели,

данные накопленные логгерами; механизмы изменения модели, добавление и удаление ФА.

Функциональные агенты AGNES

Функциональные агенты моделируют поведение исследуемой модели. AGNES ориентирована на создание моделей систем, которые легко декомпозируются на простые элементы, взаимодействующие друг с другом. Примерами подобных систем могут служить: сенсорные сети, пчелиный улей или дорожное движение. За исключением проблемно-ориентированных задач, каждый агент должен выполнять некоторые функции, которые обеспечивают процесс моделирование (периодическое резервное копирования своих данных, синхронизация, передачи сообщений между агентами и т. д.). Эти функции реализуются следующими методами: [Sleep()] — остановка моделирования; [Wakeup()] — продолжение моделирования после остановки; [Start (array InitParameters)] — инициализация и запуск моделирования; [RestoreFrom(time Moment)] — восстановление состояния агента в заданный момент времени; [CreateBackup(time Moment, agent Receiver)] — создание резервной копии состояния агента и отправка этих данных для хранения указанному агенту; [RestoreBackup(array BackupParameters)] — восстановление состояния агента из резервных данных; [GetStatus()] — получение статуса состояния модели. Следующие методы должны быть реализованы у AGNES агента независимо от модели: [SaveBackup(agent Sender, time BackupMoment, array BackupData)] — сохранение резервных данных другого агента в своем хранилище; [SendBackup(agent Receiver, time BackupMoment, agent BackupAgent)] — отправка резервных данных из своего хранилища указанному агенту; [SetTime(time ModelTime)] — синхронизация модельного времени у агента; [GetAgentTime()] — получить данные о внутреннем времени агента; [SendLog()] — широковещательная рассылка своего состояния всем подписанным на это событие агентам; [Ping()] — команда для проверки работоспособности агента. Преимуществами AGNES являются: малый объем кода имитационных программ; балансировка нагрузки при исполнении, доступность проблемно-ориентированных библиотек, возможность динамического изменения модели в ходе эксперимента.

Результаты моделирования

Рассмотрим примеры реализации отображения алгоритма на архитектуру экзафлопсной ЭВМ с использованием системы AGNES. Первая задача связана с изучением возможности масштабирования распределенного статистического моделирования на большое число вычислительных ядер. Это задачи, требующие моделирования экстремально большого количества независимых реализаций [11]. К числу таких проблем относятся задачи моделирования с использованием прямого статистического моделирования (ПСМ) течений разреженного газа с учетом химических реакций, задачи переноса излучения и теории дисперсных систем. Общая схема вычислений по методу Монте Карло (см. рис. 2):

Шаг 1: Подготовка к моделированию независимых реализаций на группах ядер.
Шаг 2: Моделирование реализаций, вычисление выборочных средних для группы.
Шаг 3: Сбор и осреднение данных.

Имитационное моделирование проводилось с использованием мультиагентной системы AGNES. Для имитации вычислений методов Монте Карло созданы два класса функциональных агентов: DataAgregator: ядро-сборщик, собирает информацию о вычислениях, обрабатывает и агрегирует ее. Возможно иерархическое построение сборщиков, которые

