ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Московский институт электроники и математики
Департамент прикладной математики

Зиганурова Лилия Фаилевна

АНАЛИЗ АЛГОРИТМОВ ПАРАЛЛЕЛЬНОГО МОДЕЛИРОВАНИЯ ДИСКРЕТНЫХ СОБЫТИЙ

Выпускная квалификационная работа - МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
по направлению 01.04.04 Прикладная математика
студента группы МА-21 (образовательная программа «Системы управления и обработки информации в инженерии»)

Рецензент
кан. хим. наук
_____
С. А. Крашаков

Научный руководитель
д-р физ.-мат. наук, проф.
_____
Л. Н. Щур

Москва 2015

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данная выпускная квалификационная работа посвящена анализу масштабируемости алгоритмов параллельного моделирования дискретных событий.

Моделирование – это метод решения задач, при использовании которого исследуемая система заменяется более простым объектом, описывающим реальную систему. Моделирование применяется в случаях, когда проведение экспериментов над реальной системой невозможно или нецелесообразно, например, по причине хрупкости или дороговизны создания прототипа, либо из-за длительности проведения эксперимента в реальном масштабе времени.

Имитационная модель – это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных.

По причине сложности моделируемых объектов наряду с последовательными системами имитационного моделирования появились параллельные системы моделирования. Это означает, что во время проведения имитационного эксперимента используется одновременно ресурсы нескольких вычислительных узлов. Моделируемая система разбивается на подсистемы, которые параллельно моделируются на исполняющих элементах.

Существуют такие системы, изменение состояния которых происходит в дискретные моменты времени. Эти изменения называются событиями. Примером такой системы может являться глобальная сеть Интернет, а события в ней – это прием и передача пакетов. Для моделирования таких систем применяется специальный метод параллельного моделирования дискретных событий (ПМДС).

Суть ПСДС состоит в том, что система моделируется как набор логических процессов (ЛП), взаимодействия между которыми осуществляется посредством передачи сообщений об изменении состояния ЛП со штампами времени, без прямого доступа к разделяемой памяти.

Таким образом, каждый логический процесс характеризуется своим

локальным виртуальным временем. Набор таких локальных виртуальных времен образуют так называемый профиль времен, который эволюционирует в системе отчета глобального времени по мере выполнения параллельного моделирования (17).

Использование ресурсов нескольких вычислительных узлов во время имитационного эксперимента сокращает время его проведения, однако алгоритм управления объектами, распределенными по вычислительным узлам, является гораздо более сложным, чем алгоритм продвижения времени в последовательном моделировании. Каждый процесс должен выполнять события в порядке неубывания своих временных меток для того, чтобы не нарушались причинно-следственные связи. Для обеспечения соблюдения причинности используются алгоритмы синхронизации. Традиционно известны два подхода к реализации алгоритмов объектов моделирования: консервативный и оптимистический. Менее известен метод Freeze and Shift (32).

Принципиальная задача консервативного алгоритма – определить время, когда обработка очередного события из списка необработанных событий является «безопасным». Событие является безопасным, если можно гарантировать, что процесс в дальнейшем не получит от других процессов сообщение с меньшей временной меткой. Различают также различные виды консервативных схем синхронизации. При оптимистическом алгоритме допускается нарушение причинно-следственных связей, при этом существует специальный механизм, позволяющий устранить ошибки.

Необходимость соблюдать причинно-следственные связи в процессе моделирования может приводить к увеличению времени моделирования. С увеличением количества ЛП все больше времени так же затрачивается на обмен сообщениями между ними. Изучение эффективности и масштабируемости при различных алгоритмах синхронизации является ключевым аспектом при изучении ПМДС. Алгоритм является масштабируемым, если профиль локальных времен растет с ненулевой скоростью (иными словами, в системе нет мертвых состояний), а также, если ширина профиля является постоянной при увеличении количества логических процессов. Разные схемы синхронизации имеют различные значения скорости роста профиля локальных времен и изменения ширины профиля.

Постановка задачи

В выпускной квалификационной работе ставится задача анализа и изучения параллельных систем моделирования при локальных консервативном и оптимистическом алгоритмах синхронизации.

Методы исследования

Для того чтобы изучить масштабируемость параллельных систем моделирования, применяется метод «моделирование модели». Считается, что наступление событий в системе распределено по пуассоновскому закону.

Моделируется рост горизонта времени при консервативном и оптимистическом алгоритмах синхронизации, а затем исследуется зависимость среднего значения высоты профиля локальных времен, ее дисперсии (т.е. ширины профиля) и скорости эволюции профиля от времени.

G. Korniss с соавторами (37) с обнаружили сходство процесса роста профиля локальных времен при консервативном алгоритме синхронизации с процессом роста молекулярной поверхности. Они показали, что фактически, процесс роста профиля времен относится к универсальному классу, к которому также относится дифференциальное уравнение в частных производных Кардара-Паризи-Жанга (20). Это наблюдение позволяет применять для изучения ПМДС также методы и средства статистической механики.

ГЛАВА 1

Моделирование. Виды моделирования.

В общем случае понятие моделирование можно сформулировать следующим образом. Моделирование – это метод исследования, при котором изучаемая система заменяется моделью с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели путем проведения эксперимента с моделью. В настоящее время моделирование широко используется, чтобы проанализировать поведение крупных систем без непосредственной их реализации, а зачастую оно является единственным методом исследования сложных динамических систем. Примерами таких систем могут служить система воздушно-транспортного сообщения, телекоммуникационные системы, финансовые рынки и проч. Реализация физических прототипов систем имеет существенные недостатки по сравнению с компьютерным моделированием, одним из которых является дороговизна изготовления физических моделей. Именно поэтому метод компьютерного моделирования широко используется в таких областях, как инженерия, информатика, экономика и военная отрасль.

Моделирование может применяться не только для получения статистических данных о моделируемой системе, но и для создания так называемой «виртуальной среды». Такое моделирование часто используется для тренировки пилотов. Программа симулирует процесс полета, а пилот может использовать встроенное оборудование для управления виртуальным полетом.

Более узкое определение моделирования было дано R.M. Fujimoto в книге «Parallel Discrete Event Simulation» (12): «моделирование - это система, которая представляет или эмулирует поведение другой системы с течением времени. В компьютерном моделировании моделирующей системой выступает компьютерная программа, а моделируемая система называется физической системой. Физическая система может быть как настоящей реализованной системой, так и гипотетической».

Согласно (12), физическую систему можно описать набором состояний, которые сменяются с течением времени. Например, количество самолетов, ожидающих посадку в аэропорту можно считать состоянием системы, так как оно будет меняться в зависимости от времени и характери-

зовать состояние аэропорта на конкретное время. Таким образом, модель должна отражать:

1. Состояние физической системы на конкретный момент времени;
2. Характер изменения модели при эволюции физической системы;
3. Некоторое представление времени.

В компьютерном моделировании состояния физической системы характеризуются набором переменных, изменение которых называется событием и отображает изменение состояний физической системы. Выполнение моделирования в данном случае - это обработка череды событий, которые происходят в определенное время. Время в физической системе представлятся некоторой абстракцией, называемой временем моделирования (simulation time).

Рассмотрим более подробно одно из важнейших понятий моделирования - время. Различают три типа времени:

1. Физическое время (physical time). Физическое время представляет собой время в физической системе.
2. Время моделирования (simulation time). Время моделирования это абстракция, представляющая время физической системы в модели.
3. Глобальное время (wallclock time). Глобальное время относится ко времени выполнения программы моделирования, то есть время, установленное в операционной системе.

Продемонстрировать разницу этих понятий можно на примере моделирования транспортной системы в Сочи в 2014 году во время зимних Олимпийских игр. Физическое время данной системы - с 7 по 23 февраля 2014 года. Время моделирование может быть представлено как переменная типа double, где единица - это один день. В данном случае время моделирования будет изменяться во время выполнения программы от 0.00 до 17.00. Если программа моделирования была запущена 20 декабря 2013 года в 12:00 и выполнялась в течение трех часов, то глобальное время моделирования принимало значения 12:00 - 15:00 того дня.

1.1 Непрерывное и дискретное моделирование

По типу механизма протекания времени (time flow mechanism) моделирование можно классифицировать на непрерывное и дискретное.

В непрерывном моделировании состояние системы изменяется непрерывно с течением времени. Система представляется как функция от времени, а поведение такой системы обычно описывается набором дифференциальных уравнений. Типичными примерами непрерывного моделирования могут служить моделирование погодных условий, потоков воздуха вокруг самолета, напряжения в сети электрического тока.

В дискретном моделирование события происходят в фиксированные дискретные моменты времени. В программе моделирования определяется набор переменных и правила изменения этих переменных в течение времени моделирования.

В некоторых случаях непрерывную модель можно заменить дискретной, зафиксировав время начала и окончания события.

1.2 Последовательное и параллельное моделирование

Как было сказано ранее, моделирование сокращает временные и денежные расходы. Тем не менее проведение компьютерного моделирование также требует большого количества времени. «Во-первых, процесс построения дизайна и разработки модели является трудоемким и требует глубоких знаний в области моделирования. Во-вторых, само выполнение программы моделирования может занимать немало времени. В большинстве в моделях используются большое количество стохастических параметров, поэтому для получения как можно более точных статистических результатов, требуется многократное цикличное выполнение программы моделирования.

Сокращение времени моделирования может быть достигнуть разными способами. Один из них - увеличение количества вычислительных ресурсов, в частности, произведение вычислений на параллельных компьютерах» (9).

Параллельное моделирование - это технология, позволяющая выполнение программы моделирования в параллельной компьютерной системе, а именно системе, состоящей из множества компьютеров, соединенных между собой.

Преимущества моделирования, выполненного на параллельном (распределенном) компьютере очевидны:

1. Уменьшение времени моделирования.
2. Возможность географического распределения.
3. Использование неоднородных процессорных элементов.
4. Повышение надежности системы.

Более подробно об каждом из этих пунктов написано в книге «PMDS» (12).

Уровни параллелизма/распределения

Согласно (9) программу моделирования можно выполнять в параллель на разных уровнях, начиная с уровня всей программы, заканчивая уровнем конкретных событий. Рассмотрим более подробно каждый из уровней.

1. Уровень программы

   Наиболее очевидный способ ускорить выполнение нескольких экспериментов моделирования - это запуск независимых экземпляров программ моделирования, возможно, с различными входными параметрами, на отдельных процессорах. Поскольку координация между процессорами в данном случае не требуется, можно ожидать высокую эффективность данного метода. Еще одно преимущество применения параллелизма на уровне приложения - это отсутствие необходимости создавать специальный параллельный код, а также хорошая масштабируемость. Тем не менее использование данного метода на распределенном компьютере может быть ограничено памятью одного процессорного узла.
2. Уровень подпрограммы

   Системы, в которых наблюдается зависимость между каждой итерацией программы моделирования, не могут быть распределены на вышестоящем уровне. Например, если входные параметры экземпля-

ра программы i зависят от выходных параметров экземпляра $i-1$, то выполнить эти два экземпляра программы одновременно на разных процессорных элементах невозможно. В этом случае подойдет ускорение работы программы на уровне подпрограммы. Генерация случайных чисел, обработка событий, обновление состояние системы, сбор статистических данных также могут быть эффективно распараллелены на уровне подпрограммы.

3. Уровень компонентов

   Модель разбивается на компоненты или подмодели так, чтобы отражать присущую физической модели параллельность. Например, рассмотрим такую сеть массового обслуживания, как документооборот в организации. Каждый офис или сотрудник организации могут быть смоделированы как различные очереди. Обработка документов так называемым агентом (сотрудником или офисом) может быть представлена работой процессора, а передача документа другому агенту в физической системе - передачей сообщения от одного процессора к другому.

