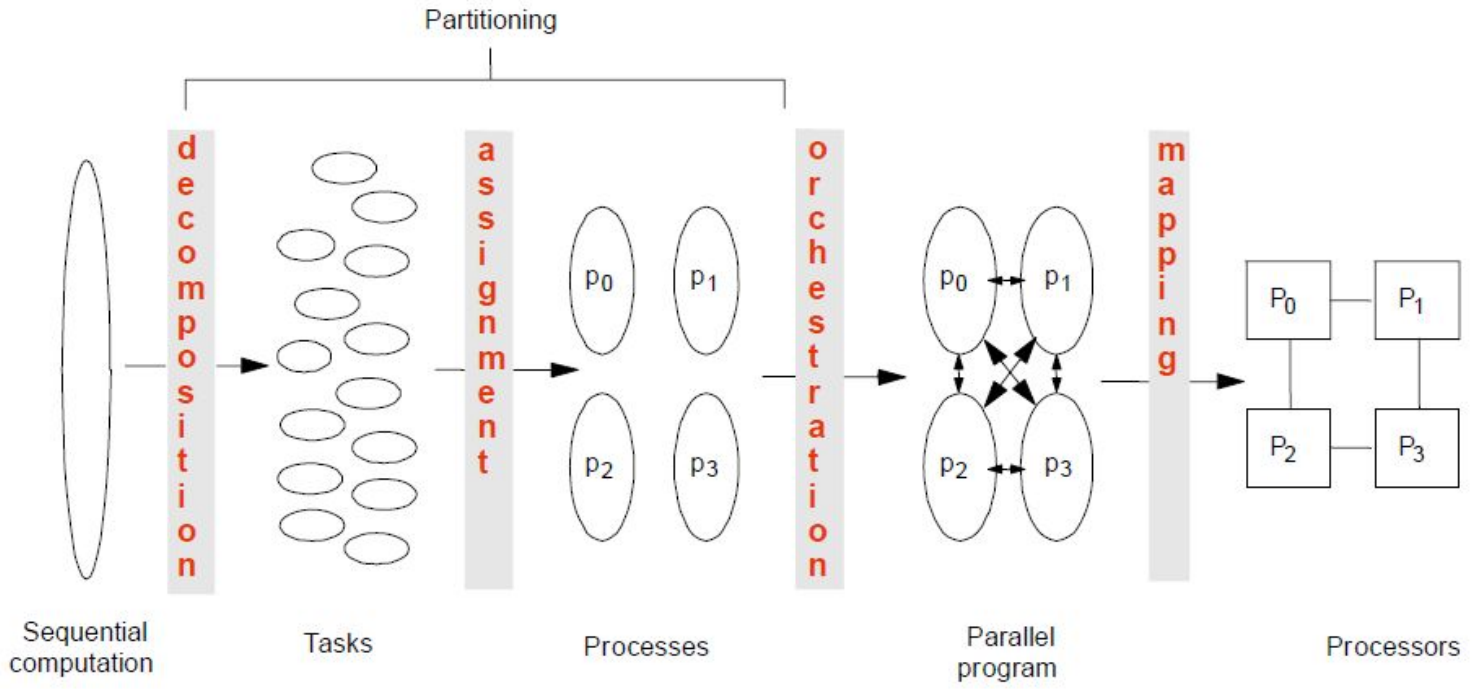
Шаблоны параллельного проектирования

Этапы разработки параллельной программы



Поиск параллельности (1/4)

Задача – помыть окна в кабинете. Вы приходите – один. Окон десять.

Нужно определить участки программы, которые можно выполнять параллельно

- Задачи должны иметь начало и окончание
- Количество задач может меняться
- Задач должно достаточно для получения ускорения (одна задача это не очень хорошо)
- Работает закон Амдала

Можно ли мыть все окна одновременно – в принципе да. Даже если мы будем все окна мыть одновременно – мы помоем 10 окон за время, равное времени, необходимого, чтобы вымыть одно окно.

Выбор шаблона реализации (2/4)

Проектирование структуры параллельной программы

Закладывается возможность масштабирования

Выбор инструментов реализации

Отдельно задачи, отдельно исполнители

Вы приглашаете друга, у каждого есть ведро и тряпка. Каждый из вас - «боевая единица», способная помыть окно. Есть окна, есть вы. Пока есть грязные окна вы всегда заняты.

Пришла бабушка ученика, которая может мыть окна только до середины. Это не очень хороший вариант. Исполнители должны быть равнозначны по возможностям, пусть даже и отличаться по скорости.

Реализация алгоритма (3/4)

Синхронизация

Взаимодействие

Если пришло достаточное количество народу, то вам нужно договориться один раз, кто какое окно моет и все. Далее каждый будет следовать простому алгоритму:

Простая линейная программа, которую сложно выполнить параллельно. *Сложно, но можно. Каждый взрослый берет себе в пару ребенка. Родители моют, ребенок таскает воду. Есть определенное ускорение. Но возникает сложность – родителю необходимо договариваться с ребенком. Поверьте, это сложно. Количество коммуникаций увеличивается. Если и второй минус, ребенок «простаивает». Получается, что часть команды отдыхает, пока вторая трудится. В случае программы это может означать простой ресурсов.*

- *Налить воду в ведро*
- *Помыть первое стекло*
- *Протереть первое стекло*
- *Поменять воду*
- *Помыть второе стекло*
- *Протереть второе стекло*
- *Вылить воду*
- *Пойти домой*

Выполнение (4/4)

Общие ресурсы

Гонки

Если кто-то забыл ведро дома, то ведро становится разделяемым ресурсом и тут уже придется договариваться. Либо ждать пока сосед помоет первым и пойти домой позже, либо попробовать использовать одно ведро на два или более окон. Опять приходится договариваться. А если кто-то еще забыл и ребенка, то..

Шаблоны параллельного программирования

1977 вышла книга «Язык шаблонов. Города. Здания. Сооружения.»

