

# Моделирование распределённых вычислений

Востокин Сергей Владимирович

# План

- Обзор моделей распределённых вычислений
- Обзор параллельных и распределённых алгоритмов
- Формальные теории параллельных процессов. Пример: темпоральная логика действий Лампорта (TLA)
- Спецификация параллельных/распределённых алгоритмов. Пример: модель акторов в терминах темпоральной логики Лампорта
- Пример: алгоритм метода Гаусса-Зейделя, сравнение акторного и последовательно-параллельного параллельных алгоритмов
- Порядок выполнения самостоятельных заданий



# Обзор моделей распределённых вычислений

# Проблемы моделирования параллельных и распределённых вычислений

- Не очевидна связь между реализацией алгоритма и его свойствами – формальные методы служат для установления этой связи
- Методы, пригодные для последовательных вычислений, не подходят – моделирование функций не является адекватным методом, так как не учитывает взаимодействие между процессом и его окружением
- Не разработано единой фундаментальной концепции, аналогичной имеющейся для последовательных вычислений (Тезис Чёрча — Тьюринга)

# Классификация формальных методов: подходы на основе синтеза и анализа

- Методы синтеза (алгоритм строится на основе спецификации )
  - Распараллеливание последовательных алгоритмов (динамические, для ациклических и циклических участков кода, для отдельных операторов программы)
  - Методы, использующие непроцедурные спецификации
  - Построении кода алгоритма совместно с доказательством корректности
  - Эквивалентные алгебраические преобразования
- Методы анализа (исследуется готовый алгоритм)
  - Точные
    - Верификация (метод утверждений)
    - Спецификация (темпоральные логики)
  - Имитационные
    - Стратегии планирования ресурсов
    - Дискретно событийные модели, управляемые событиями
    - Методы построения и анализа трасс алгоритма по его спецификации

## Классификация формальных методов: понятие состояния и локальности

- В основе моделирования лежит понятие **локальности**: действия алгоритмов изменяют лишь часть общего состояния системы
- Смена состояний может описываться последовательностью состояний или действий (порождаемых событиями)
- Вычислительный процесс может представляться либо множеством цепочек, либо деревом действий (событий)

## Классификация формальных методов: подходы к описанию состояний

- **Глобальное состояние** – подход основан на возможности упорядочивания всех событий в системе из чего делается вывод о наблюдаемости глобального состояния
- **Локальное состояние** – подход основан на том, что при наличии отказов процессы не могут согласовать свои действия, процесс может только накапливать локальную информацию о системе в целом (за счёт поступающих сообщений)
  - Понятие локального состояния полезно при доказательстве невозможности построения алгоритмов по некоторым спецификациям при наличии отказов

# Методы описания действий и их синхронизации

- Методы, использующие понятие потока управления
  - Последовательно-параллельный метод
  - С взаимодействием параллельных ветвей
    - Атомарные операции; процедурные методы; сообщения
- Асинхронные методы
  - С событийным управлением
  - Поточковые
  - Динамические
  - Сети Петри и их подклассы
- Другие: SPMD, MPMD, CSP (Оссам)



# Особенности моделей распределённых вычислений

- Обычно основаны на концепциях процессов и обмена сообщениями
  - Другие модели процессов относят к распределённым, если они могут быть описаны в терминах процессов и сообщений
- Содержат два ограничения:
  - Время работы алгоритма существенно зависит от времени передачи сообщений
  - Корректная работа любого процесса не зависит от отказов других процессов с которыми он взаимодействует

# Классификация моделей процессов, обменивающихся сообщениями (1/3)

- Сетевая топология
  - Неориентированные / ориентированные коммуникационные графы
  - Связанность (граф полностью связанный или нет)
- Синхронность
  - Полностью асинхронная модель
  - Модель с таймерами
  - Модель с часами
  - Синхронная модель

# Классификация моделей процессов, обменивающихся сообщениями (2/3)

- **Виды отказов**

- **Коммуникационные отказы**

- Потери сообщений
    - Отказ линка
    - Потеря связанности коммуникационного графа

