Современные средства разработки ПО

Модуль 3. Введение в разработку параллельных программ.

Фетисов Михаил Вячеславович
fetisov.michael@bmstu.ru

Разработка параллельных программ
Литература

- С. В. Борзунов, С. Д. Кургалин, А. В. Флегель. Практикум по параллельному программированию: учеб. пособие — СПб.: БХВ, 2017. — 236с. *(часть теории и немного OpenMP с MPI)*
- Энтони Уильямс. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. — М.: ДМК Пресс, 2012. — 672с. *(основная часть)*
- Гримм Р. Параллельное программирование на современном языке C++ — М.: ДМК Пресс, 2022. — 616с. *(для углублённого самостоятельного изучения)*
- Камерон Хьюз, Трейси Хьюз. Параллельное и распределенное программирование с использованием C++ — Вильямс, 2004. — 672с.

Разработка параллельных программ как часть курса «Введение в распределённые вычисления»

- Область ***распределенных вычислений*** (англ. *distributed computing*) представляет собой раздел теории вычислительных систем, изучающий теоретические вопросы организации ***распределенных систем*** (англ. *distributed systems*).
- **Распределенная система** — это такая система, в которой взаимодействие и синхронизация программных компонентов, выполняемых на независимых сетевых компьютерах, осуществляется посредством передачи сообщений.

Начнём с параллельных вычислений

- Параллельные вычисления являются частью распределённых вычислений.
- Мы начнём с параллельных вычислений, т. к. там нам придётся с толкнуться с частью проблем, свойственных и распределённым вычислениям.

Параллельные вычисления
Определение

- **Параллельные вычисления** — это форма вычислений, при которой несколько вычислений выполняются одновременно (в течение перекрывающихся периодов времени), вместо последовательных (с завершением одного до начала следующего).
- **Параллельная система** — это система, в которой вычисления могут выполняться без ожидания завершения других вычислений.

Параллельные вычисления и операционная система

- Современные ОС поддерживают параллелизм как минимум на уровне запускаемых программ.
- Работу программы под управлением ОС мы будем называть процессом.

Процесс и поток
Определения

- **Процесс (операционной системы)** — это выполнение программных инструкций в рамках независимого виртуального адресного пространства, объединяющего код и данные компьютерной программы.
- **Поток (выполнения)** — программный механизм, предоставляемый ОС для параллельного выполнения программных инструкций в адресном пространстве компьютерной программы.

Адресное пространство пользователя Определение

- **Адресное пространство пользователя** (англ. *user space*) — объединение кода и данных выполняемой программы в рамках запущенного в ОС процесса, защищённые от области ядра ОС и других процессов.

Механизм распараллеливания вычислений предоставляется ОС

- При запуске программы, современные ОС создают адресное пространство пользователя, загружают туда программу, и передают управление точке входа в эту программу (функция main в C/C++)
- ОС также предоставляет механизмы последующего распараллеливания выполнения программных инструкций (помимо прочих механизмов доступа к общим ресурсам компьютера).
- Применение этих механизмов мы и будем изучать в данном курсе.
- Но сначала немного абстрактной теории.

Классификация архитектур вычислительных систем
Классификация Флинна

- По Флинну поток определяется как последовательность команд или данных, выполняемых или обрабатываемых процессором.
- В этой модели программа может предоставлять процессору не только поток инструкций для исполнения, но и поток данных для обработки.
- При этом, потоки команд и потоки данных предполагаются независимыми.