4. Уровень событий с централизированным списком событий

   Это более низкий уровень параллелизма, где обработка параллельных событий распределяется главным процессором (master processor) по другим вычислительным узлам (slave processors). Как правило, события при таком алгоритме являются одновременными. Главный процессор в данном случае согласовывает работу остальных процессоров, предупреждая возникновение ошибок причинности в результате наложений обрабатываемых событий (27).

   Такой вид параллелизма подходит для процессоров с разделяемой памятью. Список событий может быть реализован как структура данных, разделяемая всеми процессорами.

5. Уровень событий с децентрализированным списком событий

   При реализации данного типа параллелизма события, происходящие в разные моменты времени, могут быть распределены по вычислительным узлам в упорядоченном, либо случайном виде. Такая схема требует специальный протокол локальной синхронизации, который увеличивает «накладные расходы» на обмен специальными сообще-

ниями между процессорами. Протоколы синхронизации являются основным объектом изучения параллельного и распределенного моделирования в связи с распространением широких параллельных и распределенных платформ.

1.3 Моделирование на основе событий и моделирование с постоянным шагом

Существует два основных типа дискретного моделирования: так называемые моделирование с постоянным шагом и моделирование на основе событий (time-stepped и event-driven соответственно).

В моделировании с постоянным шагом время разбивается на последовательность временных отрезков равной длины, каждый из которых также называется шагом Монте-Карло (Monte-Carlo step, MCS). Обновление всех переменных состояния системы происходит каждый MCS. При таком механизме возможен лишь переход от одного значения дискретного времени к ближайшему следующему, при этом изменяются абсолютно все переменные состояния, даже если некоторые из них не нуждаются в обновлении.

«Действия, происходящие в ходе исполнения программы в одно и то же время, считаются не зависящими друг от друга. Это важная особенность, благодаря которой программу моделирования можно запускать параллельно на нескольких компьютерах. В данной парадигме, если два события имеют причинно-следственную связь, действия должны происходить в разные временные шаги. Таким образом, размер одного временного шага должен выбираться с учетом конкретной физической системы, поскольку от него зависит точность конечных вычислений» (12).

От выбора длины шага зависит также длительность всего процесса моделирования: чем короче шаг, тем дольше выполняется программа. Очевидно, что для систем, в которых события происходят нерегулярно, моделирование с постоянным шагом является неэффективным, несмотря на другие преимущества этого вида моделирования. Для таких систем больше подойдет алгоритм моделирования на основе событий.

Вместо того, чтобы обновлять переменные состояния системы каждый шаг времени, гораздо эффективнее производить обновление только в

те моменты, когда происходят реальные изменения в физической системе. Эта идея является ключевой при моделировании дискретных событий.

Например, рассмотрим моделирование полета от пункта А до пунтка Б с временным шагом 10 минут. Позиция самолета будет обновляться каждые 10 минут в течение всего времени моделирования. С точки зрения ресурсозатрат, гораздо эффективнее вычислить полное время полета, а затем обновить позицию самолета только в тот момент, когда он достигнет пункта Б. Тогда момент прибытия самолета в пункт Б будет смоделировано как событие. В событийном моделировании обновление переменных происходит только в результате каких-либо событий, при этом предполагается, что точная информация о нахождении самолета в процессе полета из пункта А в пункт Б необязательна, а в случае, если она все же понадобится, ее можно будет вычислить, имея такие данные, как скорость и направление движения самолета.

Событие - это абстрактное понятие, отражающее мгновенное действие в физической системе. Каждое событие имеет временную метку, обозначающую время происхождения этого события. При происхождении события происходит обновление одной или нескольких переменных состояния системы.

«При выполнении программы моделирования берется первое событие из списка событий (event list) (т.е. событие с наименьшей временной меткой), моделируется выполнение этого события путем изменения переменных состояния системы, и планируется новое событие (т.е. создается новый элемент в списке событий). Процесс повторяется до тех пор, как не произойдет какое-либо специальное предопределенное событие, означающее конец моделирования, либо пока не в списке событий не останется элементов» (9).

1.4 Моделирование с синхронной и асинхронной динамикой

Физическая система рассматривается как набор физических процессов, взаимодействующих в различные моменты времени моделирования. В свою очередь, модель такой системы строится как набор логических процессов $ЛП_0$, $ЛП_1$, ... , где каждый логический процесс соответсвует фи-

зическому. Взаимодействие между физическими процессами моделируется как обмен сообщений со штампами времени между логическими процессами.

Как говорилось ранее, программа моделирования оперирует со следующими структурами данных:

1. Переменные состояния системы (state variables);
2. Список событий, которые еще не произошли (event list);
3. Глобальное время (global time).

Глобальное время может быть задано явно как централизованная структура данных, так и неявно - как процедура выполнения программы с постоянным временным шагом. Ключевая характеристика синхронной схемы заключается в том, что все ЛП (в каждый момент глобального времени) имеют одинаковое виртуальное время. Этого ограничения нет в асинхронном моделировании: каждый ЛП может иметь свое локальное виртуальное время, обычно отличное от остальных.

Синхронизация с глобальным временем может осуществляться различными способами. В (21; 36) предлагается выделение одного процессора для контроля глобального времени. На случай отсутствия событий длительный промежуток времени для каждого ЛП определено время, в которое произойдет следующее взаимодействие с другим ЛП. Как только минимальная временная метка следующего возможного внешнего события достигнута, глобальное время увеличивает на определенную величину $\Delta(S)$, зависящую от конкретного состояния системы S. Еще один вариант реализации глобального времени, предлагаемый J. K. Peacock, J. W. Wong и E.G. Manning (29), это использование иерархической структуры организации ЛП. В такой структуре реализуется операция оповещения корневого ЛП о минимальном времени следующего событий, а распределение этой информации вниз по дереву к другим ЛП.

Помимо использования глобальной синхронизации в моделировании с постоянным шагом, ЛП могут быть синхронизированы с глобальным временем и при моделировании на основе событий. Хотя в процессе моделирование глобальное время увеличивается только до времени ближайшего следующего события, ЛП могут выполнять обработку только тех событий, которые лежат в определенном интервале времени. Такой интервал назы-

вают по-разному. Так, Lubachevsky был предложен термин ограниченной задержки (a bounded lag) (22), а L.M. Sokol (33) назвал это же явление движущимся временным окном (a moving time window).

Синхронизация с глобальным временем упрощает реализацию корректного моделирования, так как здесь не возникает проблем, типичных для моделирования систем с асинхронной динамикой, таких, как возникновения в процессе моделирования мертвых состояний, перегруз трафика сообщениями, а также не возникает проблем с памятью. Однако, с другой стороны, такой механизм бывает неэффективен с точки зрения времени моделирования.

События в асинхронной системе происходят с различными интервалами времени. Синхронизация в таких системах происходит за счет взаимодействия процессорных элементов между собой. Каждое событие имеет свой штамп времени. В процессе моделирования события обрабатываются в порядке неубывания их временных меток. Это позволяет соблюсти причинно-следственные связи между событиями. Так, например, если сначала обработать событие E_x, имеющее больший штамп времени, чем событие E_{min}, то это может привести к изменению переменных состояния системы, используемых в событии E_{min}, что вызывает так называемую ошибку причинности (causality errors).

Как было показано в (24) и (13), ошибки причинности можно избежать, если каждый ЛП будет обрабатывать события в порядке неубывания их временных меток. Однако, если существует два независимых ЛП, то несоблюдение этого правила для этих процессов не приведет к нарушению причинности. В то же время даже при соблюдении этого правила существуют ситуации, когда ошибки все же могут возникнуть. Например, рассмотрим два события E_1 логического процесса 0 и E_2 процесса 1 со штампами времени 10 и 20 соответственно. Если E_1 произведет еще одно событие E_3 со штампом времени меньше, чем 20, то событие E_3 может повлиять на событие E_2 (12). Таким образом, ПМДС является сложнее последовательного моделирования и сильно зависит от моделируемых данных.

Существует множество различных алгоритмов синхронизации ПМДС, которые можно разделить на две основные категории: консерва-

тивные и оптимистические. Принципиальная задача консервативного алгоритма – определить время, когда обработка очередного события из списка необработанных событий является «безопасным». Событие является безопасным, если можно гарантировать, что процесс в дальнейшем не получит от других процессов сообщение с меньшей временной меткой. Различают также различные виды консервативных схем синхронизации. При оптимистическом алгоритме используется механизм обнаружения и исправления ошибок. Алгоритм позволяет обрабатывать события, не обращая внимания на их временные метки, а при возникновении ошибок причинности запускается специальный протокол исправления ошибок, когда система «откатывается» назад по времени и заново обрабатывает события.

Сравнение эффективности синхронной и асинхронной схемы

Сравнение производительности синхронной и асинхронной схемы было показано в (8). В анализе времени моделирования с использованием N логических процессов продолжительностью одного временного шага считалась случайная величина, распределенная по экспоненциальному закону:

$$T_{step,i} \sim exp(\lambda) E[T_{step,i}] = \frac{1}{\lambda} \tag{1.1}$$

Следовательно, ожидаемое время синхронного моделирования $E[T^{sync}]$ для k шагов по времени выглядит следующим образом:

$$E[T^{sync}] = kE[max_{i=1..N}(T_{step,i})] = k\frac{1}{\lambda}\sum_{i=1}^{N}\frac{1}{i} \leq \frac{k}{\lambda}log(N) \tag{1.2}$$

Теперь избавимся от ограничений для синхронного моделирования, и получим ожидаемое время асинхронного моделирования $E[T^{async}]$:

$$E[T^{async}] = E[max_{i=1..N}(kT_{step,i})] = k\frac{1}{\lambda}\sum_{i=1}^{N} frac1i > \frac{k}{\lambda} \tag{1.3}$$

Таким образом, мы имеем:

$$\lim_{k\to\infty,N\to\infty} = \frac{E[T^{sync}]}{E[T^{async}]} \approx log(N) \tag{1.4}$$

Это означает, что при большом количестве ЛП асинхронное моделирование может быть быстрее синхронного в $log(N)$ раз. Ясно, что такие результаты получены с учетом допущения, что события происходят с интервалами, распределенными по экспоненциальному закону. Для равномерного распределения мы получим:

$$\lim_{k\to\infty,N\to\infty} = \frac{E[T^{sync}]}{E[T^{async}]} \approx 2 \tag{1.5}$$

1.5 Заключение к Главе 1

Итак, в данной главе был исследован такой метод изучения систем, как компьютерное моделирование. Были рассмотрены основные классификации моделирования с акцентом на параллельное моделирование с асинхронной динамикой. В дальнейшем речь пойдет о локальном консервативном и оптимистическом алгоритме синхронизации параллельного моделирования дискретных событий (ПМДС).

ГЛАВА 2

Параллельное моделирование дискретных событий

Как было сказано ранее, параллельное моделирование дискретных событий (ПМДС) - это подкласс параллельного и распределенного моделирования, в котором изменение компонентов системы происходит мгновенно от одного состояния к другому. Такие изменения называются дискретными событиями. Основная задача ПМДС - проведение моделирования сложной системы без изменения оригинальной физической динамики. В физике, химии и биологии такой вид моделирования часто называют динамическим моделированием Монте-Карло. Примеры реальных систем с дискретными событиями: сотовая связь, магнитные системы, пространственная эпидемиология, интернет траффик. Событиями в таких системах, соответственно, будут являться входящие/исходящие звонки, изменения спинов, инфекции, прием/передача пакетов (35).

2.1 Основные понятия ПМДС

ПМДС состоит из двух основных компонентов: набор локальных виртуальных времен ЛП, составляющий так называемый профиль времен (Рисунок 2.1), и схемы синхронизации. Для того, чтобы понять, как устроен ПМДС, необходимо сначала разобраться с концепцией виртуального времени, впервые предложенной Jefferson (17), поскольку она лежит в основе ПМДС.

