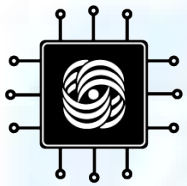
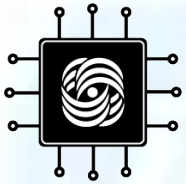


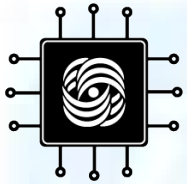
АРХИТЕКТУРА КОМПЬЮТЕРНЫХ СИСТЕМ

Лекция 13:
Параллельные вычисления



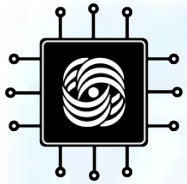
План лекции

- Показатели эффективности параллельного алгоритма
- Оценка максимально достижимого параллелизма
- Анализ масштабируемости параллельных вычислений
- Методика распараллеливания вычислительных алгоритмов



Введение

- Принципиальный момент при разработке параллельных алгоритмов - **анализ эффективности использования параллелизма**:
  - *Оценка эффективности распараллеливания конкретных выбранных методов выполнения вычислений,*
  - Оценка максимально возможного ускорения процесса решения рассматриваемой задачи (анализ всех возможных способов выполнения вычислений) Показатели эффективности параллельного алгоритма



Показатели эффективности (Ускорение)

Ускорение получаемое при использовании параллельного алгоритма для ***p*** процессоров, по сравнению с последовательным вариантом выполнения вычислений, определяется величиной

$$S_p(n) = T_1(n) / T_p(n)$$

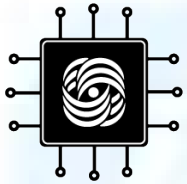
(величина ***n*** используется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи)

Показатели эффективности (Эффективность)

Эффективность использования параллельным алгоритмом процессоров при решении задачи определяется соотношением:

$$E_p(n) = T_1(n)/(pT_p(n)) = S_p(n)/p$$

(величина эффективности определяет среднюю долю времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи)

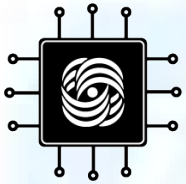


Сверхлинейное ускорение

Сверхлинейное (*superlinear*) ускорение $S_p(n)>p$ может иметь место в силу следующего ряда причин:

- неравноправность выполнения последовательной и параллельной программ (например, недостаток оперативной памяти)
- нелинейный характер зависимости сложности решения задачи от объема обрабатываемых данных
- различие вычислительных схем последовательного и параллельного методов

Показатели качества параллельных вычислений являются противоречивыми: попытки повышения качества параллельных вычислений по одному из показателей (ускорению или эффективности) может привести к ухудшению ситуации по другому показателю



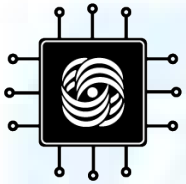
Оценка максимально достижимого параллелизма

- Оценка качества параллельных вычислений предполагает знание *наилучших* (*максимально достижимых*) значений показателей ускорения и эффективности
- Получение идеальных величин $S_p=p$ для ускорения и $E_p=1$ для эффективности может быть обеспечено не для всех вычислительно трудоемких задач

Закон Амдаля

- Достижению максимального ускорения может препятствовать существование в выполняемых вычислениях последовательных расчетов, которые не могут быть распараллелены.
- Пусть ***f*** есть *доля последовательных вычислений* в применяемом алгоритме обработки данных.
- Ускорение процесса вычислений при использовании ***p*** процессоров ограничивается величиной:

$$S_p \le \frac{1}{f + (1 - f)/p} \le S^* = \frac{1}{f}$$



Закон Амдаля (Замечания)

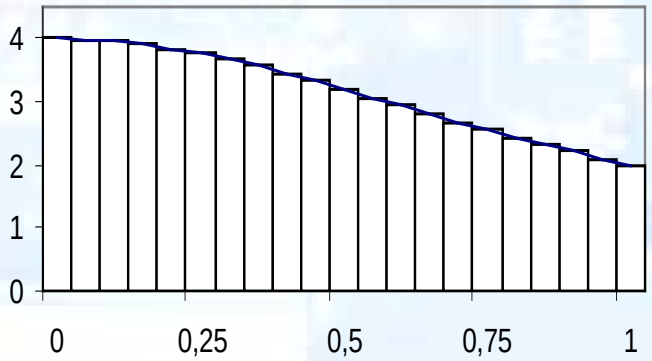
- Доля последовательных вычислений может быть существенно снижена при выборе более подходящих для распараллеливания методов
- Для большого ряда задач доля последовательных вычислений $f=f(n)$ является убывающей функцией от n, и в этом случае ускорение для фиксированного числа процессоров может быть увеличено за счет увеличения вычислительной сложности решаемой задачи.
- В этом случае, ускорение $Sp= Sp(n)$ является возрастающей функцией от параметра n.

