

67-я студенческая научная конференция ЮУрГУ

# Реализация теста Graph500 для параллельной СУБД PargreSQL

Сафонов А. Ю. группа ВМИ-456

Научный руководитель:  
кандидат физ.-мат. наук Цымблер М. Л.



Home

Complete Results

Benchmarks

Green Graph 500

Log In

# The Graph 500 List

## Top 10 (November 2013)

Rank	Machine
1	DOE/NNSA/LLNL Sequoia - Lawrence Livermore National Laboratory (65536 nodes, 1048576 cores)
2	DOE/SC/Argonne National Laboratory Mira - Argonne National Laboratory (49152 nodes, 786432 cores)
3	JUQUEEN - Forschungszentrum Juelich (FZJ) (16384 nodes, 262144 cores)
4	K computer - RIKEN Advanced Institute for Computational Science (AICS) (65536 nodes, 524288 cores)
5	Fermi - CINECA (8192 nodes, 131072 cores)
	Tianhe-2

## November 2013

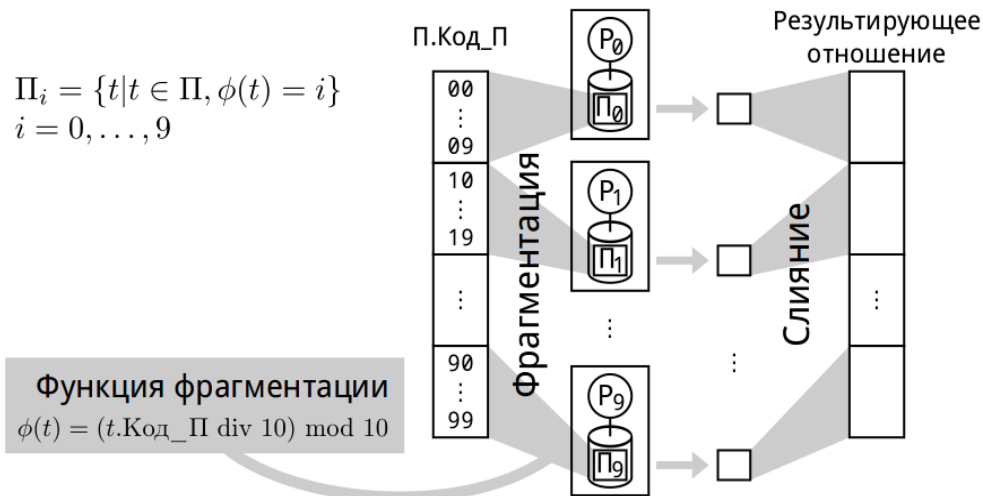
No.	Rank	Machine	Installation Site	<a href="#">Number of nodes</a>	<a href="#">Number of cores</a>	<a href="#">Problem scale</a>	<a href="#">GTEPS</a> ▼
1	1	DOE/NNSA/LLNL Sequoia (IBM - BlueGene/Q, Power BQC 16C 1.60 GHz)	Lawrence Livermore National Laboratory	65536	1048576	40	15363
2	2	DOE/SC/Argonne National Laboratory Mira (IBM - BlueGene/Q, Power BQC 16C 1.60 GHz)	Argonne National Laboratory	49152	786432	40	14328
3	3	JUQUEEN (IBM - BlueGene/Q, Power BQC 16C 1.60 GHz)	Forschungszentrum Juelich (FZJ)	16384	262144	38	5848
4	4	K computer (Fujitsu - Custom supercomputer)	RIKEN Advanced Institute for Computational Science (AICS)	65536	524288	40	5524.12
5	5	Fermi (IBM - BlueGene/Q, Power BQC 16C 1.60 GHz)	CINECA	8192	131072	37	2567
6	6	Tianhe-2 (MilkyWay-2) (National University of Defense Technology - MPP)	Changsha, China	8192	196608	36	2061.48
		Turing (IBM -					

# Цель и задачи исследования

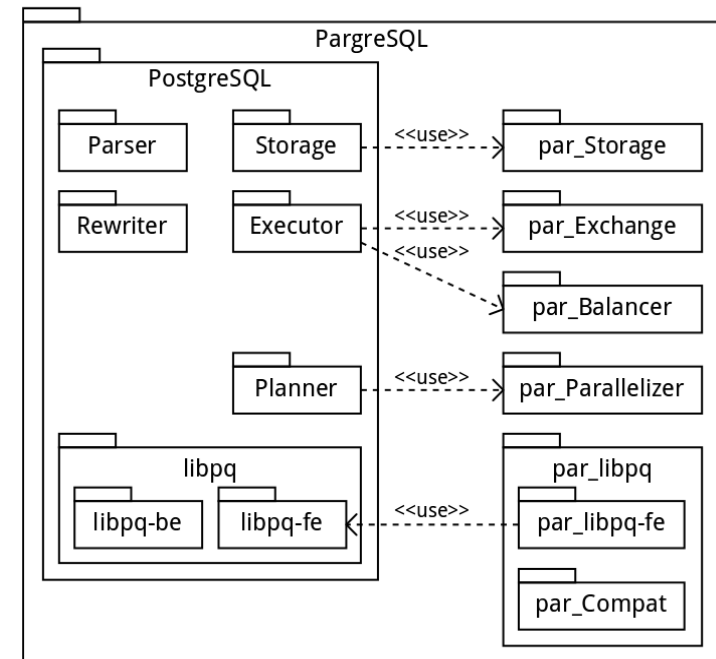
- Цель
  - Оценка эффективности параллельной СУБД PargreSQL на задачах интенсивной обработки данных с помощью сравнительного теста Graph500
- Задачи
  - Изучить архитектуру параллельной СУБД PargreSQL и спецификацию теста Graph500
  - Разработать схему базы данных для хранения графа и промежуточных данных в соответствии со спецификацией теста Graph500
  - Выполнить проектирование и разработку алгоритмов на языке SQL, реализующих тест Graph500 для параллельной СУБД PargreSQL
  - Провести вычислительные эксперименты на суперкомпьютере “Торнадо ЮурГУ”, исследующие эффективность параллельной СУБД PargreSQL на тесте Graph500

