ИНСТИТУТ
ЦИФРОВОГО
ОБРАЗОВАНИЯ
МГПУ

Лекция 11-1. Обработка графов
Apache Giraph

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Примеры графов

- Компьютерные сети
- Социальные сети
- Семантические графы
- Карты
- Маршруты

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Алгоритмы

- Вычисление кратчайшего пути
- Определение наиболее значимых узлов с PageRank
- Предсказание рейтингов
- Определение сообществ

ИНСТИТУТ
ЦИФРОВОГО
ОБРАЗОВАНИЯ
МГПУ

Apache Giraph

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Основные темы

- Назначение и особенности
- Модель данных
- API
- Архитектура
- Запуск вычислений
- Отказоустойчивость

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Словарь

- Vertex
- Edge
- Vertex State
- BSP model
- Superstep
- Partition
- Master
- Worker
- Coordination service

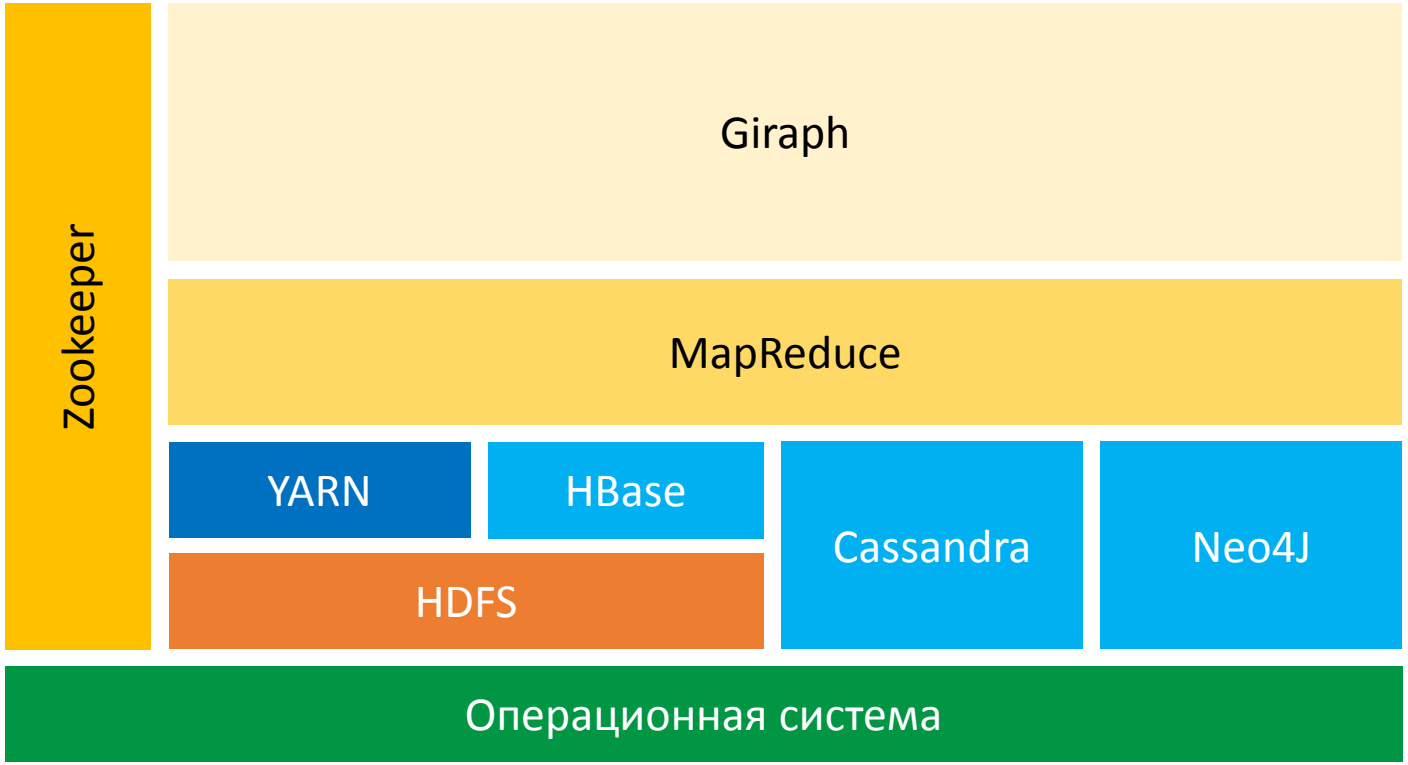
ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Назначение и особенности

- Giraph предназначен для реализации итеративных алгоритмов над графами
- с возможностью распараллеливания процессов обработки
- отказоустойчив и
- способен обрабатывать графы больших размеров
- Используется для batch вычислений (запускается после предварительного накопления значительного объема исходных данных)
- Не предназначен для быстрых интерактивных ad hoc запросов

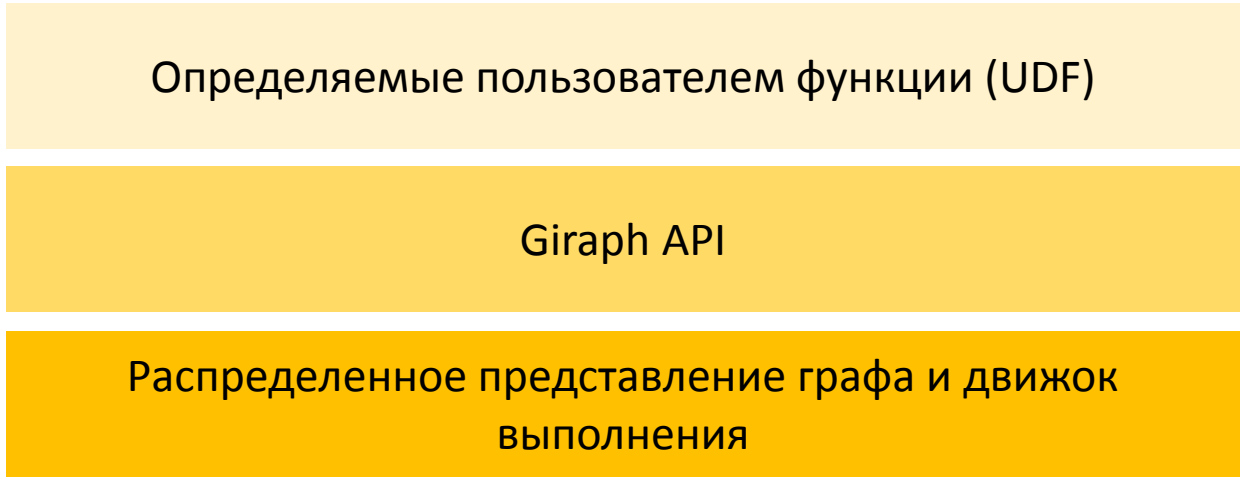
ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Стек



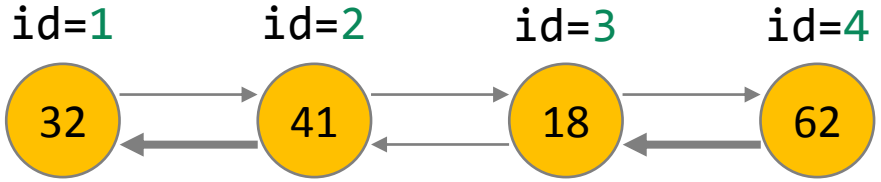
Программная модель

- Giraph использует программную модель, ориентированную на вершины
- Разработчику необходимо написать код, который будет выполняться на каждой вершине
- Программная модель скрывает сложности программирования параллельной и распределенной системы
- Giraph выполняет ваш код прозрачно на множестве доступных машин в параллельной манере