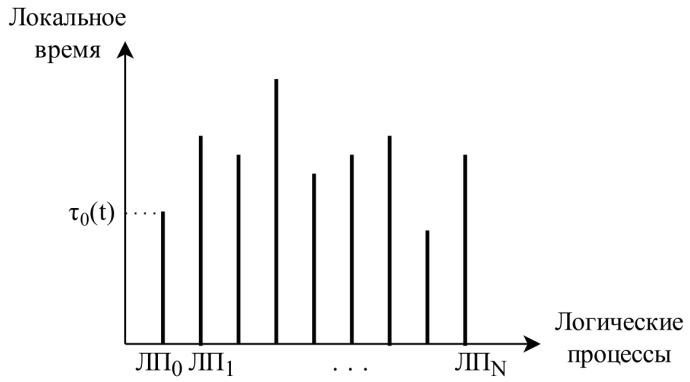


Рисунок 2.1 — Профиль локальных виртуальных времен

Согласно (17), системы с виртуальным временем – это параллель-

ные системы, координирующиеся с виртуальными часами, отсчитывающими виртуальное время. Само по себе виртуальное время является глобальной одноразмерной временной координатой системы. Оно используется для измерения вычислительного прогресса и для синхронизации, при этом виртуальное время не обязательно должно быть связано с реальным временем. Считается, что виртуальное время принимает значения больше 0 в неубывающем порядке. С точки зрения программной реализации, глобальное виртуальное время всегда прогрессирует вперед (либо остается неизменным) с некоторой скоростью. В то же время, существует множество локальных виртуальных часов, которые не синхронизированы между собой и могут иногда идти назад.

Рассмотрим системы, состоящие из множества (до десятков тысяч) процессов, выполняющихся одновременно на параллельном компьютере. Удобно рассматривать каждый процесс, как точку в виртуальном пространстве с уникальным именем и пространственной координатой. Каждое примитивное действие (изменение переменной, отправка сообщения и т.д.) также имеет некое значение виртуального времени и виртуальной координаты в пространстве, а набор таких действий, происходящих в одном и том же виртуальном месте x в одно и то же виртуальное время t называется событием (x, t).

Процессы взаимодействуют между собой путем передачи сообщений. Считается, что для взаимодействия любых двух процессов (в том числе взаимодействие процесса с самим собой) нет необходимости в специальном канале связи, а также в специальных протоколах открытия/закрытия.

Все сообщения должны содержать информацию об отправителе, виртуальном времени отправки, имени получателя и виртуальном времени доставки, т.е. времени, когда сообщение должно быть получено. Иными словами, сообщение имеет метку с координатами отправления и получения.

Системы с виртуальным временем должны следовать двум фундаментальным правилам:

Правило 1: Виртуальное время отравления каждого сообщения должно быть меньше или равно виртуальному времени получения данного сообщения.

Правило 2: Виртуальное время каждого события должно быть мень-

ше или равно виртуальному времени следующего события.

Эти два правила обеспечивают соблюдение причинности в ходе выполнения параллельных вычислений, а именно, они гарантируют, что сообщения будут отправлены в порядке возрастания виртуального времени отправки, и приняты в порядке в возрастания виртуальных времен доставки.

Событие (x, t) – это последовательность вычислений, производимых процессом x и включающая 0 или более следующих операций:

1. Процесс x может получать любое количество сообщений с указанным отправителем x и виртуальным временем получения t и читать содержимое этих сообщений;
2. Процесс x может читать свое виртуальное время;
3. Процесс x может обновлять свои переменные;
4. Процесс x может отправлять любое количество сообщений, которые автоматически будут содержать метки отправителя x и виртуального времени отправки t.

Говорится, что событие А вызывает событие Б, если существует последовательность событий А $= E_0, E_1, \ldots, E_n =$ Б такая, что для каждой пары соседних событий E_i и E_{i+1} выполняется одно из следующих условий:

А) События выполняются в одном процессе (то есть в одном и том же виртуальном пространстве) и виртуальное время E_i меньше виртуального времени E_{i+1}

Б) Событие E_i отправляет сообщение, которое должно быть получено во время события E_{i+1}.

Таким образом, главное ограничение, накладываемое на реализацию систем с виртуальным временем, можно сформулировать так: если событие А вызывает событие Б, тогда выполнение события А должно завершится раньше, чем начнется событие Б. Ясно, что если такой причинной связи между двумя событиями нет, то их можно выполнять в различной последовательности или одновременно, даже если они имеют разные временные координаты.

Итак, для получения корректных результатов программы моделирования каждый ЛП характеризуется своим локальным виртуальное время (ЛВВ). Набор всех ЛВВ образует профиль локальных времен. Важны-

ми статистическими показателями ПМДС являются скорость эволюции и средняя ширина профиля локальных времен. Средняя скорость эволюции показывает величину приращения локального времени за один шаг. Ширина профиля - это средне-статистическое отклонение виртуального времени каждого ЛП от средней высоты профиля. В дальнейшем будем пользоваться следующими обозначениями:

t - глобальное дискретное время (MCS);

$\tau_i(t)$ - локальное время i-того процесса;

$w^2(t)$ - ширина профиля;

$u(t)$ - скорость эволюции профиля.

Поскольку количество процессорных элементов при параллельном моделировании может достигать десятки тысяч, то возникает вопрос о масштабируемости ПМДС. Согласно (37), PDES-схема называется полностью масштабируемой (fully scalable), если:

1. Профиль локальных времен растет с ненулевой скоростью;
2. Ширина горизонта ограничена конечным числом, в то время как количество ЛП стремится к бесконечности.

При выполнении только первого условия для достаточно большого значения t, ПМДС-схема называется computationally scalable, что означает, что при росте количества вычислений, средняя скорость вычисления каждого отдельного ЛП остается неизменной, что обеспечивает отличный от ноля прогресс (скорость) моделирования.

Однако, как будет показано далее, при росте размера системы ширина профиля может сильно рассинхронизироваться, препятствуя постоянному сбору данных о состоянии системы. В частности, когда одному ЛП необходимы некоторые данные о системе в момент физического времени τ, ему необходимо ждать, пока все ЛП дойдут до этого значения. Таким образом, время ожидания пропорционально разбросу флуктуаций или ширине профиля времен. В зависимости от размера для разных ПМДС-схем ширина профиля и время ожидания будут отличаться. Такие системы не являются measurement scalable. Когда выполняется условие 2, мы считаем, что ПМДС-схема является measurement scalable.

Критерий масштабируемости может просто быть формализован в терминах свойств локального профиля времени. Среднее значение профи-

ля времени после t параллельных шагов вычисляется по формуле:

$$\overline{\tau}(t) = \frac{1}{N}\sum_{i=1}^{N}\tau_i(t) \tag{2.1}$$

Measurement масштабируемость PDES-схемы характеризуется разбросом профиля локальных времен. Вместо того чтобы работать с разницей между минимальным и максимальным значением, мы рассмотрим понятие средней ширины профиля, определяемого по формуле:

$$\langle w^2(t)\rangle = \langle\frac{1}{N}\sum_{i=1}^{N}[\tau_i(t) - \overline{\tau}(t)]^2\rangle. \tag{2.2}$$

PDES-схема является measurement масштабируемой, когда существует некая константа $M > 0$ такая, что:

$$\lim_{t\to\infty, N\to\infty}\langle w^2(t)\rangle > M \tag{2.3}$$

В реальности количество ЛП N или время моделирования t никогда не уходят в бесконечность, поэтому на практике масштабируемость выводится из поведения величин в течение долгого времени и большого количество ЛП.

Интересно, что для изучения поведения профиля локальных времен можно использовать некоторые методы из статистической механики. Для этого проводятся аналогии между эволюцией профиля и другими физическими явлениями. Так, эволюция профиля при консервативном алгоритме схожа с процессом роста молекулярной поверхности, а при оптимистичеком - с направленной перколяцией. Подробнее об этих явлениях и их свойствах написано в следующих секциях текущей главы.

2.2 Задача роста поверхности

Эволюция профиля локальных времен в базовой одноразмерной консервативной схеме ПМДС схожа с процессом неравновесного роста поверхности (10; 34). Это сходство было впервые обнаружено G. Korniss и описано в статье «Virtual Time Horizon Control via Communication Network Design»

(37). Термины и обозначения, приведенные выше, хорошо подходят для установления соответствий между неравновесным (non-equilibrum) ростом поверхности и консервативной ПМДС-схемы.

Рассмотрим подробнее процесс роста поверхности и понятие шероховатости. Шероховатость поверхности – это важный феномен науки и технологий. Осаждение из паровой фазы, рост бактериальных колоний, вытеснение жидкости могут служить яркими примерами данного феномена. Исследование таких неравновесных процессов, как рост дендритов, электроосаждение, пробой диэлектрика, фильтрация жидкости в пористой среде, разрушение и многие другие также сводится к описанию формирования и развития поверхности раздела между фазами. Несмотря на разную физическую природу перечисленных явлений, оказалось, что во многих случаях поверхности подчиняются универсальным закономерностям. Одной из наиболее интересных особенностей является нетривиальное скейлинговое поведение «динамической» ширины поверхности (7).

Обычно поверхность растет с d-размерной плоской подложки размером L и со временем становится шероховатой. Становление результирующей поверхности может быть интерпретировано в терминах динамического скейлинга (dynamic scaling) и фрактальной геометрии (1). При данном подходе ширина w поверхности $h(x,t)$ может быть задана так называемым отношением Family-Vicsek (6):

$$w(L,t) = \Big\langle \frac{1}{L}\sum_{i=1}^{L}(h_i - \overline{h})^2 \Big\rangle \sim L^{2\alpha} g(\frac{t}{L^{\alpha/\beta}}), \tag{2.4}$$

где g - это функция скейлинга, h_i - высота поверхности в i-ом узле, $\overline{h}$ - средняя по длине высота, показатели α и β отвечают за поведение ширины поверхности как функции пространства (за длительное время) и времени соответственно.

Для функции скейлинга $g(x)$ получены следующие оценки:

$$g(x) \sim \begin{cases} x^{2\beta} & x \ll 1 \\ const & x \gg 1 \end{cases} \tag{2.5}$$

При малых значениях аргумента функция растет по степенному закону, а при больших значениях приближается к константе.

Показатели α и $z = \alpha/\beta$ задают классы универсальности и являются аналогами критических индексов в теории фазовых переходов второго рода. Заметим, что величина, определяемая соотношением 2.4, является статистическим усреднением дисперсии функции распределения высот и является мерой шероховатости поверхности.

Идея скейлинга (гипотезы подобия) не является новой, и раньше она широко использовалась в физике фазовых переходов. Характерный вид функции шероховатости представлен на рисунке 2.2.

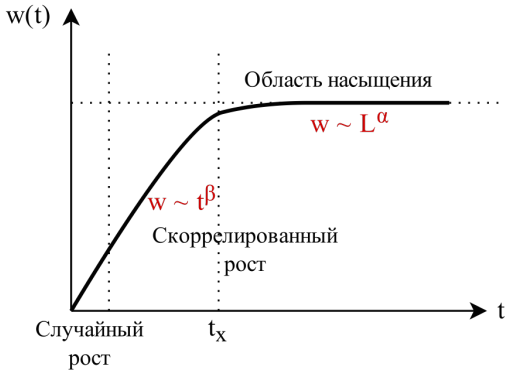


Рисунок 2.2 — Вид функции шероховатости

В начальный момент времени, при формировании первых монослоев шероховатость возрастает пропорционально корню квадратному из времени. При малом заполнении рост колонок является некоррелированным, что соответствует режиму случайного роста. В последующие моменты времени, при возможен скоррелированный рост и шероховатость возрастает по закону $x^{\alpha/z}$.

Существуют различные классы универсальности, характеризующиеся определенными значениями скейлинговых показателей α и z . Понятие класса универсальности фактически предполагает, что физическое поведение совершенно разных систем может описываться подобными математическими уравнениями.