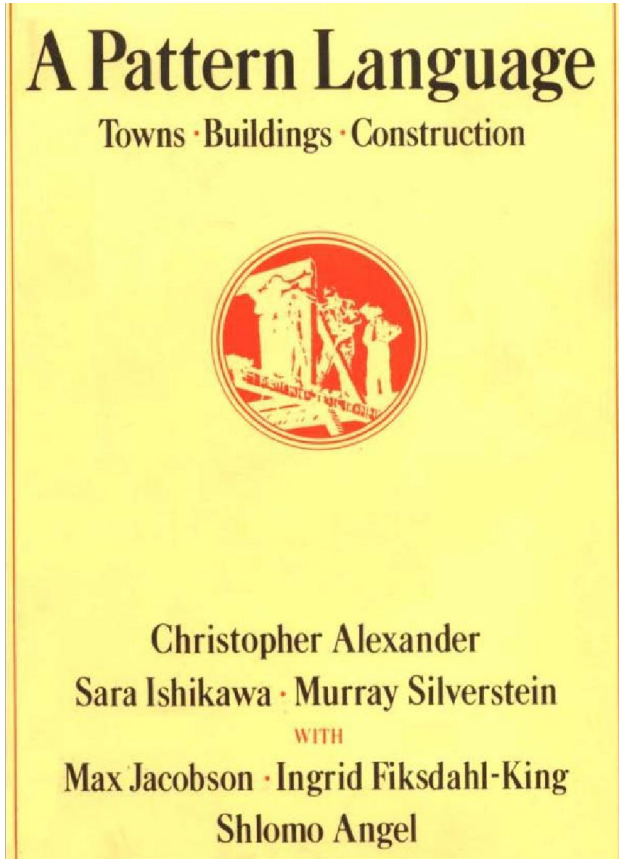
Кристофер Александер

англ. Christopher Alexander

Кристофер Александер

Основные сведения

Имя при рождении	Кристофер Вольфганг Александер
Страна	США
Дата рождения	4 октября 1936 (80 лет)
Место рождения	Вена, Австрия
Учёба:	Гарвардский университет и Тринити-колледж

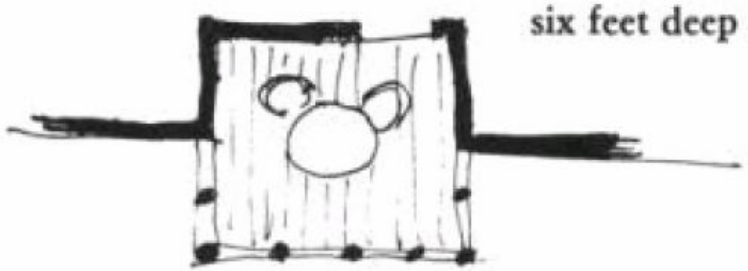


Шаблоны в строительстве

I started with PRIVATE TERRACE ON THE STREET (140). That pattern calls for a terrace, slightly raised, connected to the house, and on the street side. SUNNY PLACE (161) suggests that a special place on the sunny side of the yard should be intensified and made into a place by the use of a patio, balcony, outdoor room, etc. I used these two patterns to locate a raised platform on the south side of the house.

Шаблоны в строительстве

Whenever you build a balcony, a porch, a gallery, or a terrace always make it at least six feet deep. If possible, recess at least a part of it into the building so that it is not cantilevered out and separated from the building by a simple line, and enclose it partially.



Not this . . . *. . . this.*

Шаблоны в программировании

Adapter

Builder

Decorator

Facade

Flyweight

…

Task parallelism

Data parallelism

Recursive decomposition

Geometric decomposition

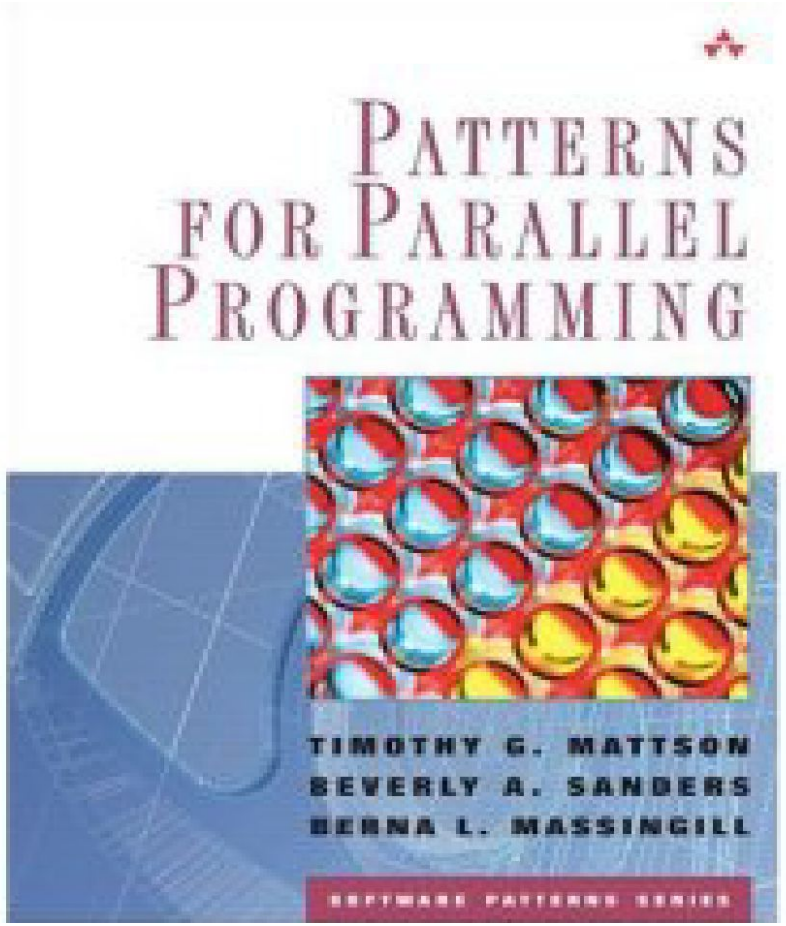
Divide and conquere

SPMD

Master/Worker

…

Patterns for parallel programming



Patterns for Parallel Programming. Mattson, Sanders, and Massingill (2005).