Классификация Флинна
Классы вычислительных систем

- **Один поток команд, один поток данных** (Single Instruction stream, Single Data stream; **SISD**). Например: одноядерные IBM PC с DOS.
- **Один поток команд, несколько потоков данных** (Single Instruction stream, Multiple Data stream; **SIMD**). В таких системах одна и та же операция выполняется одновременно над различными данными. Например, векторные вычислительные системы, CUDA, шейдеры.
- **Несколько потоков команд, один поток данных** (Multiple Instruction stream, Single Data stream; **MISD**). Машины MISD могут быть полезны в некоторых узкоспециальных задачах.
- **Несколько потоков команд, несколько потоков данных** (Multiple Instruction stream, Multiple Data stream; **MIMD**). Системы MIMD составляют наиболее обширную и разнообразную группу в классификации Флинна. Большинство современных многопроцессорных вычислительных систем относится именно к этому классу.

Классификация Флинна

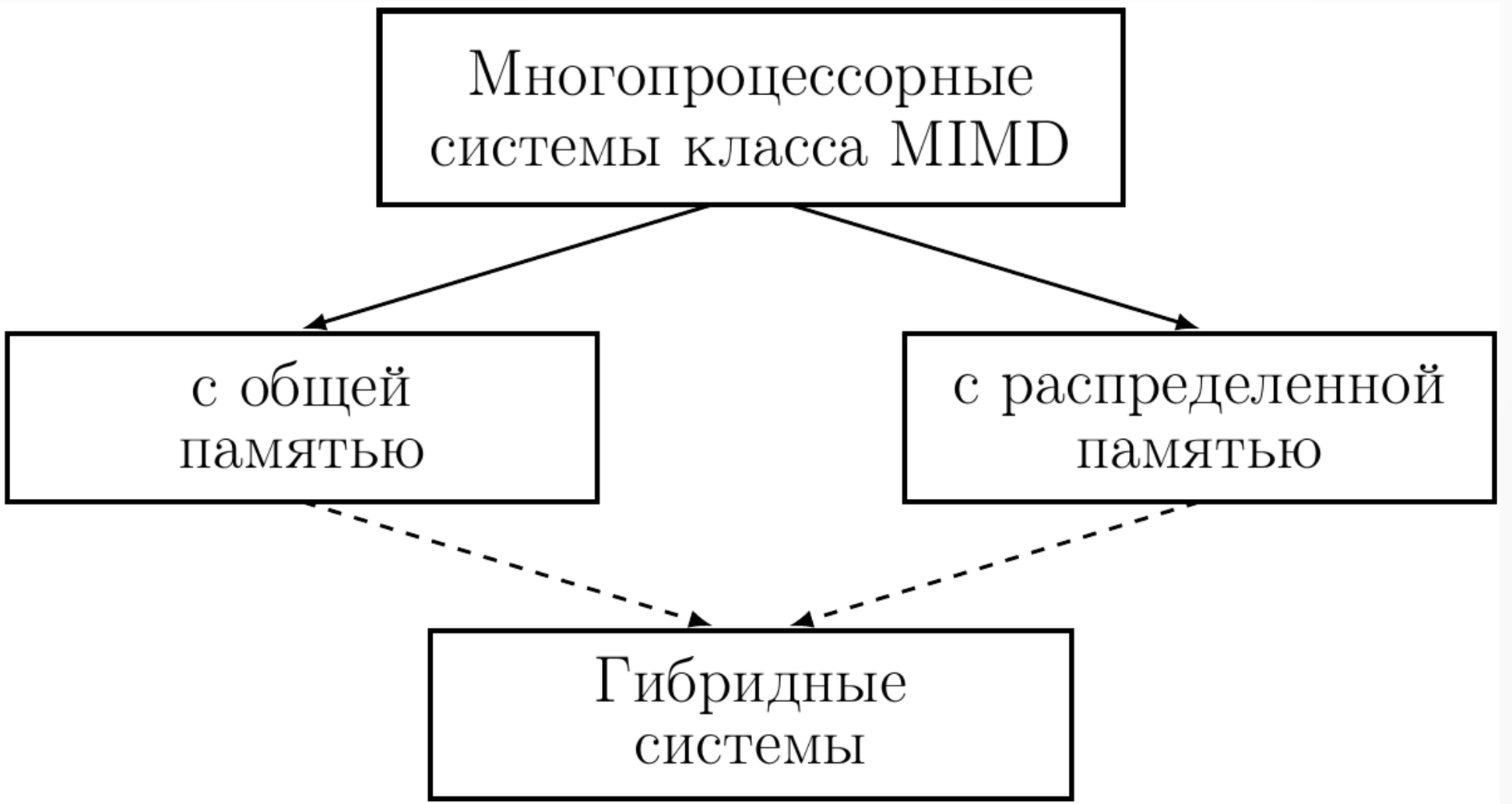
В ячейках записаны примеры арифметических операций, доступных для одновременного выполнения соответствующими системами

