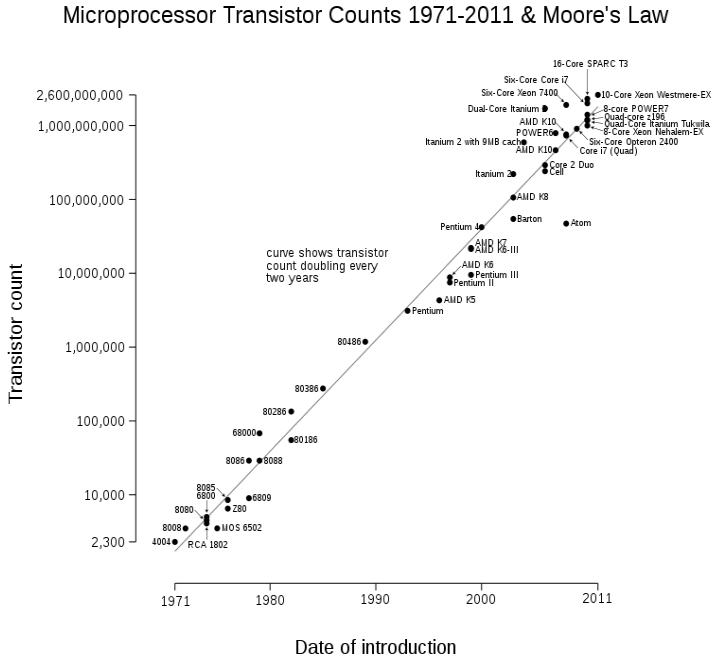
Многопоточное программирование на C++

О курсе, аллокаторах и задачах

О параллелизме, многопоточности и лени

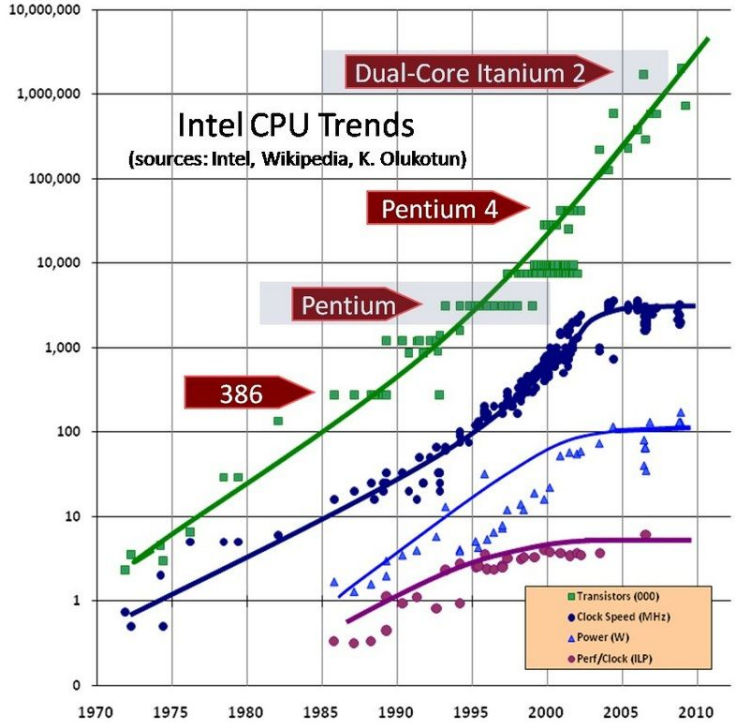
- Закон Мура
  - Примерно с середины десятилетия 2000-х годов по разным причинам производители процессоров предпочитают многоядерные архитектуры

- Закон Амдала
  - В случае, когда задача разделяется на несколько частей, суммарное время её выполнения на параллельной системе не может быть меньше времени выполнения самого длинного фрагмента»



О неизбежности многопоточности

- Количество задач растет
  - Количество данных растет
  - А так же число методов обработки
- Вычислительная мощность?



Чего не будет

- Задач о филосовах
- Рассказов о API (почти)
- Распределенных систем

Чем будем заниматься

- Посмотрим какие бывают практические задачи
- Рассмотрим разные варианты их решения
- Попробуем понять недостатки и достоинства каждого из методов