Вычисление числа π...

- Значение числа π может быть получено при помощи интеграла

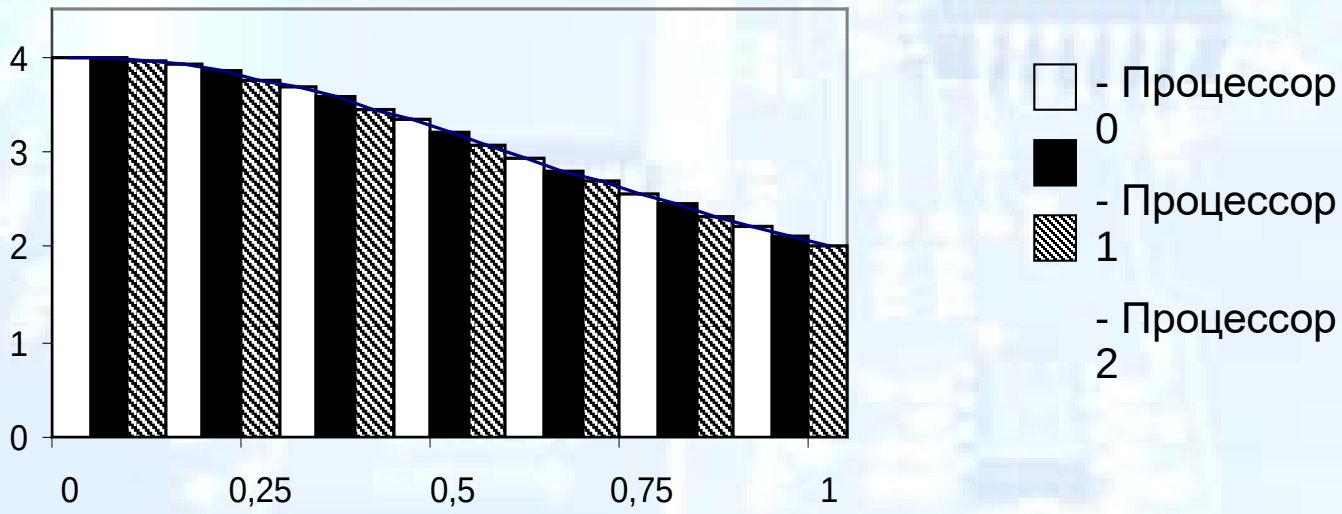
$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

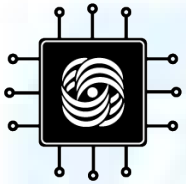
- Для численного интегрирования применим метод прямоугольников



Вычисление числа π…

- Распределим вычисления между ***p*** процессорами (циклическая схема)
- Получаемые на отдельных процессорах частные суммы должны быть просуммированы





Вычисление числа π...

Анализ эффективности…

- n – количество разбиений отрезка [0,1]
- Вычислительная сложность задачи

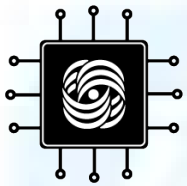
$$W = T_1 = 6n$$

- Количество узлов сетки на отдельном процессоре

$$m = \lceil n/p \rceil \leq n/p + 1$$

- Объем вычислений на отдельном процессоре

$$W_p = 6m = 6n/p + 6.$$



Вычисление числа π...