# Параллельная СУБД PargreSQL

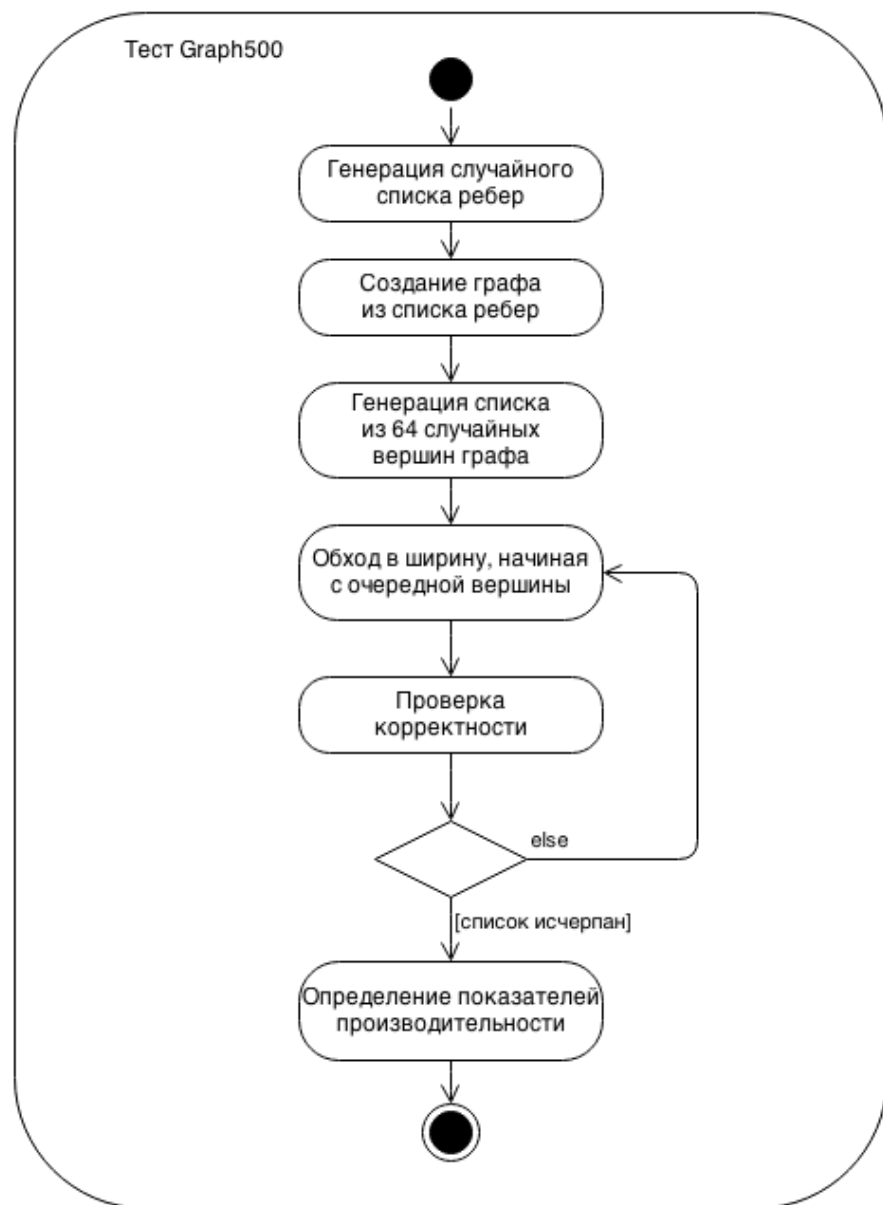
- Фрагментный параллелизм



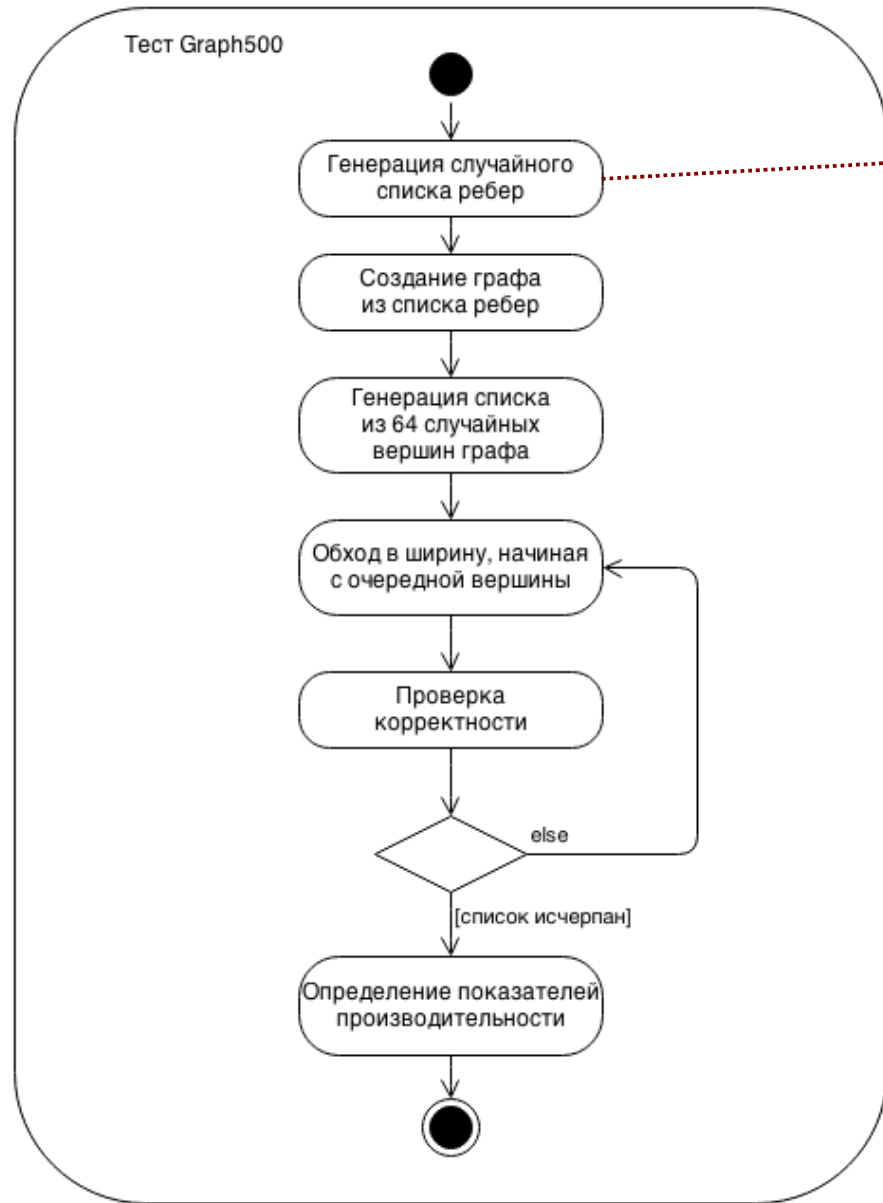