- **Отказы процессов**

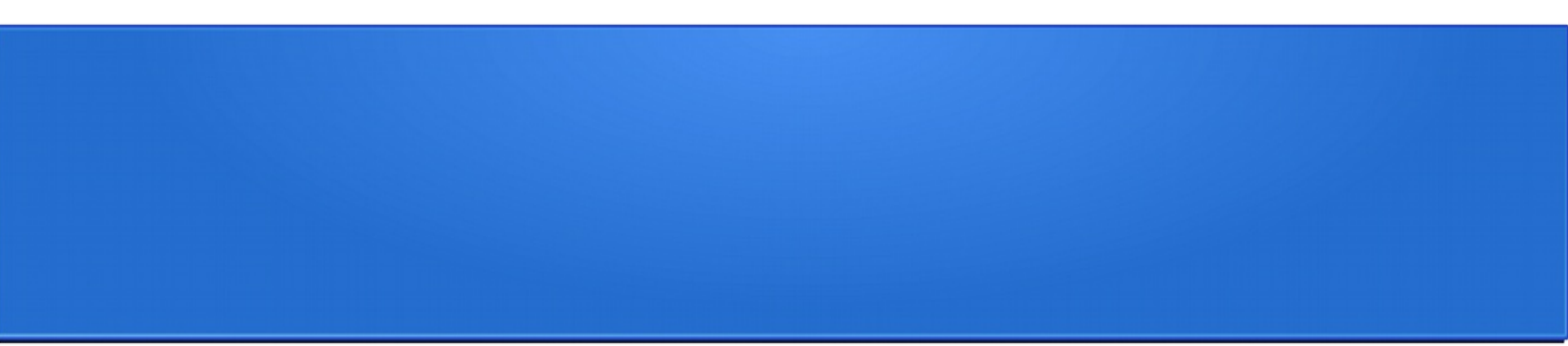
- Византийский отказ
    - Отказ-пропуск
    - Отказ-остановка

## Классификация моделей процессов, обменивающихся сообщениями (3/3)

- Способы буферизации сообщений
  - Неограниченная буферизация
  - Ограниченная буферизация
    - С ожиданием процесса-отправителя
    - Со сбоем процесса-отправителя
  - FIFO буферизация

## Другие модели процессов

- Глобальные разделяемые переменные (не отказоустойчивы)
- Локальные разделяемые переменные (чтение переменной из отказавшего процесса предполагает возврат специального значения)
- Модель Хоара (CSP) (не отказоустойчива)
- Удалённый вызов процедуры (RPC) и рандеву (предусматривают обработку ошибок)



# Обзор параллельных и распределённых алгоритмов

# Группы алгоритмов

- Алгоритмы, использующие разделяемые переменные
- Алгоритмы достижения соглашения
- Сетевые алгоритмы
- Алгоритмы управления параллельными базами данных

# Алгоритмы с разделяемыми переменными (1/3)

- **Задача о критической секции**
  - Алгоритм Дейкстры (Деккера) – не учитывает эффект отталкивания процессов
  - Алгоритм кондитерской Лампорта – учитывает отталкивание
  - Алгоритмы, основанные на примитиве TEST-AND-SET
  - Стохастические алгоритмы
  - Алгоритмы, основанные на обмене сообщениями
  - С произвольным числом процессов в критической секции



# Алгоритмы с разделяемыми переменными (2/3)

- Задача об обедающих философах
  - Алгоритм Дейкстры, использующий семафоры
  - Стохастические и другие алгоритмы
- Проблемы кооперации процессов типа «поставщик-потребитель»
  - Задача об ограниченном буфере
  - Маркированные графы – дают общее решение задачи

# Алгоритмы с разделяемыми переменными (3/3)

- Задача о параллельном сборе мусора
- Задачи об отказоустойчивых кооперативных алгоритмах
- Задача писатели-читатели
  - удалось определить «универсальные» классы переменных, для которых можно построить алгоритмы, реализующие атомарную операцию чтения–записи
  - показано, что при некоторых условиях одновременное чтение–запись не позволяет реализовать примитив TEST-AND-SET

# Алгоритмы достижения соглашения

- Задача о двух генералах
  - При наличии коммуникационных отказов в системе из двух процессов невозможно достичь соглашения
- Задачи выбора общего значения
  - Задача о византийских генералах
  - Задача об одновременном действии (задача о распределенных стрелках)
  - Задача о синхронизации часов
  - Задача о распределённой транзакции

# Алгоритмы достижения соглашения (результаты)

- Найдены предельные значения количества ненадежных процессов и минимальное число циклов обмена сообщениями, при которых имеет решение задача выбора общего значения
- Показано, что стохастические алгоритмы позволяют сократить число циклов обмена сообщениями
- Найден общий способ преобразования алгоритма выбора значения в алгоритм об одновременном действии
- Установлено, что для асинхронных моделей задача выбора общего значения не разрешима даже при наличии одиночного отказа-остановки
- Стохастические алгоритмы решают проблему для полностью асинхронной модели с византийскими отказами, но завершаются с вероятностью, стремящейся к единице
- Для полностью асинхронной модели с византийскими можно получить решение в вещественных числах с заданной точностью