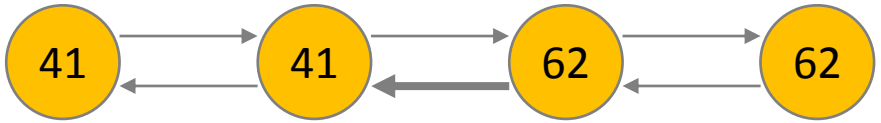


Пример. Поиск наибольшего значения

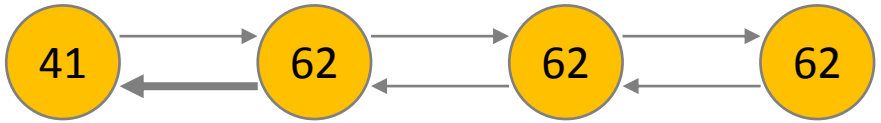
Итерация 1



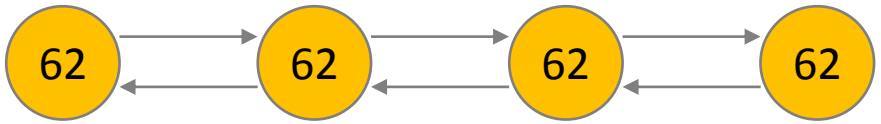
Итерация 2



Итерация 3



Итерация 4

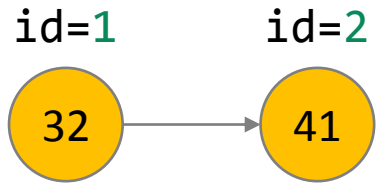


ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Модель данных

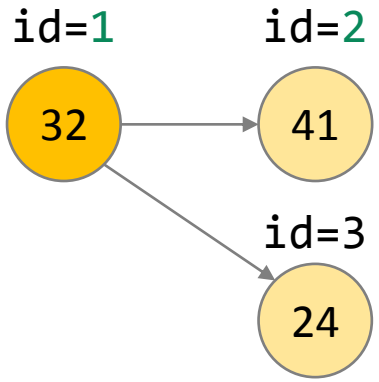
Каждая вершина имеет:

- уникальный идентификатор (id): integer, string и пр.
- значение: integer, string и пр.
- некоторое количество исходящих ребер, которые указывают на другую вершину



Каждое ребро может иметь значение

Вершина в Giraph




```
class Vertex:
    function getId() # id вершины
    function getValue() # значение вершины
    function setValue(value) # установка значения вершине
    function getEdges() # все исходящие ребра
    function getNumEdges() # количество исходящих ребер
    # значение первого ребра, соед. с вершиной targetId
    function getEdgeValue(targetId)
    # присвоение значения первому ребру, соед. с вершиной targetId
    function setEdgeValue(targetId, value)
    function getAllEdgeValues(targetId) # все значения ребер, соед. с вершиной targetId
    function voteToHalt() # остановка вычисления для вершины
    function addEdge(edge) # добавление ребра
    function removeEdges(targetId) # удаление всех ребер, соед. с вершиной targetId
```

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

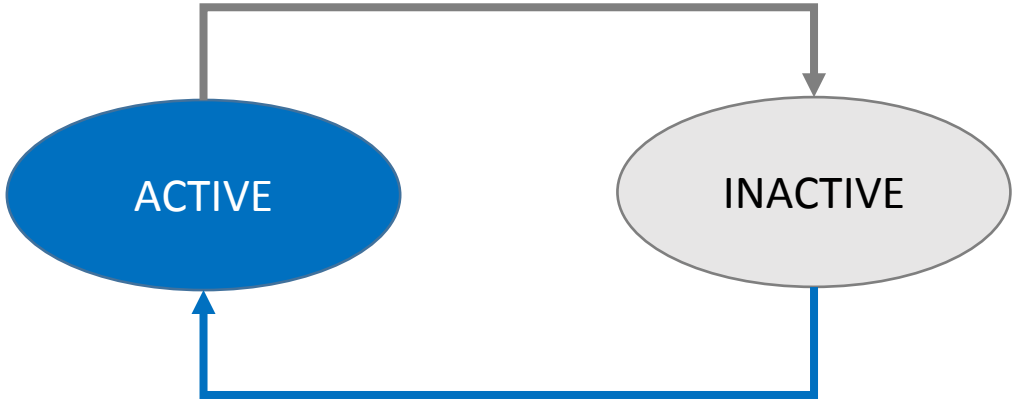
Ребро в Giraph

```
class Edge:
    function getTargetVertexId() # id целевой вершины (с которой соед.)
    function getValue() # значение ребра
    function setValue(value) # установка значения ребра
```

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

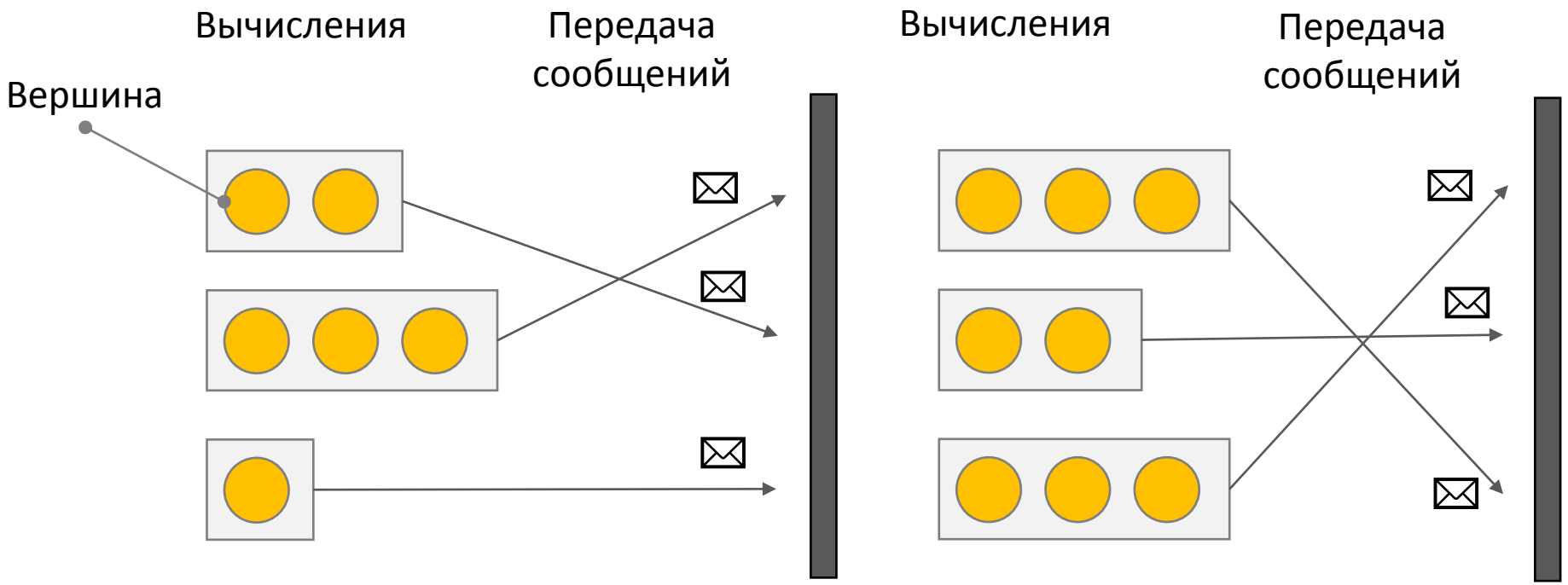
Состояния вершины

Вершина выполняет команду остановки (vote to halt)