Мощный теоретический аппарат к кинетической шероховатости представлен так называемым уравнением Кардара-Паризи-Жанга (20), описывающим временное развитие высоты $h(x,t)$:

$$\frac{\partial h}{\partial t} = \nu \nabla^2 h + \lambda (\nabla h)^2 + \eta, \tag{2.6}$$

где ν - это эффективный рост поверхности (effective surface growth), а λ - сила бокового роста (strength of lateral growth), η - это некоторый (например, гауссовский) шум.

Для универсальный класс КПЖ показатели α, β, z соответственно равны 1/2, 1/3, 3/2.

Уравнение КПЖ описывает множество компьютерных моделей шероховатости, в том числе, как будет показано далее, и модель роста профиля локальных времен при консервативном алгоритме ПМДС.

2.3 Задача направленной перколяции

Обнаружение фазового перехода в законе изменения средней скорости эволюции профиля времен при оптимистическом сценарии ПМДС было описано в статье (41). Найденные критические показатели совпадают с показателями универсального класса направленной перколяции (DP universality class).

Теория перколяции - это наука о формировании областей связности элементов с определенными свойствами (кластеров) при условии, что связь каждого элемента со своими соседями носит случайный характер (но осуществляется вполне определенным способом) (15).

Можно привести массу примеров явления перколяции в жизни: так, например, проникновение воды сквозь тонкую бумагу имеет ту же математическую интерпретацию, что и движение электричества сквозь двухразмерную случайную цепочку резисторов в физике, а в химии - хроматография. Примеры могут быть найдены не только в физике и химии, но и в биологии, экологии, экономике и т.д.

Явления, описываемые теорией перколяции, также относятся к разряду так называемых критических явлений, которые характеризуются особой критической точкой (обозначим ее в данной работе как q_c), в которой наиболее важное с точки зрения рассматриваемого процесса свойство системы качественно меняется. По существу, это есть фазовый переход второго рода, характеризуемый с количественной стороны набором универсальных критических показателей. Универсальный характер этих показателей заключается в том, что они не зависят от конкретного вида модели, т.е.

типа решетки, а определяются лишь размерностью пространства. Этот основной постулат теории перколяции базируется на анализе огромного количества результатов численного моделирования процессов на решетках различного типа. Однако в простейших случаях, например в случае плоской квадратной решетки, они могут быть получены и аналитическим методом.

Направленная перколяция – это класс моделей, описывающих протекание жидкостей сквозь пористые материалы в заданном направлении. Выделяют отдельный класс универсальности направленной перколяции (DP university class). Данный класс универсальности в одноразмерной решетке характеризуется значением критического показателя $\nu \approx 1.733847(6)$, а значение критической точки $q_c \approx 0.276486(8)$ (26).

В данной ВКР планируется найти и посчитать эти показатели и соотнести с результатами, полученными в (41).

2.4 Заключение к Главе 2

В данной главе были сформулированы такие основные понятия параллельного моделирования дискретных событий, как ширина профиля локальных времен, средняя скорость роста профиля, масштабируемость ПМДС-схем и т.д. Помимо этого приведена информация о росте поверхности и направленной перколяции. Эта информация потребуется далее при обработке результатов моделирования разных ПМДС-схем и анализе их свойств масштабируемости.

ГЛАВА 3

Консервативная схема ПМДС

3.1 Подходы к реализации консервативной схемы

В общем случае консервативный подход ПМДС должен подчиняться следующему правилу: «Для любых i и j логический процесс i никогда не получит сообщения от логического процесса j с временной меткой меньшей, чем время логического процесса i. Другими словами, ЛП не должен получать сообщения из логического «прошлого»» (31).

Есть множество подходов, гарантирующих соблюдение данного правила (16). Первый подход заключается в том, что каждый ЛП должен получить список событий от всех других ЛП, с которыми он может взаимодействовать. Получающий сообщения ЛП выбирает и обрабатывает события с наименьшей временной меткой из своего списка событий. К сожалению, такой метод иногда приводит к возникновению ситуаций, когда несколько ЛП ждут сообщений от друг друга и оказываются заблокированными, иными словами, к возникновению мертвых состояний.

В целях избежания возникновения мертвых состояний были разработаны другие консервативные схемы. Самая простая была предложена Chandy и Misra (2; 3). Они предложили алгоритм, который обнаруживает мертвые состояния и находит логический процесс, который в этом состоянии имеет достаточно данных для безопасного продолжения работы программы.

Существуют также алгоритмы, которые совсем избегают мертвых состояний. Например, при помощи отправки пустых сообщений (null messages). Однако такой метод также имеет ряд существенных недостатков, например, отправка пустых сообщений занимает процессорное время и снижает скорость работы программы. Альтернативный способ отправки пустых сообщений, впервые предложенный в (28) и затем расширенный в (30), заключается в сокращении количества пустых сообщений, когда они высылаются только в ответ на специальный запрос заблокированного ЛП.

Для ряда физических систем, которые могут быть смоделированы методом Монте-Карло, используется специальная схема, называемая базовой консервативной схемой (10), впервые представленная Lubachevsky (23).

Самым ярким примером такой системы является спиновая модель Изинга. Важным свойством такого метода параллельного моделирования является то, что он не изменяет оригинальной динамики моделируемой системы. Каждому элементу физической системы соответствует свой ЛП, который взаимодействует с другими ЛП так же, как и соответствующие элементы физической системы. Поскольку динамика событий в ПМДС является асинхронной, синхронизация происходит за счет взаимодействия между этими ЛП. В базовой одноразмерной схеме взаимодействие между узлами в сети происходит только с ближайшими соседями.

В данной выпускной квалификационной работе проводится анализ только базовой одоразмерной консервативной схемы, которая представляет собой регулярную одноразмерную решетку. С периодическими граничными условиями топология сети представляет собой кольцо (Рисунок 3.1), где каждый узел связан только с ближайшими правым и левым узлом.

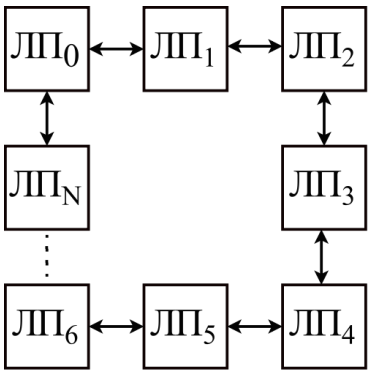


Рисунок 3.1 — Базовая одноразмерная схема с периодическими граничными условиями

3.2 Моделирование консервативной схемы

Пусть N - количество логических процессов в базовой консервативной одноразмерной схеме, а t - количество дискретных параллельных шагов по времени. Каждый ЛП характеризуется значением своего локального виртуального времени τ_i. Поскольку в базовой одноразмерной схеме взаимодействие происходит только с ближайшими соседями, на каждом дискретном шаге моделирования логические процессы синхронизируются только с двумя соседними процессами. Так, если локальное время ЛП не боль-

ше локальных времен его соседей, то такой процесс считается активным (Рисунок 3.2). Это означает, что на данном шаге он может производить вычисления, при этом его локальное время увеличивается на некоторую величину η. В противном случае никаких обновлений в работе ЛП не происходит.

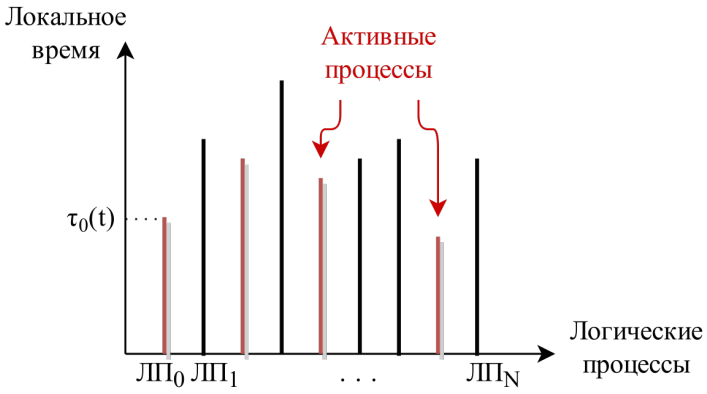


Рисунок 3.2 — Профиль локальных виртуальных времен

Во многих сложных системах динамика случайных событий может быть охарактеризована функцией распределения Пуассона. Распределение Пуассона - это вероятностное распределение дискретного типа, моделирует случайную величину, представляющую собой число событий, произошедших за фиксированное время, при условии, что данные события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

При фиксированном параметре $\lambda > 0$, который также иногда называют интенсивностью, функция вероятности имеет вид:

$$p(k) \equiv \mathbb{P}(Y = k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{3.1}$$

Итак, считая, что случайная величина η распределена по закону Пуассона, можно записать уравнение эволюции локальных времен в данной схеме следующим образом:

$$\tau_i(t+1) - \tau_i(t) = \eta_i(t)\theta(\tau_{i+1}(t) - \tau_i(t))\theta(\tau_i(t) - \tau_{i-1}(t)), \tag{3.2}$$

где η_i - это случайная величина, а θ - ступенчатая функция Хэвисайда.

Ступенчатую функция Хэвисайда также называют единичной ступенчатой функцией или функцией единичного скачка (Рисунок 3.3). Она представляет собой кусочно-постоянную функции, равную нулю для отрицательных значений аргумента и единице - для положительных. В нуле эта функция не определена, однако её обычно доопределяют в этой точке некоторым числом, чтобы область определения функции содержала все точки действительной оси. В этой работе используется следующее определение функции Хэвисайда (39):

$$\theta(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{3.3}$$

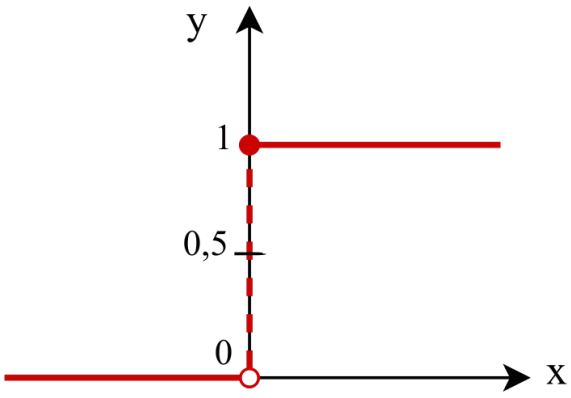


Рисунок 3.3 — Ступенчатая функция Хэвисайда

Введем понятие эффективности эволюции профиля локальных времен. В базовой консервативной одноразмерной схеме эффективности соответствует плотности локальных минимумов профиля времен и может быть записана следующим уравнением:

$$\langle u(t) \rangle = \langle \theta(\tau_{i+1}(t) - \tau_i(t)) \theta(\tau_i(t) - \tau_{i-1}(t)) \rangle. \tag{3.4}$$

Описание практической работы

Для изучения поведения профиля локальных времен была написана программа, моделирующая эволюцию профиля при локальном консервативном алгоритме синхронизации.

Алгоритм программы:

1. Инициализируется массив локальных времен из N элементов, в начальный момент времени каждый элемент массива равен 0.
2. Задается количество шагов по времени M = 1 000 000.
3. На каждом дискретном шаге моделирования проверяется условие активности ЛП. Значение локальных времен активных процессов увеличивается на случайную величину η.
4. Рассчитываются такие показатели, как средняя высота и ширина профиля и средняя скорость эволюции профиля времен.
5. Поскольку в работе программы используются случайные величины, производится несколько циклов вычислений для последующего усреднения результатов.

Было произведено 50-100 циклов вычислений для разных N процессов, усредненные результаты занесены в файл в виде таблицы: $\langle t \rangle$, $\langle \tau(t) \rangle$, $\langle w^2(t) \rangle$, $\langle u(t) \rangle$, где t - дискретное время. По полученным данным были построены графики зависимости ширины и скорости эволюции профиля локальных времен от времени и количества процессов.

Ширина профиля времен растет со временем, а затем прекращается (Рисунок 3.4). Рост осуществляется по степенному закону. Полученный показатель степени равен 0,5830(2). Таким образом, $\langle w^2 \rangle \sim t^{0,5830(2)}$.