- Архитектура PargreSQL



# Тест Graph500



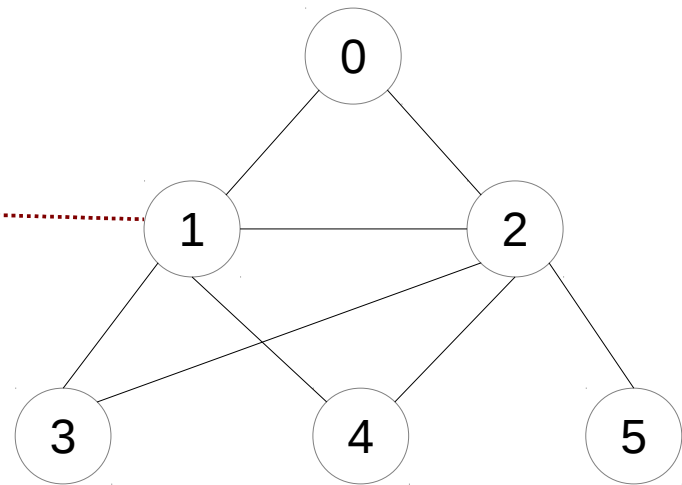
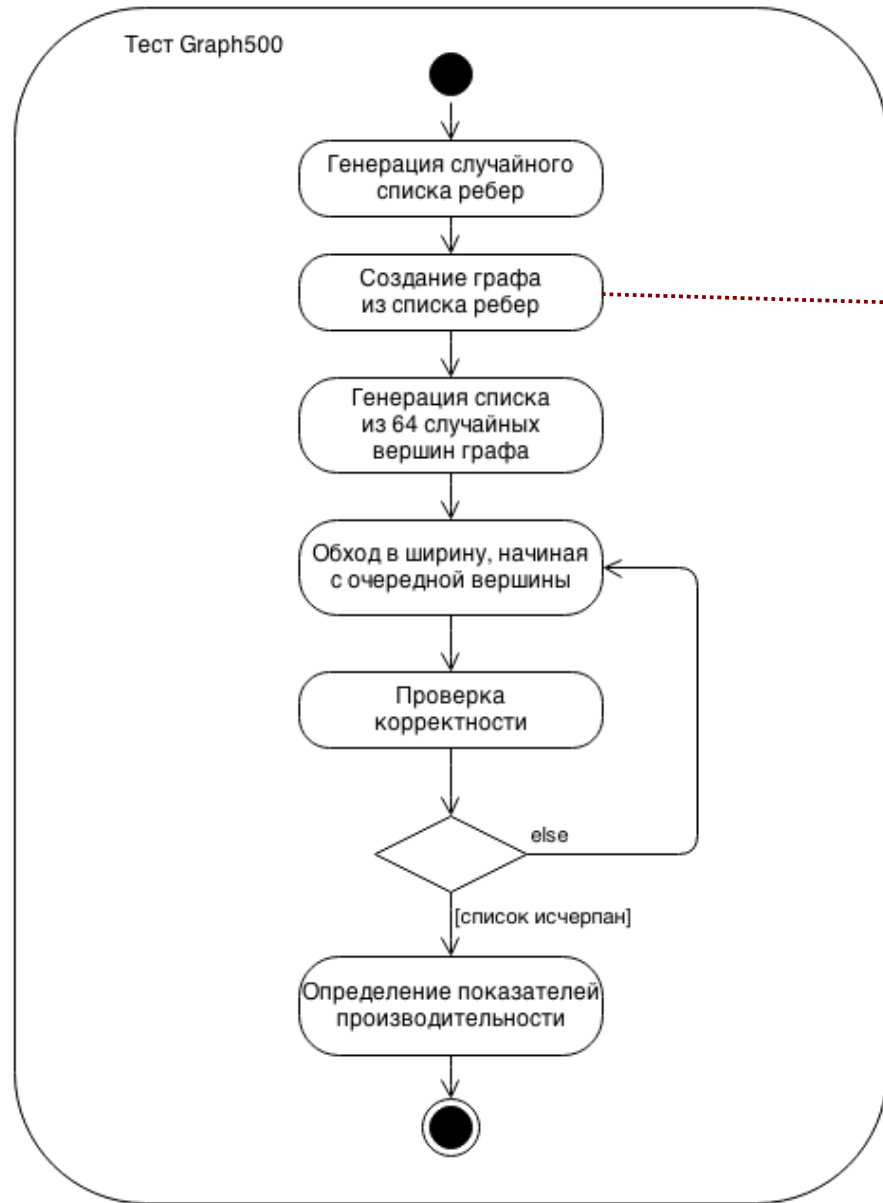
# Тест Graph500