		Поток данных	
		один	несколько
Поток команд	один	SISD $a_1 + b_1$	SIMD $a_1 + b_1$ $a_2 + b_2$ $a_3 + b_3$
	несколько	MISD $a_1 + b_1$ $a_1 - b_1$ $a_1 * b_1$	MIMD $a_1 + b_1$ $a_2 - b_2$ $a_3 * b_3$

Уточнение классификации Флинна

- Широко используется уточнение классификации Флинна, согласно которому происходит разделение категории MIMD по способу организации памяти вычислительной системы. Среди MIMD-систем выделяют:
  - **мультипроцессоры** — машины с общей памятью (Uniform Memory Access, UMA), и
  - **мультикомпьютеры** — машины, не обладающие удаленным доступом к памяти (NO Remote Memory Access, NORMA).
- Взаимодействие между процессорами в мультикомпьютерах осуществляется с помощью механизма передачи сообщений.

Классификация многопроцессорных вычислительных систем класса MIMD по принципу организации памяти



MIMD по принципу организации памяти Модели

- Вопросы одновременного доступа нескольких вычислительных узлов к участкам общей памяти занимают центральное место в модели и решаются с помощью механизмов синхронизации, к которым относят барьеры, замки, семафоры и др.
- В системах с распределенной памятью каждый вычислительный узел имеет доступ только к принадлежащему ему участку памяти — локальной памяти. В этом случае для межпроцессорного обмена данными предусматривается возможность отправки и приема сообщений по коммуникационной сети, объединяющей вычислительную систему. Соответствующая модель программирования носит название **модели передачи сообщений**.
- Необходимо отметить, что программирование гибридных систем основано на использовании вышеперечисленных моделей или их комбинаций.

MIMD по принципу организации памяти Примеры

- Укажите примеры известных программ:
  - с общей памятью (UMA);
  - с распределённой памятью (NORMA);
  - гибридные системы.
- Для двух последних укажите **модель передачи сообщений**.