Вершина получает сообщение

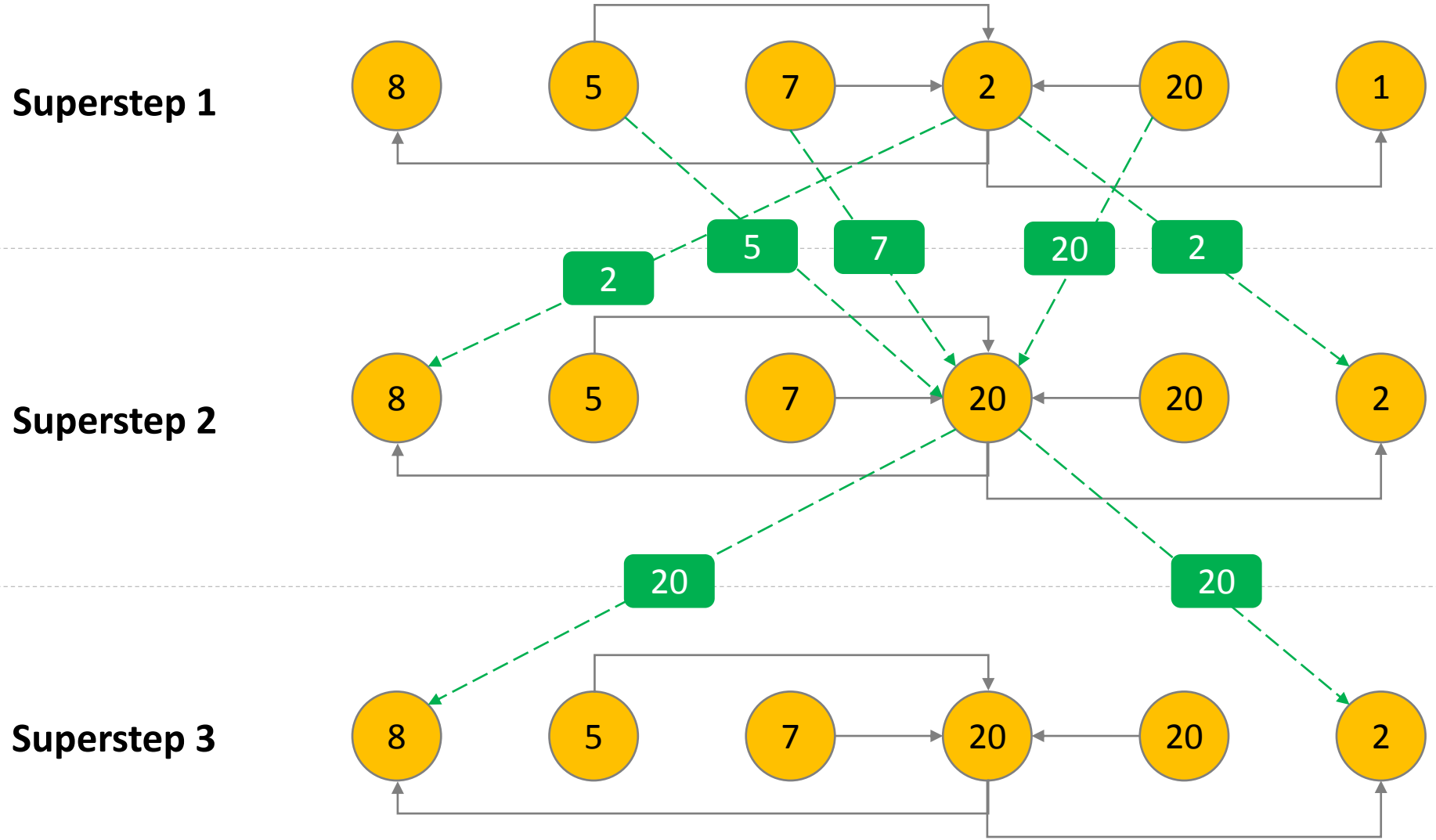
Синхронизирующая модель Giraph



Superstep 1

Superstep 2

Пример


Superstep 1
8
5
7
2
20
1
Superstep 2
2
5
7
20
2
8
5
7
20
20
2
Superstep 3
20
20
8
5
7
20
20
2

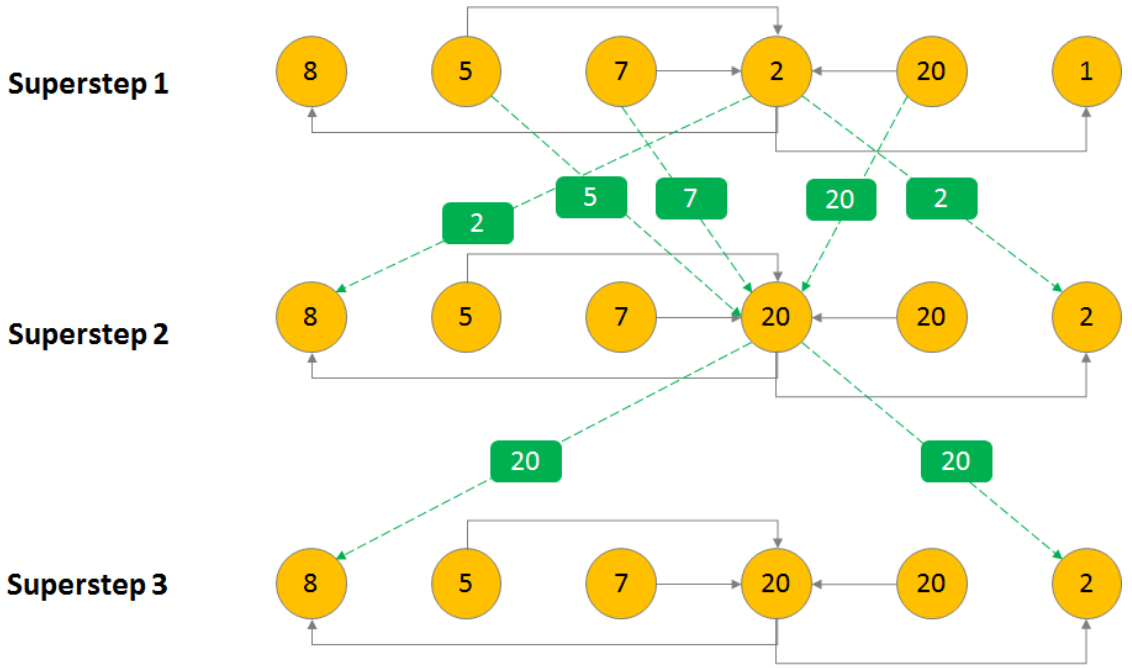
Compute

На каждом superstep’е вызывается метод compute() на всех активных вершинах (обрабатываются сообщения отправленные на предыдущем superstep’е)

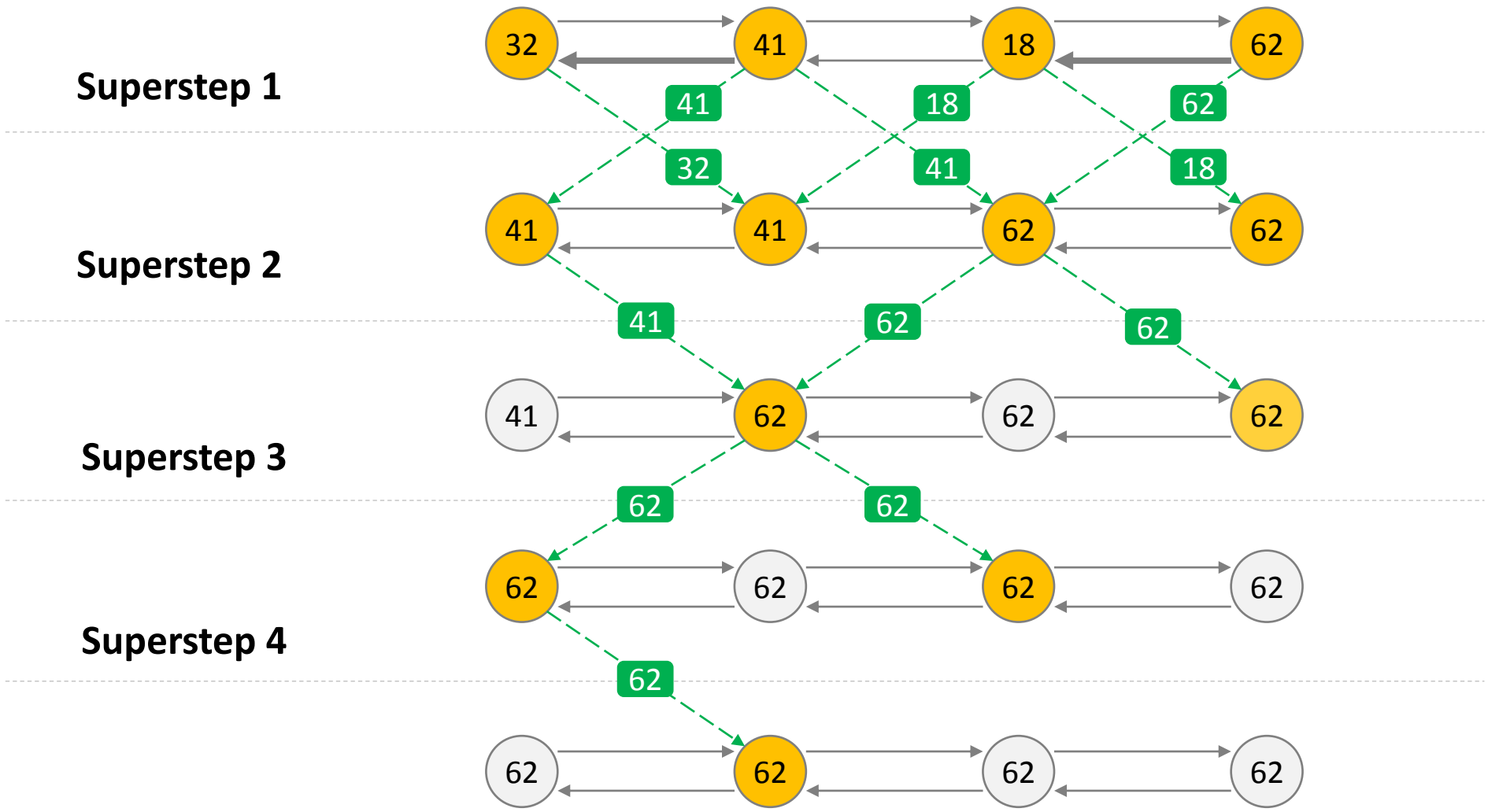
```
class BasicComputation:
    function compute(vertex, messages) # определяемая пользователем обработка
    function getSuperstep() # текущий superstep
    function getTotalNumVertices() # суммарное количество вершин в графе
    function getTotalNumEdges() # суммарное количество ребер в графе
    function sendMessage(targetId, message) # суммарное количество вершин в графе
    function sendMessagetoAllEdges(vertex, message) # отправка сообщения вершине
    vertex
    function addVertexRequest(vertexId) # запрос на добавление вершины в граф
    function removeVertexRequest(vertexId) # запрос на удаление вершины из графа
```

