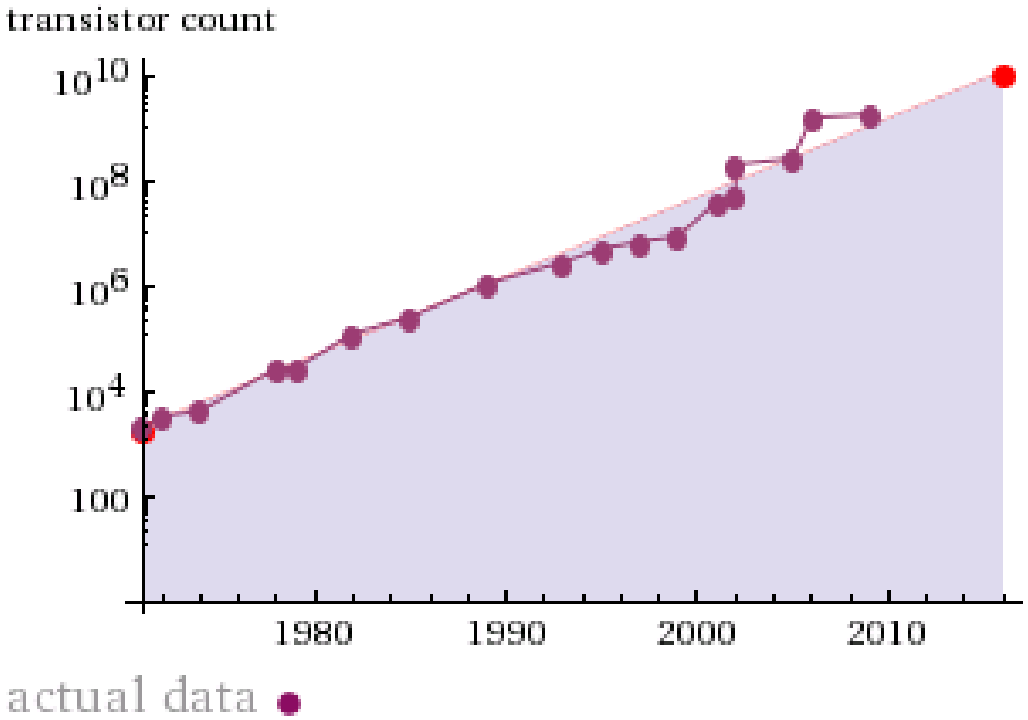
Теория и практика многопоточного программирования

ЛЕКЦИЯ 1. ВВЕДЕНИЕ

Темы лекции

- Предпосылки возникновения
- Состояние дел на настоящий момент
- Парадигмы мышления
- Содержание курса
- Компетенции после курса

Предпосылки возникновения



Проблемы закона Мура других «законов»

- Тепловыделение
  - Повышение числа транзисторов (рост разрядности и числа ядер). Принцип Ландауэра
  - Увеличение тактовой частоты приводит к росту теплоотделения
- Скорость света ближе, чем хотелось бы
- Скорость памяти
- Сложность алгоритмов не позволяет оценивать производительность «в уме»

Состояние дел на сегодня

Процессор – штука сложная, поэтому современая система – это...

- «простые» команды
- много ядер и hyper-threading
- много кэшей и медленная память

Последовательная программа на Core i7 – менее 13% от пиковой производительности

Масштабирование программ

	Общая (разделяемая) память	Распределённая память
Один поток	XIX век	Распределённые вычислительные системы: MPI и кластеры, GRID, RPC и наседники, Load balance
Много потоков	Многопоточное программирование: OpenMP, native threads, async-await	Комбинированные технологии (кластеры с OpenMP + MPI), GPGPU

«Линейка» систем

Многопоточные программы

- Быстрее не придумаешь (КПД распараллеливания высок)
- Ограниченность сверху уровнем технологии
- Боится физических неполадок

Распределённая память в кластере

- Медленнее
- Боится физических неполадок

Распределённая память в GRID

- Ещё медленнее (крупные «атомы»)
- Не боится неполадок

Load Balancing:

- Узкий класс задач
- Не боится неполадок
- Самый простой вариант

Парадигмы мышления

Последовательная

- Я (программа), выполняюсь одна, забираю всё процессорное время
- Никого (процессов, планировщиков) нет, никто не вмешается в мою работу, не отнимет процессор и не меняет данные
- Программист должен думать о **корректности** алгортима

Параллельная

- Таких как я (потоков) на машине много.
- Есть ещё ОС, другие программы, другие экземпляры таких как я.
- Программист думает о корректности алгоритма и влиянии **многопоточного окружения**
- Программист думает не только о скорости алгоритма, но и о выравнивании данных, устройствах, обращениях к памяти разных типов, кэше

Содержание курса

Архитектура ЭВМ как препятствие и инструмент при разработке эффективных параллельных алгоритмов

- Способы явного и неявного манипулирования аппаратурой

Теория: «математика» параллельного программирования (анализ производительности, корректности)

- Понятия атомарности, времени, истории, линеаризуемости, и т.п.
- Сравнение примитивов синхронизации
- Задача о консенсусе, числа консенсуса для различных структур данных

Содержание курса [2]

Практика: написание программ, проверка теории

Неблокирующие и неожидающие алгоритмы и структуры данных

- Понятия свободы от блокировок и от ожидания
- Существующие неожидающие алгоритмы

Специфика работы с разделяемой памятью

- Проблемы параллельности и совместного доступа

Знания после курса

- Умение анализировать работу параллельных потоков, использующих общую память
- Понимание проблем и терминологии параллельного программирования
- Знание современных подходов организации высокопроизводительных параллельных вычислений
- Знание принципиальных ограничений алгоритмов
- Умение создавать эффективные алгоритмы