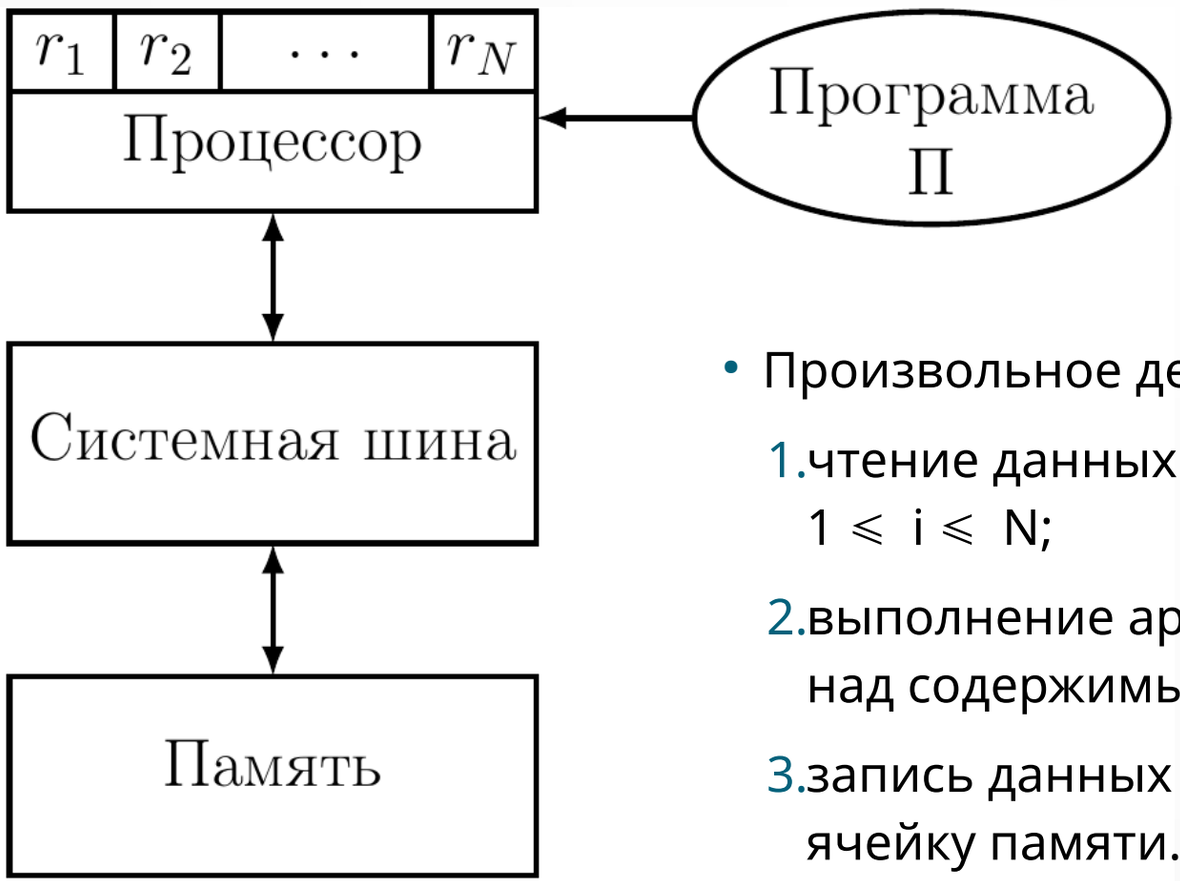
Средства параллельного программирования

- Сформулированы несколько подходов к использованию ресурсов параллелизма предоставляемых ОС в разрабатываемой программе:
  - автоматическое распараллеливание последовательной версии программы средствами компилятора;
  - использование специализированных языков для параллельного программирования (см. Erlang);
  - использование библиотек, предоставляющих возможности параллельного исполнения кода;
  - программирование с использованием особых расширений языка — средств распараллеливания.

Модели RAM и PRAM

- **Машина с произвольным доступом к памяти** (Random Access Machine, **RAM**). Перечислим основные свойства RAM:
- Система, состоит из процессора, устройства доступа к памяти (системной шины) и памяти, состоящей из конечного числа ячеек (см. рисунок).
- Процессор последовательно выполняет команды, заложенные в программе П; ему доступны основные арифметические и логические операции и чтение/запись данных в памяти. При этом постулируется, что каждая команда выполняется за фиксированное время.