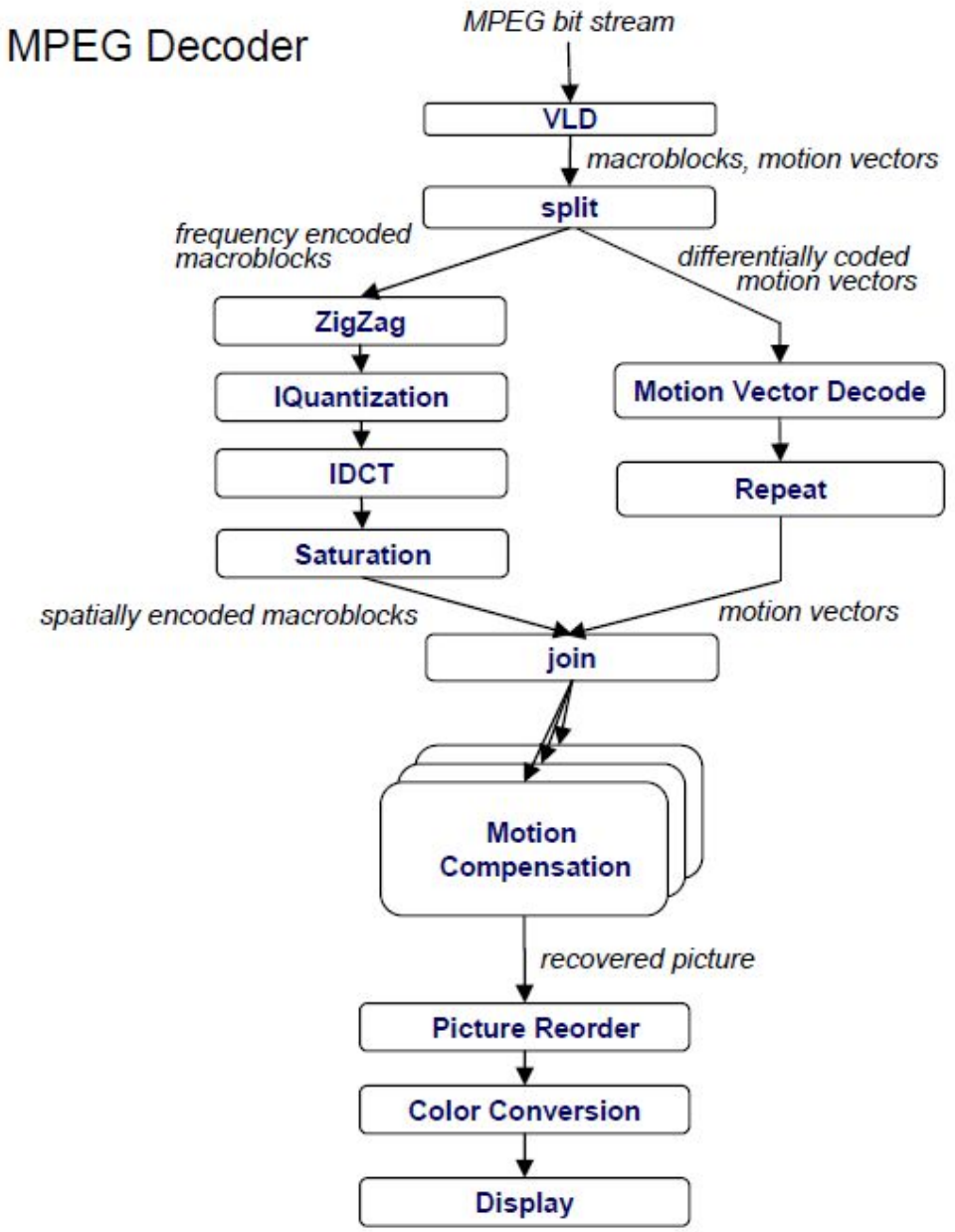
Есть pdf`ка..

Начало книги

“Could you just tweak my serial code to make it run in parallel?”

Что параллелить?

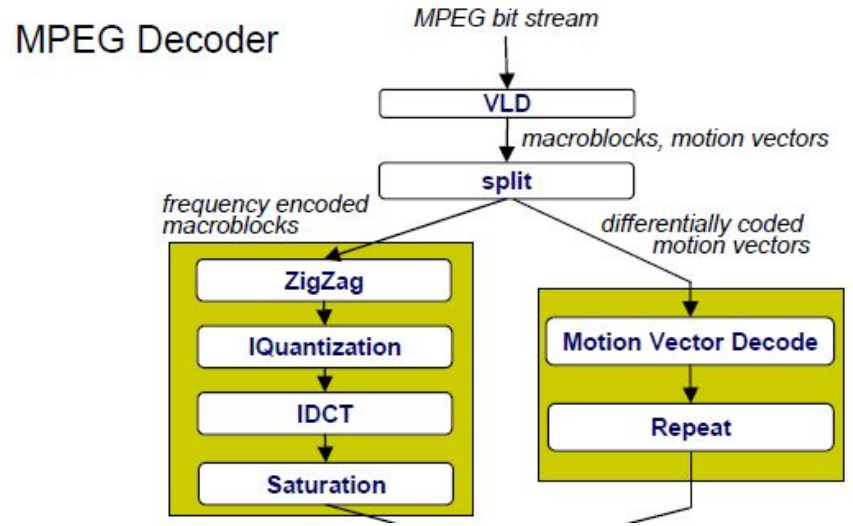
Вот алгоритм – что можно вычислить параллельно?



Параллелизм задач

Параллельные задачи

- Гибкость
- Эффективность
- Простота



Что там про окна: если кроме окон нужно покрасить парты, то можно выполнять эти операции независимо друг от друга. Ресурсы не пересекаются, последовательность не важна.

Гибкость (1/3)

Не привязывайтесь к железу

Издали закон, по которому окна можно мыть только по три человека.

Если окон десять и родителей десять, что делать?

Если окно одно – двое будут простаивать?

Гибкость (2/3)

Задачи не должны решать какие данные им обрабатывать

Например, приходит в класс мыть десять окон десять человек. В идеале можно помыть все окна зха время равное мытью одного окна. Но тут Иван Петрович говорит, а дай-к я вымою все окна сам. В принципе все клево, развернулись и пошли домой. Но с точки зрения времени и ресурсов такая задача будет выполнена неэффективно.

Гибкость (3/3)

Задачи должны уметь считать разные объемы данных

Например у нас три окна и один мойщик окон, который умеет за раз быть сразу два окна. Два он помыл. Но осталось еще одно, которое он мыть не умеет. Что делать? Рисовать окно на стене или оставить его не мытым? Такие задачи неудобно масштабировать.