Анализ эффективности

- Время параллельного решения задачи

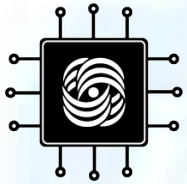
$$T_p = 6n/p + 6 + \log_2 p$$

- Ускорение

$$Sp = T_1 / T_p = 6n / (6n/p + 6 + \log_2 p)$$

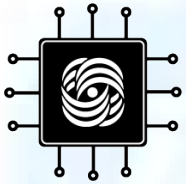
- Эффективность

$$Ep = 6n / (6n + 6p + p\log_2 p)$$



Определение эффективности параллельных вычислений

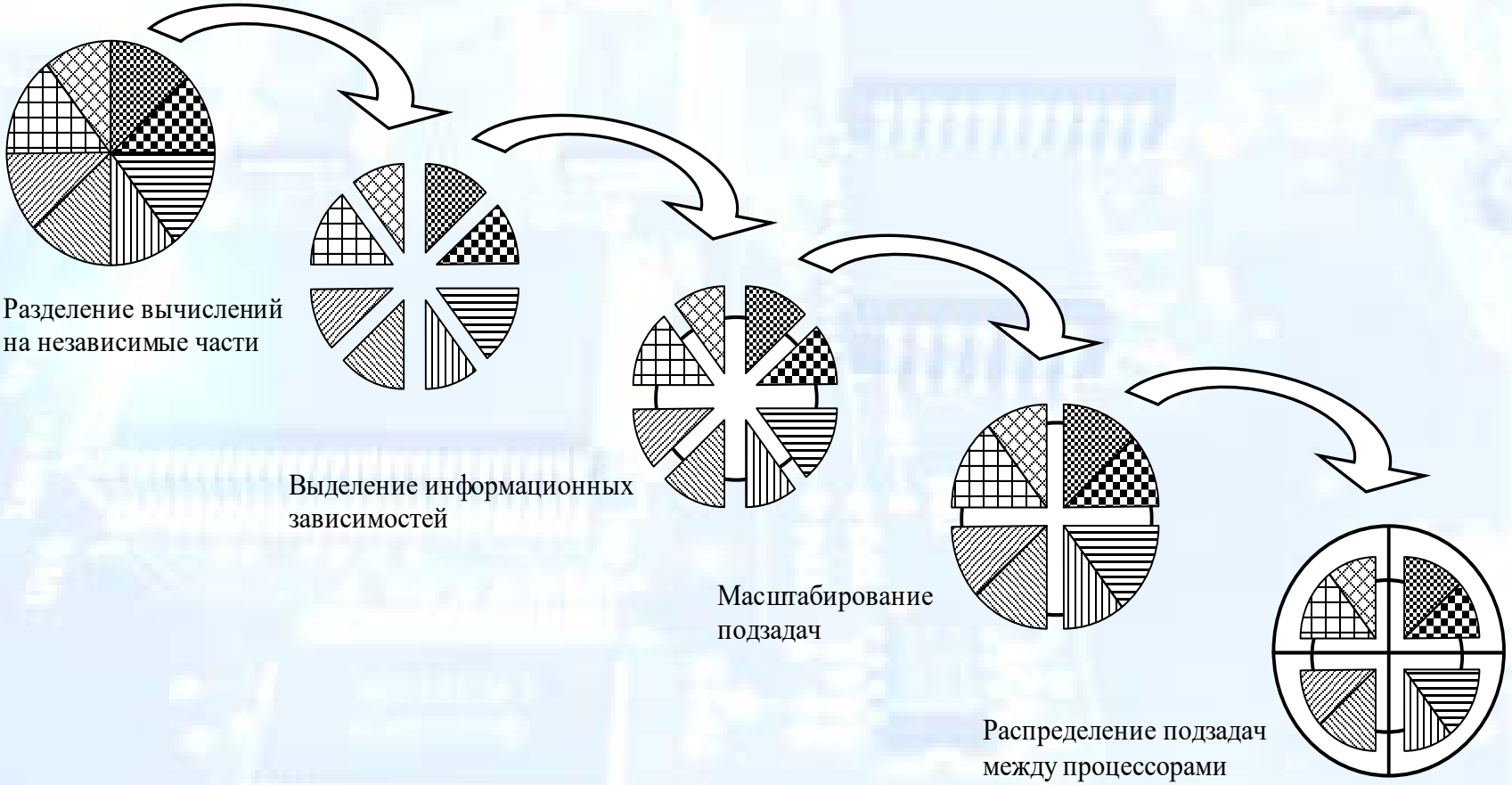
- Выполнение анализа имеющихся вычислительных схем и декомпозиция их на подзадачи, которые могут быть реализованы независимо друг от друга
- Выделение для набора подзадач *информационных взаимодействий*
- Определение необходимой (или доступной) для решения задачи *вычислительной системы* и выполнение *распределения* имеющего набора подзадач между процессорами системы.

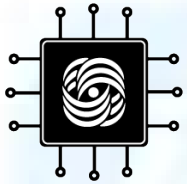


Основные критерии

- Объем вычислений для каждого процессора должен быть примерно одинаков – это позволит обеспечить равномерную вычислительную загрузку (*балансировку*) процессоров
- Распределение подзадач между процессорами должно быть выполнено таким образом, чтобы наличие информационных связей (*коммуникационных взаимодействий*) между подзадачами было минимальным

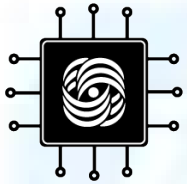
Схема разработки параллельных алгоритмов





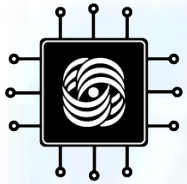
Разработка параллельных алгоритмов

- После выполнения всех этапов проектирования необходимо оценить эффективность разрабатываемых параллельных методов
- По результатам проведенного анализа может оказаться необходимым повторение некоторых (или всех) этапов разработки:
  - корректировка состава сформированного множества задач - подзадачи могут быть укрупнены (*агрегированы*) или *детализированы*. Данные действия могут быть определены как *масштабирование* разрабатываемого алгоритма