# Алгоритмы достижения соглашения (результаты, продолжение)

- Задача о синхронизации часов неразрешима, если треть и более процессов подвержена византийским отказам
- Из невозможности решения задачи о двух генералах следует неразрешимость задачи о транзакции при условии потери сообщений
- Невозможно решение задачи о транзакции в асинхронных системах при наличии отказов процессов даже в отсутствии потери сообщений (окно отказа)
- трехфазный протокол решения задачи о транзакции, не имеющий окна отказа, предполагает надежную доставку сообщений и возможность обнаружения отказа процесса

# Сетевые алгоритмы

- Алгоритмы определения оптимальных маршрутов передачи сообщений
  - Алгоритмы широковещательной передачи сообщения
- Алгоритмы выбора лидера
  - Разные подходы различаются: свойствами коммуникационных графов; начальной информацией процессов; синхронностью; применением случайных или детерминированные методов; использованием уникальных идентификаторов процессов
- Алгоритмы вычисления функции на сети процессов (например, вычисление среднего значения)
- Алгоритмы определения завершения вычислений и тупика

## Сетевые алгоритмы (продолжение)

- Динамические алгоритмы
  - Алгоритм снимка глобального состояния
  - Алгоритмы-синхронизаторы (методы преобразования синхронных сетевых алгоритмов в асинхронные)
- Протоколы линка
  - Протокол АВР (Alternating Bit Protocol)

# Управление параллельными базами данных

- Особенности

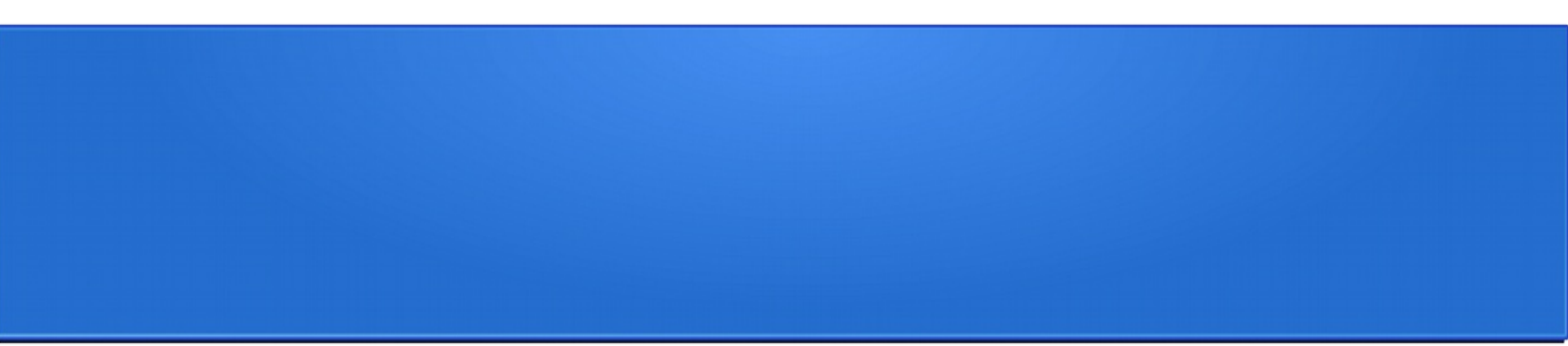
- Цель алгоритмов – обеспечить неделимость и упорядоченность: результат нескольких одновременных транзакций должен быть таким, как если бы они выполнялись автономно в некотором порядке
- Алгоритмы допускают возможность отмены результата транзакции

- Алгоритмы, использующие блокировки

- Алгоритмы, использующие отметки времени

- Алгоритмы вложенных транзакций





Формальные теории параллельных процессов.  
Пример: темпоральная логика действий  
Лампорта (TLA)

# Вычислительный процесс как объект моделирования

- в качестве наблюдаемого поведения дискретной системы будем рассматривать множество историй её выполнения бесконечной длины  $\sigma$  вида

$$s_0 \xrightarrow{A_1} s_1 \xrightarrow{A_2} s_2 \xrightarrow{A_3} s_3 \xrightarrow{A_4} \dots$$