Модель RAM

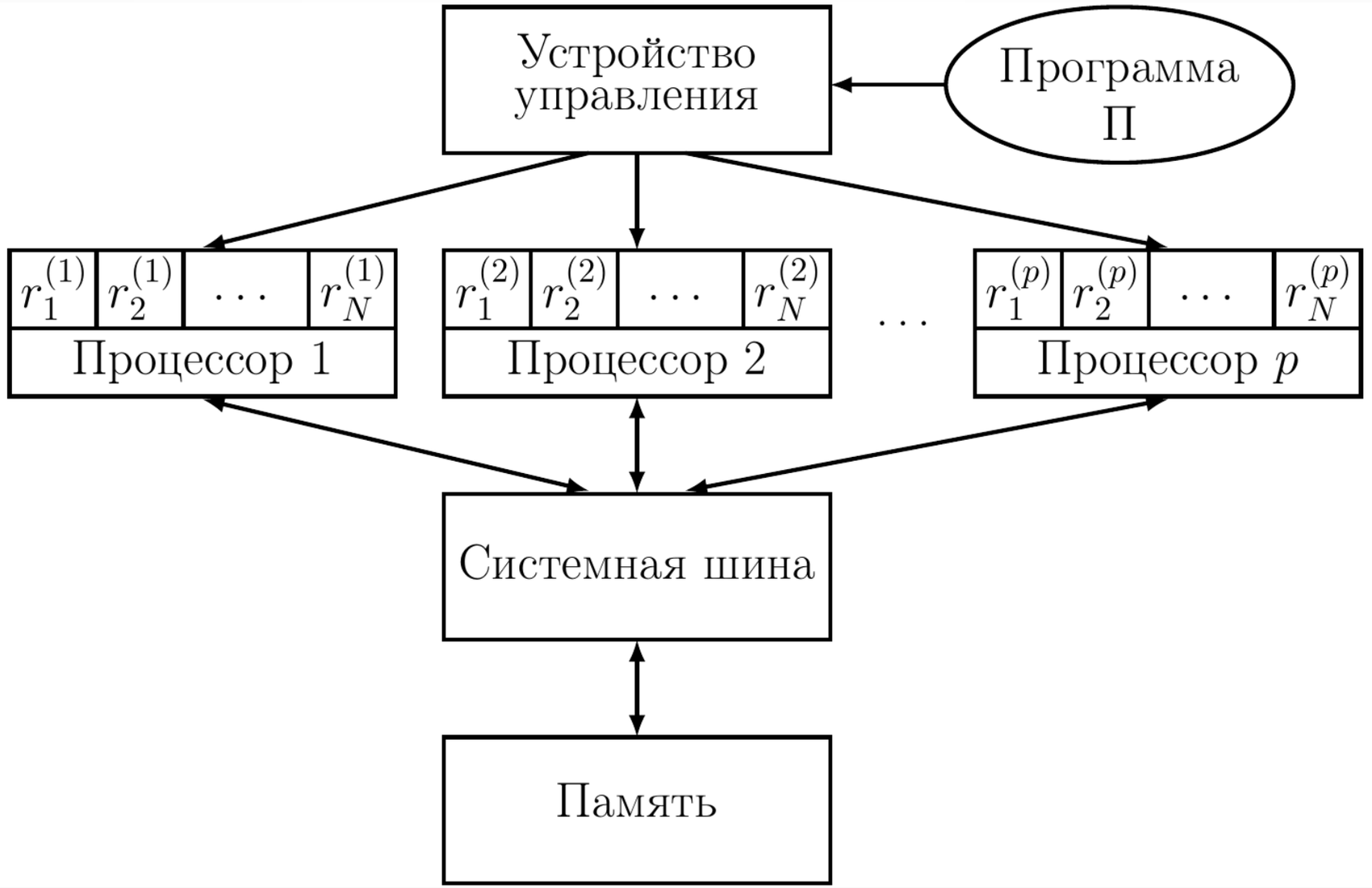


- Произвольное действие процессора состоит из трех этапов:
  1. чтение данных из памяти в один из своих регистров r_i, где $1 \leqslant i \leqslant N$;
  2. выполнение арифметической или логической операции над содержимым своих регистров;
  3. запись данных из регистра r_j, где $1 \leqslant j \leqslant N$, в некоторую ячейку памяти.
- Считается, что исполнение трех перечисленных шагов требует времени $\Theta(1)$.

Модель PRAM

- Одна из самых распространенных моделей параллельных компьютерных систем — **параллельная машина с произвольным доступом к памяти** (Parallel Random Access Machine, **PRAM**).
- В PRAM объединены *p* процессоров, общая память и устройство управления, которое передает команды программы П процессорам.