Моделирование параллельных программ (1)

- На стадии проектирования параллельный метод может быть представлен в виде ***графа "подзадачи – сообщения"*** (*агрегированное* представление графа *"операции-операнды"*)
- Модель "подзадачи - сообщения" позволяет:
  - Определить подзадачи одинаковой вычислительной сложности,
  - Обеспечить низкий уровень информационной зависимости между подзадачами

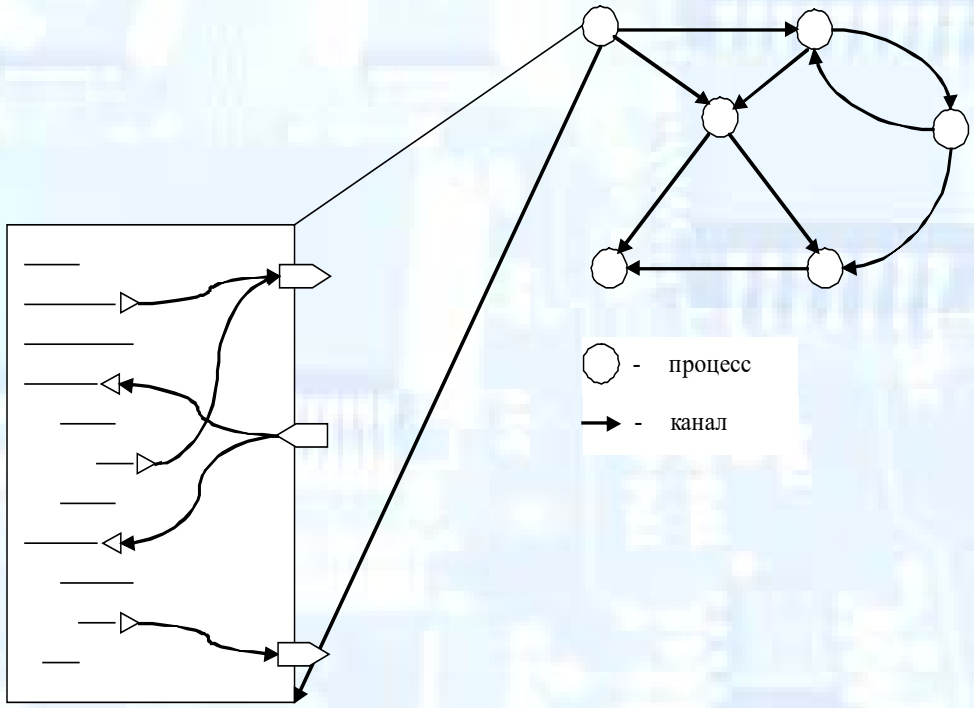


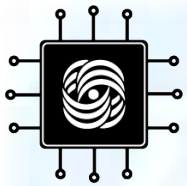
Моделирование параллельных программ (2)

- На стадии выполнения для описания параллельной программы может быть использована модель в виде ***графа "процессы – каналы"***, в которой вместо подзадач используется понятие процессов, вместо информационных зависимостей - каналы передачи сообщений)
- Модель *"процессы – каналы"* позволяет:
  - Осуществить оптимальное распределение подзадач по процессорам,
  - Выполнить анализ эффективности разработанного параллельного метода,
  - Обеспечить возможность контроля и управления процессом выполнения параллельных вычислений

Моделирование параллельных программ (3)

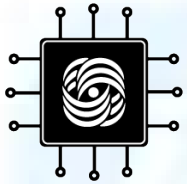
- Модель параллельной программы в виде графа "процессы-каналы"...





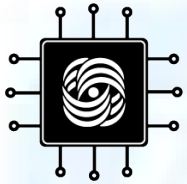
Модель "процессы-каналы"

- *Процесс* - выполняемая на процессоре *программа*, использует для свой работы часть локальной *памяти* процессора и содержит ряд *операций приема/передачи* данных для организации информационного взаимодействия между выполняемыми процессами параллельной программы



Модель "процессы-каналы"

- *Канал передачи данных* - *очередь сообщений*, в которую один или несколько процессов могут отправлять пересылаемые данные и из которой процесс-адресат может извлекать сообщения:
  - каналы возникают динамически в момент выполнения первой операции приема/передачи с каналом
  - канал может соответствовать одной или нескольким командам приема/передачи данных различных процессов
  - емкость канала неограничена
  - операции приема сообщений могут приводить к *блокировкам* (запрашиваемые данные еще не были отправлены процессами-источниками)