Условия игры

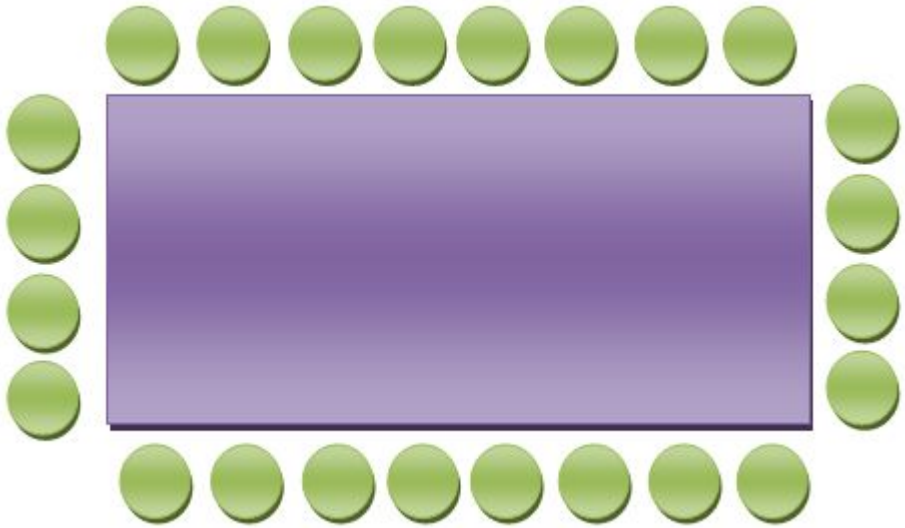
- Linux: CentOS 7, ArchLinux
- Колличество домашек:
  - Обязательных:
  - За остальные даем баллы
    - На 4 нужно:
    - На 5 нужно:
- Сроки сдачи:

Аллокатор

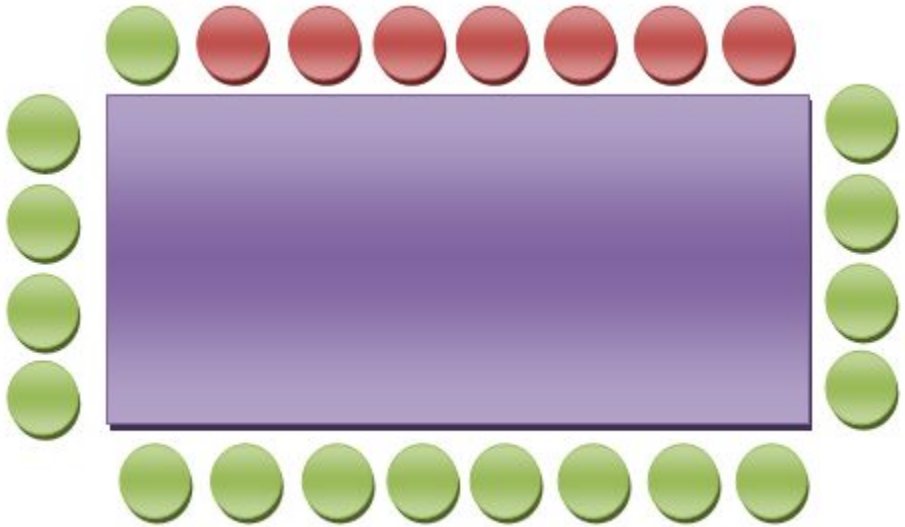
Тот самый кирпич

- В среднем проекте аллокации случаются постоянно
- В каждом из потоков
- Память процесса - разделяемый ресурс
- Есть множество стратегий выделения памяти

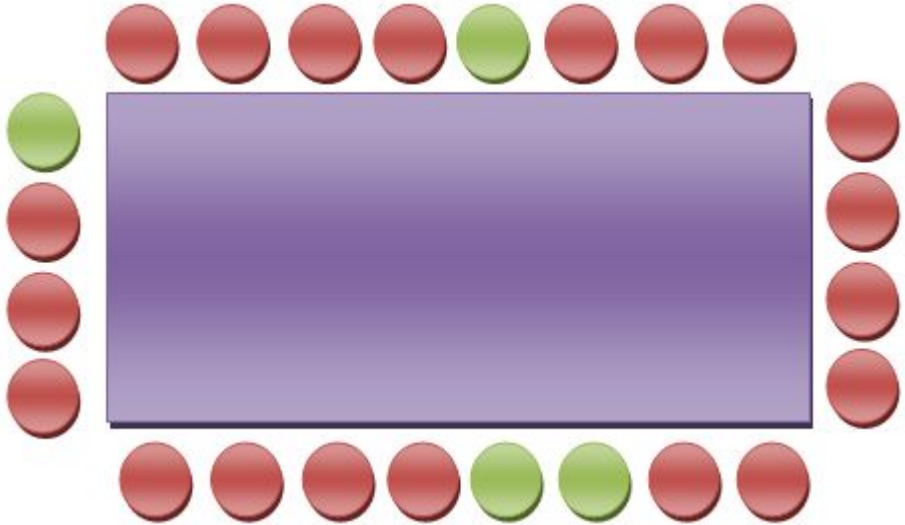
Что такое malloc/free или немного о суши



Что такое malloc/free или немного о суши



Что такое malloc/free или немного о суши



Но мы же не в ресторане...

```
template <class Entry> class LockFreeStack {
    struct Node {
        Entry *data;
        Node *next;
    }

    Node *_head;

    void Push(Entry *d) {
        Node *n = new Node();
        n->data = d;
        do {
            n->next = _head;
        } while (!cas(&_head, n->next, n));
    }

    Entry *Pop() {
        Node *old_head;
        do {
            old_head = m_head;
            if (old_head == nullptr) {
                return nullptr;
            }
        } while (!cas(&m_head, old_head, old_head->next))

            Entry *result = old_head->data;
        delete old_head;

        return result;
    }
};
```

Тут код домашки и обсуждение