Модель PRAM
Схема



Модель PRAM
Конфликты доступа

- Важной особенностью PRAM является ограниченное время доступа любого из процессоров системы к произвольной ячейке памяти. Как и в случае RAM, шаг алгоритма здесь также соответствует трем действиям процессора. Как было отмечено выше, любой шаг алгоритма выполняется за время $\Theta(1)$.
- Одновременное обращение двух и более процессоров к одной и той же ячейке памяти приводит к **конфликтам доступа**.
- Они подразделяются на **конфликты чтения** и **конфликты записи**.

Модель PRAM
Конфликты чтения

- Если несколько процессоров пытаются прочесть данные из одной ячейки, то возможны два варианта дальнейших действий:

1. **Исключающее чтение** (Exclusive Read, **ER**). В данный момент времени чтение разрешено только одному процессору, в противном случае происходит ошибка выполнения программы.
2. **Одновременное чтение** (Concurrent Read, **CR**). Количество процессоров, получающих доступ к одной ячейке памяти, не ограничивается.

Модель PRAM
Конфликты записи

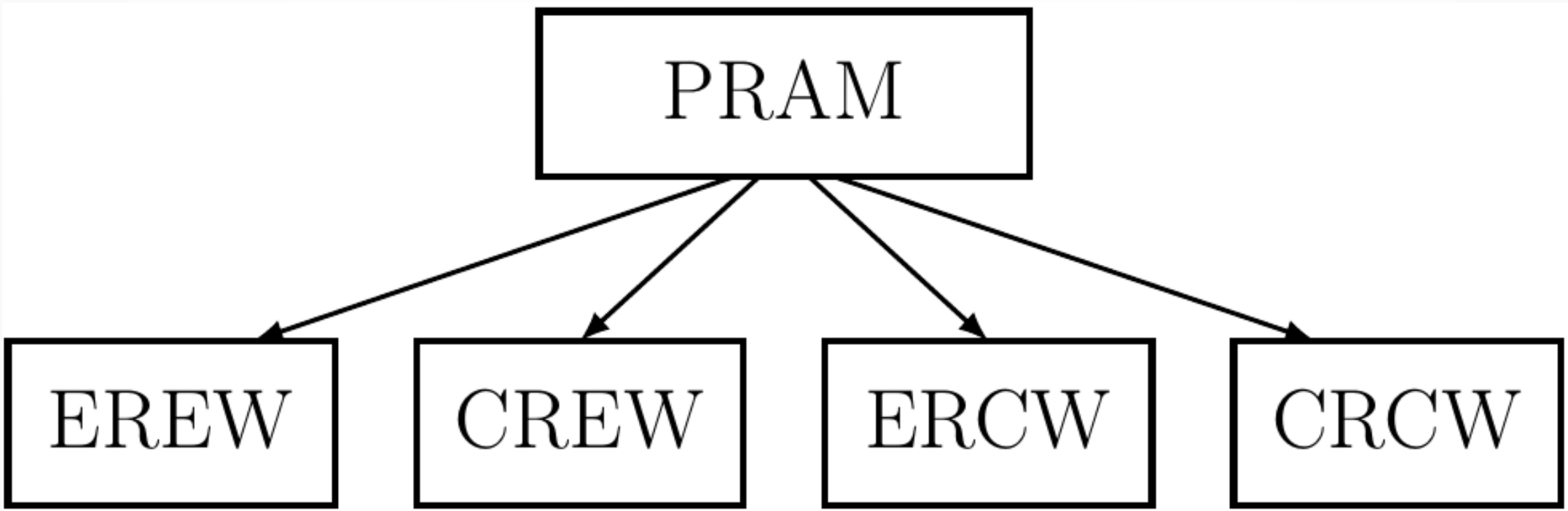
- Если более одного процессора пытаются записать данные по одному адресу, разделяют два способа действия.