Эффективность (1/2)

Задачи должны эффективно использовать ресурсы

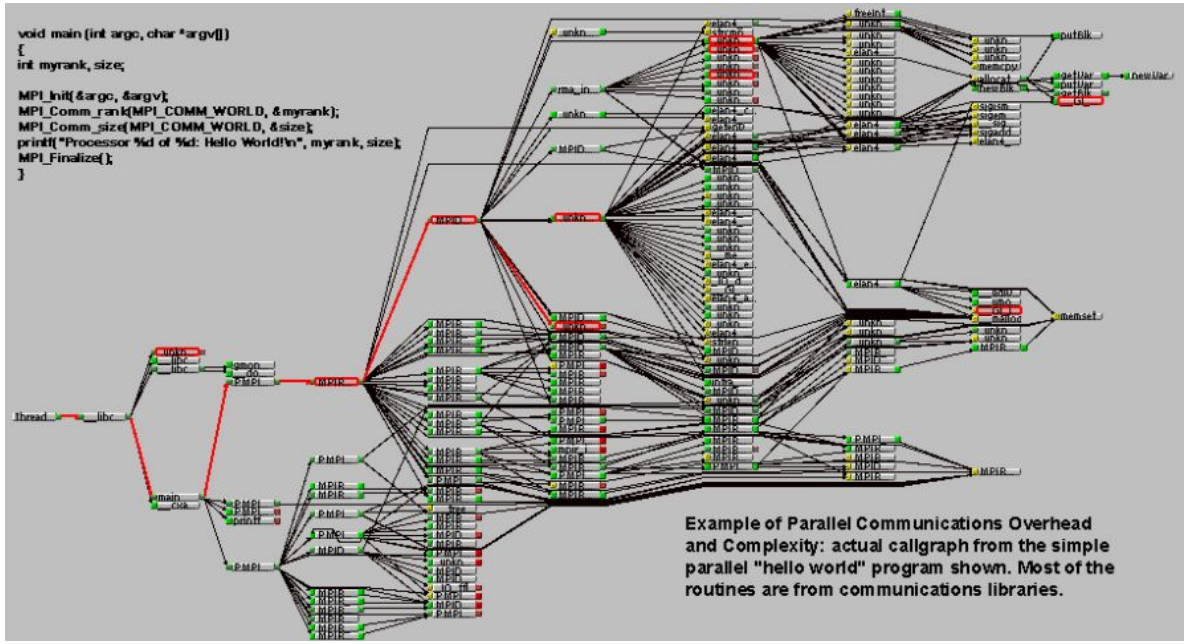
В классе десять окон. Вспомним алгоритм:

- *налить ведро*
- *помыть окно*
- *вылить воду*
- *повторить*

А теперь окна заменим на плитки. И вымоем таким образом медицинский кабинет. Время инициализации и очистки ресурсов будет несравненно больше времени полезной работы. Такая задача будет работать неэффективно.

Эффективность (2/2)

Создавайте независимые задачи

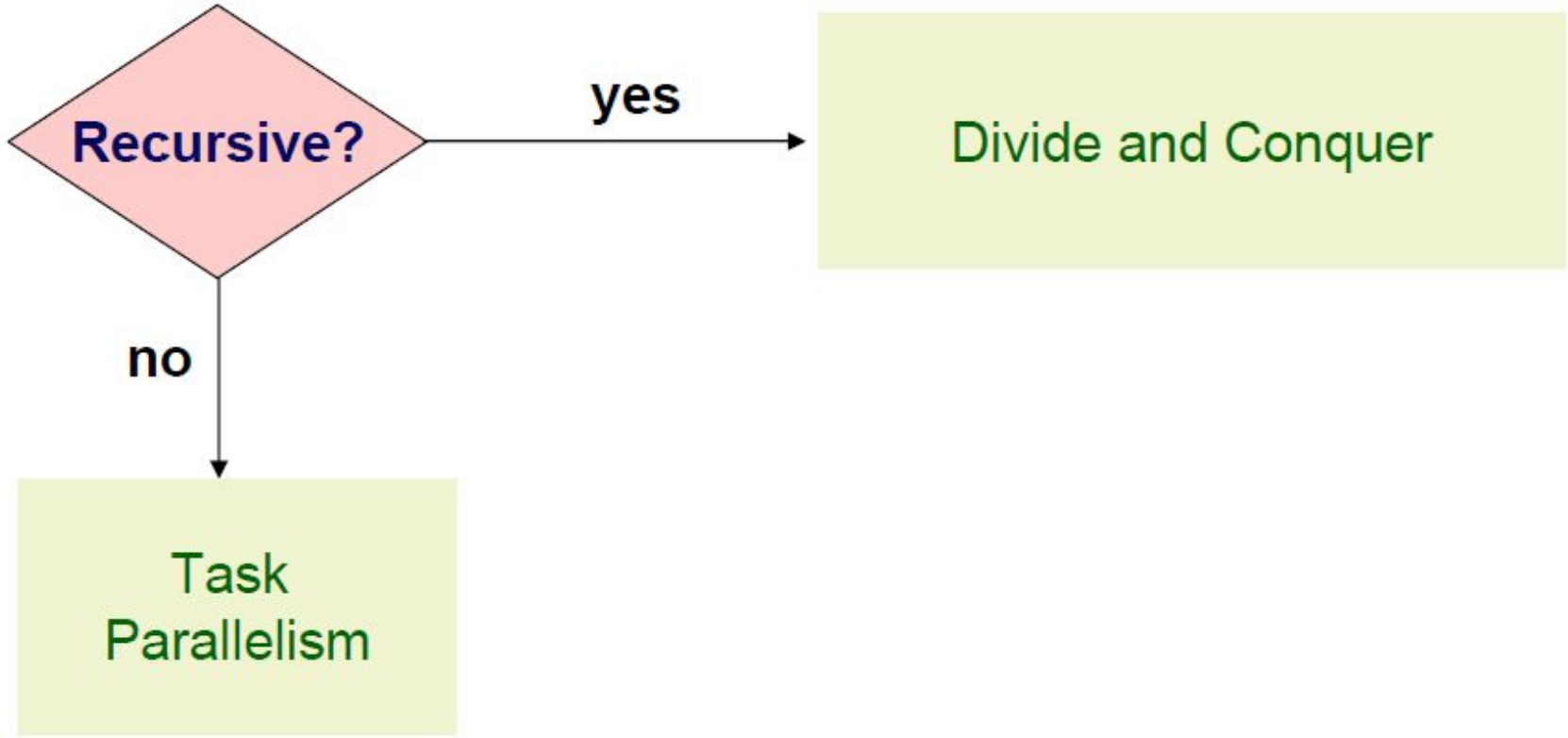


Just “Hello world!”