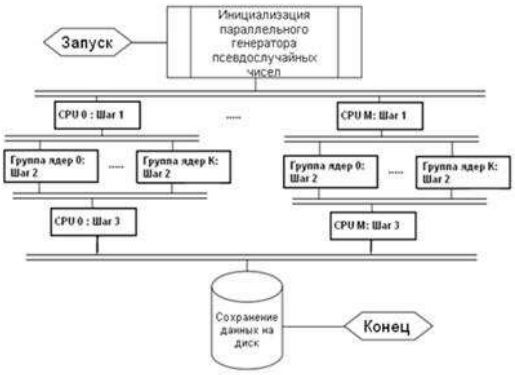

Рис. 2. Схема параллельных вычислений методов Монте Карло

на нижнем уровне обрабатывают данные непосредственно вычислителей, а затем передают их вышестоящему агенту DataAgregator. На вершине этой пирамиды всегда стоит одно главное ядро-сборщик, подготавливающее итоговые данные обо всех вычислениях и сохраняющее их на жесткий диск. MonteCarlo: агент, имитирующий расчет независимых реализаций методов Монте Карло на нескольких вычислительных узлах, группа ядер-вычислителей. Каждый агент проводит независимые вычисления согласно схеме вычислений и взаимодействует только с соответствующим DataAgregator. Основными характеристиками агента являются временные и статистические свойства, оценки которых получены на основе реальных вычислений. В результате работы модели собираются следующие отчеты.

• Набор времен, потраченных на каждую итерацию вычислений каждым агентом. Эти времена позволяют получить статистические характеристики протекающих в модели вычислений, для оценки правдоподобия модели.

• Информация о количестве итераций вычислений, совершенных каждым агентом MonteCarlo. При помощи данной статистики можно, например, отследить, как влияет количество вычислителей на скорость расчетов.

• Информация об интенсивности получения данных агентами DataAgregator от вычислителей либо нижестоящих DataAgregator, в данном случае регистрируется количество полученных за равные промежутки времени пакетов.

Исходные данные для имитационного моделирования получены с использованием библиотеки PARMONC, предназначенной для использования на современных суперкомпьютерах тера- и петафлопсного уровня [11]. Область применения библиотеки: «большие» задачи статистического моделирования в естественных и гуманитарных науках (физика, химия, биология, медицина, экономика и финансы, социология и др.). Библиотека PARMONC установлена на кластерах Сибирского суперкомпьютерного центра (ЦКП ССКЦ СО РАН) и может использоваться на вычислительных системах с аналогичной архитектурой. При этом использование библиотеки не привязано к каким-то определенным компиляторам языков C и FORTRAN или MPI. Инструкции по использованию библиотеки с примерами можно найти по ссылкам [12]. Как известно, теоретическое ускорение при распараллеливании для методов статистического моделирования практически идеальное, что подтверждается численными расчетами при числе вычислительных ядер порядка нескольких тысяч [13]. Тем не менее при числе ядер порядка сотен тысяч или нескольких миллионов вопросы организации счета требуют серьезного исследования, поскольку при этом возникают проблемы с большой загрузкой ядер-сборщиков, которые периодически

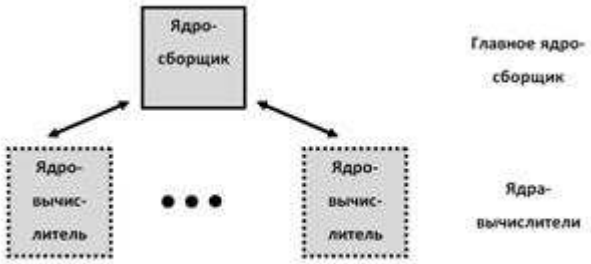

Рис. 3. Вариант организации связей между ядрами: одно главное ядро-сборщик

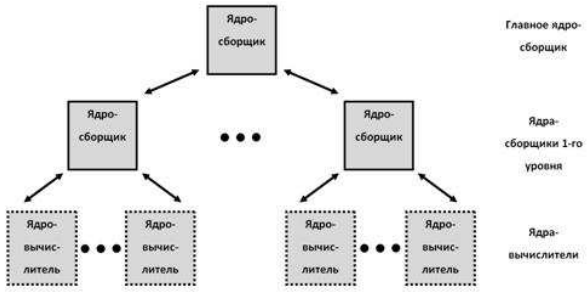

Рис. 4. Вариант организации связей между ядрами: один уровень промежуточных ядер-сборщиков