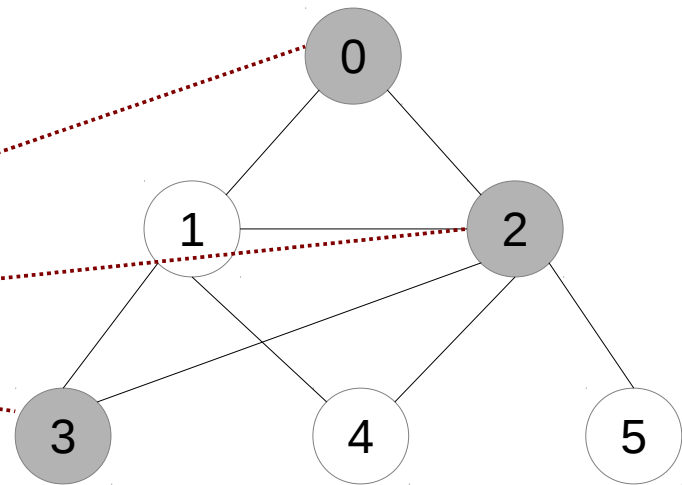
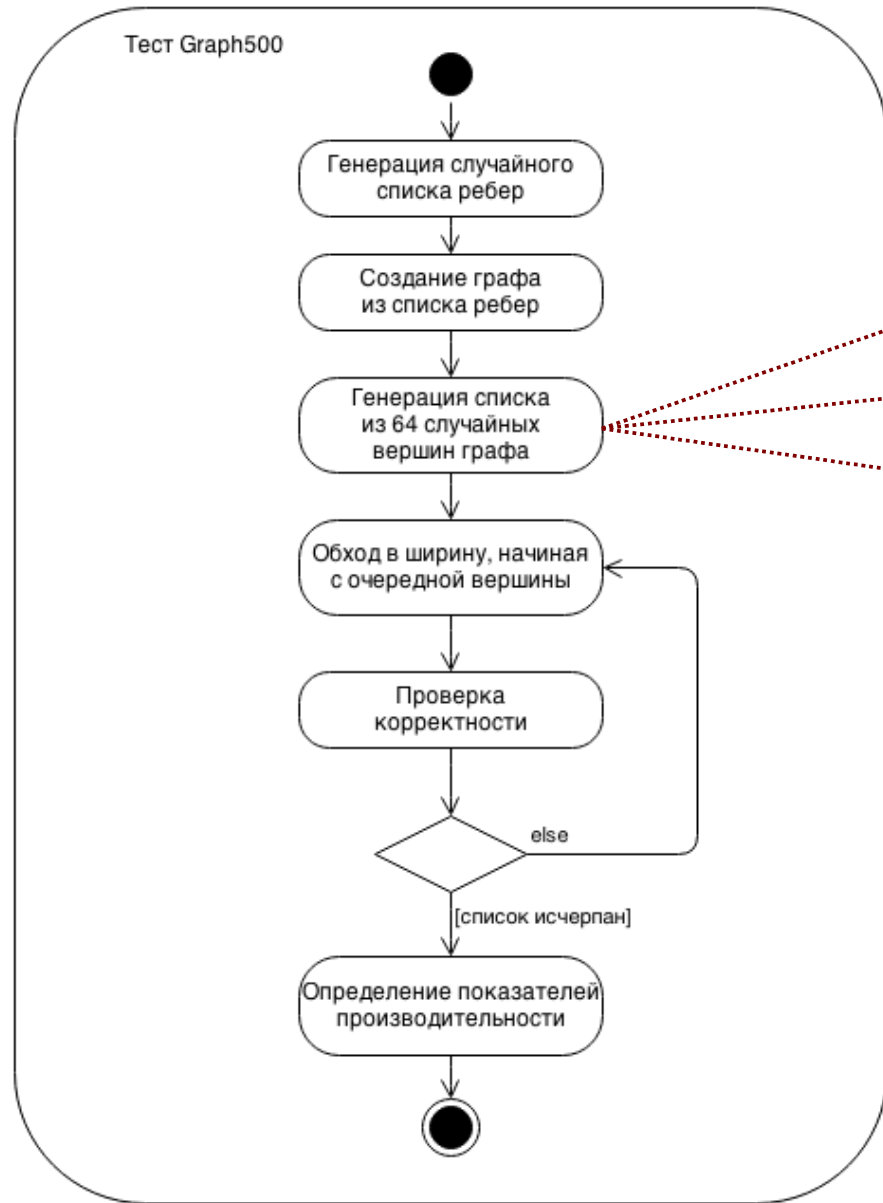
Начальная вершина	Конечная вершина
0	1
2	0
1	2
3	1
4	2
4	1
2	5
3	2