Методика разработки параллельных алгоритмов (1)

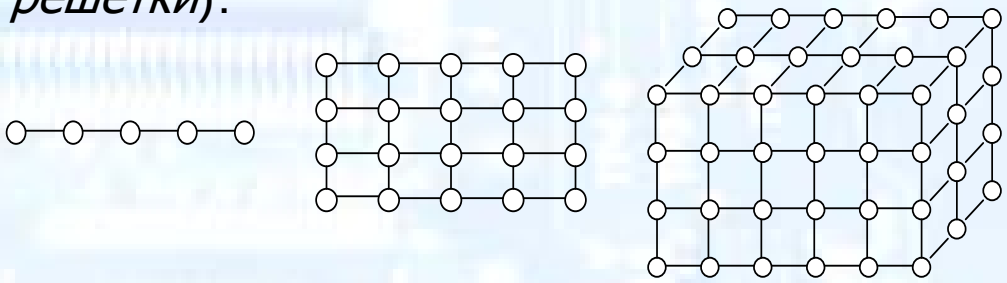
- Для разработки параллельных алгоритмов необходимо выполнить:
  - выделение подзадач,
  - определение информационных зависимостей,
  - масштабирование,
  - распределения подзадач по процессорам вычислительной системы
- Для демонстрации рекомендаций будем использовать задачу поиска максимального значения среди элементов матрицы ***A***:

$$y = \max_{1 \le i, j \le N} a_{ij}$$

Разделение вычислений на независимые части

Типовые вычислительные схемы:

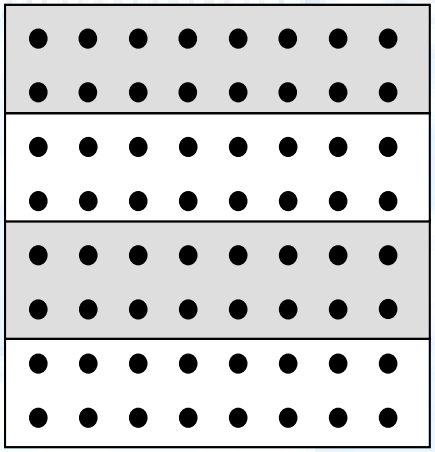
- выполнение однотипной обработки большого набора данных - ***параллелизм по данным***. В этом случае выделение подзадач сводится к разделению имеющихся данных.
  - Для большого количества решаемых задач разделение вычислений по данным приводит к порождению одно-, двух- и трех- мерных наборов подзадач, для которых информационные связи существуют только между ближайшими соседями (*сетки* или *решетки*):



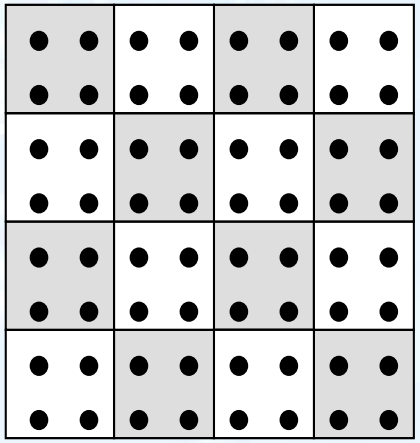
- выполнение разных операций над одним и тем же набором данных - ***функциональный параллелизм***.

Разделение вычислений на независимые части

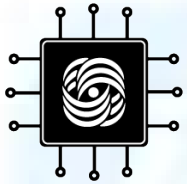
- **Пример**: нахождение максимального элемента матрицы
  - исходная матрица ***A*** может быть разделена на отдельные строки (или последовательные группы строк) – *ленточная схема* разделения данных
  - прямоугольные наборы элементов – *блочная схема* разделения данных



а)

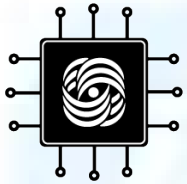


б)



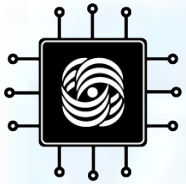
Разделение вычислений на независимые части (2)

- Уровень декомпозиции вычислений:
  - Формирование максимально возможного количества подзадач*:*
    - Обеспечивает использование предельно допустимого параллелизма,
    - Затрудняет анализ параллельных вычислений.
  - Использование достаточно крупных подзадач:
    - Приводит к ясной схеме параллельных вычислений,
    - Затрудняет эффективное использование большого числа процессоров.
  - Промежуточный уровень - использование в качестве элементов декомпозиции только тех подзадач, для которых методы параллельных вычислений известны. Выбираемые подзадачи при таком подходе будем именовать далее *базовыми*, которые могут быть *элементарными* (не допускают дальнейшего разделения) или *составными.*



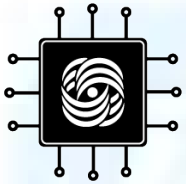
Разделение вычислений на независимые части (3)

- Для оценки корректности этапа разделения вычислений на независимые части можно воспользоваться контрольным списком вопросов:
  - Не увеличивает ли выполненная декомпозиция объем вычислений и необходимый объем памяти?
  - Возможна ли при выбранном способе декомпозиции равномерная загрузка всех имеющихся процессоров?
  - Достаточно ли выделенных частей процесса вычислений для эффективной загрузки имеющихся процессоров (с учетом возможности увеличения их количества)?