собирают статистику с ядер-вычислителей. А именно: проведенное имитационное моделирование показало, что при большом числе используемых вычислительных ядер (больше 10 000) реальное ускорение от распараллеливания существенно отличается от теоретического, что связано с большой загрузкой выделенных ядер-сборщиков, которые обрабатывают поступающие пакеты данных с ядер-вычислителей. При этом до 1000 ядер ускорение в модели совпадает с ускорением в реальных расчетах. С целью повышения эффективности распараллеливания исследовались различные варианты организации обмена данными между ядрами. А именно: целесообразно осуществлять периодическую пересылку результатов промежуточного осреднения реализаций, независимо полученных на загруженных ядрах (ядрах-вычислителях), на выделенные ядра (ядра-сборщики), объединенные в многоуровневую структуру. Ядра-сборщики будут периодически получать переданные им данные и осреднять их, передавая затем результаты на ядро (с номером 0), соответствующее вершине многоуровневой структуры (рис. 3 и 4). Будем называть такое ядро главным ядром-сборщиком; в числе его задач — сохранение осредненных данных на диск.

Рассчитанные на главном ядре-сборщике осредненные значения будут соответствовать выборке, полученной совокупно на всех ядрах-вычислителях. Распределенное статистическое моделирование на разных вычислительных ядрах-вычислителях производится в асинхронном режиме. Отправка и получение результатов статистического моделирования также осуществляется в асинхронном режиме [8]. На кластере НКС-30Т Сибирского суперкомпьютерного центра с использованием библиотеки PARMONC был произведен ряд расчетов для общего числа ядер, примерно равного 1000. Реальные затраты машинного времени на независимое моделирование реализаций на ядрах-вычислителях и обмен данными (выборочными средними) с главным ядром-сборщиком были использованы для

калибровки имитационной модели в AGNES. По результатам расчетов был сделан вывод, что требуемый уровень относительной статистической погрешности в 0,1% достигается при объеме выборки равном $L = 240\,000$. Среднее время моделирования одной реализации — примерно 12 с. Для ядер-вычислителей обмен данными с главным ядром-сборщиком происходил после каждой смоделированной на них реализации. Отображение модели на вычислительный кластер выглядит следующим образом: на каждом сервере запускается JVM и отдельный контейнер JADE. На каждом контейнере запускаются агенты имитирующие расчет методов Монте Карло. Каждый агент MonteCarlo содержит внутри себя группу циклических поведений, каждое из которых имитирует расчет независимых реализаций на одном вычислительном узле. Подобное представление позволяет моделировать расчеты по методу Монте Карло на 10^7 вычислительных узлах с использованием 10^4 агентов. При имитационном моделировании расчетов по методу Монте Карло за основу взята MPP-архитектура кластера НКС-30Т. При этом исследовалась величина относительного ускорения от распараллеливания при расчетах на M ядрах, определенная следующим образом:

$$S_L(M) = \frac{T_L(M_{\min})}{T_L(M)},$$

где $T_L(M)$ — машинное время на главном ядре-сборщике, затраченное на моделирование и сохранение выборочных средних для задачи, в которой моделируется L реализаций случайной оценки; $M_{\min}$ — наименьшее число ядер, использованных при расчетах. В первой серии экспериментов рассматривались два варианта организации обмена данными между ядрами-вычислителями и главным ядром-сборщиком:

— без использования промежуточных ядер-сборщиков (рис. 3);
— с использованием одного уровня промежуточных ядер-сборщиков (рис. 4).