Пример compute

```
function compute(vertex, messages):
    maxValue = max(messages) # определение максимального значения
    if maxValue > vertex.getValue(): # сравнение с текущим значением вершины
        vertex.setValue(maxValue) # установка нового значения текущей вершины
        sendMessageToAllEdges(vertex, maxValue) # отправка сообщения всем связанным
        вершинам
    vertex.voteToHalt() # остановка текущей вершины (переход в неактивное состояние)
```

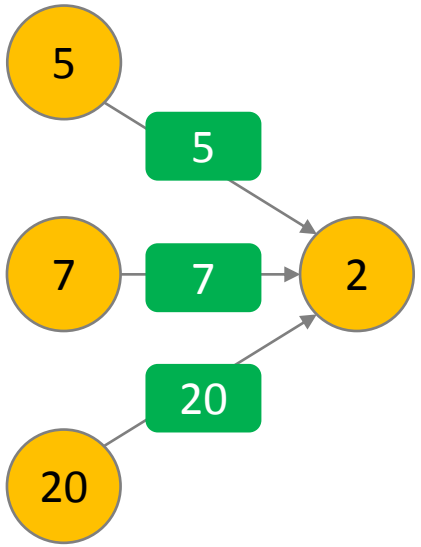


Пример compute

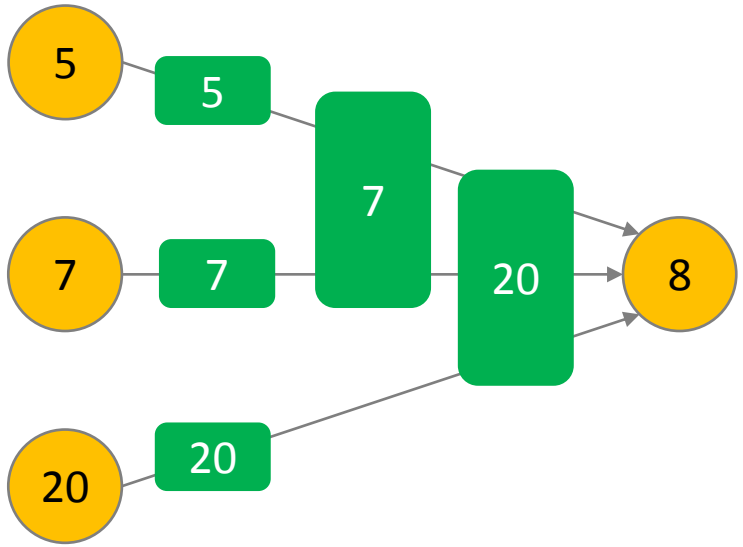


Combiner

```
class MessageCombiner:
    # возвращает комбинацию двух сообщений
    function combine(id, message1, message2)
```



Без Combiner



С Combiner

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Aggregator

```
class Aggregator:
    function aggregate(value) # агрегирование значений
    function getAggregatedValue() # агрегированное значение
    function setAggregatedValue(value) # установка значения
    function reset() # сброс значения
```

Пример Aggregator

Агрегатор

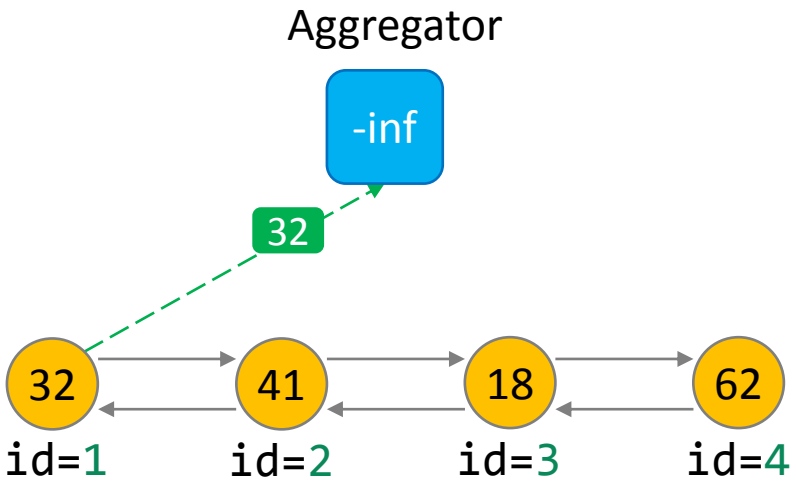
```
maxValue = -Inf
function aggregate(value):
    maxValue = max(maxValue, value)
function getAggregatedValue():
    return maxValue
function setAggregatedValue(value):
    maxValue = value
function reset()
    maxValue = -Inf
```

Метод выполнения (вершина)

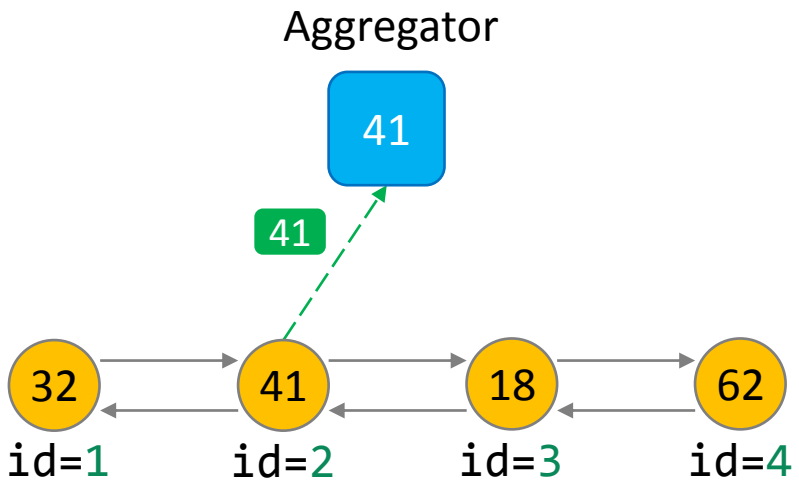
```
function compute(vertex, messages):
    maxValueAggregator.aggregate(vertex.getValue())
    vertex.voteToHalt()
```

Пример Aggregator

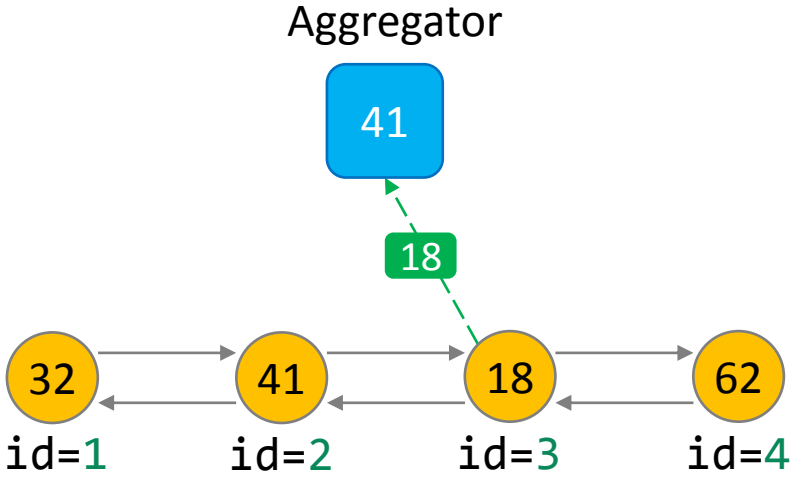
1.



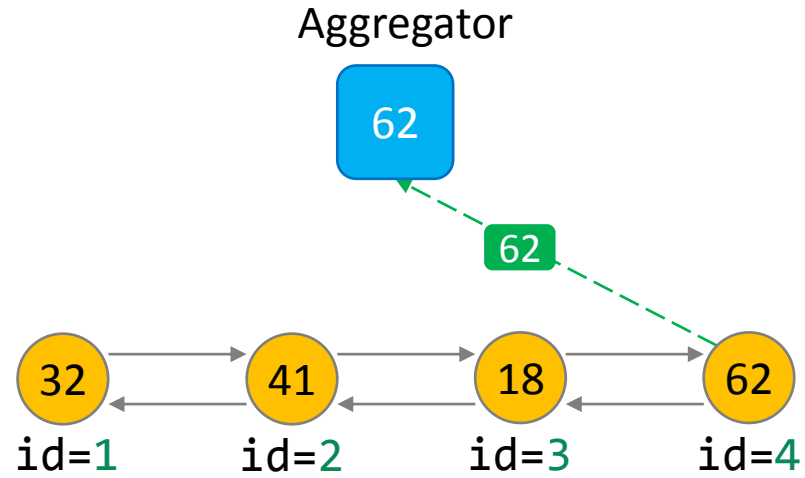
2.