Выделение информационных зависимостей

- *Локальные и глобальные схемы* передачи данных
  - для локальных схем в каждый момент передачи данных выполняются только между небольшим числом подзадач (располагаемых, как правило, на соседних процессорах),
  - для глобальных операций передачи данных в процессе коммуникации принимают участие все подзадачи
- *Структурные и произвольные способы взаимодействия* –
  - для структурных способов организация взаимодействий приводит к формированию некоторых стандартных схем коммуникации (например, в виде кольца, прямоугольной решетки и т.д.),
  - для произвольных структур взаимодействия схема выполняемых операций передач данных не носит характер однородности

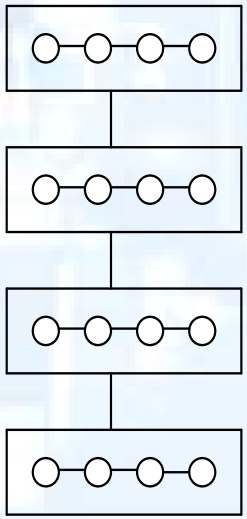


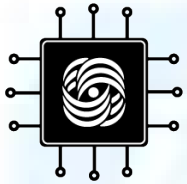
Выделение информационных зависимостей

- *Статические или динамические схемы* передачи данных –
  - для статических схем моменты и участники информационного взаимодействия фиксируются на этапах проектирования и разработки параллельных программ,
  - для динамического варианта взаимодействия структура операции передачи данных определяется в ходе выполняемых вычислений
- *Синхронные и асинхронные способы взаимодействия* –
  - для синхронных способов операции передачи данных выполняются только при готовности всех участников взаимодействия и завершаются только после полного окончания всех коммуникационных действий,
  - при асинхронном выполнении операций участники взаимодействия могут не дожидаться полного завершения действий по передаче данных

Выделение информационных зависимостей

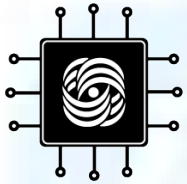
- **Пример**: нахождение максимального элемента матрицы
  - Достаточный уровень декомпозиции может состоять в разделении матрицы *A* на множество отдельных строк и получении на этой основе набора *подзадач поиска максимальных значений в отдельных строках*,
  - Структура информационных связей имеет вид:





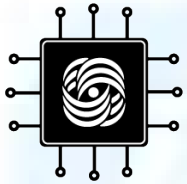
Выделение информационных зависимостей (2)

- Для оценки правильности этапа выделения информационных зависимостей можно воспользоваться контрольным списком вопросов:
  - Соответствует ли вычислительная сложность подзадач интенсивности их информационных взаимодействий?
  - Является ли одинаковой интенсивность информационных взаимодействий для разных подзадач?
  - Является ли схема информационного взаимодействия локальной?
  - Не препятствует ли выявленная информационная зависимость параллельному решению подзадач?



Масштабирование набора подзадач

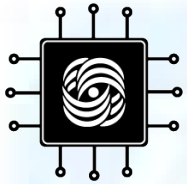
- Масштабирование проводится в случае, если количество имеющихся подзадач не совпадает с числом доступных процессоров
- Для сокращения количества подзадач необходимо выполнить укрупнение (*агрегацию*) вычислений:
  - подзадачи должны иметь одинаковую вычислительную сложность, а объем и интенсивность информационных взаимодействий между подзадачами должны быть минимально возможными,
  - первыми претендентами на объединение являются подзадачи с высокой степенью информационной взаимозависимости
- При недостаточном количестве подзадач для загрузки всех доступных к использованию процессоров необходимо выполнить *декомпозицию* вычислений



Масштабирование набора подзадач (2)

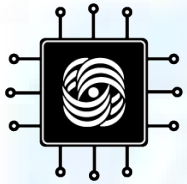
Список контрольных вопросов для оценки правильности этапа масштабирования:

- Не ухудшится ли локальность вычислений после масштабирования имеющегося набора подзадач?
- Имеют ли подзадачи после масштабирования одинаковую вычислительную и коммуникационную сложность?
- Соответствует ли количество задач числу имеющихся процессоров?
- Зависят ли параметрически правила масштабирования от количества процессоров?



