Задача 4. Рекурсивный параллелизм с контролем числа процессов. Общая схема алгоритма. Контроль числа процессов. Обоснование предложения.

Исследование провёл студент группы 22207 Гордеев Никита

Дата выполнения работы 11.12.2022 (Вариант 4)

Проблема:

При рекурсивном параллелизме необходимо регулировать количество процессов, которые мы создаем, погружаясь в рекурсию, иначе это приведет к превышению числа доступных процессов.

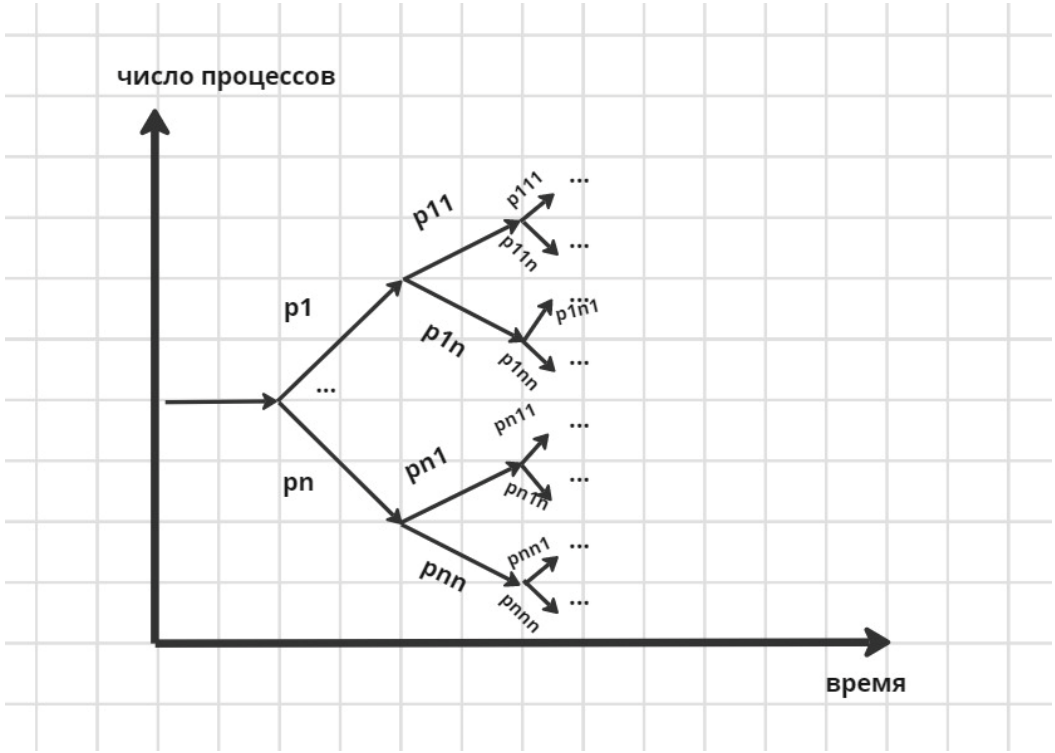
Постановка задач:

На основе рассмотренной на лекции программы с рекурсивным параллелизмом, где для каждого шага рекурсии распараллеливаем текущий процесс, запуская несколько новых:

1. **Предложить общую схему алгоритма рекурсивного параллелизма.**
2. **Расширить эту схему механизмом контроля числа процессов:**
    1. **Задана граница N, (число одновременно выполняемых параллельных процессов не должно превосходить N).**
    2. **Обосновать предложенную схему (какие коллизии возможны из-за параллельности).**