Простота (1/1)

- Понятность
- Читаемость
- Шаблоны ООП
- Стили кодирования
- Тестирование

Шаблоны параллелизма по задачам



Task parallelism (1/2)

Примеры

- Ray tracing - вычисление каждого луча независимо по данным и последовательности
- Молекулярная физика - движение несвязанных частиц, слабое взаимодействие

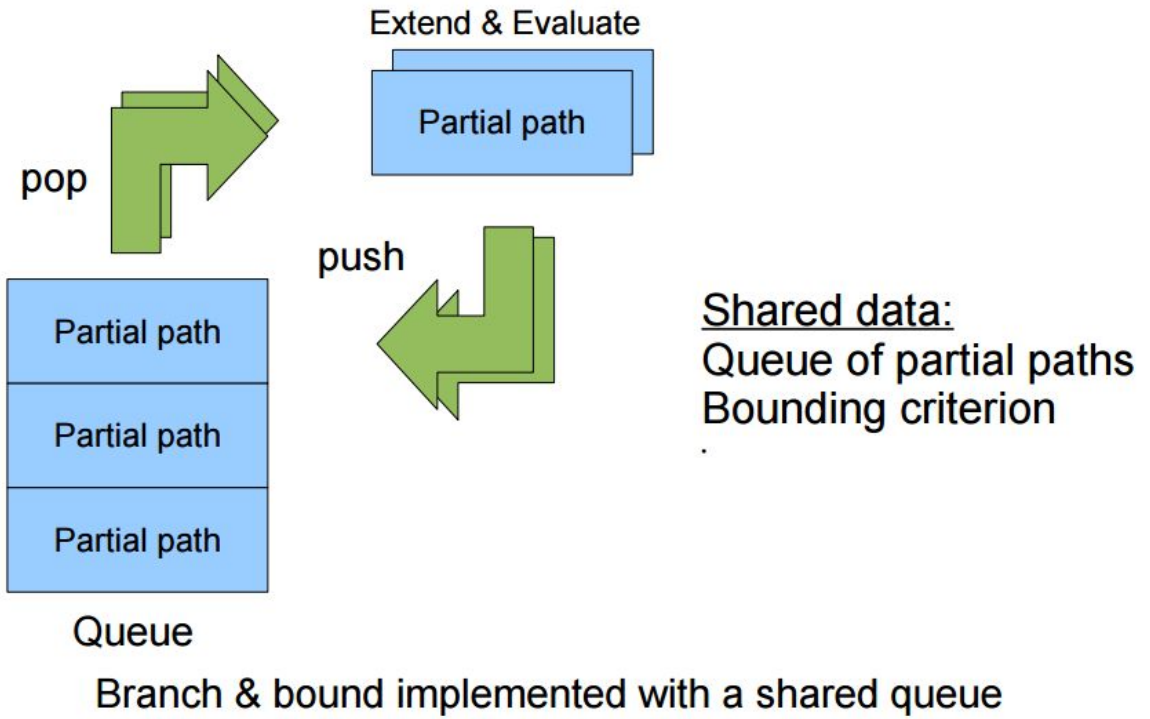
Основные особенности

- Задачи связаны с определенными циклами
- Список задач в основном известен в начале вычисления
- **Не обязательно все задачи должны быть выполнены для получения части конечного решения**

Task parallelism (2/2)

Используйте очереди:

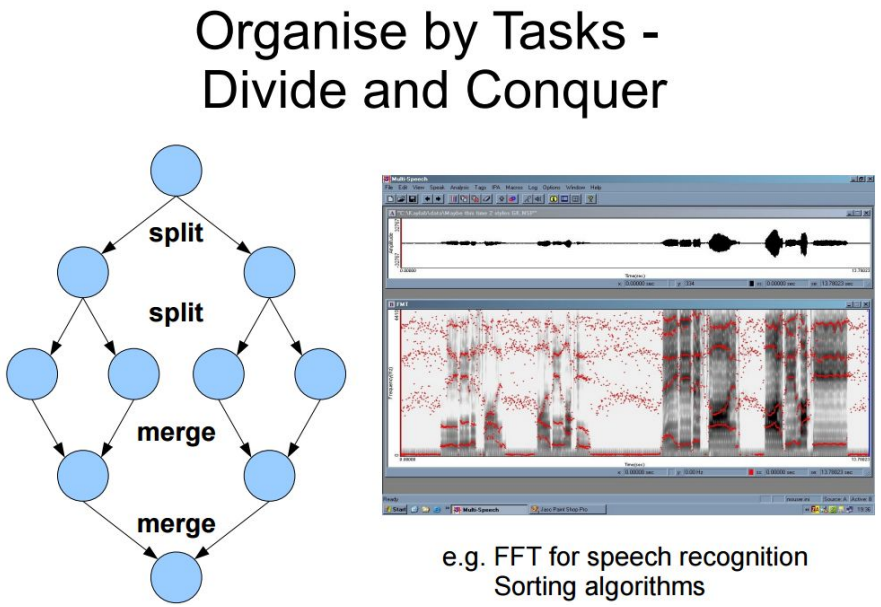
- RabbitMQ
- MSMQ
- Amazon SQS
- IronMQ
- StormMQ
- Windows Azure Queues



Branch & bound implemented with a shared queue

Divide and Conquer

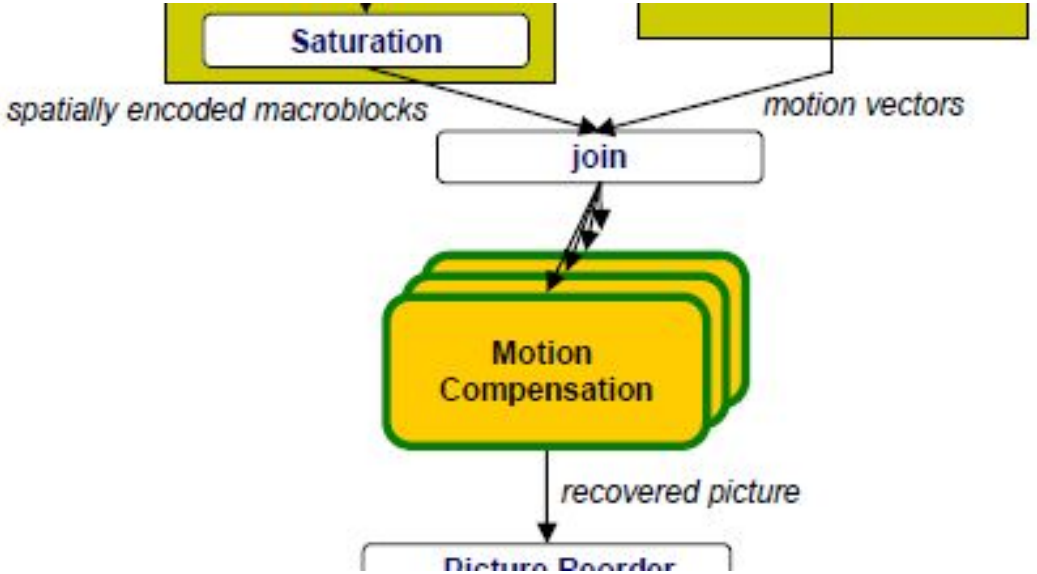
- Задачи выполняют разные действия
- Важна последовательность задач



Параллелизм данных

Параллелизм данных

- Основные вычисления выполняются на большом объеме данных;
- Одни и те же операции применяются в разным частям данных.