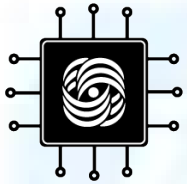
Масштабирование набора подзадач

- Управление распределением нагрузки для процессоров необходима только для вычислительных систем с распределенной памятью. Для мультипроцессоров распределение вычислительной нагрузки между процессорами обычно выполняется операционной системой автоматически
- Этап распределения подзадач между процессорами является избыточным, если количество подзадач совпадает с числом имеющихся процессоров, а топология сети передачи данных вычислительной системы представляет собой полный граф



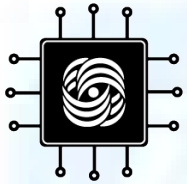
Распределение подзадач между процессорами

- *Эффективность использования процессоров* - относительная доля времени, в течение которого процессоры использовались для вычислений, связанных с решением исходной задачи
- Пути достижения хороших показателей эффективности:
  - равномерное распределение вычислительной нагрузки между процессорами,
  - минимальное количество сообщений, передаваемых между процессорами.
- Оптимальное решение проблемы распределения подзадач между процессорами основывается на анализе информационной связности графа "подзадачи - сообщения"



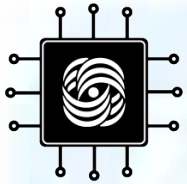
Распределение подзадач между процессорами (2)

- *Динамическое распределение* вычислительной нагрузки необходимо в ситуациях, когда количество подзадач может изменяться в ходе вычислений.
- Одна из часто используемых схем - *схема "менеджер - исполнитель"* (*manager-worker scheme*):
  - Для управления распределением нагрузки в системе выделяется отдельный процессор-менеджер, которому доступна информация обо всех имеющихся подзадачах
  - Остальные процессоры системы являются исполнителями, которые для получения вычислительной нагрузки обращаются к процессору-менеджеру
  - Порождаемые в ходе вычислений новые подзадачи передаются обратно процессору-менеджеру и могут быть получены для решения при последующих обращениях процессоров-исполнителей
  - Завершение вычислений происходит в момент, когда процессоры-исполнители завершили решение всех переданных им подзадач, а процессор-менеджер не имеет каких-либо вычислительных работ для выполнения



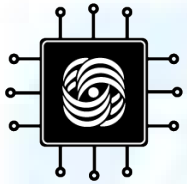
Распределение подзадач между процессорами (3)

- Перечень контрольных вопросов для проверки этапа распределения подзадач состоит в следующем:
  - Не приводит ли распределение нескольких задач на один и тот же процессор к росту дополнительных вычислительных затрат?
  - Существует ли необходимость динамической балансировки вычислений?
  - Не является ли процессор-менеджер "узким" местом при использовании схемы "менеджер-исполнитель"?



Пример применения методики: *Гравитационная задача N тел*

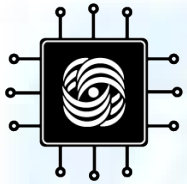
- ***Гравитационная задача N тел*** (или просто *задача N тел*), как и многие задачи в области физики, сводится к операциям обработки данных для каждой пары объектов имеющейся физической системы:
  - Дано большое количество тел (планет, звезд и т.д.), для каждого из которых известна масса, начальное положение и скорость
  - Под действием гравитации положение тел меняется
  - Требуемое решение задачи состоит в моделировании динамики изменения системы *N* тел на протяжении некоторого интервала времени



Гравитационная задача N тел (2)

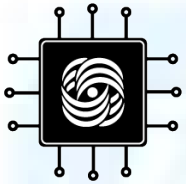
- Для проведения моделирования интервал времени разбивается на временные отрезки небольшой длительности, на каждом шаге моделирования вычисляются силы, действующие на каждое тело, и обновляются скорости и положения тел
- Очевидный алгоритм решения задачи *N* тел состоит в рассмотрении на каждом шаге моделирования всех пар объектов физической системы и выполнении для каждой получаемой пары всех необходимых расчетов
- При таком подходе время выполнения одной итерации моделирования (*т.е.* время перевычисления параметров одной пары тел):

$$T_1 = \tau N(N-1)/2$$



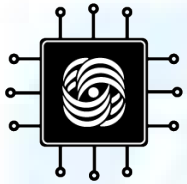
Гравитационная задача N тел (3)

- **Разделение вычислений на независимые части**
  - *Базовая подзадача* - весь набор вычислений, связанных с обработкой данных одного какого-либо тела физической системы
- **Выделение информационных зависимостей**
  - Выполнение вычислений в подзадаче возможно только в случае, когда имеются данные обо всех телах физической системы,
  - Перед началом каждой итерации моделирования каждая подзадача должна получить все необходимые сведения от всех других подзадач системы,
  - Такая процедура передачи данных именуется *операцией обобщенного сбора данных* (*multi-node gather or all gather*).