3.



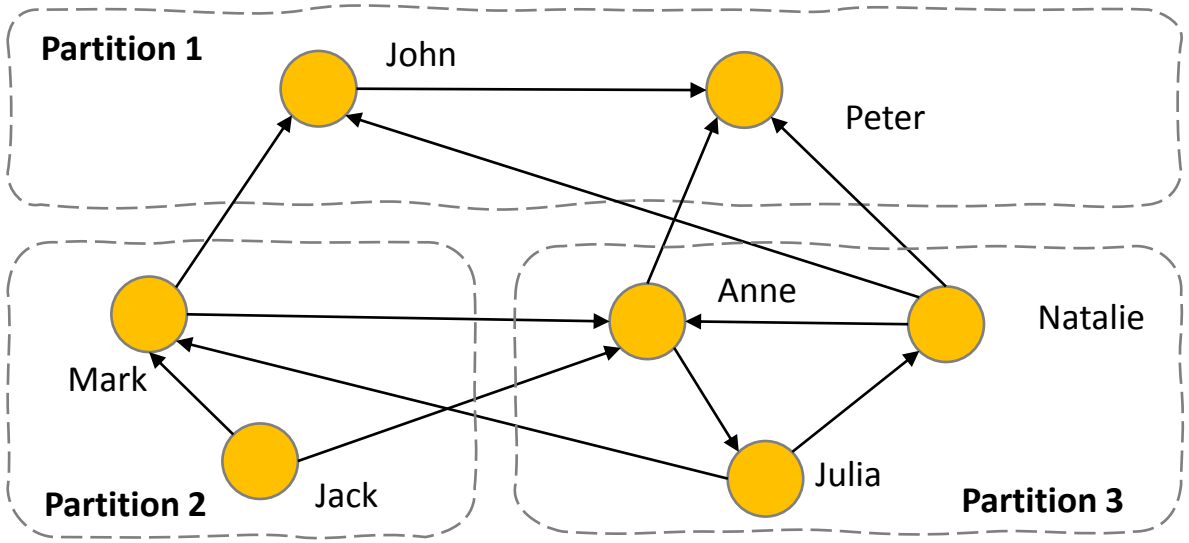
4.



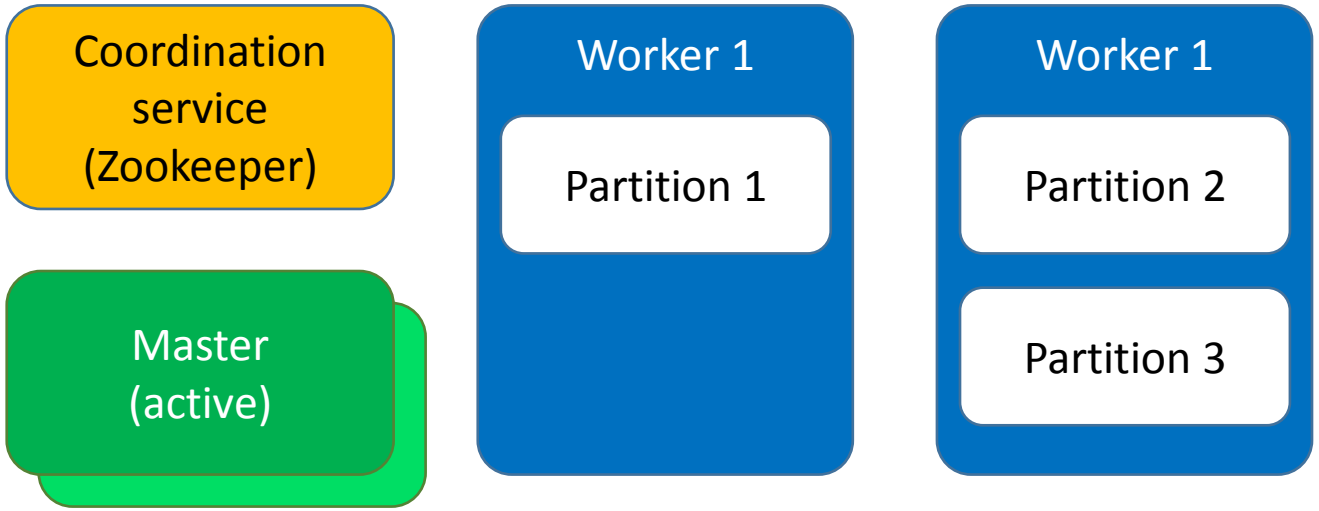
ИНСТИТУТ
ЦИФРОВОГО
ОБРАЗОВАНИЯ
МГПУ

Архитектура Giraph

Разбиение графа на части



Компоненты Giraph



ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Master

Master (active)

- Управление superstep’ами
- Назначение частей (partitions) графа worker’ам
- Запуск compute для агрегаторов
- Мониторинг работоспособности и статистики worker’ов

Worker и Координатор

Worker 1

- Управление состояниями частей графа
- Выполнение compute для всех вершин частей на worker’е
- Сохранение состояния (checkpoint)

Coordination service (Zookeeper)