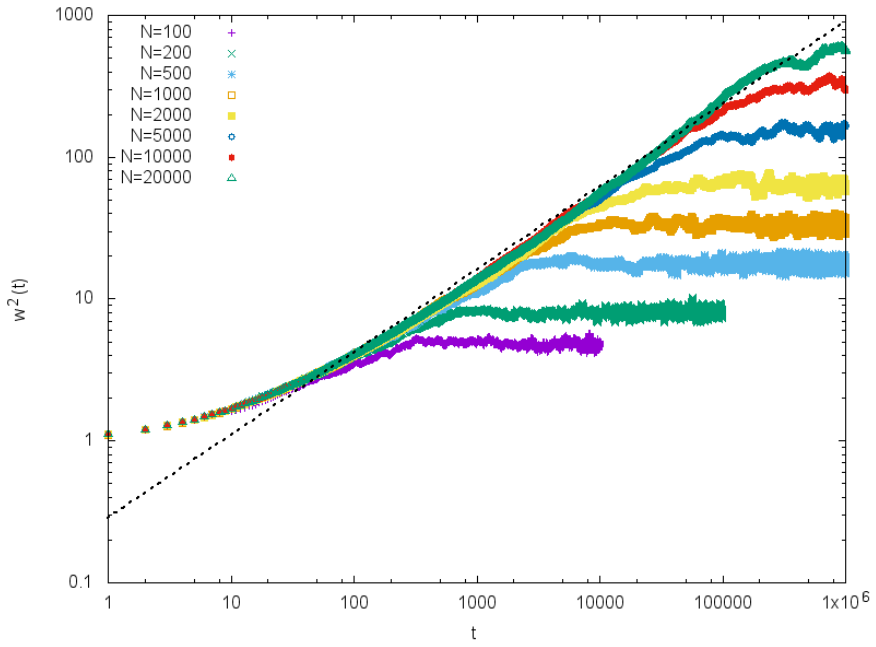


Рисунок 3.4 — Зависимость ширины профиля локальных времен от времени

Зависимость ширины профиля от количества процессов также осуществляется по степенному закону с показателем 0,8(6) (Рисунок 3.5).

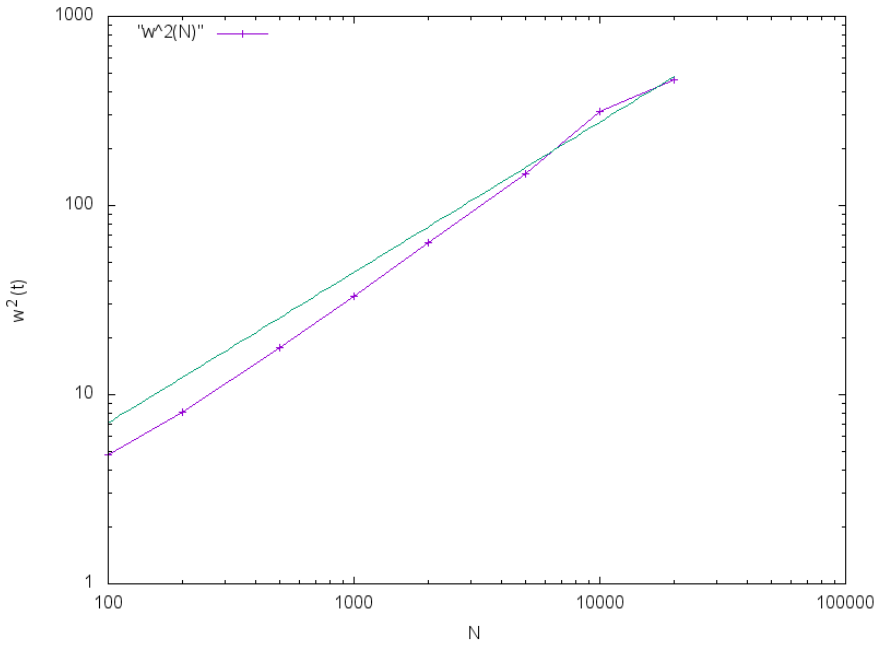
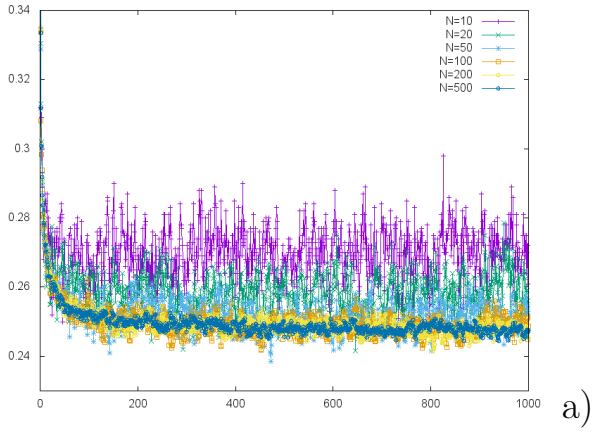
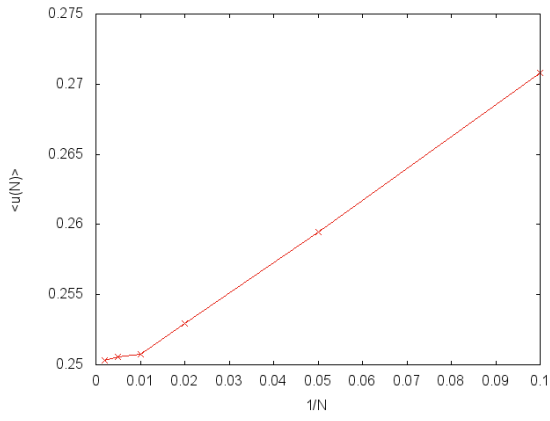


Рисунок 3.5 — Зависимость ширины профиля локальных времен от количества процессов

Средняя скорость эволюции профиля является ненулевой и равна 0.24644(3) (Рисунок 3.6 а). Для меньшего числа процессов этот показатель достигает 0.270(8). Зависимость скорости эволюции от количества процессов продемонстрирована на рисунке 3.6 б.



а)



б)

Рисунок 3.6 — а) Средняя скорость эволюции профиля времен б) Зависимость скорости эволюции от количества процессов

3.3 Аналогия с растущей поверхностью

G. Korniss с соавторами обнаружили сходство процесса роста профиля локальных времен при консервативном алгоритме синхронизации с процессом роста молекулярной поверхности (14; 25; 37). Они показали, что процесс роста профиля относится к универсальному классу, к которому также относится дифференциальное уравнение в частных производных Кардара-Паризи-Жанга (20).

Поверхность растет по степенному закону $\langle w^2(N,t)\rangle \sim t^{2\beta}$ до некоторого времени $t_\times$, после чего достигает устойчивого состояния, при этом ширина поверхности зависит от N следующим образом: $\langle w^2(N,\infty)\rangle \sim N^{2\alpha}$. Экпонента β называется экпонентой роста, α - экпонентой шероховатости, а z - динамической экпонентой, $z = \alpha/\beta$. Для уравнения КПЖ аналитически найденные точные значения этих экпонент следующие: $\alpha = 1/2, \beta = 1/3, z = 3/2$.

В результате моделирования мною были получены следующие показатели роста: $\alpha = 0.4(3)$, $\beta = 0.2915(1)$. Это подтверждает, что с учетом погрешности рост ширины профиля локальных времен можно отнести к классу растущих поверхностей КПЖ (4).

Помимо этого, можно получить уравнение КПЖ из уравнения 3.2 эволюции профиля локальных времен. Рассмотрим вывод формул более подробно. Для начала введем понятие наклонных $\phi_i = \tau_i - \tau_{i-1}$. Тогда:

$\phi_i(t) = \tau_i(t) - \tau_{i-1}(t)$

$\phi_i(t-1) = \tau_i(t) + \eta_i(t) - \tau_{i-1}(t)$

$\phi_i(t+1) = \tau_i(t+1) - \tau_i(t) - \eta_i(t)$

И, наконец,

$\tau_i(t+1) - \tau_i(t) = \phi_i(t+1) - \phi_i(t)$

Производя замену переменной τ на ϕ из уравнения 3.2 получаем:

$$\phi_i(t+1) - \phi_i(t) = \eta_i(t)[\theta(-\phi_i(t)\theta(\phi_{i+1}(t)) - \theta(-\phi_{i-1}(t)\theta(\phi_i(t))] \tag{3.5}$$

Далее заменим функцию Хэвисайда на гиперболический тангенс:

$$\theta(\phi) = \lim_{k\to 0}\frac{1}{2}[1 + th(\frac{\phi}{k})] \tag{3.6}$$

Подставляем 3.6 в 3.5 и для удобства опускаем t:

$$\phi_i(t+1) - \phi_i(t) = \frac{1}{4}\eta_i(t)\big[(1 - th(\frac{\phi_i}{k})(1 + th(\frac{\phi_{i+1}}{k}) - (1 - th(\frac{\phi_{i-1}}{k})(1 + th(\frac{\phi_i}{k})\big] \tag{3.7}$$

Разложив гиперболический тангенс в ряд Тейлора, заменяем $th(\phi)$ на ϕ:

$$\phi_i(t+1) - \phi_i(t) = \frac{1}{4k}\eta_i(t)\big[\phi_{i+1} - 2\phi_i + \phi_{i-1} - \frac{1}{k}\phi_i(\phi_{i+1} - \phi_{i-1})\big] \tag{3.8}$$

Итак, получили:

$$\phi_i(t+1) - \phi_i(t) = \eta_i(t)\big[\frac{1}{4k}\langle\phi_{i+1}(t) - 2\phi_i(t) + \phi_{i-1}(t)\rangle - \frac{1}{4k^2}\langle\phi_i(t)(\phi_{i+1}(t) - \phi_{i-1}(t))\rangle\big] \tag{3.9}$$

Теперь заменим конечные разности на бесконечно малые следующим образом:

$\phi_i(t+1) - \phi_i(t) = \frac{\partial\hat{\phi}}{\partial t}$

$\phi_{i+1}(t) - 2\phi_i(t) + \phi_{i-1}(t) = (\phi_{i+1}(t) - \phi_i(t)) - (\phi_i(t) - \phi_{i-1}(t)) = \frac{\partial^2\hat{\phi}}{\partial x^2}$

$\phi_i(t)(\phi_{i+1}(t) - \phi_{i-1}(t)) = \frac{\partial\hat{\phi}^2}{\partial x}$

В итоге получается уравнение, похожее на уравнение КПЖ 3.11:

$$\frac{\partial\hat{\phi}}{\partial t} = \frac{\partial^2\hat{\phi}}{\partial x^2} - \lambda\frac{\partial\hat{\phi}^2}{\partial x}, \tag{3.10}$$

где λ - это параметр огрубления (coarse-graining procedure). Выразив $\hat{\phi}$ через $\partial\hat{\tau}/\partial x$, приходим к уравнению Кардара-Паризи-Жанга в частных производных (20):

$$\frac{\partial\hat{\tau}}{\partial t} = \frac{\partial^2\hat{\tau}}{\partial x^2} - \lambda\big(\frac{\partial\hat{\tau}}{\partial x}\big)^2 \tag{3.11}$$

3.4 Заключение к Главе 3

Проведен анализ масштабируемости консервативного алгоритма ПМДС. Моделирование показало, что базовая консервативная одноразмер-

ная схема ПМДС является полностью масштабируемой, так как скорость роста профиля локальных времен с течением времени ненулевая и равна 0.24644(3), а ширина профиля ограничена величиной, зависящей от количества логических процессов. Помимо этого было проверено, что найденные показатели роста профиля соответствуют показателям роста поверхности, описываемым уравнением КПЖ.