Класс задачи	Количество вершин	Количество ребер	Память
Игрушечный (уровень 10)	$2^{26}$	$2^{30}$	17 ГБ
Минимальный (уровень 11)	$2^{29}$	$2^{33}$	137 ГБ
Маленький (уровень 12)	$2^{32}$	$2^{36}$	1 ТБ
Средний (уровень 13)	$2^{36}$	$2^{40}$	17 ТБ
Большой (уровень 14)	$2^{39}$	$2^{43}$	140 ТБ
Огромный (уровень 15)	$2^{42}$	$2^{46}$	1 ПБ

# Тест Graph500

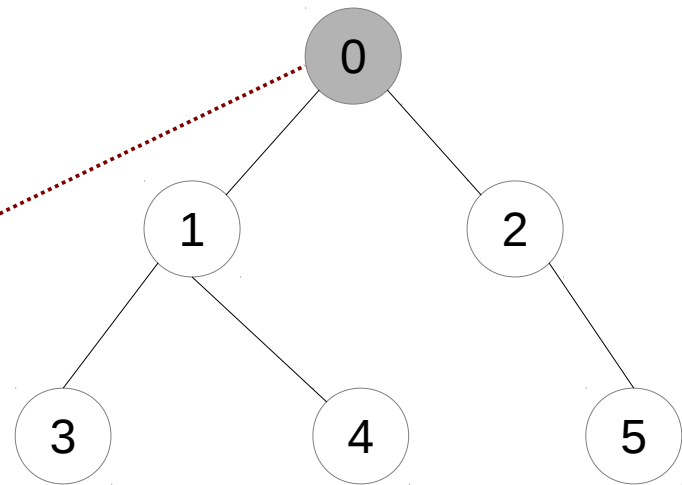
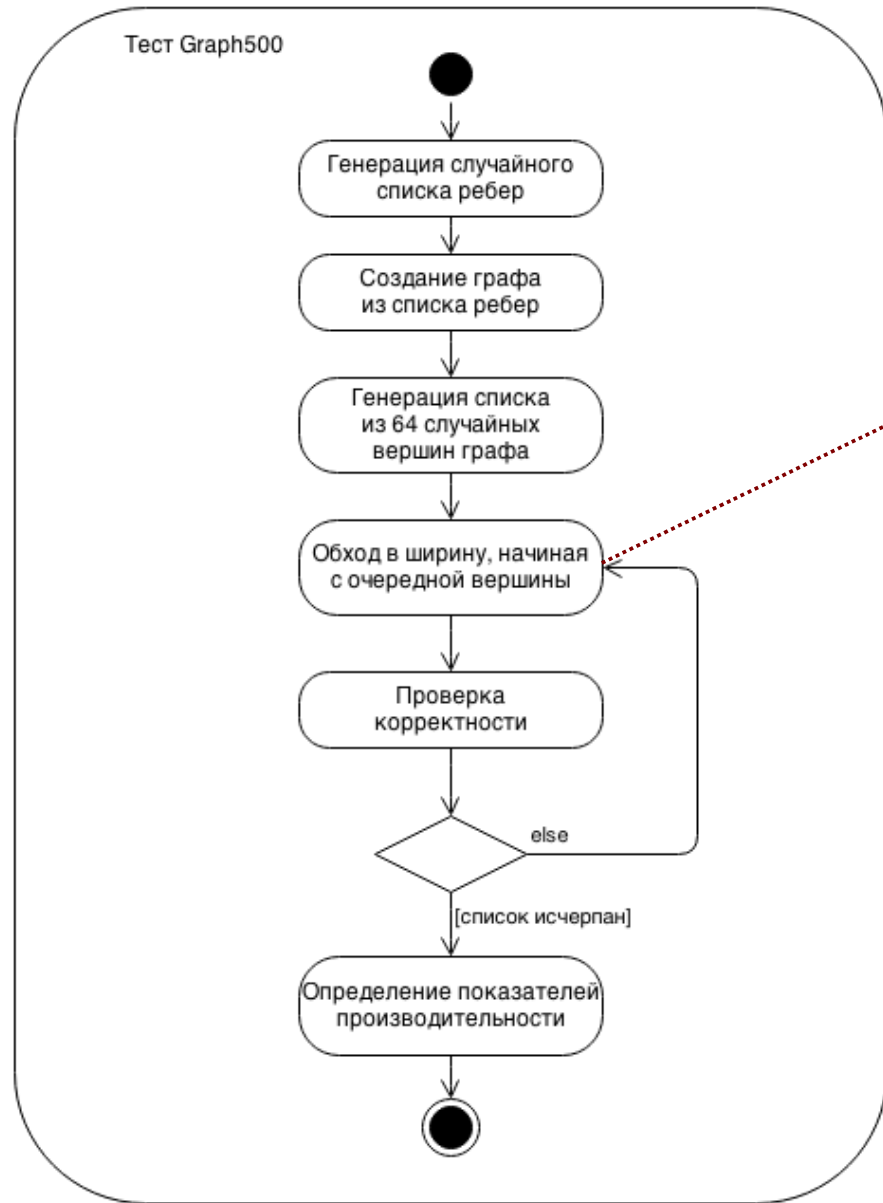