В первом варианте ядра-вычислители были поделены на M_1 равных частей ($M_1 = 10$, 20, 100), для каждой из которых данные с ядер-вычислителей всегда отправлялись на «свое» ядро-сборщик. В свою очередь, M_1 ядер-сборщиков отправляли данные на главное ядро-сборщик. Во втором варианте для определенности будем считать, что параметр $M_1 = 0$. На рис. 5 приведена зависимость относительного ускорения $S_L(M)$ от общего числа моделируемых ядер M. Моделирование проводилось до $M = 5 \cdot 10^5$ ядер, но для показательности на рисунке приведены данные до $M = 10^5$. На рисунке ясно видна закономерность: увеличение числа ядер-сборщиков приводит к увеличению относительного ускорения. На начальном участке до 1000 ядер модельные данные хорошо совпадают с фактическими расчетами на гибридном кластере НСК-30+GPU, однако с увеличением количества ядер и в зависимости от количества ядер-сборщиков происходит отклонение от теоретической кривой. Следует отметить, что чем больше ядер-сборщиков, тем ближе модельная кривая к теоретической кривой. Подробнее эти вычислительные эксперименты описаны в работе [10]. В этой связи интересно исследовать вопрос об оптимальном (в смысле максимального значения относительного ускорения) числе ядер-сборщиков при фиксированном общем числе моделируемых ядер. Во второй серии экспериментов при фиксированном числе моделируемых ядер $M = 10^6$ варьировалось число ядер-сборщиков M_1, а также соответствующее число ядер-вычислителей в каждой группе, связанной со «своим» ядром-сборщиком. На рис. 6 приведен график зависимости относительного ускорения S_L от числа ядер-сборщиков M_1. Из рисунка видно, что максимальная величина относительного ускорения достигается при M_1, приближенно равном 1000. Это говорит о том, что при меньшем значении M_1 ядра-сборщики перегружены обработкой данных,

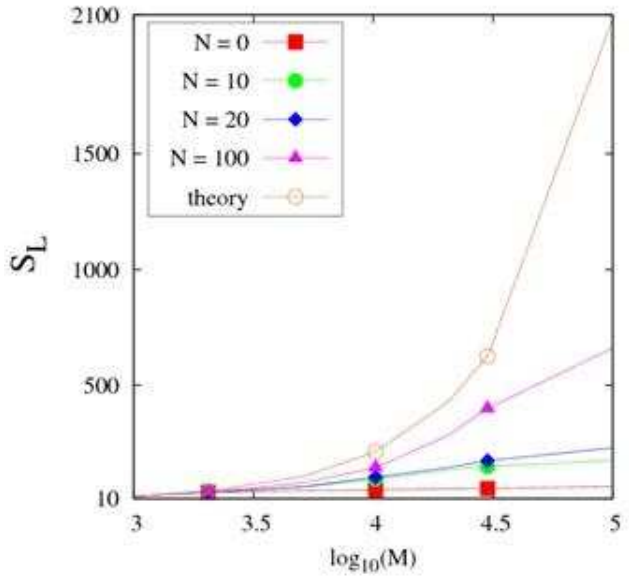

Рис. 5. Зависимость относительного ускорения S_L от общего числа моделируемых ядер M при разном числе ядер-сборщиков M_1 (горизонтальная ось — в логарифмическом масштабе)

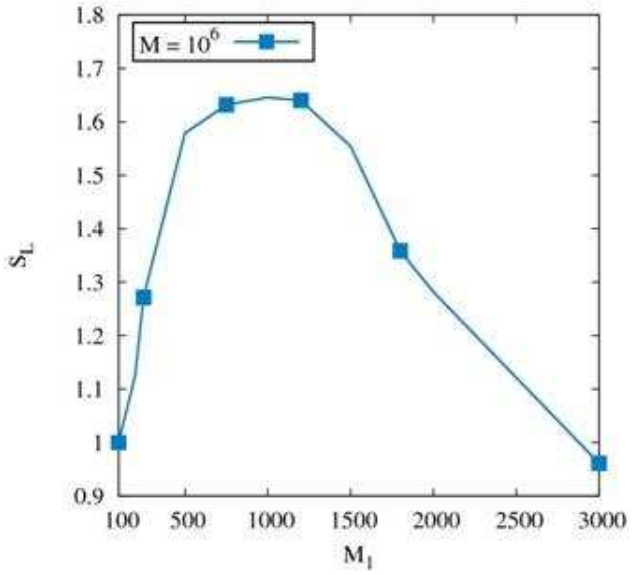

Рис. 6. Зависимость относительного ускорения S_L от числа ядер-сборщиков M_1 при общем числе моделируемых ядер $M = 10^6$

поступающих от ядер-вычислителей, а при большем числе – перегружено главное ядро-сборщик, занятое обработкой поступающих данных от ядер-сборщиков.