ГЛАВА 4

Оптимистическая схема ПМДС

4.1 Подходы к реализации оптимистической схемы

Оптимистический алгоритм обнаруживает ошибки причинности, но строго не запрещает их возникновения. Одним из преимуществ данного подхода является то, что он позволяет эффективно распараллеливать те части программы, в которых ошибки могут возникать, но фактически не возникают. Также этот подход позволяет динамическое создание или удаление логических процессов.

Наиболее известный протокол, основанный на парадигме виртуального времени, это протокол Time Warp (5). Впервые его определение было дано в статье Jefferson и Sowizral (19). Фундаментальный механизм Time Warp основан на lookahead – rollback протоколах. Каждый процесс работает самостоятельно, вне зависимости от конфликтов синхронизации с другими процессами. Как только конфликт обнаружен неким процессом, этот процесс производит откат по времени назад до нужного момента, а затем заново производит вычисления. И механизм обнаружения ошибок, и механизм их исправления, являются прозрачными для пользователя (17).

Обычный откат транзакций традиционно использовался для обеспечения устойчивости последовательной системы к ошибкам, однако ранее его никто не использовал для синхронизации параллельных систем. Возможно, причина этому – кажущаяся сложность реализации механизма откатов, либо большая трата времени в ходе выполнения программы модели. Однако Jefferson показал, что на самом деле механизм Time Warp имеет естественную простую реализацию, время на осуществление отката соразмерно со временем, в течение которого в других алгоритмах синхронизации процессы находятся заблокированными, а также, что ошибки причинности возникают в системах относительно редко.

Механизм Time Warp

Напомним, что каждое событие имеет 2 временных штампа: время отправки и время доставки (Глава 2). Будем называть термином «временная метка» виртуальное время доставки сообщения. Для корректной реализации виртуального времени необходимо и достаточно, чтобы в каждом процессе сообщения обрабатывались в порядке неубывания временных меток. В каждый момент реального времени процессы могут иметь различные временные метки.

Неочевидно, как обрабатывать события в порядке неубывания их временных меток, если сообщения приходят случайным образом (в беспорядочной последовательности), то есть каждое следующее сообщение всегда может прийти с меньшей временной меткой, чем сообщение, полученное ранее. Даже если приходит сообщение, которое действительно является «следующим», понять этого нельзя. Это является центральной проблемой реализации виртуального времени, которую решает механизм Time Warp.

Механизм Time Warp не зависит от конкретной компьютерной архитектуры и может быть эффективно запущен на мультипроцессорных компьютерных системах. Предполагается, что коммуникации между процессорами являются надежными, при этом совсем необязательно сообщения должны быть доставлены в порядке их отправки.

Механизм Time Warp имеет две основные части: локальный механизм контроля, необходимый для того, чтобы удостовериться, что обработка событий и доставка сообщений происходят в правильном порядке, и глобальный механизм контроля, работающий с такими процессами, как контроль потока, I/O, обработка ошибок, завершение работы и проч.

Локальный механизм контроля

Хотя существует единый стандарт к продвижению виртуального времени, в реализации нет глобальных виртуальных часов, напротив, каждый процесс имеет свое виртуальное время. Это время не изменяется в процессе обработки события, оно обновляется лишь между событиями и принимает значение, равное временной метки следующего сообщения во входящей

очереди сообщений. В каждый момент времени некоторые локальные часы могут быть впереди других, но это не имеет значения, поскольку процессы могут читать только свое локальное время. Отправленные сообщения автоматически маркируются временной меткой, равной локальному времени отправляемого процесса. Отправителем может быть любой процесс, в зависимости от конкретной реализации конкретной системы.

Каждый процесс имеет свою входную очередь, в которой хранятся входящие сообщения в порядке неубывания виртуального времени получения. В идеале работа процесса – это простой цикл, в котором он принимает сообщения и обрабатывает события в правильном порядке. Однако в связи с разной скоростью вычисления на каждом узле, либо в связи с задержкой в передаче сообщения, иногда приходят сообщения с «прошедшей» временной меткой. Единственный способ, чтобы исправить такие происшествия – это откат по времени назад и отмена всех промежуточных изменений, а затем обработка событий заново. Как только заканчиваются сообщения во входной очереди, локальное время процесса устанавливается на плюс бесконечность и процесс завершает свою работу. Тем не менее, если будет получено сообщение, вызывающее откат, процесс возобновит свою работу.

Важно отметить, что в реальности количество процессов обычно превосходит количество процессоров, поэтому только некоторое подмножество процессов выполняются одновременно. Считается, что на каждом процессоре выполняется тот процесс, который имеет наименьшее время, и в дальнейшем не происходит никаких перераспределений процессов по процессорным элементам.

Реализация отката по времени назад в параллельных системах сложна по ряду причин. Процесс, нуждающийся в откате, может отправить сообщения другим процессам, вызывая при этом цепочку событий, требующих отмены. Некоторые из этих сообщений могут привести к необратимым последствиям (например, отправка денег или запуск ракеты). Некоторые из них могут быть физически в пути и поэтому неподконтрольны. Также может образоваться бесконечный цикл из сообщений, или мертвые состояния. Однако механизм Time Warp может эффективно отменять такие события, вызванные ложно отправкой сообщений.

Антисообщения и механизм отката.

Чтобы понять механизм отката, необходимо рассмотреть структуру процессов и сообщений. Каждый процесс представляет собой набор следующих структур:

1) Имя процесса (координата виртуального пространства), которое должно быть уникально

2) Локальное виртуальное время (координата виртуального времени)

3) Состояние - свои локальные процедуры, переменные и счетчики.

4) Очередь состояний, хранящий копии предыдущих состояний. Для того, чтобы была возможность выполнить откат, механизм Time Warp должен сохранять состояния процессов с некоторой регулярностью, например, после каждого события. В зависимости от конкретной реализации можно изменять частоту сохранения и длину очереди.

5) Входящая очередь, содержащая все входящие сообщения, хранящиеся в порядке неубывания временных меток. При этом сообщения не удаляются даже после обработки на случай отката.

6) Исходящая очередь, состоящая из «негативных» копий сообщений, которые уже были отправлены, хранящихся в порядке неубывания временных меток. Такие сообщения отправляются при необходимости их отмены.

Для каждого сообщения существует свое антисобщение, которое отличается от оригинального только одним полем – знаком. Два одинаковых сообщения, но с разными знаками называется антисообщениями. Все сообщения, которые отправляются имеют знак «+» и помещаются во входящую очередь целевого процесса, а копии этих сообщений остаются в исходящей очереди со знаком «-». Если антисообщения обнаруживаются в одной и той же очереди, то они немедленно уничтожаются.

Такой протокол крайне устойчив и работает правильно при любых условиях. Нет никаких блокирующих событий, а следовательно, нет вероятности наступления мертвых состояний. А также невозможно наступление «эффекта домино» (каскадное продвижение по времени назад) – в худшем случае все процессы в системе откатятся до времени, на котором обнаружилась ошибка причинности.

Как было сказано ранее, затраты на выполнение отката сравнимы с временем блокировки процессов в других алгоритмах синхронизации. «Стоимость отката» - затраты на выполнение специальных операций, кото-

рые не происходят в течение нормального прогресса моделирования. Так, откат влечет за собой исправление следующих структур: 1) входящей очереди, 2) очереди состояний, 3) исходящей очереди. Изменение входящей очереди и состояния несет незначительную стоимость, так как необходимо лишь переместить указатель на другое событие в очереди. Стоимость же изменения исходящей очереди гораздо больше, так как требует отправки анти-сообщений.

Можно предположить, что стоимость отката пропорциональна длине отката (то есть длине промежутка времени, на который система должна отодвинуться назад). Тем не менее, откат назад на 7 временных шагов занимает меньше времени, чем продвижение вперед по времени на те же 7 шагов, поскольку отправка антисообщений занимает меньше времени, чем отправка позитивных сообщений. Это объясняется тем, что антисообщения по сути являются уже сформированными ранее сообщениями, но имеют только другой флаг. Позитивные же сообщения должны сохраняться в буфер и быть заполнены данными, которые необходимо отправить. Плюс прогресс вперед несет затраты на планирование новых событий, обработку входящих сообщений, сохранение состояния процессов и сами вычисления. Таким образом, стоимость отката назад по времени является менее ресурсозатратным, чем продвижение вперед. В худшем случае стоимость отката составляет примерно одну десятую стоимости прогресса (11). Поэтому будем считать, что он пропорционален длине отката, но с коэффициентом 0,1. Более того, обычно длина отката является небольшой, поэтому стоимость отката назад может быть принята за небольшую константу.

Глобальный механизм контроля

Глобальный механизм контроля отвечает за выполнение глобального прогресса моделирования, за обнаружение окончания моделирования, отслеживает ошибки ввода-вывода, переполнения памяти и т.д.

Центральная концепция глобального механизма контроля – это концепция глобального виртуального времени, описанная в Главе 2. По определению, данному в (17), глобальное виртуальное время в момент реального времени r – это минимальное из всех виртуальных времен процессов

и виртуального времени отправки всех сообщений, которые уже были отправлены, но еще не обработаны в момент реального времени r.

ГВВ никогда не уменьшается, несмотря на то, что локальное время процессов часто уходит назад. А раз так, то значение ГВВ является максимальной, до которого процессы могут произвести откат. Фактически если каждое событие завершается нормально, если сообщения доставляются надежно (что предполагалось в самом начале), если обработка событий не откладывается на неопределенное время (что также было оговорено), а также если достаточно памяти, ГВВ должно увеличиваться со временем.

Благодаря этому свойству ГВВ может служить показателем общего прогресса моделирования. Для эффективной работы программы моделирования необходимо периодически обновлять значение ГВВ. Частота должна быть выбрана оптимальным образом, поскольку, с одной стороны, чем чаще обновляется ГВВ, тем лучше используется память, однако тем больше загружаются процессорные элементы и каналы связи, а следовательно, замедляется общий прогресс.

Использование ГВВ также помогает при управлении памятью. Так, любое сообщение во входящей и исходящей очередях, чье виртуальное время меньше глобального, может быть удалено. Похожим образом удаляются все сохраненные состояния каждого логического процесса до текущего глобального времени.

4.2 Моделирование оптимистической схемы

Для изучения поведения профиля локальных времен при оптимистическом алгоритме ПМДС была написана программа, моделирующая эволюцию профиля для одноразмерного случая с периодическими краевыми условиями.

Пусть N - количество логических процессов, а τ_i - ЛВВ i-того ЛП. В начальный момент значение виртуального времени всех ЛП равно 0. На каждом временном шаге случайным образом выбирается процесс и его виртуальное время увеличивается на некоторую случайную величину J из распределения Пуассона. Этот процесс повторяется N раз. Это соответствует работе оптимистического алгоритма до момента обнаружения ошибки

причинности - каждый ЛП обрабатывает события без какой-либо синхронизации с другими процессами. Далее моделируется сам процесс отката процессоров по времени. При этом он происходит также асинхронно: выбирается случайный ЛП и проверятся, нарушена ли причинность. Проверка синхронизации происходит только с одним из ближайших соседей. Если ошибка была обнаружена, т.е. виртуальное время выбранного ЛП превосходит время другого ЛП, то уменьшается до значения виртуального времени этого ЛП. Эта операция производится K·N раз, где K - случайная величина. После этого считается средняя скорость эволюции профиля ЛВВ, ширина профиля и ее дисперсия. Программа моделирования запускается с различными параметрами случайных величин J и K.

Основная исследуемая величина - это средняя скорость эволюции профиля ЛВВ $\langle u \rangle$. Она рассчитывается как производная значения ЛВВ по времени, то есть на каждом шаге моделирования усредненное по процессам значение скорости профиля равно:

$$\langle u \rangle = \langle \tau(t+1) - \tau(t) \rangle \tag{4.1}$$

Помимо усредненной скорости оценивалась скорость отстающего процессорного элемента (т.е. минимальная скорость $< u_{min} >$).