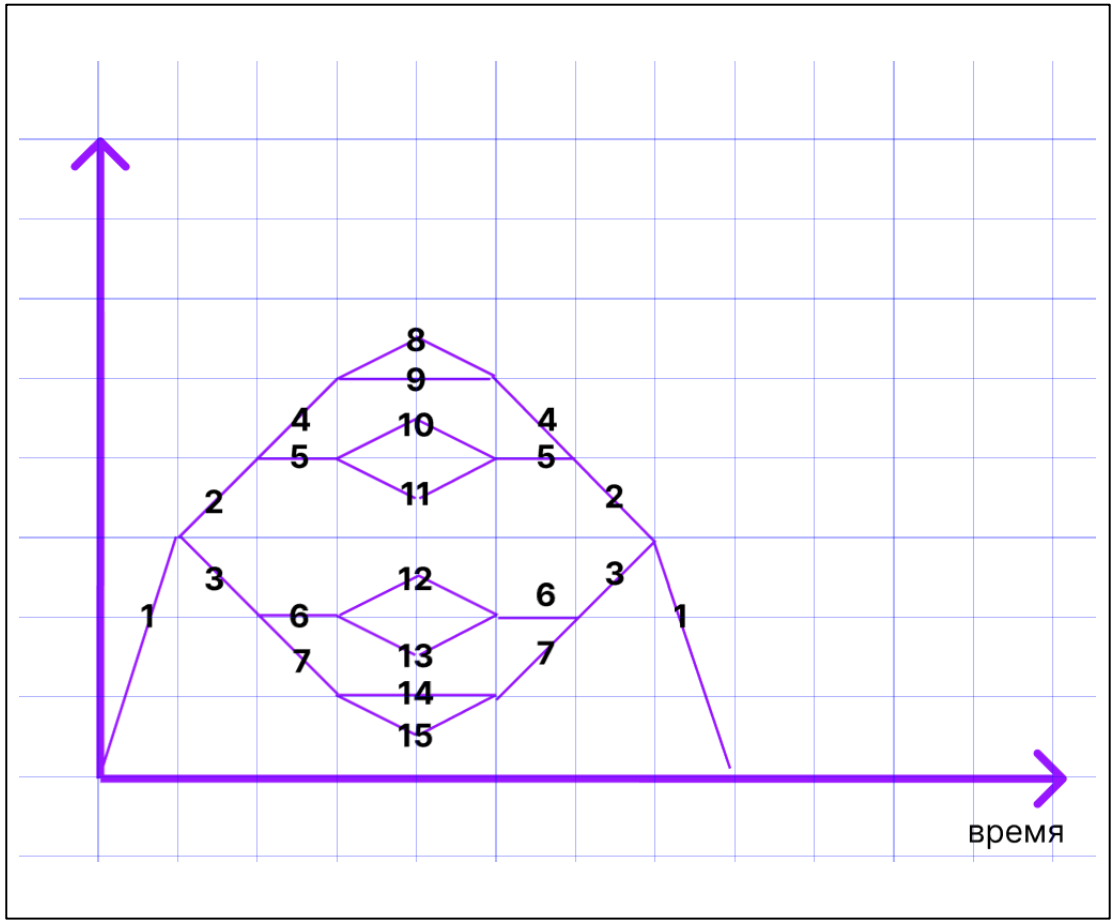
Программа с рекурсивным параллелизмом

```
co
код 1
// код 2
// код 3
// …
// код n
oc
```