1. **Исключающая запись** (Exclusive Write, **EW**). Только одному процессору разрешено осуществлять запись в данную ячейку в конкретный момент времени.
2. **Одновременная запись** (Concurrent Write, **CW**). Несколько процессоров сразу получают доступ для записи к одной ячейке памяти.

Модель PRAM
Правила записи для случая CW

- **Запись общего значения**. Предполагается, что все процессоры, готовые производить запись в одну и ту же ячейку памяти, обязаны записывать только общее для всех них значение, иначе инструкция записи в отведенную область памяти считается ошибочной.
- **Произвольный выбор**. Процессор, осуществляющий запись, выбирается случайным образом.
- **Запись с учетом приоритетов**. Каждому из конкурирующих процессоров присваивается некоторый приоритет, например, его порядковый номер, при этом сохраняется только то значение, которое поступило от процессора с заранее определенным приоритетом (например, наименьшим).
- **Комбинированный выбор**. Все процессы выдают значения для записи, из них по определенному правилу формируется результат (например, сумма значений, максимальное значение и др.), который и записывается.

Модель PRAM
Методы разрешения конфликтов



Модель PRAM
Проблемы при разрешения конфликтов

- Таким образом, системы EREW обладают существенными ограничениями, налагаемыми на работу с ячейками памяти.
- С другой стороны, системы CREW, ERCW, CRCW с большим количеством процессоров трудно построить по техническим причинам, поскольку количество вычислительных ядер, одновременно получающих доступ к некоторому участку памяти, ограничено.
- Однако существует важный и несколько неожиданный результат, позволяющий моделировать работу CRCW-машины на системе, построенной в соответствии с принципом EREW.
- Он представлен в теореме об эмуляции.

Модель PRAM
Теорема об эмуляции

- *Пусть алгоритм для CRCW-машины решает некоторую задачу с параметром размера N за время T(N), используя p процессоров. Тогда существует алгоритм для той же задачи на EREW-системе с p процессорами, который может быть исполнен за время $O(T(N)\log_2 N)$.*
- При этом объем памяти PRAM должен быть увеличен в O(p) раз.

Модель PRAM
Ускорение вычислений

- Ускорение $S_p(N)$, получаемое при использовании параллельного алгоритма на машине с p процессорами, равно

$$S_p(N) = \frac{T_1(N)}{T_p(N)}$$

- Это мера прироста производительности по сравнению с наилучшим последовательным алгоритмом. Чем больше ускорение, тем больше отличается время решения задачи на многопроцессорной системе от продолжительности работы алгоритма на системе с одним процессором.

Модель PRAM
Закон Амдала

- Пусть f — доля последовательных вычислений в алгоритме A. Тогда ускорение S_p при использовании A на системе из p процессоров удовлетворяет неравенству