Аналогичные расчеты были проведены для другого класса алгоритмов, связанного с сеточными методами. Решалась задача численного моделирования распространения сейсмических полей в 3D изотропной неоднородной упругой среде [13]. В этом случае предполагаем, что архитектура гипотетического кластера является гибридной, вычислительные узлы состоят из нескольких CPU и GPU. Под масштабируемостью понимаем следующее: время счета алгоритма меняется незначительно при следующих допущениях: размер 3D модели увеличивается пропорционально количеству вычислительных узлов; каждый вы-

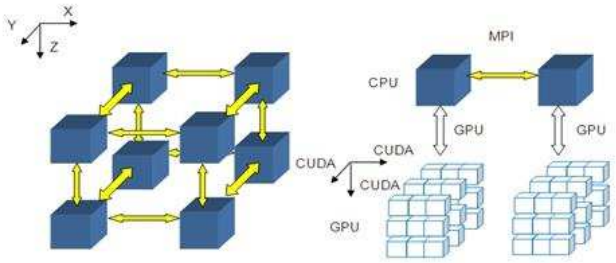

Рис. 7. Схема организации параллельных вычислений на гибридном кластере

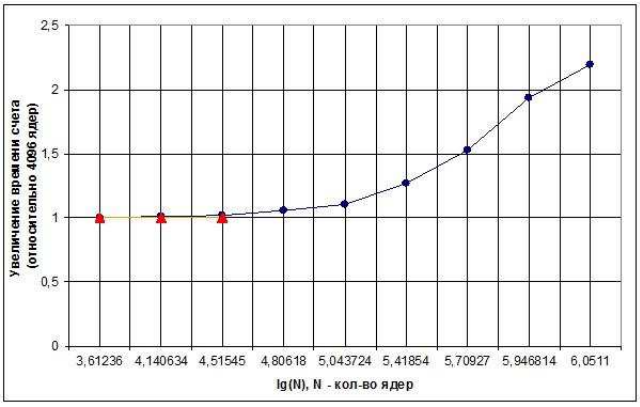

Рис. 8. Изменение времени расчета алгоритма численного моделирования в зависимости от числа вычислительных ядер (горизонтальная ось — в логарифмическом масштабе), треугольник — при проведении расчетов на суперкомпьютере, овал — при исследовании алгоритма с помощью имитационного моделирования

числительный узел совершает одно и то же количество итераций для своей подобласти. Разработана программа на основе масштабируемого параллельного алгоритма численного моделирования при использовании комбинации CUDA и MPI. Для проведения расчетов различных 3D моделей была рассмотрена следующая организация параллельного алгоритма и программы: 3D область моделирования разделяется на трехмерные подобласти по направлениям координатных осей; каждая из подобластей рассчитывается независимо на выделенном GPU, а обмены данными, между соседними GPU проводятся посредством CPU с использованием MPI (рис. 7). При этом вычисления для подобластей на GPU производятся посредством CUDA в 3D.

Для имитации сеточных методов реализован класс функциональных агентов Grid — узел-вычислитель, имитирующий расчет сеточных методов на одном вычислителе. Моделируются вычисления, когда область исследования делится вдоль осей на 3D подобласти, и полученные области загружаются на вычислители. Таким образом, получается, что у каждого вычислителя есть пересечение по данным максимум с двумя вычислителями по каждой из осей. Общие результаты изменения времени счета в зависимости от количества доступных ядер GPU (при пропорциональном увеличении размера 3D модели) в логарифмическом масштабе приведены на рис. 8. Показано хорошее соответствие экспериментальных и модельных результатов на начальном участке кривой (до 32 768 ядер).