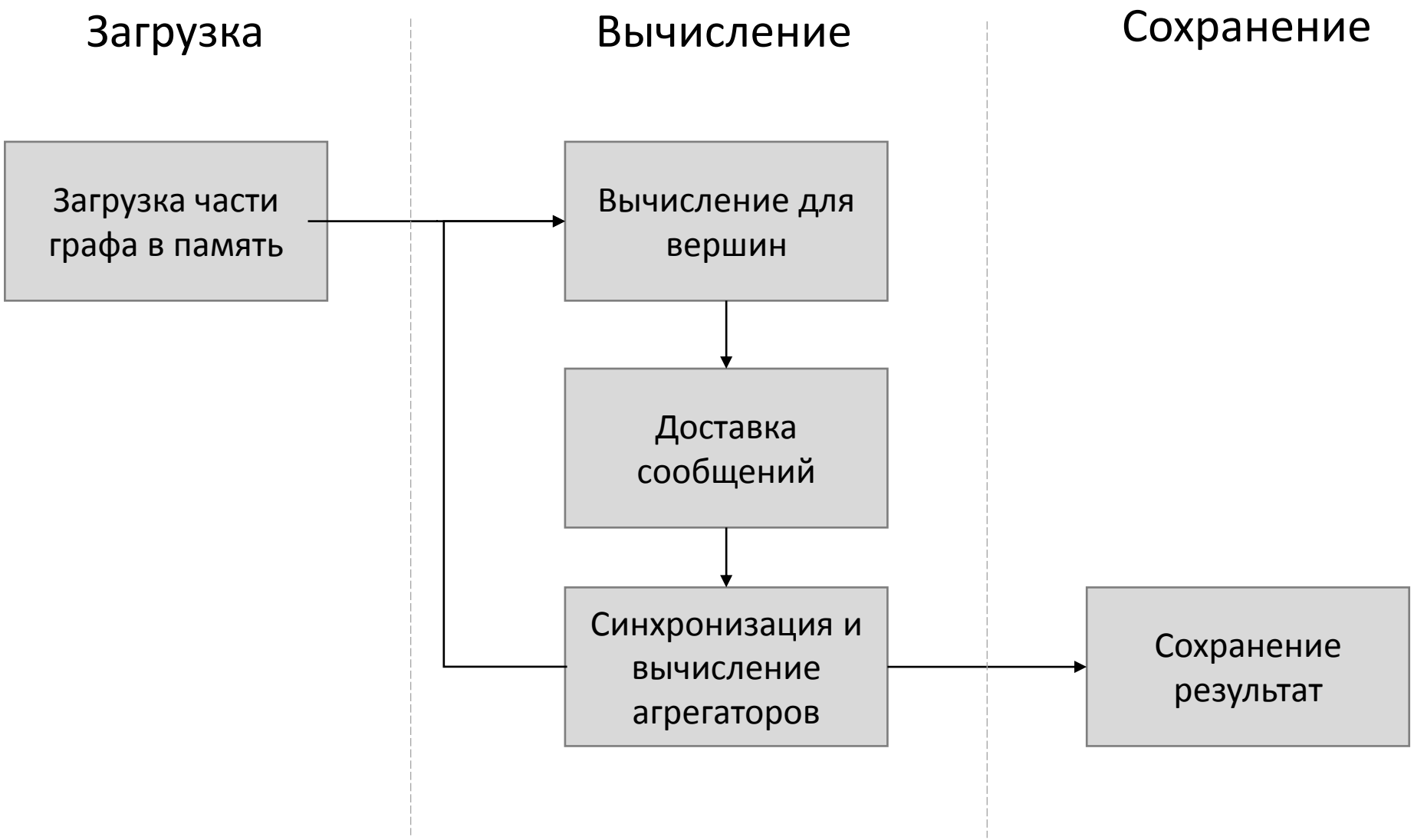
- Конфигурирование
- Синхронизация

ИНСТИТУТ
ЦИФРОВОГО
ОБРАЗОВАНИЯ
МГПУ

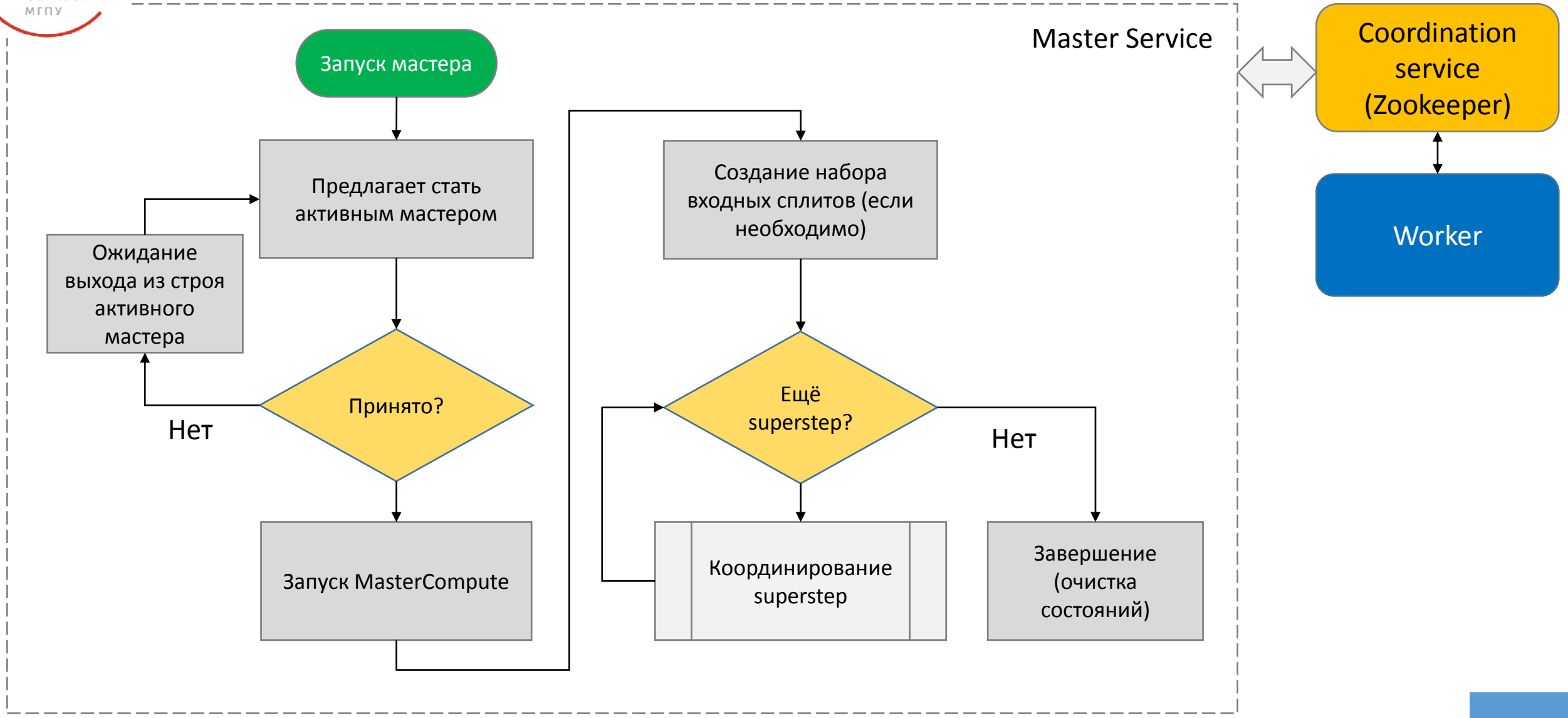
Запуск вычислений

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

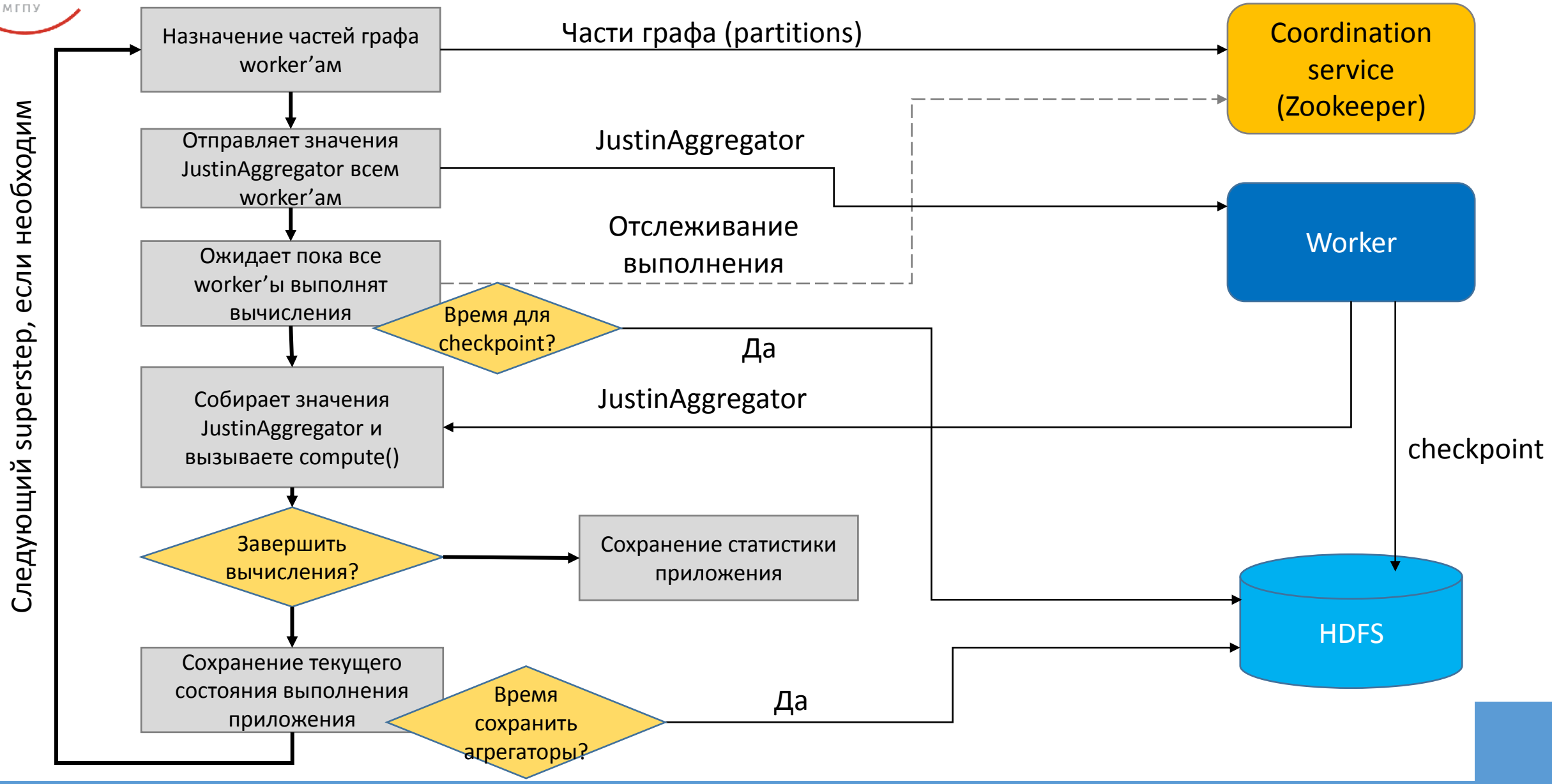
Основные стадии Giraph



Запуск Master Service



Master Service. Superstep



ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Master Service. Условия завершения вычислений

- Установлен внешний флаг остановки вычислений
- Все вершины перешли в состояние остановки (halt) и нет сообщений для вершин
- Превышение максимального количества superstep’ов

