iSCALARE

Лаборатория суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур

Параллельная симуляция

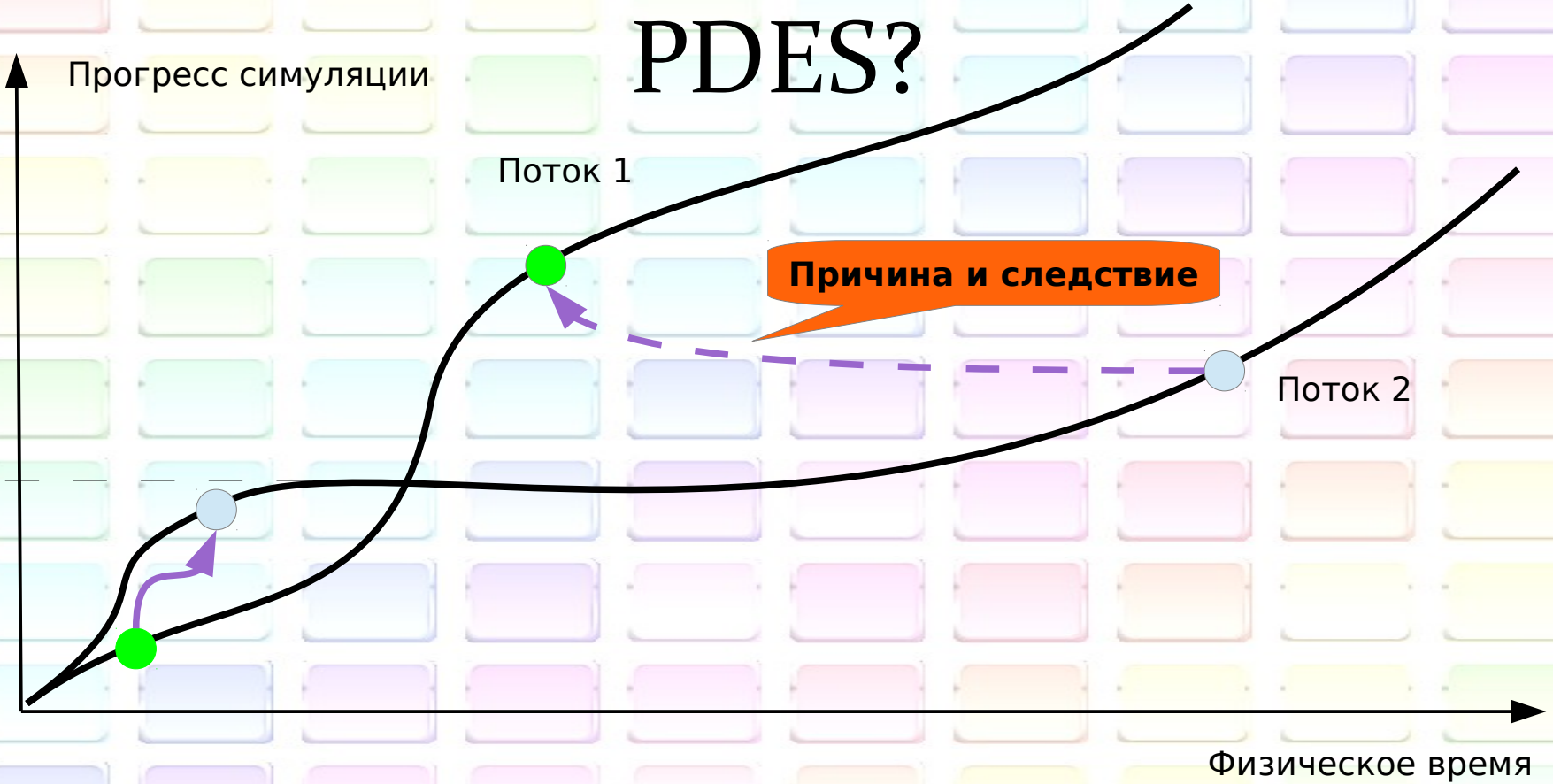
часть 2

Григорий Речистов

grigory.rechistov@phystech.edu

- Parallel Discrete event simulation
- Консервативные модели
- Оптимистические модели
- Time Warp

PDES?