Параллелизм данных

Гибкость

- Данные должны хорошо дробиться, чтобы поддерживать высокий уровень параллелизма

Эффективность

- Размер данных должен обеспечивать достаточное количество вычислений

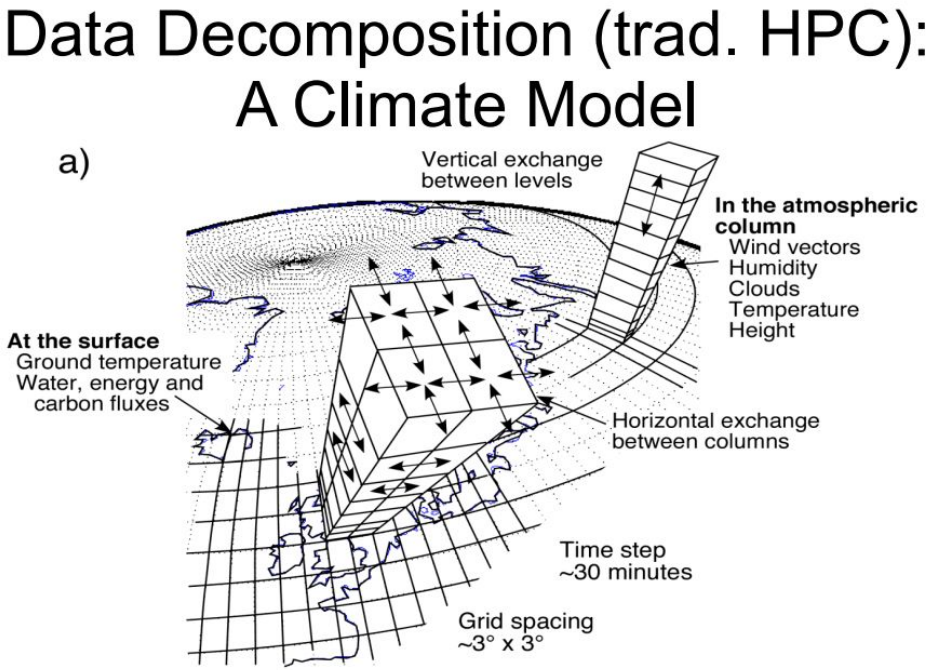
Простота

- Сложные структуры данных сложно отлаживать и поддерживать

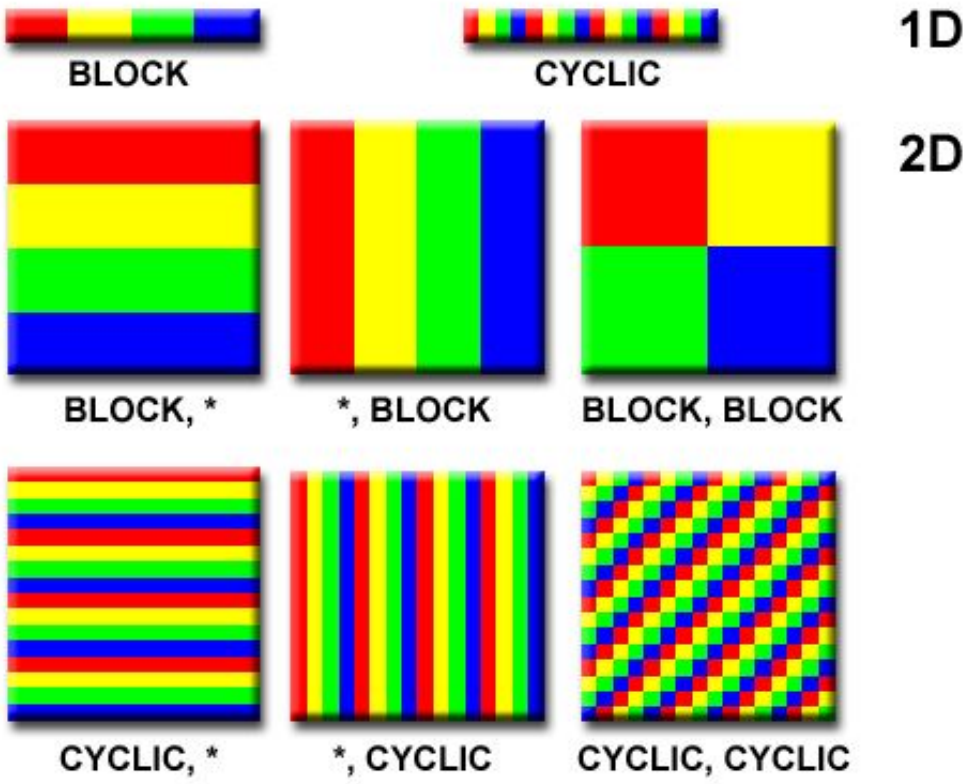
Геометрическое разбиение

Хорошо работает на таких типах данных как:

- Массив
- Список
- Справочник



Способы геометрического разбиения



Рекурсивное разбиение

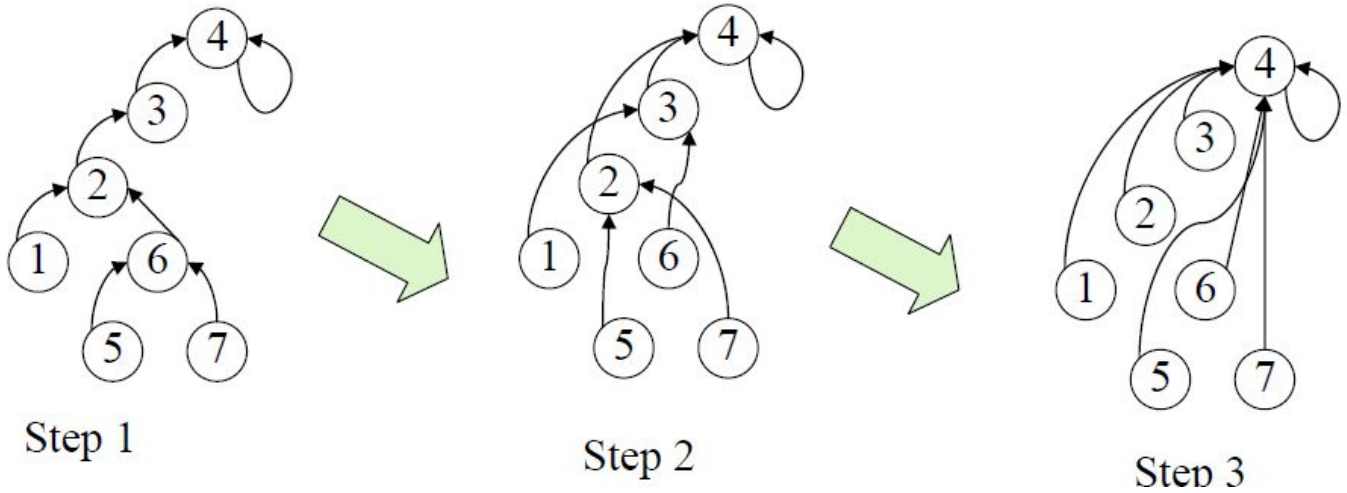
Хорошо работает на таких типах данных как:

- Массив
- Список
- Деревья
- Графы

- *Подсчет среднего возраста по каждому континенту*
  - *Подсчет среднего возраста по каждой стране континента*
    - *Подсчет среднего возраста по каждому городу страны*
    - *Подсчет среднего возраста между городами страны*
  - *Подсчет среднего возраста между странами континента*
- *Подсчет среднего возраста между континентами*

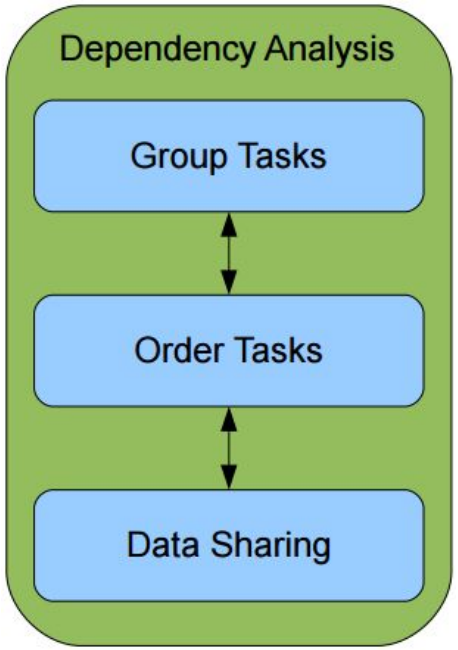
Рекурсивное разбиение

Времени **меньше**: O(log n), вместо O(n)
Вычислений **больше**: O(n * log n), вместо O(n)



Стратегия выполнения

Какой стратегии придерживаться



Окна есть, работники есть..

- Чо делать то?

Если наряду с окна нужно вымыть пол, то окна моются в первую очередь, пол во вторую. Можно вымыть окна половину класса и начать мыть там пол, тем временем домывая окна во второй половине, но тогда придется как-то делить ведра.

SPMD

Single program multiple data

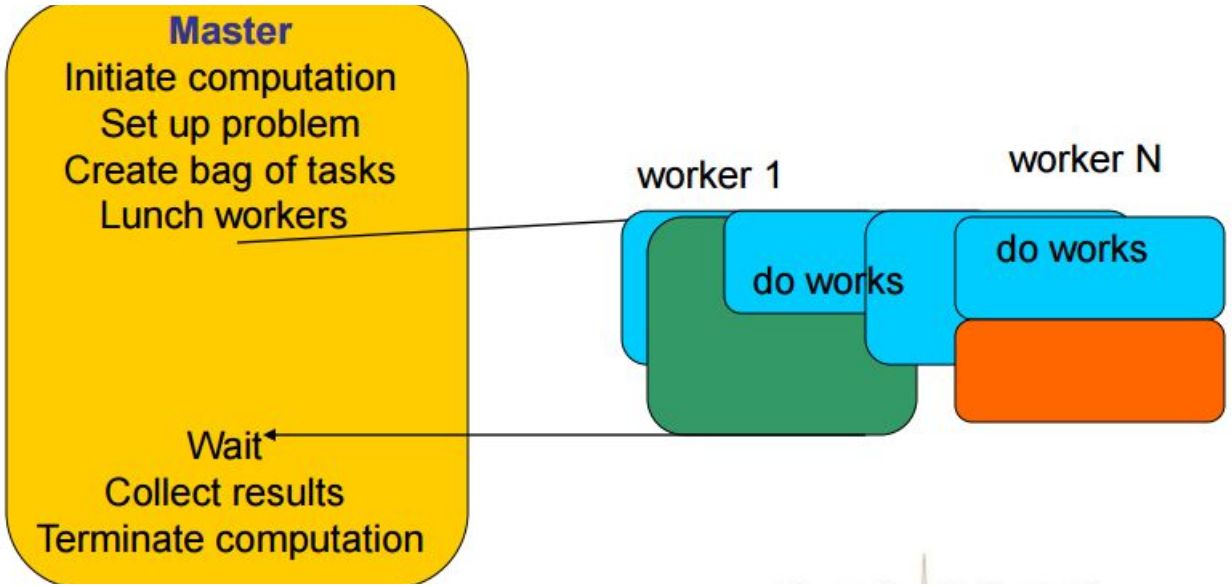
Каждый процесс выполняет одну задачу, но со своим набором данных

1. Инициализация
2. Получение идентификатора задачи
3. Выполнение вычислений
4. Возврат значения
5. **Завершение работы**

Master/Worker (1/2)

Мастер

- создаёт пул задач и исполнителей. Следит, чтобы исполнители работали, а задачи создавались.



Пришла классная руководитель позвонила и собрала родителей, определила какие окна сегодня нужно мыть, а в конце попросила вымыть окна еще и в соседнем классе. Дождалась, когда все закончат, закрыла класс, пошла домой.

Master/Worker (2/2)

Исполнитель

- Получает задачу из очереди задач
- Выполняет задачу
- Помечает задачу как исполненную
- **Идемпотентные задачи рулят**

Послали мыть окна в соседней школе. Ушел и не вернулся. Повторить?

Послали положить денег на счет. Ушел и не вернулся. Повторить?