# Определение модели

- Модель  $F$  – это формула, смысл которой определен в терминах историй бесконечной длины, однозначно определяющая заданное множество историй
- Интерпретация формулы  $F$  — это функция  $[[\cdot]]$ , которая ставит в соответствие формуле  $F$  истинное или ложное высказывание об историях вычислительного процесса

$$\langle s_0, s_1, s_2, \dots \rangle [[F]] \equiv \sigma[[F]] : \text{St}^\infty \rightarrow \{И, Л\}$$

# Определение функции интерпретации модели

- Функция интерпретации строится на основе понятия **состояния**: функции, отображающей множество переменных  $Var$  на множество значений переменных  $Val$

$$St : Var \rightarrow Val$$

- Таким образом, смысл переменных в формуле состоит в представлении значений в некотором состоянии вычислений

# Функции состояния

- Математические выражения, построенные из переменных, знаков математических операций, знаков отношений, логических связок, кванторов и т.п. называются **функциями состояния** и интерпретируются согласно выражению:

$$s[[f]] \stackrel{def}{=} f(\forall' v': s[[v]] / v)$$

- Частный случай функции состояния – **предикат состояния**

# Действие

- Действие — это выражение, аналогичное функции состояния по своей структуре, которое дополнительно включает переменные с символом апострофа, принимает значения истина или ложь и интерпретируется в виде

$$\overset{def}{s[[A]]t} = A(\forall' v': s[[v]] / v, t[[v]] / v')$$

- Формула для действия дает ответ на вопрос: могут ли два состояния  $s$  и  $t$  быть соседними в некоторой истории

## Расширение интерпретации $F$ для историй бесконечной длины

- Для описания историй бесконечной длины с использованием действий вводится специальный модальный оператор необходимости, который читается «всегда»

$$\sigma[[\Box F]] \stackrel{def}{=} \forall n \in \mathbb{N} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle [[F]]$$

- Где  $F$  – предикат состояния или действие

# Расширение интерпретации $F$ для историй бесконечной длины (продолжение)

- Смысл функций состояния и действий расширяют на бесконечные истории следующим образом

$$\begin{aligned} \langle s_0, s_1, s_2, \dots \rangle [[A]] &\stackrel{def}{=} s_0 [[A]] s_1, \\ \langle s_0, s_1, s_2, \dots \rangle [[P]] &\equiv s_0 [[P]] \end{aligned}$$

- Тогда

$$\begin{aligned} \langle s_0, s_1, s_2, \dots \rangle [[\Box A]] &\stackrel{def}{=} \\ \forall n \in \mathbb{N} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle [[A]] &\equiv \forall n \in \mathbb{N} : s_n [[A]] s_{n+1} \\ \langle s_0, s_1, s_2, \dots \rangle [[\Box P]] &\equiv \forall n \in \mathbb{N} : s_n [[P]] \end{aligned}$$



# Интерпретация произвольной формулы

- Сложные формулы строятся путем комбинирования предикатов состояний и действий с использованием логических связок и кванторов. Для передачи смысла таких формул достаточно определить семантику логической основы формулы для произвольной системы базисных функций, например так:

$$\sigma[[\neg F]] \stackrel{def}{=} \neg \sigma[[F]]$$

$$\sigma[[F \wedge G]] \stackrel{def}{=} \sigma[[F]] \wedge \sigma[[G]]$$

# Специализация модели (1/2)

- Пусть процесс  $X$  периодически увеличивает переменную  $x$   
 $[x=0] \rightarrow [x=1] \rightarrow [x=2] \rightarrow [x=3] \rightarrow [x=4] \rightarrow [x=5] \rightarrow [x=6] \rightarrow [x=7] \dots$
- Пусть процесс  $Y$  периодически увеличивает переменную  $y$   
 $[y=0] \rightarrow [y=1] \rightarrow [y=2] \rightarrow [y=3] \rightarrow [y=4] \rightarrow [y=5] \rightarrow [y=6] \rightarrow [y=7] \dots$
- Рассмотрим композицию систем  $X$  и  $Y$ . Так как системы независимы, то допустимы истории, где  $x$  изменяется, а  $y$  – нет. Например, такая  
 $[x=0, y=0] \rightarrow [x=1, y=0] \rightarrow [x=2, y=1] \rightarrow [x=2, y=2] \rightarrow [x=2, y=3] \dots$