Прогресс симуляции
Поток 1
Причина и следствие
Поток 2
Физическое время

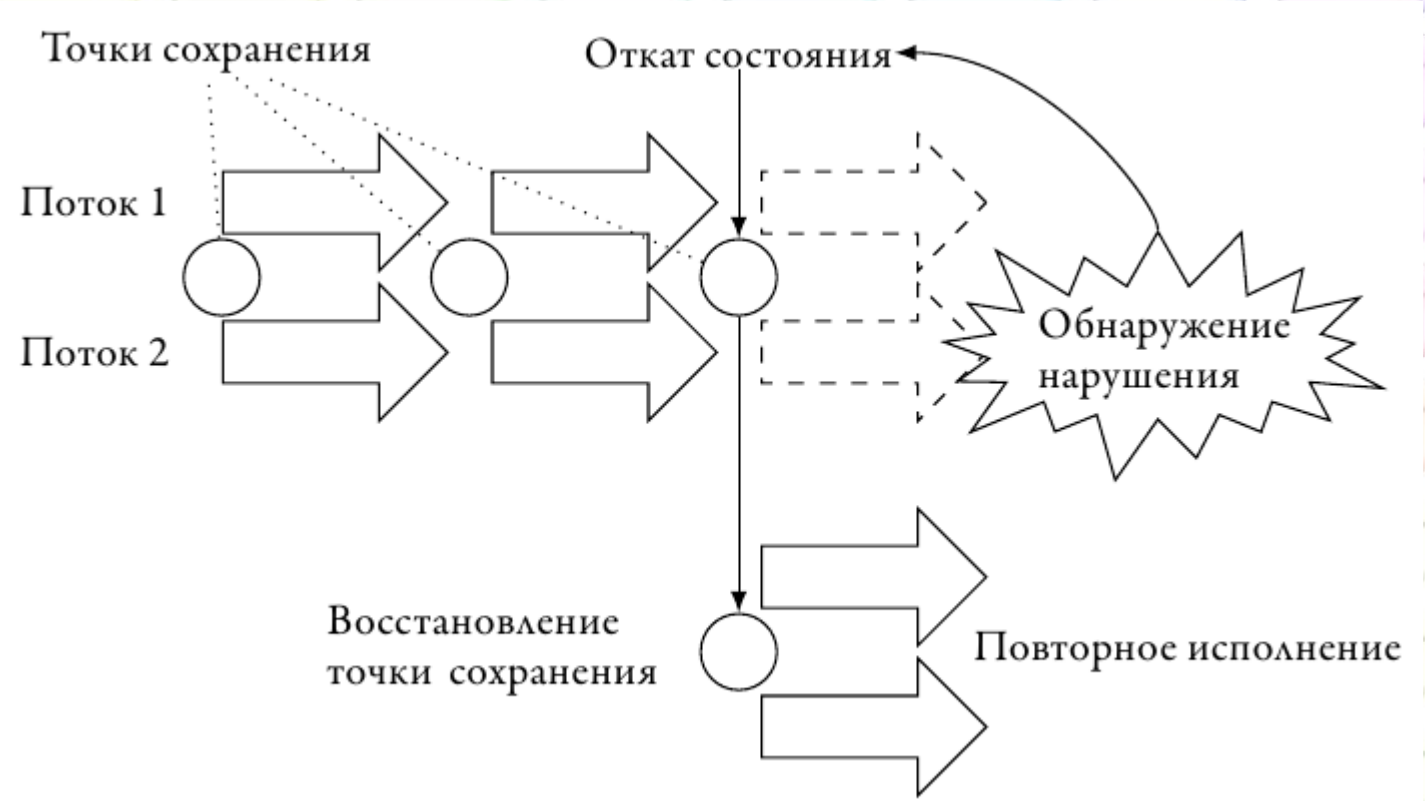
Предпосылки

- У нас нет гарантий, что потоки симулятора будут исполняться синхронно
- Однако можно предположить, что большую часть времени они будут выровнены, и порядок событий будет корректным

Оптимистичные схемы 1

- Даём параллельной программе работать самой по себе
- Периодически сохраняем (корректное) состояние всей системы
- При обнаружении каузальных ошибок откатываемся до ближайшего сохранённого состояния
- Проходим проблемный участок аккуратными методами (напр. консервативно)

Оптимистичные схемы 2



Оптимистичные схемы 3

+ Синхронизируемся только тогда, когда этого не удалось избежать

- Цена создания точек сохранения: время, память

- Цена отката

- Необходимость иметь схему симуляции для «плана Б» в случае отката

Time Warp

- Сообщение — набор данных, описывающих событие, которое должно быть добавлено в одну из очередей событий. Оно характеризуется, кроме своего непосредственного содержимого, виртуальными временами отправки tsend и обработки treceive.
- LVT (local virtual time) — значение симулируемого времени отдельного потока, участвующего в симуляции. Для создаваемых событий их время отправки tsend равно значению LVT отправителя. В отличие от консервативных схем, эта величина может как расти в процессе симуляции, так и убывать в случае отката процесса.