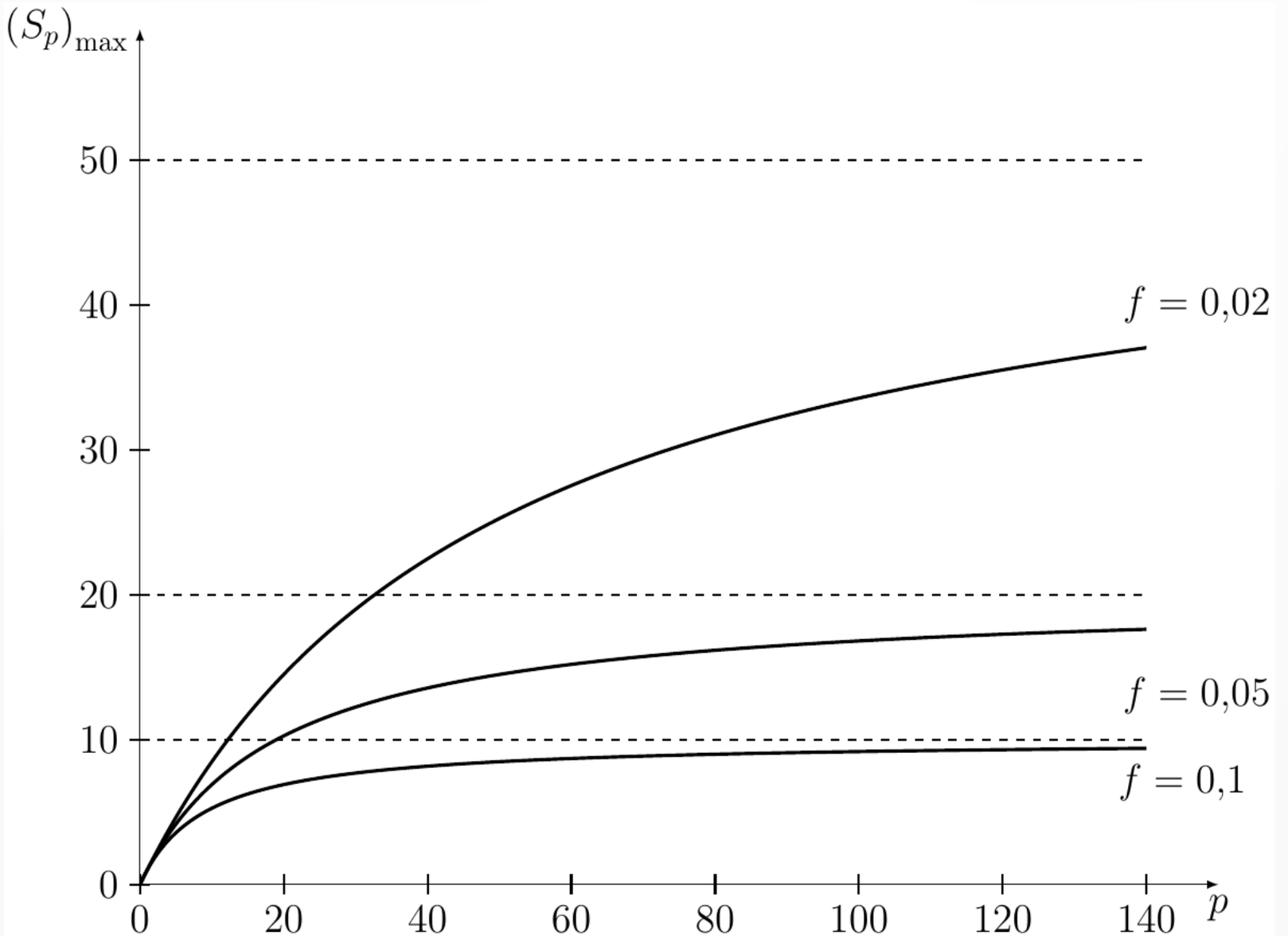
$$S_p \leqslant \frac{1}{f + (1 - f)/p}$$

Модель PRAM
Закон Амдала (некоторые выводы)

- Неравенство $S_p \leqslant (S_p)_{\max}$
- показывает, что существование последовательных вычислений, которые не могут быть распараллелены, накладывает ограничение на S_p. Даже при использовании компьютера с p = ∞ ускорение не превысит величины

$$S_\infty = \frac{1}{f}$$

Зависимости значений максимального ускорения $(S_p)_{max}$ от числа процессоров p при разных f-долях последовательных вычислений



Модель PRAM
Закон Амдала (некоторые выводы 2)

- Эмпирически установлено, что для широкого класса вычислительных задач доля последовательных вычислений убывает с ростом размера входных данных задачи.
- Поэтому на практике ускорение может быть увеличено за счет увеличения вычислительной сложности решаемой задачи.

Вопрос

- Вы проходили технологию OpenMP?

Вопросы?

Фетисов Михаил Вячеславович
fetisov.michael@bmstu.ru