# Тест Graph500

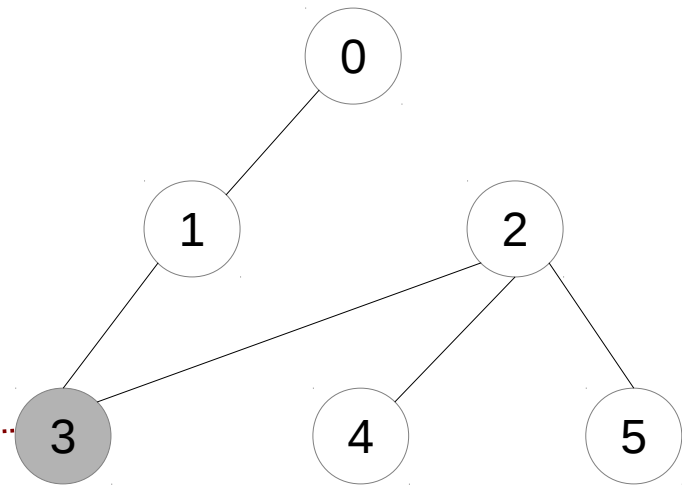
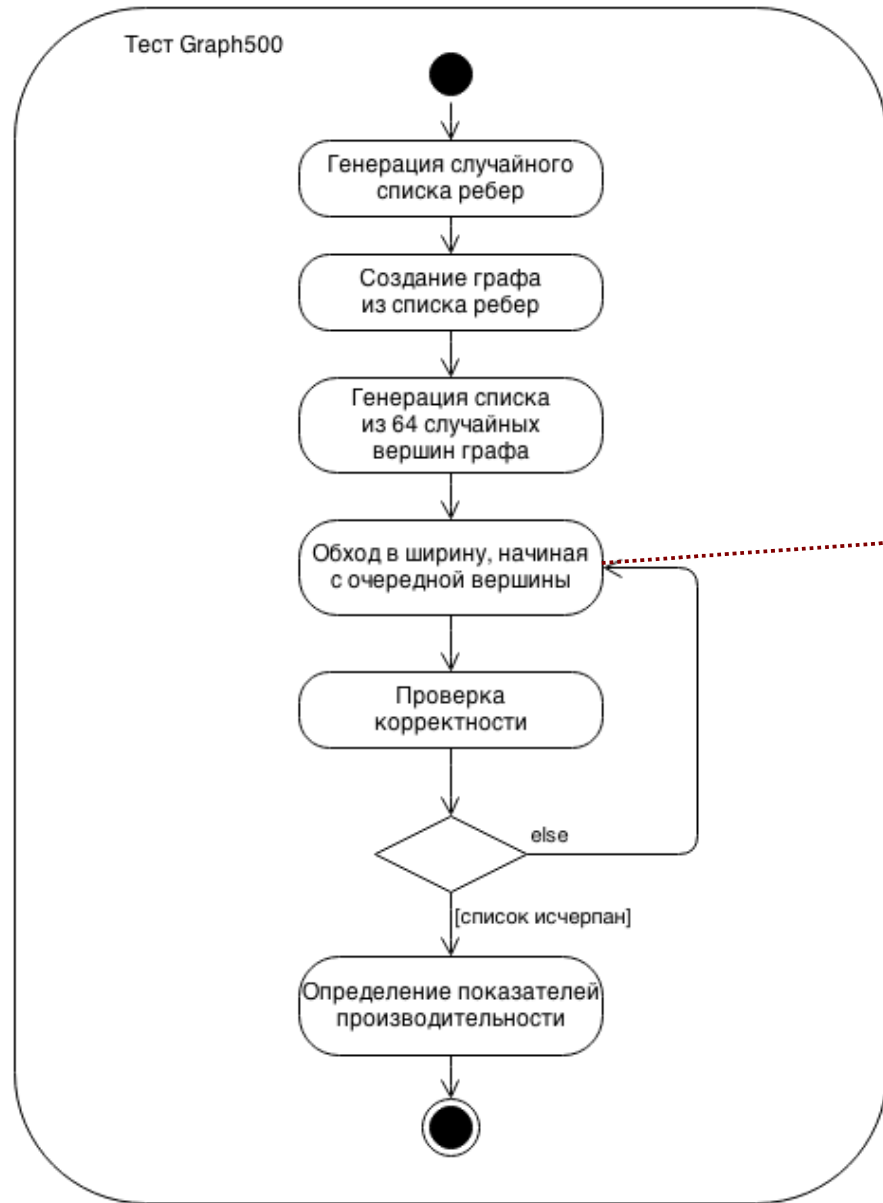