# Специализация модели (2/2)

- Композицию систем  $X$  и  $Y$  естественно рассматривать как конъюнкцию формул:  $X \wedge Y$
- История системы должна быть также историей для входящих в систему подсистем:  $X \wedge Y \rightarrow X$  и  $X \wedge Y \rightarrow Y$
- Этого можно добиться, если процесс описывается множеством историй, инвариантных по отношению к преобразованию дублирования одного состояния в соседних отсчетах времени (stuttering)
- Для этого вводятся дополнительные синтаксические ограничения на структуру темпоральных формул

# Общий вид формул, инвариантных к stuttering-преобразованию

$$\overset{def}{\Phi} ::= I \wedge \square [N]_f \wedge F$$

$F$  — формулы, обозначающие справедливое в слабом ( $WF$ ) или сильном смысле ( $SF$ ) выполнение действий из  $N$  (поясняется далее)

# Пояснения оператора stuttering преобразования

$$f \stackrel{def}{=} (v_1, v_2, v_3, \dots) \quad f' \equiv (v_1, v_2, v_3, \dots)' \equiv (v'_1, v'_2, v'_3, \dots)$$

$$unchanged(f) \stackrel{def}{=} f = f'$$

$$[A]_f \stackrel{def}{=} A \vee (f = f') \equiv A \vee unchanged(f)$$

$$\langle A \rangle_f \stackrel{def}{=} A \wedge (f \neq f') \equiv A \wedge \neg unchanged(f)$$

# Выражение понятия «справедливого выполнения» в TLA

Если допустить инвариантность относительно stuttering преобразования, то возможны истории, в которых вычисления произвольно останавливаются

Поэтому в общую формулу добавляются части  $WF$  и  $SF$ , оговаривающие отсутствие самопроизвольной остановки

$$WF_f(A) \stackrel{def}{=} (\Box \langle A \rangle_f) \vee (\Box \diamond \neg Enabled \langle A \rangle_f)$$

$$SF_f(A) \stackrel{def}{=} (\Box \langle A \rangle_f) \vee (\diamond \Box \neg Enabled \langle A \rangle_f)$$

# Пояснения операторов в формулах «справедливого выполнения» в TLA

Оператор возможности:

$$\diamond F \stackrel{def}{=} \neg \square \neg F$$

$$\langle s_0, s_1, s_2, \dots \rangle [[\diamond F]] \equiv \exists n \in \mathbb{N} : \langle s_n, s_{n+1}, s_{n+2}, \dots \rangle [[F]]$$

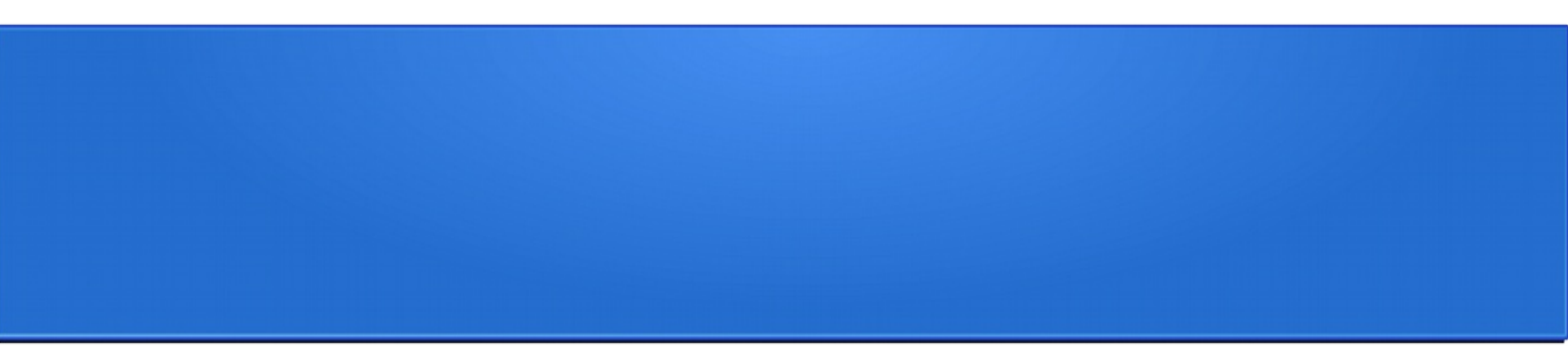
Комбинации операторов необходимости и возможности:

$$\langle s_0, s_1, s_2, \dots \rangle [[\square \diamond A]] \equiv \forall n \in \mathbb{N} : \exists m \in \mathbb{N} : s_{n+m} [[A]] s_{n+m+1}$$

$$\langle s_0, s_1, s_2, \dots \rangle [[\diamond \square A]] \equiv \exists n \in \mathbb{N} : \forall m \in \mathbb{N} : s_{n+m} [[A]] s_{n+m+1}$$