Jefferson et al. Time warp operating system. 1987

GVT

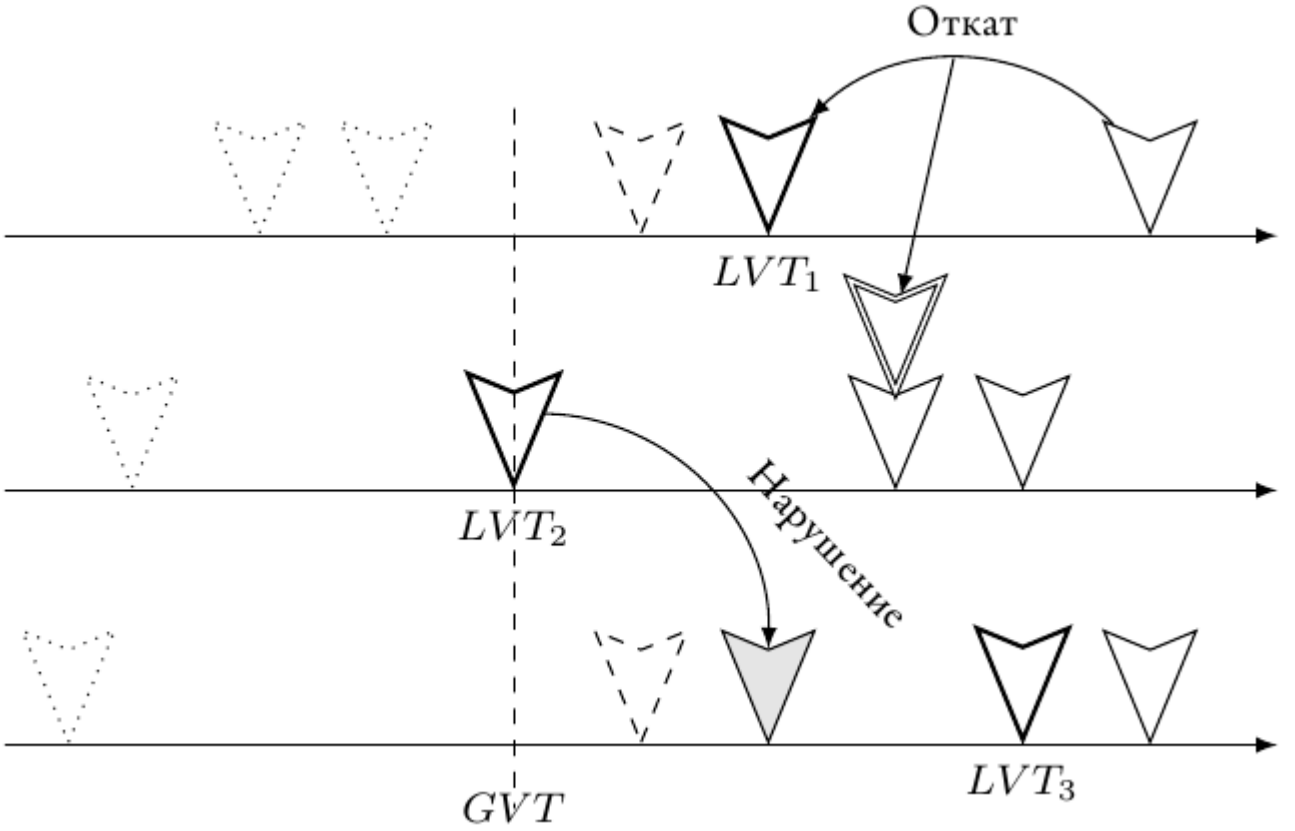
- GVT (global virtual time) — глобальное время для всей симуляции, определяющее, до какой степени возможно её откатывать. Глобальное время всегда монотонно растёт, всегда оставаясь позади локального времени самого медленного потока, а также оно меньше времени отправки самого раннего ещё не доставленного события:

$$GVT \leq \min\left(\min_i LVT_i, \min_k t_k^{receive}\right).$$

Straggler, antimessage

- Отставшее сообщение (straggler) — событие, пришедшее в очередь с меткой времени treceivestraggler, меньшей, чем LVT получателя. Его обнаружение вызывает откат текущего состояния, при этом LVT уменьшается, пока не станет меньше, чем treceivestraggler, после чего оно может быть обработано. После этого возобновляется прямая симуляция.
- Антисообщение (antimessage). Каждое антисообщение соответствует одному ранее созданному сообщению, порождённому в интервале симулируемого времени [tstraggler, LVTi] и вызывает эффект, обратный его обработке (т.е. возвращает состояние в исходное).