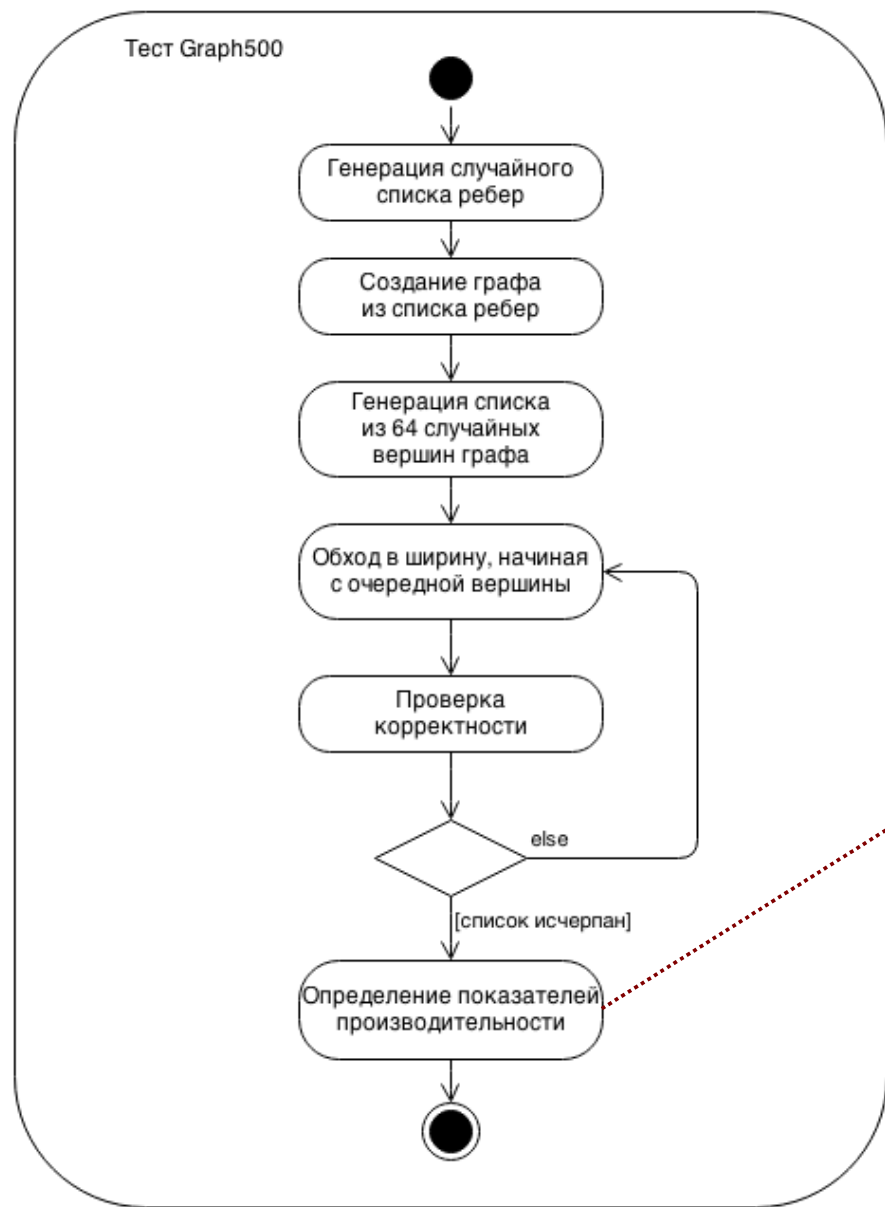
# Тест Graph500



# Тест Graph500

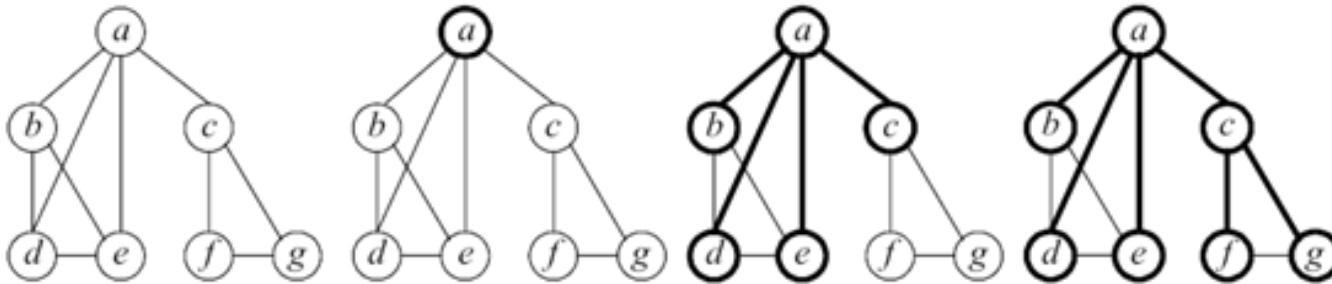


# Тест Graph500



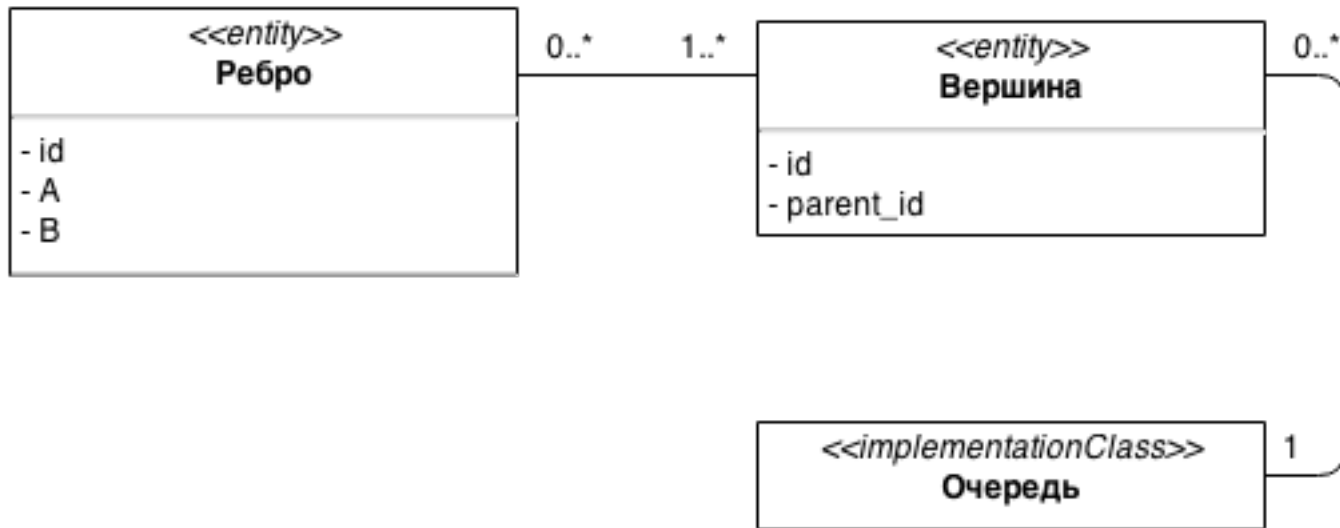
Производительность:  
412 GTEPS

# Обход в ширину



```
BFS (start_node)
{
    for (all nodes i)
        visited[i] = false;
    queue.push(start_node);
    visited[start_node] = true;
    while( !queue.empty() )
    {
        node = queue.pop();
        foreach(child in expand(node))
        {
            if(visited[child] == false)
            {
                queue.push(child);
                visited[child] = true;
            }
        }
    }
}
```

# Схема данных



# Таблица хранения

```
create table edge (  
    id bigserial primary key,      -- порядковый номер ребра  
    a bigint,                      -- первая вершина ребра  
    b bigint,                      -- вторая вершина ребра  
    parent bigint,                 -- родитель вершины  
    queue bigint                  -- номер вершины в очереди  
);
```

Функция фрагментации =  $\text{id} \% [\text{Количество узлов}]$

# Реализация поиска в ширину с использованием API PargreSQL

```
BFS (start_node) {  
    for (all nodes i) {  
        parent[i] = -1;  
    }  
  
    queue.push(start_node);  
    parent[start_node] = start_node;  
  
    while( !queue.empty() ) {  
        node = queue.pop();  
        foreach(child in expand(node)) {  
            if(parent[child] == -1) {  
                queue.push(child);  
                parent[child] = node;  
            }  
        }  
    }  
}
```

```
#include <par_libpq-fe.h>  
  
int main () {  
    conn = PQconnectdb(  
        "dbname=alexander host=localhost user=alexander password=*****");  
  
    long long lastQueueNumber = 0; // номер последнего добавленного элемента  
    long long currentQueueNumber = 0; // номер текущего элемента  
  
    PQexec(conn,"update edge set parent=-1, queue=NULL");  
    PQexec(conn,"update edge set parent=%ld, queue=%ld where a=%ld",  
        startNode,lastQueueNumber,startNode);  
  