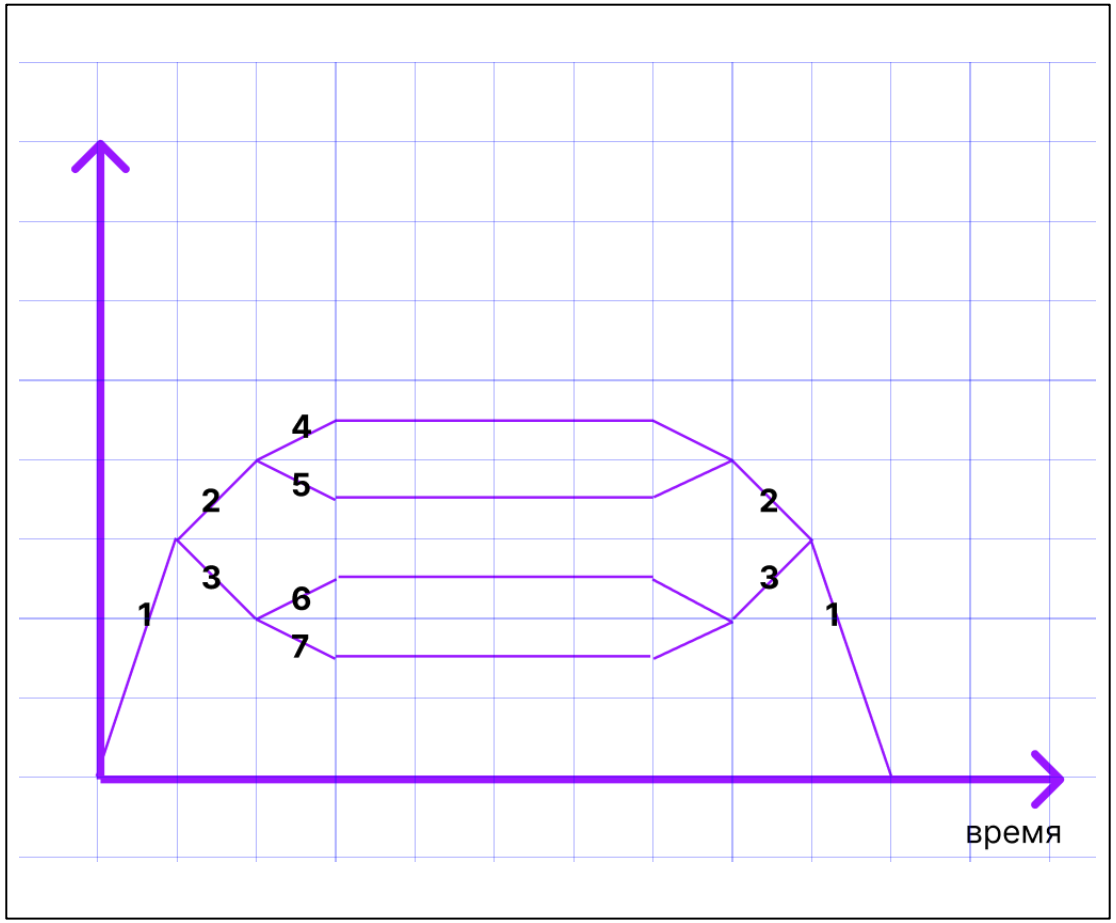
Общая схема алгоритма рекурсивного параллелизма

1) **На каждом этапе рекурсивного алгоритма из текущего процесса запускается несколько новых, которые в свою очередь запускают новые, пока не выполнится условие возврата из функции.**
2) **Одновременно при такой программе будет задействовано очень большое количество процессоров, так как количество параллельных процессов увеличивается с геометрической прогрессией.**