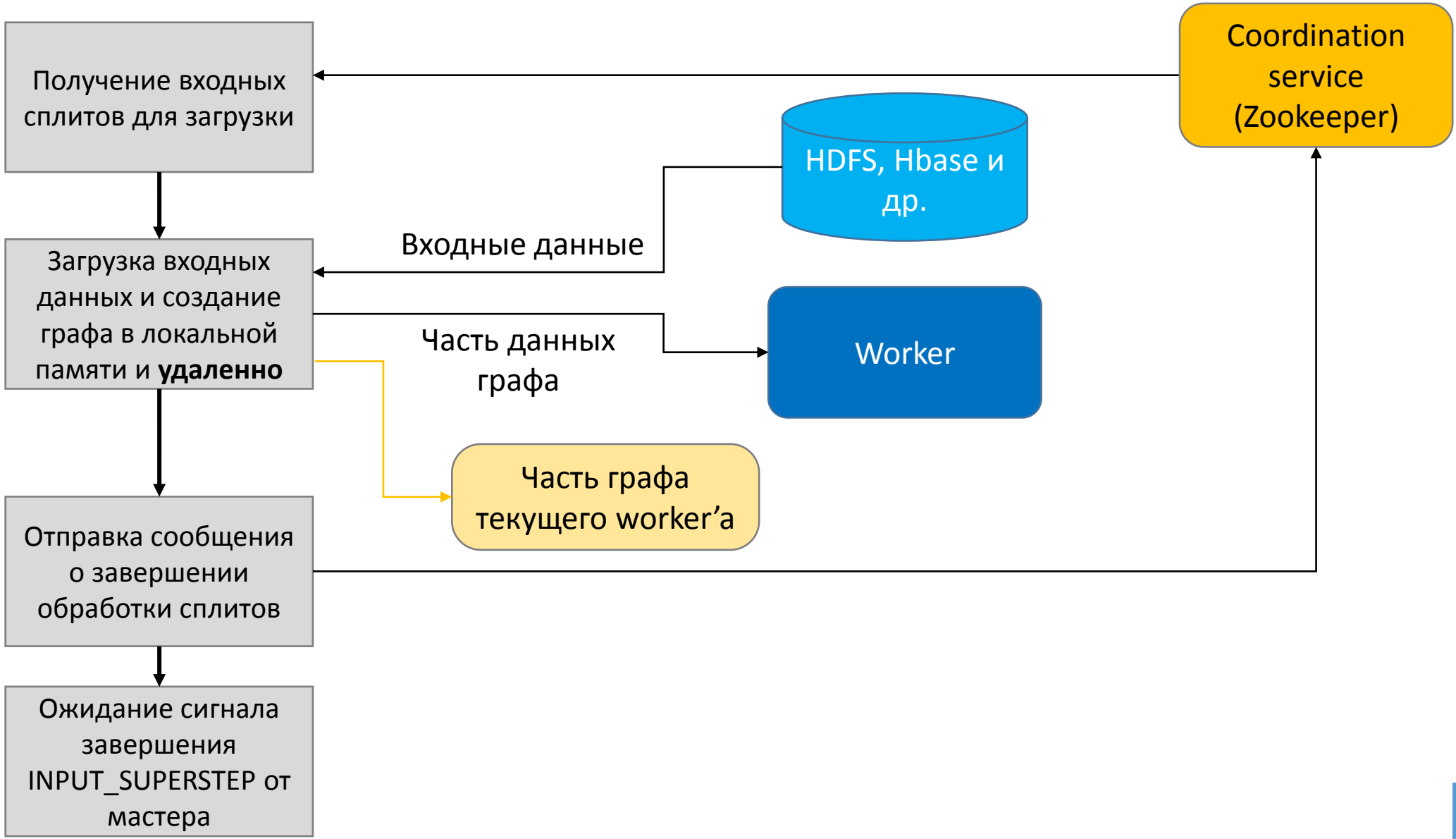
ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Worker Service

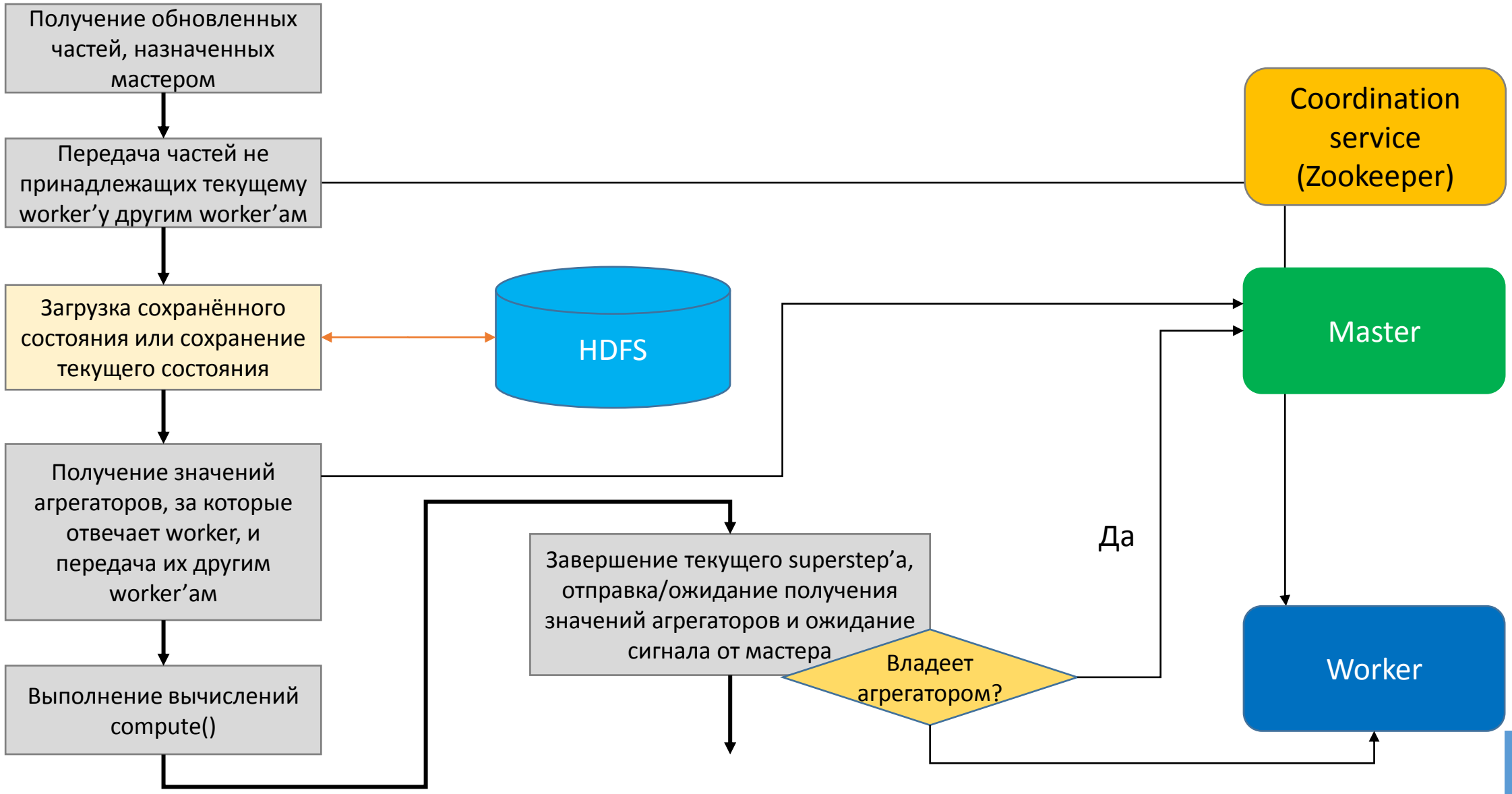
- **Стадия настройки**
  Загрузка исходных данных графа на все worker’ы
- **Стация основного цикла superstep’ов**
  Вычисления

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Worker Service. Загрузка данных. INPUT_SUPERSTEP