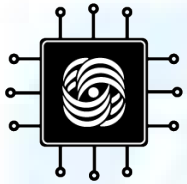
Гравитационная задача N тел (4)

- **Выделение информационных зависимостей. Операция обобщенного сбора данных.**
  - **Метод 1**: Обмен данными осуществляется в ходе последовательности шагов, на каждом из которых все имеющиеся подзадачи разбиваются попарно и обмен данными осуществляется между подзадачами образовавшихся пар (*N-1* итерация)
  - **Метод 2**: Первый шаг метода выполняется точно также, как в методе 1 - после выполнения этого шага подзадачи будут содержать свои данные и данные подзадач, с которыми они образовывали пары. Как результат, на втором шаге пары подзадач могут быть образованы для обмена данными сразу о двух телах физической системы. После завершения второго шага каждая подзадача будет содержать сведения о четырех телах системы и т.д. Тем самым, общее количество шагов для выполнения всех требуемых обменов является равным $log_2 N$



Гравитационная задача N тел (5)

- **Масштабирование и распределение подзадач по процессорам**
  - Если число тел физической системы *N* значительно превышает количество процессоров *p*, рассмотренные ранее подзадачи следует укрупнить, объединив в рамках одной подзадачи вычисления для группы (*N/p*) тел
  - После проведения подобной агрегации число подзадач и количество процессоров будет совпадать
  - При распределении подзадач между процессорами необходимо обеспечить наличие прямых коммуникационных линий между процессорами с подзадачами, между которыми имеются информационные обмены при выполнении операции сбора данных

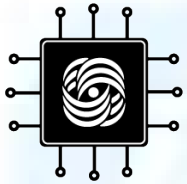


Гравитационная задача N тел (6)

- **Анализ эффективности параллельных вычислений…**
  - Предложенные варианты отличаются только методами выполнения информационных обменов и для сравнения подходов достаточно определить длительность операции обобщенного сбора данных.
  - Используем для оценки времени передачи сообщений модель, предложенную Хокни (*Hockney*), в которой трудоемкость операции коммуникации между узлами вычислительной системы оценивается в соответствии с выражением

$$t_{пд}(m) = \alpha + m / \beta,$$

где α есть задержка сети на передачу данных, m есть размер передаваемого сообщения в байтах, а β обозначает пропускную способность сети.



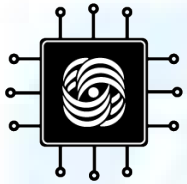
Гравитационная задача N тел (7)

- **Анализ эффективности параллельных вычислений…**
  - Длительность выполнения операции сбора данных для первого метода реализации может быть выражена как:

$$T_p^1(comm) = (p-1)(\alpha + m(N/p)/\beta)$$

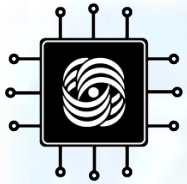
  - При использовании второго метода информационного обмена для итерации с номером *i* объем сообщений оценивается как $2^{i-1}(Nm/p)$. Тем самым, длительность выполнения операции сбора данных в этом случае является равной

$$T_p^2(comm) = \sum_{i=1}^{\log p}(\alpha + 2^{i-1} m(N/p)/\beta) = \alpha \log p + m\,(N/p)(p-1)/\beta$$



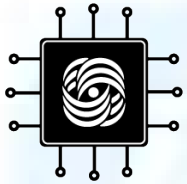
Заключение (1)

- Для оценки оптимальности разрабатываемых методов параллельных вычислений приводятся основные показатели качества
- Рассматривается вопрос построения оценок максимально достижимых значений показателей эффективности
- Определены основные требования к разрабатываемым алгоритмов параллельных вычислений:
  - Достижение равномерной загрузки процессоров,
  - Обеспечение низкого уровня информационного взаимодействия отдельных подзадач



Заключение (2)

- Представлены две модели для описания параллельных алгоритмов и программ:
  - Модель "подзадачи-сообщения" для использования на стадии проектирования параллельных алгоритмов,
  - Модель "процессы-каналы" для применения на стадии разработки параллельных программ
- Рассмотрена методика разработки параллельных алгоритмов, которая включает этапы:
  - Разделение вычислений на независимые части,
  - Выделение информационных зависимостей,
  - Масштабирование имеющегося набора подзадач,
  - Распределение подзадач между процессорами
- Приведен пример использования методики разработки параллельных алгоритмов для параллельного решения гравитационной задачи *N* тел



Спасибо за внимание!