    while (currentQueueNumber <= lastQueueNumber) {  
        children = PQexec(conn,  
            "select distinct a,b from edge where queue=%ld",  
            currentQueueNumber);  
        long long childCount = PQntuples(children);  
        long long node = PQgetvalue(children, 0, A_COLUMN);  
        for (long long i = 0; i < childCount; ++i)  
        {  
            long long child = PQgetvalue(children, i, B_COLUMN);  
            long long parent = PQexec(conn,  
                "select distinct parent from edge where a=%ld",child);  
            if (parent == -1) { // вершина еще не добавлена в очередь  
                ++lastQueueNumber;  
                PQexec(conn,  
                    "update edge set queue=%ld,parent=%ld WHERE a=%ld"  
                    lastQueueNumber,node,child);  
            }  
        }  
        ++currentQueueNumber;  
    }  
    return 0;  
}
```

# Проверка корректности

Для каждого ключа поиска после завершения обхода графа в ширину нужно проверить, что соблюдены условия:

1. Полученное BFS-дерево является деревом и не содержит циклов.
2. Каждое ребро BFS-дерева соединяет вершины, чей уровень при обходе в ширину различается ровно на 1.
3. Каждое ребро из входного списка ребер соединяет вершины, уровень которых в BFS-дереве различается не более чем на единицу или же обе эти вершины не включены в BFS-дерево.
4. BFS-дерево связывает все вершины данной компоненты связности.
5. Каждая вершина и ее родитель соединены ребром в исходном графе.



# Оценка производительности

TEPS (traversed edges per second) — количество пройденных ребер за секунду:

$$TEPS = m / time_{K2}$$

$m$  — количество ребер в итоговом BFS-дереве  
 $time_{K2}$  — время работы алгоритма обхода в ширину

# Текущие результаты

- Изучена архитектура параллельной СУБД PargreSQL и спецификация теста Graph500
- Разработана схема базы данных для хранения графа и промежуточных данных в соответствии со спецификацией теста Graph500
- Выполнено проектирование и разработка алгоритмов на языке SQL, реализующих тест Graph500 для параллельной СУБД PargreSQL