Время работы алгоритма существенно увеличивается при увеличении количества ядер (удалось получить результаты для 1 124 864 ядер). Это объясняется характерными осо-

бенностями данного алгоритма — увеличением числа обменов каждого узла с соседями на каждой итерации, таким образом, число обменов в системе растет более стремительно. Уже на 500 000 ядер время выполнения алгоритма увеличилось в 1,5 раза, а для 1 000 000 ядер почти в 2,1 раза. Видно, что эффективное использование этого алгоритма на гибридных суперкомпьютерах с количеством ядер около 1 млн требует его модификации. Проведенные численные эксперименты по имитационному моделированию показали возможность масштабирования алгоритмов на большое число (сотни тысяч и даже миллионы) вычислительных ядер предполагаемого экзафлопсного суперкомпьютера, а также возможность исследования поведения алгоритмов при таком большом масштабировании.

Заключение

В работе представлена мультиагентная система имитационного моделирования AGNES. Система может быть использована для исследования масштабируемости различных алгоритмов для суперкомпьютеров с учетом особенностей архитектуры суперкомпьютера. Таким образом, с помощью AGNES можно проводить исследования связанные с разработкой алгоритмов для суперкомпьютеров экзафлопсного класса с учетом различных предполагаемых архитектур данных суперкомпьютеров. Приведенные тесты имитационного моделирования хорошо соотносятся с реальными данными, полученными при запуске программ на кластерах ЦКП ССКЦ СО РАН.

Литература

[1] *Sivasubramaniam A., Singla A., Ramachandran U., Venkateswaran H.* A simulation-based scalability study of parallel systems // *J. Parallel Distributed Computing*, 1994. Vol. 22, no. 3. P. 411–426.

[2] *Racherla G., Killian S., Fife L., Lehmann M., Parekh R.* Parsit — a parallel algorithm reconfiguration simulation tool // *Conference (International) on High Performance Computing Proceedings*, 1995.

[3] *D'Souza R. M., Lysenko M., Marino S., Kirschner D.* Data parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units // *2009 Spring Simulation Multiconference (SpringSim'09) Proceedings*. San Diego, CA, USA: Society for Computer Simulation International, 2009.

[4] *Goodrich M. T.* Simulating parallel algorithms in the MapReduce framework with applications to parallel computational geometry CoRR. URL: `http://dblp.uni-trier.de/db/journals/corr/corr1004.html\#abs-1004-4708`.

[5] *Avetisyan A. I., Gaysaryan S. S., Ivannikov V. P., Padaryan V. A.* Productivity prediction of MPI programs based on models // *Automation and remote control*, Vol. 68, no. 5. P. 750–759 .

[6] *Ivannikov V., Avetisyan A., Padaryan V.* Evaluation of dynamic characteristics of a parallel program on a model // *Programming*, 2006. Vol. 4. P. 21–37. (In Russian.)

[7] *Глинский Б. М., Родионов А. С., Марченко М. А., Подкорытов Д. И., Винс Д. В.* Агентно-ориентированный подход к имитационному моделированию суперЭВМ экзафлопсной производительности в приложении к распределенному статистическому моделированию // *Вестник ЮУрГУ*, 2012. Т. 18(277), № 12. С. 94–99.

[8] *Podkorytov D., Rodionov A., Sokolova O., Yurgenson A.* Using agent-oriented simulation system AGNES for evaluation of sensor networks // *LNCS*. Heidelberg: Springer, 2010. Vol. 6235. P. 247–250.

[9] *Podkorytov D., Rodionov A., Choo H.* Agent-based simulation system AGNES for networks modeling: Review and researching // *6th Conference (International) on Ubiquitous Information Management and Communication (ACM ICUIMC 2012) Proceedings*, 2012. Paper 115. 4 p.