Идея: механизм контроля числа процессов

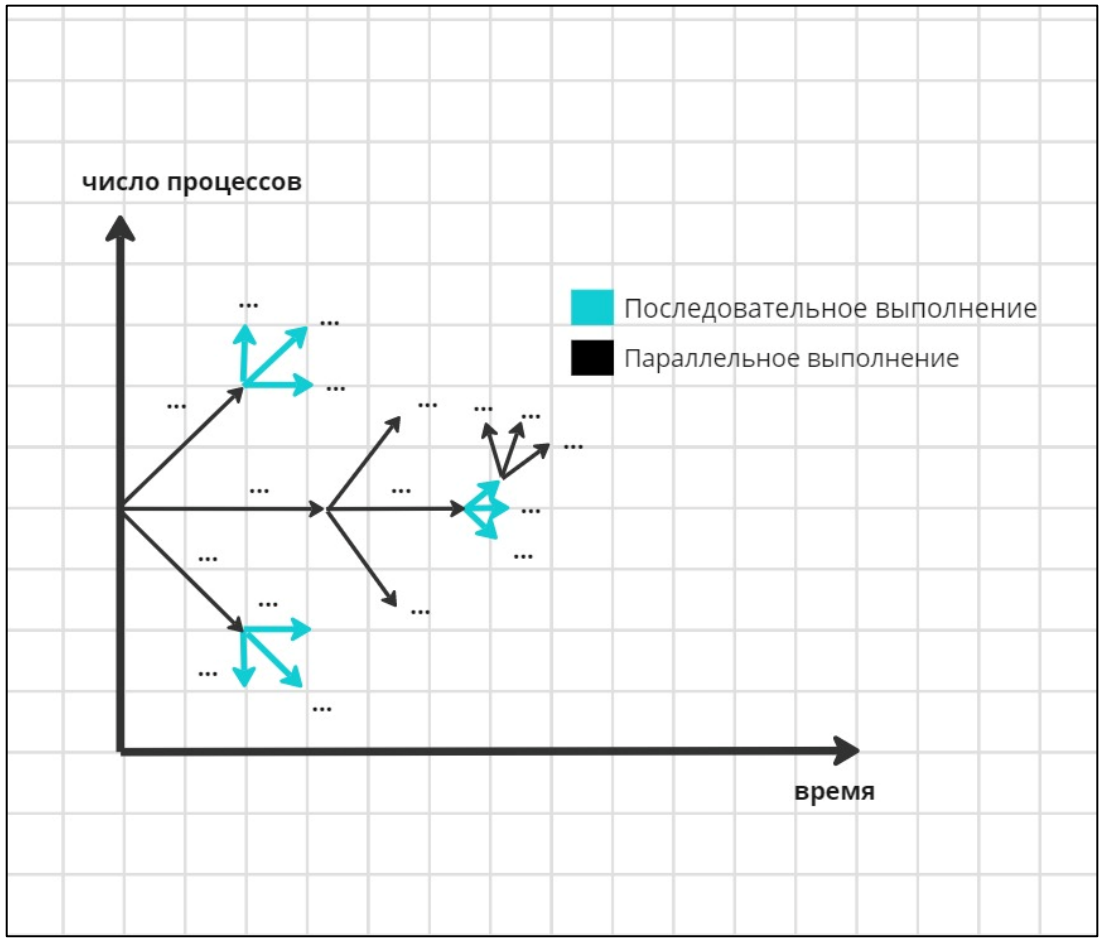
1)] N — количество процессоров
2) Алгоритм с контролем числа процессов работает аналогично версии без контроля до того момента, как количество запущенных процессов не станет равным N, после этого программа будет запускать рекурсию последовательно, пока количество процессов не станет меньше N.
3) Созданные N процессов, рекурсия выполняется последовательно



Программа с механизмом контроля числа процессов

```
function recf() {
    # m число вызовов
    # processes - число текущих процессов
    # N - максимальное количество работающих одновременно процессов

# Увеличиваем количество каждый раз, когда запускаем новый процесс и уменьшаем при выходе из него
    if [processes + m <= N] {
        co [i = 1 to m] {
            recf();
        }
    }
# Если количество процессов при новом рекурсивном вызове меньше N, запускаем рекурсию последовательно
    else {
        for [i = 1 to m] {
            recf();
        }
    }
}
```



Выводы:

1) **При распараллеливании рекурсивных программ количество параллельно работающих процессов увеличивается с большой скоростью.**
2) **Решением этой проблемы является хранение количества запущенных на данный момент процессов.**
3) **Программа распараллеливается при следующем этапе рекурсии только в случае, если есть свободные процессоры.**

Материалы:

- **Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования / Г.Р.Эндрюс. - Москва : Вильямс, 2003. - 512 с (дата обращения: 16.10.2022).**
- **Parallel Programming Languages and Systems // SCHOOL OF INFORMATICS URL: https://www.inf.ed.ac.uk/teaching/courses/ppls/overheads.pdf (дата обращения: 23.10.2022).**

Изменения:

1) **Версия 2:**
    1) **Изменил оформление титульного слайда**
    2) **Перерисовал графики**
2) **Версия 3:**
    1) **Выделил важные фрагменты кода**
    2) **Изменил размер шрифта на больший**
3) **Версия 4**
    1) **Убрал код реальной программы, заменил на псевдокод**
    2) **Перерисовал графики**