Алгоритм программы:

1. Инициализируется массив локальных времен длинной N. В начальный момент времени $\tau_i = 0$. Задается число шагов по времени M.
2. N раз время случайно выбранного логического процесса τ_i увеличивает свое значение на случайную величину J, распределенную по закону Пуассона с параметром r.
3. Генерируется случайное число K, также распределенное по пуассоновскому закону с параметром b.
4. $K \cdot N$ раз выбирается случайный логический процесс i. Его локальное время τ_i с вероятностью $1/2$ принимает значение $min(\tau_i, \tau_{i+1})$ или $min(\tau_{i-1}, \tau_i)$.
5. Рассчитывается средняя скорость эволюции профиля, средняя скорость отстающего процессорного элемента и их дисперсия.

6. Шаг 2 - 6 повторяются M раз.
7. Программа выполняется K раз для последующего усреднения результатов.

При выполнении программы были выбраны следующие значения переменных: $N = 10000, M = 10000, K = 100, r = [1, 2, 3, 4, 10], b = [0.1..10]$ с шагом 0.1. Для анализа был также введен параметр $q = 1/(1 + b)$. Построены графики зависимости средней скорости профиля от параметров r, b, q.

Обнаружено, что скорости ведут себя одинаковым образом при разных параметрах r (Рисунок 4.1). Видно, что увеличение параметра b приводит к резкому падению скорости до 0 (иными словами, рост профиля ЛВВ, а следовательно, т.е. общий прогресс программы моделирования, останавливается).

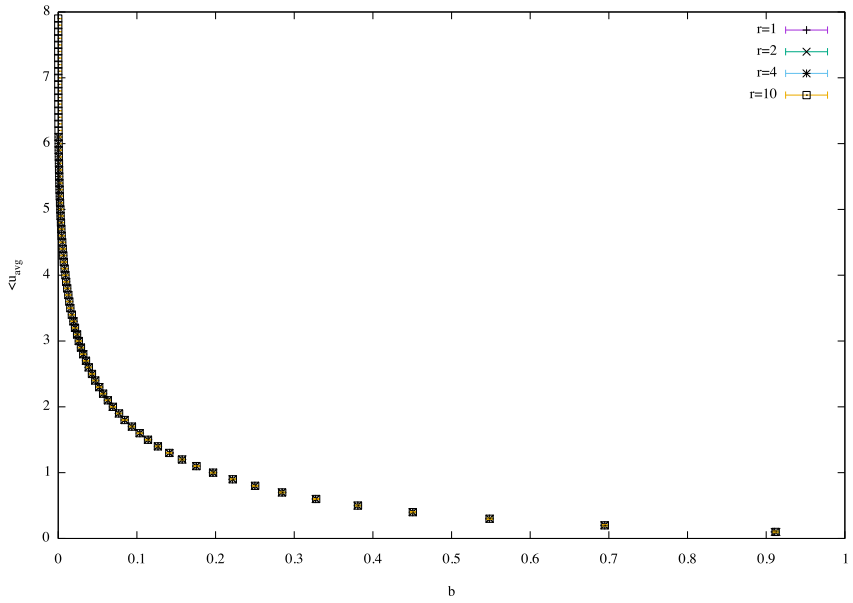
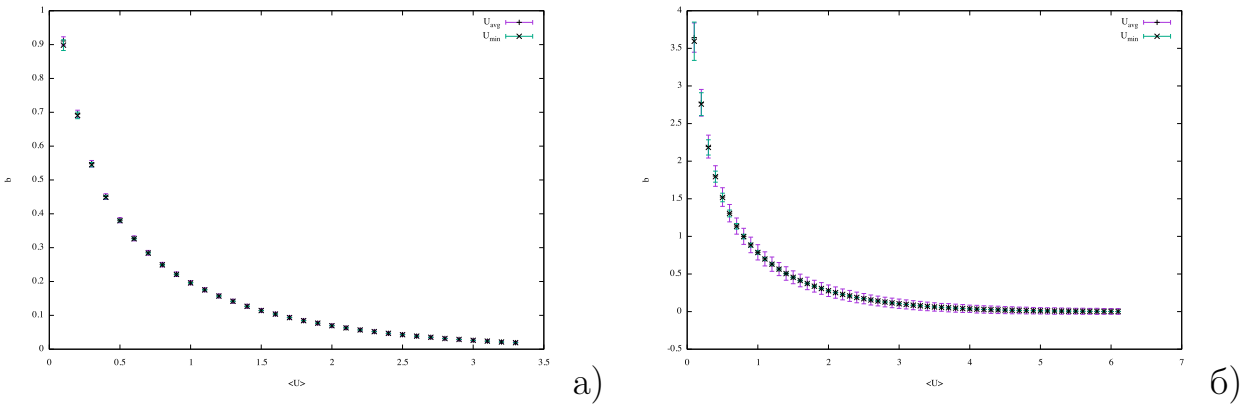


Рисунок 4.1 — Зависимость средней скорости профиля локальных времен от параметра b

При этом обнаружено, что средняя скорость $\langle u_{avg} \rangle$ и скорость отстающего ЛП $\langle u_{min} \rangle$ также практически совпадают (Рисунок 4.2). Это означает, что элемент с минимальной временной меткой прогрессирует во времени так же, как и остальные элементы (в отличие от консервативного ПМДС):

$$\frac{\langle u_{min} \rangle}{r} = \frac{\langle u_{avg} \rangle}{r} \tag{4.2}$$

На графике 4.3 можно проследить зависимость скорости от парамет-



а) б)

Рисунок 4.2 — Сравнение средней скорости профиля и скорости отстающего процессорного элемента а) r = 1 б) r = 4

ра q: виден перегиб в точке q_c. Найденное значение $q_c \approx 0.143(5)$ (Рисунок 4.4).

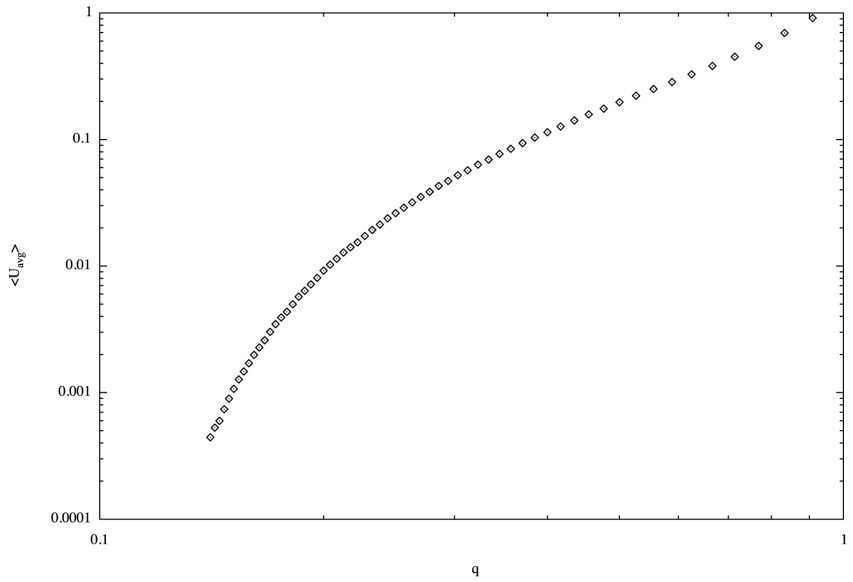


Рисунок 4.3 — Зависимость средней скорости профиля локальных времен от параметра b

Найденный показатель степени, по которой скорость уменьшается до 0 при приближении q к критическому значению q_c приблизительно равен 1.71(6) (Рисунок 4.4). Таким образом, закон, по которому изменяется средняя скорость профиля ЛВВ при оптимистическом алгоритме ПМДС выглядит следующим образом:

$$u = u_0(q - q_c)^{\nu} \tag{4.3}$$

где найденные значения $u_0 \approx 1.19(9), q_c \approx 0.143(5), \nu \approx 1.71(6)$.

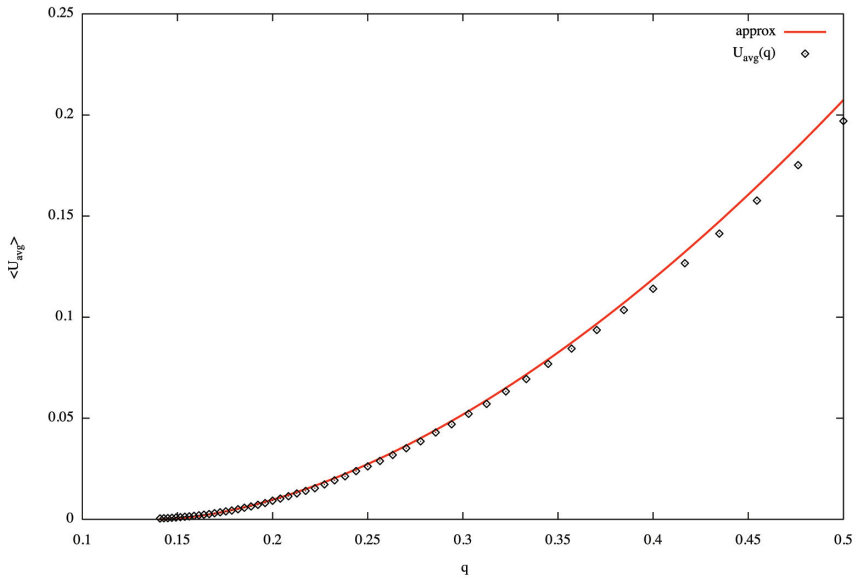


Рисунок 4.4 — Зависимость средней скорости профиля локальных времен от параметра q (данные взяты для r=4)

4.3 Аналогия с направленной перколяцией

Итак, как было сказано в Главе 2, значение показателя класса направленной перколяции $\nu = 1.733847(6), q_c = 0.276486(8)$. В статье (41) были приведены следующие значения: $q_c = 0.233$, $\nu = 1.74(2)$. «Параметр K при этом принимает критическое значение $K_c = 2,39J$. Практически это означает, что если при выполнении J шагов оптимистической эволюции понадобится K_c шагов для отхода назад по времени, то система не сможет эволюционировать во времени - горизонт времен будет флуктуировать вокруг некоторого значения. В теории роста поверхности этот эффект называется переходом шероховатости поверхности. В нашем случае средняя скорость горизонта времен обращается в нуль при критическом значении параметра по степенному закону с показателем 1.74(2)» (41).

4.4 Заключение к Главе 4

В Главе 4 бы подробно описан основой подход к реализации оптимистической схемы ПМДС - алгоритм Time Warp. Была реализована модель этого алгоритма и изучено поведение профиля ЛВВ. Был обнаружен фазовый переход в законе изменения средней скорости эволюции профиля

времен. Предполагалось, что критические показатели будут совпадать с критическими показателями универсального класса направленной перколяции. Однако гипотеза не подтвердилась. Причины этого предстоит выяснить в дальнейшем.

ГЛАВА 5

Сравнительный анализ схем ПМДС

После анализа консервативного и оптимистического алгоритмов ПМДС возникает логичный вопрос, какой же из них лучше? Ясно, что ответить однозначно на этот вопрос нельзя, поскольку ответ зависит от множества факторов: от конкретной моделируемой системы, от платформы моделирования, коммуникационной схемы между процессорными элементами, и т.д.(9)

Один из недостатков консервативного алгоритма заключаются в том, что он является блокирующим, то есть может вызывать мертвые состояния в процессе выполнения программы моделирования. Предотвращение и распознавание возникновения таких состояний требует дополнительных накладных расходов на координацию процессорных элементов между собой (например, на отправку пустых сообщений), что снижает общую производительность алгоритма. В оптимистических же алгоритмах ошибочные вычисления могут образовывать бесконечный цикл, требующий грубого прерывания работы программы моделирования, поэтому механизм Time Warp должен предотвращать появление таких ситуаций. Тем не менее, для моделирования систем, в которых ошибки причинности возникают редко, большую производительность показывает оптимистический алгоритм.

