

# Теория графов. Первая презентация

Ермолович Анна



# Введение

**Поиск в ширину (BFS)** — алгоритм обхода графа, позволяющий находить кратчайшие пути от стартовой вершины до всех остальных вершин в невзвешенном графе.

**Multiple-source BFS (MS-BFS)** — обобщение BFS, при котором поиск запускается одновременно из нескольких стартовых вершин.

**MS-Parent BFS** — вариант MS-BFS, в котором для каждой вершины и каждого источника сохраняется родительская вершина на одном из кратчайших путей.

# Multiple-source Parent BFS

## Задача:

Дан неориентированный невзвешенный граф

$$G=(V,E)$$

и задано множество стартовых вершин

$$S=\{s_0,s_1,\dots,s_{k-1}\}$$

Требуется:

для каждой вершины  $v \in V$  и для каждой стартовой вершины  $s \in S$ , если  $v$  достижима из  $s$ , определить родительскую вершину  $v$  на одном из кратчайших путей из  $s$  в  $v$ .

# Характеристики вычислительной машины

- **Процессор:** AMD Ryzen 7 5800H (8 ядер, 16 потоков)
  - **L1-кэш:** 64 КВ (на ядро)
  - **L2-кэш:** 512 КВ (на ядро)
  - **L3-кэш:** 16 МВ (общий)
- **RAM:** 32 GB
- **GPU:** RTX 3070 (8 GB видеопамяти)
- **Операционная система:** Ubuntu 24.04.2

# Dataset SNAP

Description	Nodes	Edges
Twitch gamers network	168,114	6,797,557
Gemsec Facebook dataset	134,833	1,380,293
Gemsec Deezer dataset	143,884	846,915
Twitch social networks	34,118	429,113
Github developer network	37,700	289,003
Facebook page-page network	22,470	171,002
Facebook social circles	4,039	88,234
Deezer Europe social network	28,281	92,752
LastFM Asia social network	7,624	27,806

# Эксперимент

# Эксперимент

Цель: Исследовать масштабируемость Pregel+ для алгоритма MS-Parent BFS — как изменяется время выполнения и эффективность при увеличении числа процессов — на разных графах из датасета.

Ход эксперимента:

1. Для каждого выбранного графа из датасета запускаем Pregel+ с использованием `mpirun` на 1, 2, 4, 8 и 16 процессах.
2. Измеряем время выполнения, фиксируем результаты.
3. Строим графики для каждого графа:
  - а. Зависимость времени выполнения от числа процессов
  - б. Зависимость ускорения и эффективности от числа процессов
4. Сравниваем результаты и делаем выводы о масштабируемости Pregel+.

Проблемы:

- Масштабируемость может различаться для разных графов (разная структура и размер).
- Из-за параллельной природы вычислений возможны колебания времени между запусками.

# Метрики оценки масштабируемости

*Масштабируемость* — это способность системы к увеличению производительности при добавлении новых ресурсов или способность эффективно перераспределять имеющиеся ресурсы при увеличении нагрузки.

## Ускорение:

Показывает, во сколько раз выполнение стало быстрее при  $p$  процессах, чем при 1:

$$S(p) = \frac{T(1)}{T(p)}$$

## Эффективность:

Показывает, насколько эффективно используются вычислительные ресурсы:

$$E(p) = \frac{S(p)}{p}$$

## Критерии анализа:

1. Строим графики зависимости  $T(p)$ ,  $S(p)$ ,  $E(p)$  от  $p$ .
2. Ищем, при каком  $p$  эффективность заметно падает (*например*,  $< 0.5$ ).
3. Оцениваем влияние аппаратной многопоточности (SMT) на масштабируемость Pregel-алгоритмов.



# Реализация на Pregel+

# Pregel+

**Pregel+** — библиотека для параллельной обработки графов по модели Bulk Synchronous Parallel (BSP).

Она оптимизирует передачу сообщений между вершинами за счет:

- *Vertex mirroring* — локального хранения информации о соседях,
- *Request-respond* — отправки сообщений только при необходимости, а не всем вершинам подряд.

**Почему Pregel+ подходит для реализации алгоритма:**

- Поддерживает параллельное выполнение алгоритмов на графах и легко масштабируется на разное число процессов.
- Снижает накладные расходы на передачу сообщений между процессами за счет *vertex mirroring* и *request-respond*.

# Pregel+ и MPI

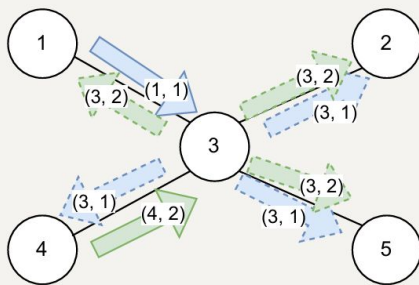
## Как будем обеспечивать распределенность на одной машине:

1. Мы будем использовать `mpirun` для запуска нескольких независимых процессов на одной машине.
2. Каждый процесс имеет свою отдельную память. Общей памяти между процессами нет.
3. Pregel+ внутри использует MPI для обмена сообщениями между worker`ами.
4. Таким образом, запрос-ответ (request-respond) и зеркалирование вершин (vertex mirroring) продолжают работать, просто передача сообщений идёт через MPI-сообщения.

Это означает, что даже без кластера мы можем корректно изучать поведение Pregel+ при разном числе процессов

# MS-Parent BFS и Pregel+

- Из всех стартовых вершин отправляем сообщение соседям (номер\_вершины, стартовая\_группа)
- При принятии сообщения, если в итоговой таблице `parents[v][startGroup]` нет записи:
  - Добавляем запись в таблицу
  - Распространяем сообщение (заменяя номер\_вершины)



# MS-Parent BFS и Pregel+

## Инициализация (супершаг 1)

каждая стартовая вершина  $s_i$   
рассылает сообщения соседям:  
(startGroup =  $i$ , parent =  $s_i$ )

Сортировка сообщений  
обеспечивает  
детерминированный выбор  
родителя при равных  
расстояниях

## Основные супершаги

**Каждая вершина:**

- 1) получает сообщения вида (startGroup, parent)
- 2) сортирует их по:
  - startGroup
  - parent id
- 3) для каждого startGroup:
  - если родитель ещё не задан:
    - сохраняет родителя
    - распространяет сообщение дальше
  - иначе — игнорирует сообщение