Работа Time Warp



Fossil Collection

- Освобождение места, занятого сообщениями, расположенными левее GVT

Рекомендуемая литература (1/2)

- Fujimoto Richard M. Parallel discrete event simulation // Commun. ACM. — 1990. — Окт. — Т. 33, No 10. — С. 30–53. http://doi.acm.org/10.1145/84537.84545
- Liu Jason Parallel Discrete-Event Simulation. — 2009. http://www.cis.fiu.edu/~liux/research/papers/pdes-eorms09.pdf
- Lantz Robert E. Parallel SimOS: Performance and Scalability for Large System Simulation. — 2007 http://cs.stanford.edu/~rlantz/papers/lantz-thesis.pdf

Рекомендуемая литература (2/2)

Параллельная симуляция. IDZ

http://software.intel.com/ru-ru/blogs/2013/09/22/0

http://software.intel.com/ru-ru/blogs/2013/09/22/1

http://software.intel.com/ru-ru/blogs/2013/09/22/2

http://software.intel.com/ru-ru/blogs/2013/09/22/3

На следующей лекции:

- Параллельная симуляция
  - Особенности моделирования процессоров
  - Атомарные инструкции
  - Модели консистентности памяти

Спасибо за внимание!

Все материалы курса выкладываются на сайте лаборатории:
http://iscalare.mipt.ru/material/course_materials/

Почему всё так

- Отсутствует порядок исполнения в привычном нам понимании
  - Необходимо балансировать гранулярность критических секций, сложность кода и скорость работы
- Консервативные модели — перестраховка: не все точки синхронизации на самом деле нужны
- Оптимистические модели — надеемся на быстрый «частный случай», в (должно быть) редких исключениях возвращаемся к общей схеме.
  - Гранулярность критических секций большая, но мы можем иметь более одного потока внутри каждой: транзакции.

Транзакции (1/3)

- Блок кода, эффектов от исполнения которого не видно до его (успешного) окончания
- Если по каким-то причинам транзакция прерывается в середине, то её эффектов вообще нет. Как будто она и не начиналась
- Несколько транзакций могут исполняться параллельно, но если они конфликтуют по ресурсам, то выживет только одна — остальные будут отменены

Транзакции (2/3)

- Упрощается параллельное программирование
  - Не приходится мельчить критические секции — даже большие Tx могут быть исполнены эффективно
  - Алгоритмы выглядят проще (Tx комбинируемы, в отличие от блокировок)
- Чисто программная реализация может быть медленной
  - Точки сохранения, откаты
- Неклассическая парадигма программирования
  - Нет общепринятой поддержки в языках

Транзакции (3/3)

- Существуют программные реализации параллельного программирования с использованием Tx (напр. в Java, C++ Boost, Haskell, .NET)
- Аппаратная поддержка встречается реже. Пример: Intel Haswell TSX.
  - **http://www.hwsw.hu/kepek/hirek/2012/09/SF12_ARCS004_100.pdf**

Детерминированность симуляции

- Пересылка сообщений между потоками должна происходить в моменты, определённые заранее
- Схема с доменами отвечает этому условию
- Существуют ли какие-то ещё схемы?

Что можно параллелить в симуляторе

- Модели процессоров
  - Подсистемы модели ЦПУ
- Модели периферии
  - Осмысленно для I/O интенсивных сценариев
    - База данных, пишущая на несколько дисков

Что может облегчить симуляцию?

- Особенности моделируемой памяти
  - Модель когерентности
  - Совпадение моделей у гостя и хозяина
- Особенности моделируемой программы
  - Корректные программы симулировать легче

Модель когерентности памяти (1/3)

- Thread#1
  - Read-Write1-Write2
- Thread#2
  - Read
  - Может ли увидеть результаты Write1?