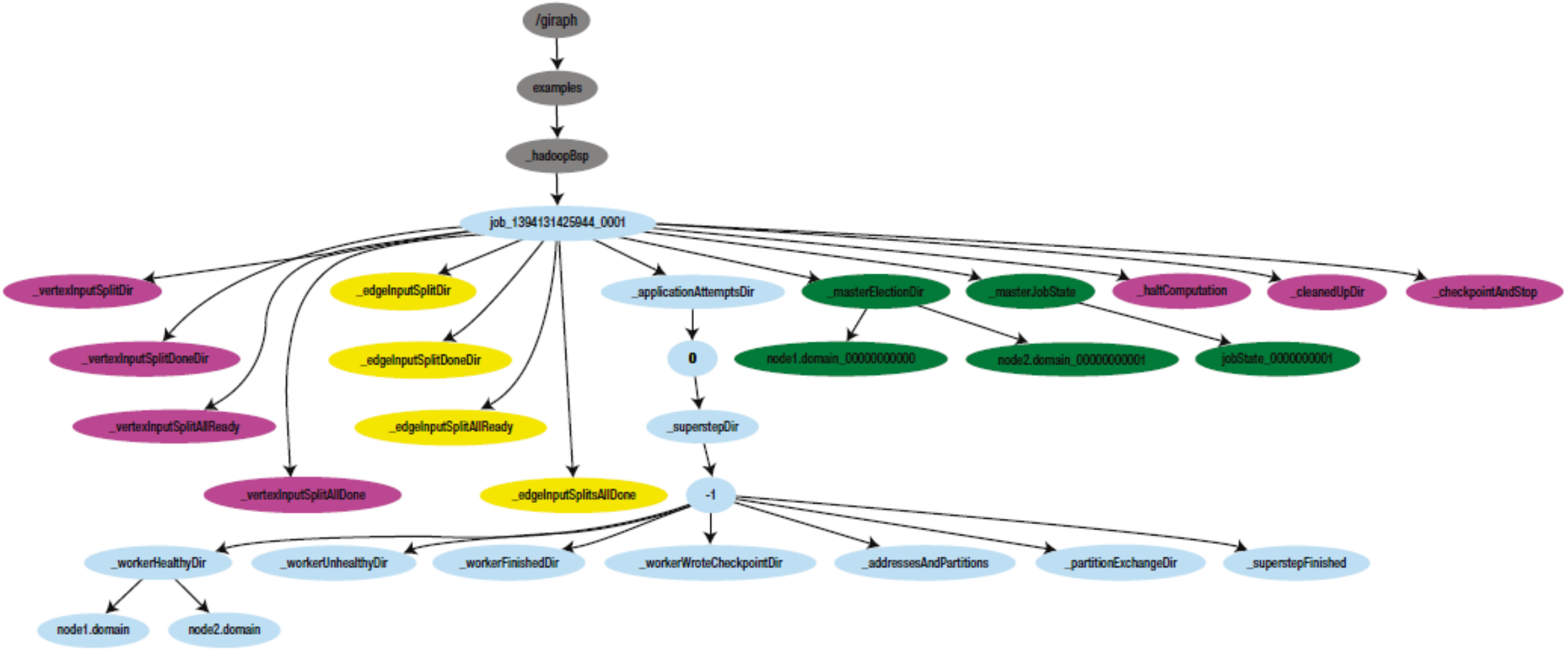


Worker Service. Основной вычислительный цикл



ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Координирующий Service/Zookeeper

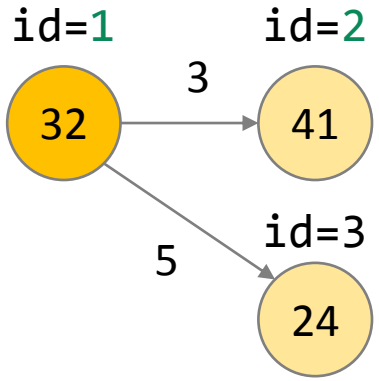


Формат входных данных

Формат входных данных определяет как читать данные из системы хранения и как затем преобразовать их в объекты Vertex и Edge

- VertexInputFormat

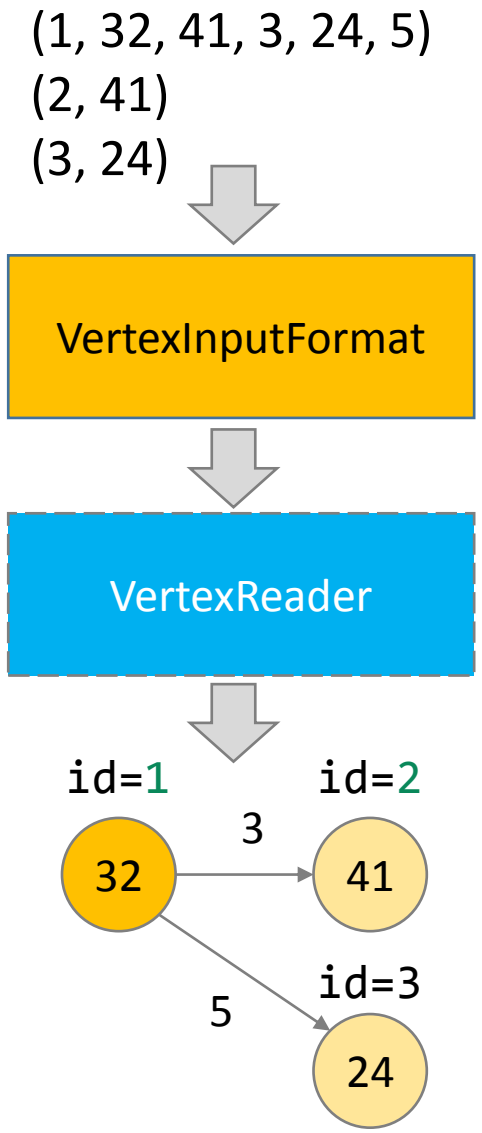
(1, 32, 41, 3, 24, 5)
(2, 41)
(3, 24)



- EdgeInputFormat

(32, 41, 3)
(32, 24, 5)

VertexInputFormat



Загружает данные из системы хранения

Разбивается входные данные на сплиты (InputSplits)

Каждый сплит обрабатывается VertexReader

Каждая запись преобразуется в объект Vertex

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

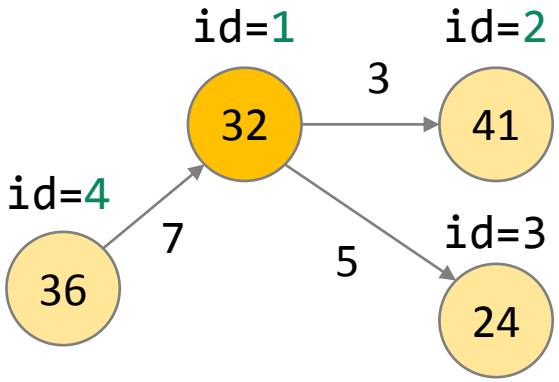
Формат выходных данных

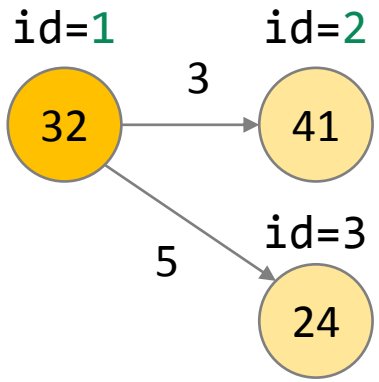
Формат выходных данных определяет как преобразовать граф к отдельные записи и сохранить их в системе хранения

- VertexOutputFormat
- EdgeOutputFormat

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

VertexInputFormat





Граф

VertexOutputFormat

Отвечает за сохранение данных в системе хранения

Назначает вершине VertexWriter

VertexWriter

Каждая вершина преобразуется в запись

(1, 32, 41, 3, 24, 5)

Отказоустойчивость

Master

Если вышел из строя активный мастер, то

- Исчезает эфемерный znode в zookeeper’е, соответствующий мастеру
- Происходит выбор нового активного мастера
- Новый активный мастер получает состояния из zookeeper’а
- Значения агрегаторов будут получены от worker’ов при завершении текущего superstep’а

Отказоустойчивость

Worker

- Если сбой произошел на стадии загрузки сплитов, то мастер отменит запуск приложения.

- Если сбой произошел на стадии основного цикла и checkpoint включен, то

  ✓ исходный граф гарантированно сохранен в HDFS

  - Эфемерный znode worker’а исчезает и об этом уведомляется мастер
  - Мастер помечает текущий superstep как неудачный (failed)
  - Мастер запускает основной цикл с последнего сохраненного в HDFS superstep’а и поручает рабочим worker’ам загрузить состояния, соответствующие последнему checkpoint’у

ИНСТИТУТ ЦИФРОВОГО ОБРАЗОВАНИЯ МГПУ

Отказоустойчивость

Координатор (Zookeeper)

Zookeeper – это распределенный координирующий сервис со встроенным механизмом отказоустойчивости. Как правило, ансамбль zookeeper’а состоит из 3 серверов и будет в рабочем состоянии при выходе из строя одного сервера (т.к. в этом случае будет обеспечен кворум из 2 узлов)

Источники

Practical Graph Analytics with Apache Giraph by Claudio Martella, Roman Shaposhnik, Dionysios Logothetis (book)

Apache Giraph (official website)