[10] *Glinsky B., Rodionov A., Marchenko M., Podkorytov D., Weins D.* Scaling the distributed stochastic simulation to exaflop supercomputers // *2012 IEEE 9th Conference (International) on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th Conference (International) on High Performance Computing and Communication Proceedings*, 2012. IEEE. P. 1131–1136.

[11] *Марченко М. А., Михайлов Г. А.* Распределенные вычисления по методу Монте-Карло // *Автоматика и телемеханика*, 2007. № 5. С. 157–170.

[12] *Marchenko M. A.* PARMONC — A software library for massively parallel stochastic simulation // *LNCS*, 2011. Vol. 6873. P. 302–315.

[13] *Глинский Б. М., Караваев Д. А., Ковалевский В. В., Мартынов В. Н.* Численное моделирование и экспериментальные исследования грязевого вулкана «Гора Карабетова» вибросейсмическими методами // *Вычислительные методы и программирование*, 2010. Т. 11, № 1. С. 99–108.

References

[1] *Sivasubramaniam A., Singla A., Ramachandran U., Venkateswaran H.* 1994. A simulation-based scalability study of parallel systems. *J. Parallel Distributed Computing* 22(3):411–426.

[2] *Racherla G., Killian S., Fife L., Lehmann M., Parekh R.* 1995. Parsit — a parallel algorithm reconfiguration simulation tool. *International Conference on High Performance Computing Proceedings*.

[3] *D'Souza R. M., Lysenko M., Marino S., Kirschner D.* 2009. Data parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units. *2009 Spring Simulation Multiconference (SpringSim'09) Proceedings*. San Diego, CA, USA: Society for Computer Simulation International.

[4] *Goodrich M. T.* Simulating parallel algorithms in the MapReduce framework with applications to parallel computational geometry CoRR. Available at: `http://dblp.uni-trier.de/db/journals/corr/corr1004.html\#abs-1004-4708`.

[5] *Avetisyan A. I., Gaysaryan S. S., Ivannikov V. P., Padaryan V. A.* Productivity prediction of MPI programs based on models *Automation Remote Control* 68(5):750–759 .

[6] *Ivannikov V., Avetisyan A., Padaryan V.* 2006. Evaluation of dynamic characteristics of a parallel program on a model. *Programming* 4:21–37. (In Russian.)

[7] *Glinskiy B. M., Rodionov A. S., Marchenko M. A., Podkorytov D. I., Vins D. V.* 2012. Agent oriented approach to imitation of distributed statistical modeling in application of exaflops supercomputer. *Vestnik YURGU* 12(18):94–99.

[8] *Podkorytov D., Rodionov A., Sokolova O., Yurgenson A.* 2010. Using agent-oriented simulation system AGNES for evaluation of sensor networks. *LNCS*. Heidelberg: Springer. 6235:247–250.

[9] *Podkorytov D., Rodionov A., Choo H.* 2012 Agent-based simulation system AGNES for networks modeling: review and researching // *6th Conference (International) on Ubiquitous Information Management and Communication (ACM ICUIMC 2012) Proceedings*. Paper 115. 4 p.

[10] *Glinsky B., Rodionov A., Marchenko M., Podkorytov D., Weins D.* 2012. Scaling the distributed stochastic simulation to Exaflop supercomputers. *2012 IEEE 9th Conference (International) on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th Conference (International) on High Performance Computing and Communication Proceedings*. IEEE. 1131–1136.

[11] *Marchenko M. A., Mikhailov G. A.* 2007. Distributed computing of Monte Carlo method. *Automation Remote Control* 5:157–170.

[12] *Marchenko M. A.* 2011. PARMONC — A software library for massively parallel stochastic simulation. *LNCS* 6873:302–315.

[13] *Glinskiy B. M., Karavaev D. A., Kovalevskiy V. V., Martynov V. N.* 2010. Numerical modeling and experimental studies of "Karabetova Mount" salse by vibroseismical methods. *Computational Methods Programming* 11(1):99–108.