Модель когерентности памяти (2/3)
Strict consistency

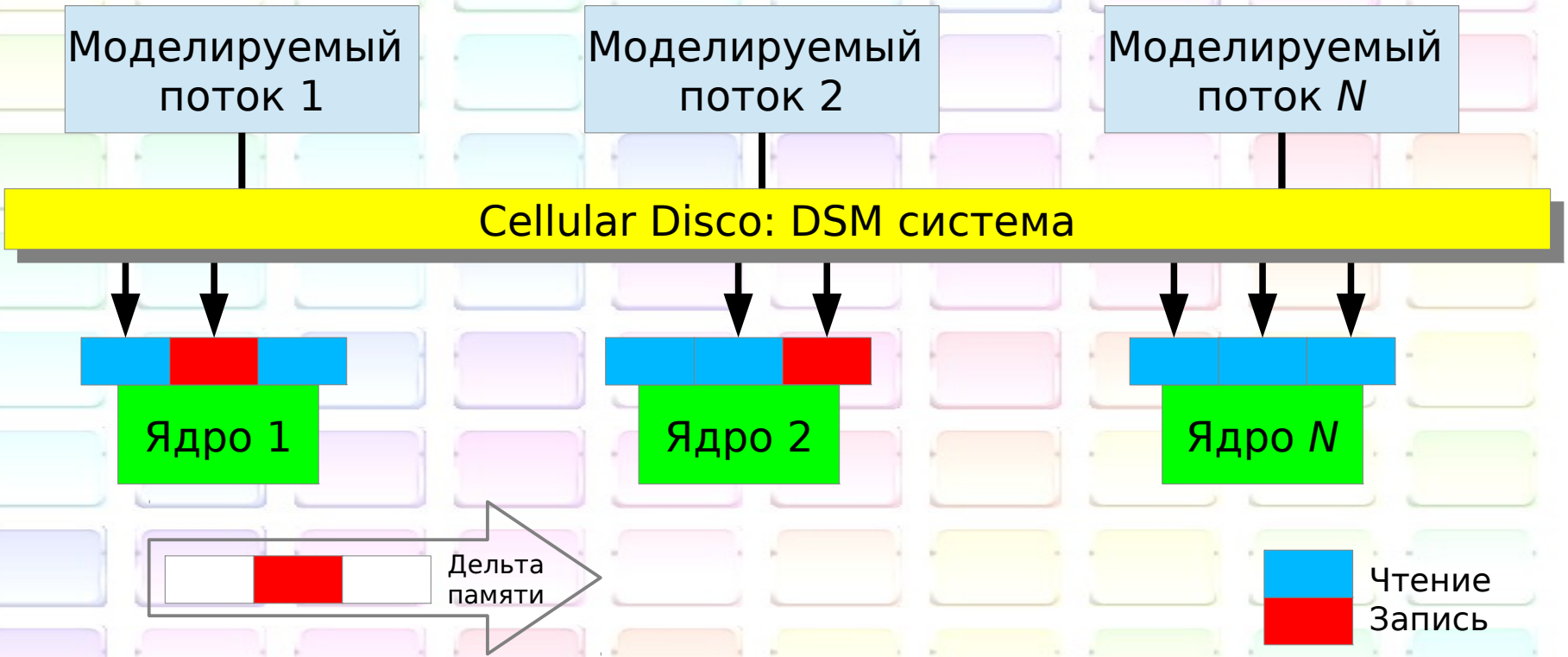
- Thread#1
  - Read-(sync)-Write1-(sync)-Write2-(sync)
- Thread#2
  - После каждой операции новое значение видно всем агентам в системе

Модель когерентности памяти (3/3)

Release consistency

- Thread#1
  - Read-Write1-Write2-Sync
- Thread#2
  - Обновление общего значения только после **явной** синхронизации

Корректность симулируемой программы



Реализация в современных продуктах

- Многие пытались, немногие преуспели
- До сих пор нет универсального решения
- Частные решения существуют

Simics

- Доменная схема синхронизации
  - Минимальная единица параллельного исполнения — материнская плата
- Процессоры внутри исполняются последовательно
- Детерминистичная система

BigSim

- Система поддержки разработки для IBM BlueGene
- Программная модель Charm++ (AMPI)
- Оптимистическая схема

Graphite

- Симулятор уровня приложений IA-32 (Debian Lenny)
- Основан на Pin
- Консервативная схема с пересылкой меток времени
- Distributed shared memory
- Недетерминистичный

Same ISA VM

- VBox, VMWare, Qemu, Xen ...
- Способны исполнять параллельно ядра гостевой системы с помощью DEX
- Архитектура гостя ~ архитектуре хозяина
- Недерминистичны