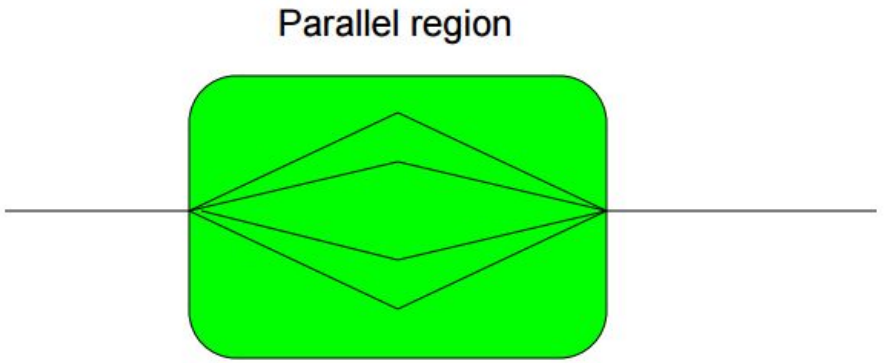
Fork/Join

Похож на Master/Worker

SPMD

Более легковесная версия

Применяется к потокам, а не процессам



Что там про окна: да ничего. Можете сами придумать.

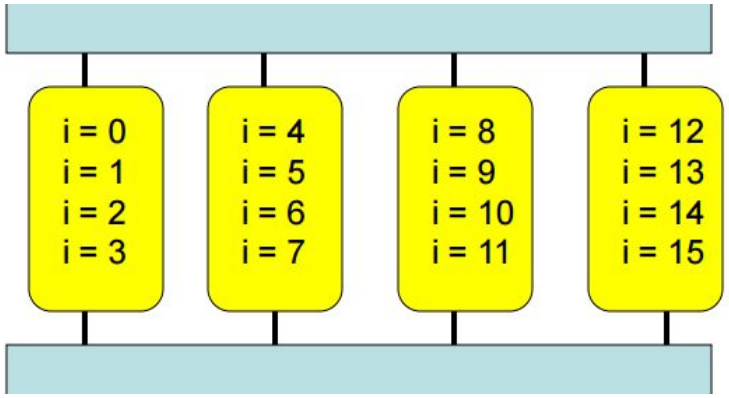
Loop parallelism

Легкий способ ускорения линейной программы

Используйте профилировщик

Есть готовые решения - OpenMP

```
#pragma omp parallel for
For (i = 0; i < 16; i++)
        c[i] = A[i]+B[I];
```

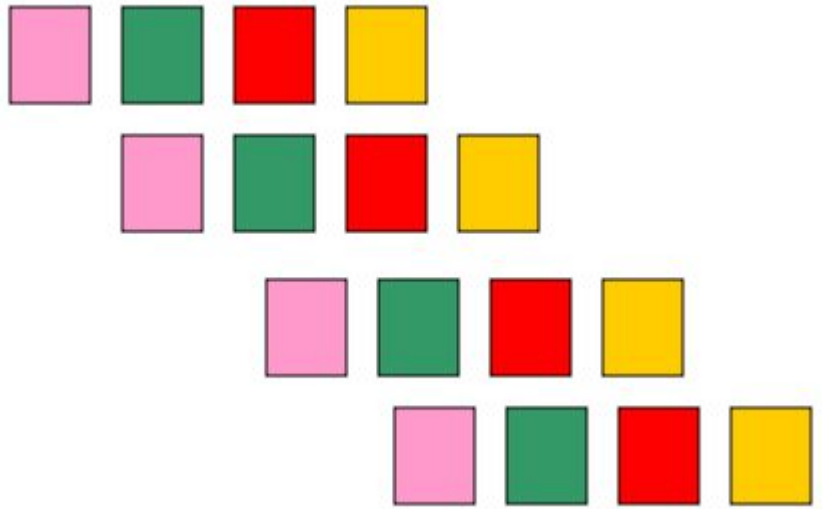


Pipeline

Графический конвейер

Обработка команд в процессоре

Shell pipeline



Вася – несет воду
Петя – моет окно
Света – протирает окно
Коля – уносит воду

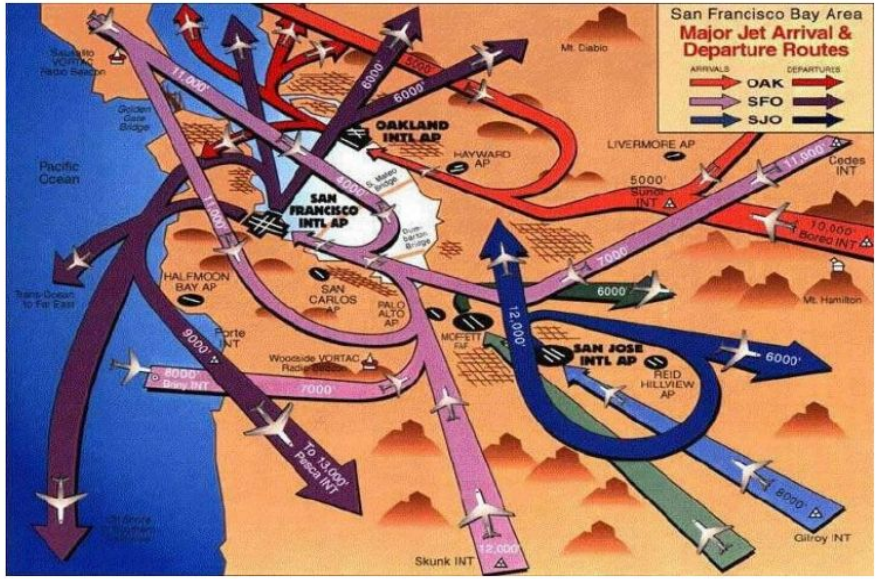
Event based

Использует другие шаблоны

Примитивы синхронизации – Event

Сложно отлаживать

Используйте логирование



Требуется помыть все окна в школе. Школа не достроена. Строители периодически что-то ломают и строят заново. Вода периодически перестает течь из кранов. Окна бьют хулиганы из школы, а вставляет дворник на полставки. Вам периодически звонит начальник с работы и жена из дома.

Стратегии и шаблоны

	Task Parallel.	Divide/ Conquer	Geometric Decomp.	Recursive Data	Pipeline	Event-based
SPMD	☺☺☺☺	☺☺☺	☺☺☺☺	☺☺	☺☺☺	☺☺
Loop Parallel	☺☺☺☺	☺☺	☺☺☺			
Master/Worker	☺☺☺☺	☺☺	☺	☺	☺	☺
Fork/ Join	☺☺	☺☺☺☺	☺☺		☺☺☺☺	☺☺☺☺

Стратегии и реализация

	OpenMP	MPI	CUDA
SPMD	☺ ☺ ☺	☺ ☺ ☺ ☺	☺ ☺ ☺ ☺ ☺
Loop Parallel	☺ ☺ ☺ ☺	☺	
Master/ Slave	☺ ☺	☺ ☺ ☺	
Fork/Join	☺ ☺ ☺		