Оператор Enabled

$$s [[Enabled A]] \stackrel{def}{=} \exists t \in St : s [[A]] t$$

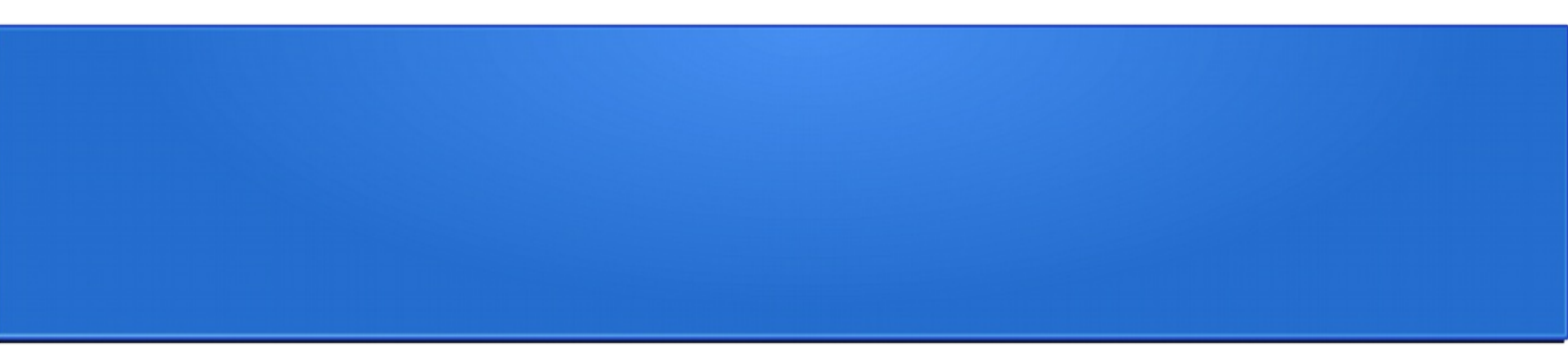


Спецификация параллельных и  
распределённых алгоритмов.  
Пример: модель акторов в терминах  
темпоральной логики Лампорта



# Тема 1

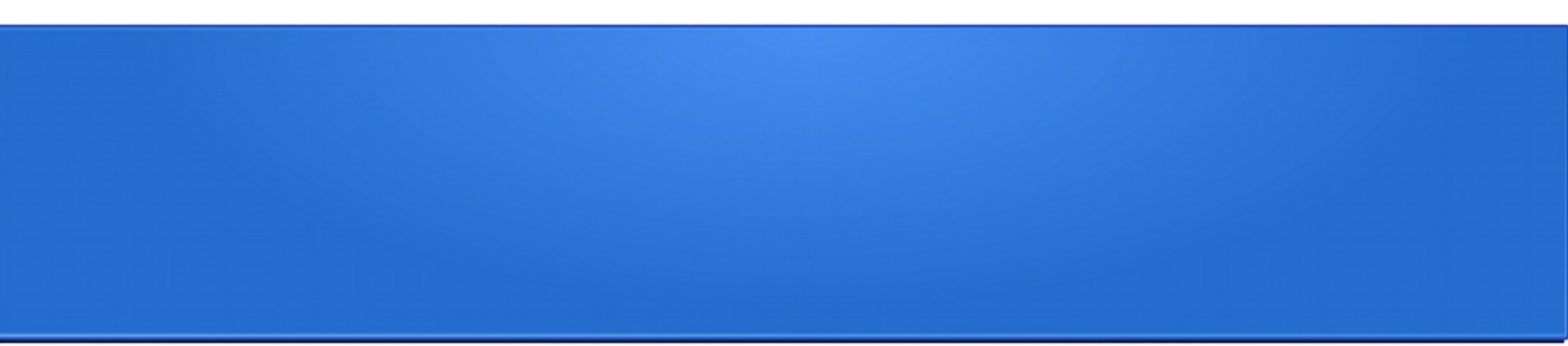
- Тезис 1
- Тезис 2
- Тезис 3



Пример: алгоритм метода Гаусса-Зейделя,  
сравнение акторного и последовательно-  
параллельного параллельных алгоритмов

# Тема 1

- Тезис 1
- Тезис 2
- Тезис 3



# Порядок выполнения самостоятельных заданий

# Тема 1

- Тезис 1
- Тезис 2
- Тезис 3