Критика оптимистического алгоритма хорошо описана в статье Fujimoto (13). Так, автор говорит, что основной критический момент оптимистического подхода заключается в том, что система тратит значительную часть времени на исправление ошибочных вычислений. Однако такое поведение встречается редко на практике, а когда встречается, то зачастую может быть объяснено слабыми местами в самой реализации, нежели фундаментальными недостатками алгоритма.

Более серьезная проблема оптимистической схемы - это необходимость периодически сохранять состояние логических процессов. Это существенно снижает производительность Time Warp программ, даже если вектор состояния имеет небольшой размер. Данная проблема усугубляется в программах с динамической памятью. Накладные расходы на сохранение вектора состояния ЛП ограничивают эффективность алгоритма Time Warp даже в таких приложениях, где затраты на обработку событий суще-

ственно превосходят затраты на сохранения состояний.

Помимо этого, оптимистические алгоритмы требуют в несколько раз больше памяти, чем консервативные. В (18) Jefferson показал, что можно реализовать Time Warp, используя такое же количество памяти, что и для соответствующего последовательного алгоритма, однако производительность такой реализации будет низкой. Он определил протокол, который откатывает процессы назад, при необходимости изменяя ресурсы памяти. Это дает возможность работать алгоритму Time Warp с любым размером предоставленной памяти. Более того, что Jefferson также показал, что консервативная ПМДС схема тоже неоптимально использует ресурсы памяти.

Наконец, оптимистические алгоритмы сложнее реализовывать. Несмотря на то, что сам код может быть не очень сложным, несущественные ошибки или неоптимальные реализации некоторых отдельных механизмов (например, планировщика событий) могут приводить к существенному снижению производительности. Помимо этого, отлаживание алгоритма является затратным по времени. Консервативная схема, напротив, является прозрачной для реализации и имеет простые структуры данных и механизм контроля.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе представлена классификация различных видов моделирования с акцентом на параллельное моделирование дискретных событий, проведен анализ консервативного и оптимистического алгоритмов синхронизации ПМДС, а также приведена их сравнительная характеристика.

Анализ алгоритмов ПМДС включал в себя моделирование этих алгоритмов и изучение поведения профиля локальных виртуальных времен.

Было показано, что консервативный алгоритм является полностью масштабируемым, то есть при увеличении числа логических процессов скорость эволюции профиля отлична от нуля, а ширина профиля ограничена некой константой, зависящей от количества процессов. Также в работе указывалось на аналогию эволюции профиля при консервативном ПМДС с растущей поверхностью, а именно, было продемонстрировано, что критические показатели роста профиля локальных времен соответствуют критическим показателям универсального класса КПЖ.

В законе изменения средней скорости эволюции горизонта времен при оптимистическом сценарии ПМДС был также показан фазовый переход. Предположение, что критические показатели будут совпадать с классом направленной перколяции, не подтвердилось. Скорее всего, допущена ошибка в реализации оптимистической модели.

Такие аналогии алгоритмов с теорией роста поверхности, а также с теорией перколяции могут быть полезны при частных реализациях обсуждаемых алгоритмов проведения моделирования методом параллельных дискретных событий. Предложенные алгоритмы учитывают многоядерность и многопотоковость, а также другие особенности развития современных вычислительных систем.

Направление дальнейших исследований

Watts и Strogatz предложили модель сети, известную под названием Small-World network (38). Такая модель представляет собой регулярную

решетку с добавлением случайных связей между узлами. Это сокращает средний путь между двумя узлами сети и, оказывается, может оказывать влияние на масштабируемость алгоритмов синхронизации ПМДС. Такое влияние было продемонстрировано в статье «Synchronisation landscapes in small-world-connected computer networks» (9). Для оптимистической схемы таких исследований не проводилось. Таким образом, одно из дальнейших направлений текущей исследовательской работы может быть изучение эволюции профиля локальны времен при оптимистическом алгоритме в сетях топологии Small-World. (40)

ЛИТЕРАТУРА

1. Barabasi, A.-L. Fractal Conepts in surface growth / A.-L. Barabasi, H. E. Stanley. — USA: Cambridge University Press, 1995.

2. Chandy, K. Asynchronous distributed simulation via a sequence of parallel computations / K.M. Chandy, J. Misra // CACM.— 1981.— Vol. 24, no. 4. — P. 198–205.

3. Chandy, K. Distributed simulation: A case study in design and verification of distributed programs / K.M. Chandy, J. Misra // Software Engineering, IEEE Transactions. — 1989. — Vol. SE-5, no. 5.

4. Corwin, I. The Kardar-Parisi-Zhang equation and universality class / I. Corwin // Random matrices: Theory and applications. — 2012. — Vol. 1, no. 01.

5. Distributed simulation and the time warp operating system / D. Jefferson, B. Beckman, F. Weiland et al. // ACM SIGOPS Operating Systems Review. — 1987. — Vol. 21, no. 5. — P. 77–93.

6. Family, F. Scaling of the active zone in the eden process on percolation networks and the ballistic deposition model / F. Family, T. Vicsek // Journal of Physics A: Mathematical and General. — 1985. — Vol. 18.

7. Family, F. Dynamics of fractal surfaces / F. Family, T. Vicsek // World Scientific. — 1991.

8. Felderman,R.Anupperboundontheimprovementofasynchronousversus synchronous distributed processing / R.E. Felderman, L. Kleinrock // Proceeding of thr SCS Multiconference on Distributed Simulation. — SCS, 1990.

9. Parallel and distributed simulation of discrete event systems: Rep. /

University of Maryland at College Park College Park; Executor: A. Fersha, S. K. Tripathi. — USA: 1994.

10. From massively parallel algorithms and fluctuating time horizons to nonequilibrium surface growth / G. Korniss, Z. Toroczkai, M. A. Novotny, P. A. Rikvold // Physical Review Letters. — 2000. — Vol. 84, no. 1351.

11. Fujimoto, R. M. Time warp on a shared memory multiprocessor / R. M. Fujimoto // Trans. Soc. for Comput. Simul. — Vol. 6. — 1989.

12. Fujimoto, R. M. Parallel Discrete Event Simulation / R. M. Fujimoto. — Commun. ACM., 1990.

13. Fujimoto, R. M. Parallel discrete event simulation / R. M. Fujimoto // Communication of the ACM. — 1990. — Vol. 33, no. 10.

14. Going through rough times: from non-equilibrium surface growth to algorithmic scalability / G. Korniss, M.A. Novotny, P.A. Rikvold et al. // Materials Research Society Symposium Proceedings Series. — 2002. — Vol. 200. — P. 297–308.

15. Hinrichsen, H. Nonequilibrium critical phenomena and phase transitions into absorbing states / H. Hinrichsen // Advances in Physis. — 2000. — Vol. 49. — P. 815–958.

16. Jafer, S. Synchronization methods in parallel and distributed discrete-event simulation / S. Jafer, Q. Liu, G. Wainer // Simulation Modelling Practice and Theory. — 2013. — Vol. 30. — P. 54–73.

17. Jefferson, D. Virtual time / David Jefferson // ACM Transactions on Programming Languages and Systems (TOPLAS). — 1985. — Vol. 7, no. 3.

18. Jefferson, D. Virtual time ii: storage management in conservative and optimistic systems / D. Jefferson // Proceeding PODC '90 Proceedings of the ninth annual ACM symposium on Principles of distributed computing. — 1990. — P. 75–89.

19. Jefferson, D. Fast Concurrent Simulation Using the Time Warp Mechanism / D. Jefferson, H. Sowizral. — Santa Monica, Calif. : Rand Corp., 1982.

20. Kardar, M. Dynamic scaling of growing interfaces / M. Kardar, G. Parisi, Y.-C. Zhang // Physical Review Letters.— 1986.— Vol. 56, no. 9.— P. 889–892.

21. Li, H. Time advancement in distributed event simulation / H.F. Li, K. Venkatesh, T. Radharkishnan // Journal of Parallel and Distributed Computing. — 1990. — Vol. 9. — P. 15–25.

22. Lubachevsky, B. Bounded lag distributed discrete event dimulation / B. Lubachevsky // Proceeding of thr SCS Multiconference on Distributed Simulation. — SCS, 1988.

23. Lubachevsky, B. Efficient parallel simulations of dynamic ising spin systems / B. Lubachevsky // Journal of Computational Physics. — 1988. — Vol. 75. — P. 103–122.

24. Misra, J. Distributed discrete-event simulation / J. Misra // ACM Computing Surveys. — 1986. — Vol. 18, no. 1.

25. Nonequilibrium surface growth and scalability of parallel algorithms for large asynchronous systems / G. Korniss, M. A. Novotny, Z. Toroczkai, P. A. Rikvold // Computer Simulation Studies in Condensed-Matter Physics XIII. — 2000. — Vol. 86.

26. Odor, G. Universality classes in nonequilibrium lattice systems / Geza

Odor // Reviews of Modern Physics. — 2004. — Vol. 76.

27. Parallel multi-objective optimization using master-slave model on heterogeneous resources / Sanaz Mostaghim, Jurgen Branke, Andrew Lewis, Hartmut Schmeck. — 2008.

28. Peacock, J. Synchronization of distributed simulation using broadcast algorithms / J.K. Peacock, E. Manning, J.W. Wong // Computer Networks. — 1980. — P. 3–10.

29. Peacock, J. Distributed simulation using a network of processors / J.K. Peacock, J.W. Wong, E.G. Manning // Computer Networks.— 1979. — Vol. 3, no. 1.

30. Reynolds, P. F. A shared resource algorithm for distributed simulation / P. F. Reynolds // Proc of the Ninth Annual Int'l Comp Arch Conf.— 1982. — P. 259–266.

31. Reynolds, P. F. A spectrum of options for parallel simulation / P. F. Reynolds // In Processing of the 1988 Winter Simulatio Conference. — 1988. — P. 325–332.

32. Shchur, L. Evolution of time horizons in parallel and grid simulations / L.N. Shchur, M.A. Novotny // Physical Review Letters. — 2004. — Vol. 70.

33. Sokol, L. Mtw: a strategy for scheduling discrte simulation events for concurrent execution / L. Sokol, D. Briscoe, A. Wieland // Proceeding of thr SCS Multiconference on Distributed Simulation. — SCS, 1988.

34. Suppressing roughness of virtual times in parallel discrete-event simulations / G. Korniss, M.A. Novotny, H. Guclu et al. // Science. — 2003. — Vol. 299, no. 5607. — P. 677–679.

35. Synchronization landscapes in small-world-connected computer networks / H. Gulcu, G. Korniss, M. Novotny et al. // Physical Review Letters. — 2000. — Vol. 84, no. 1351.

36. Venkatesh, K. Discrete event simulation in a distributed system / K. Venkatesh, T. Radhakrishnan, H.F. Li // IEEE COMPSAC, IEEE Computer Society Press. — 1986. — P. 123–129.

37. Virtual time horizon control via communication network design / G. Korniss, Z. Toroczkai, M. A. Novotny, P. A. Rikvold // Physical Review Letters. — 2006. — Vol. 73, no. 066115.

38. Watts, D. J. Collective dynamics of «small-world» networks / D. J. Watts, S.H.Strogatz // Nature. — 1998. — Vol. 393.

39. Абрамовиц М., Стиган И. Справочник по специальным функциям с формулами, графиками и математическими таблицами // Под редакцией М. Абрамовица и И. Стиган. - М.: Наука, 1979. 832 с.

40. Д.А. Гавриш, С.Н. Саранча Методы распределенного моделирования дисретно-событийных систем // Системы обработки информации. – 2004. – 5.

41. Л.Н. Щур, М.А. Новотный Эволюция горизонта времен при параллельном моделировании дискретных событий // Труды Семинара по вычислительным технологиям в естественных науках. Вып. 1. Вычислительная физика / Под ред. Р. Р. Назирова. М. : Изд-во КДУ, 2009, с. 6-13