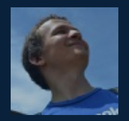
Триллион ребер и больше: обработка графов в Facebook

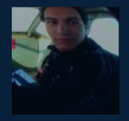
GraphHPC 2016, Москва

Avery Ching
Facebook



Sergey Edunov
Facebook

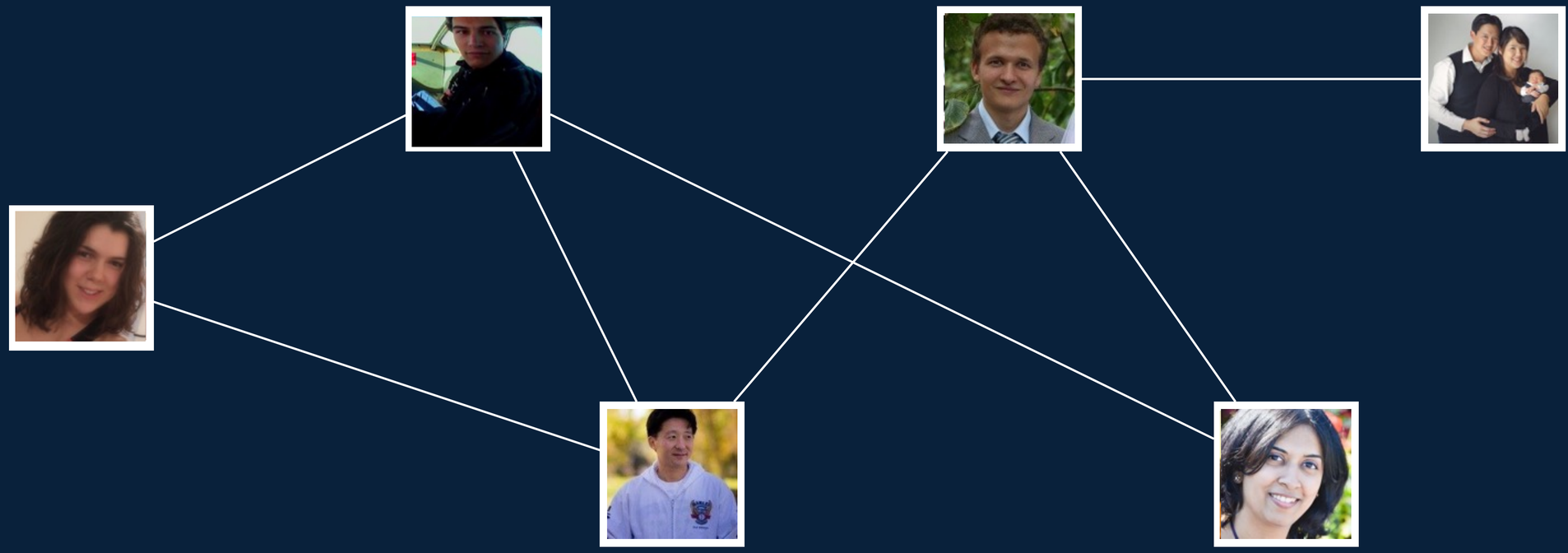
Maja Kabiljo
Facebook



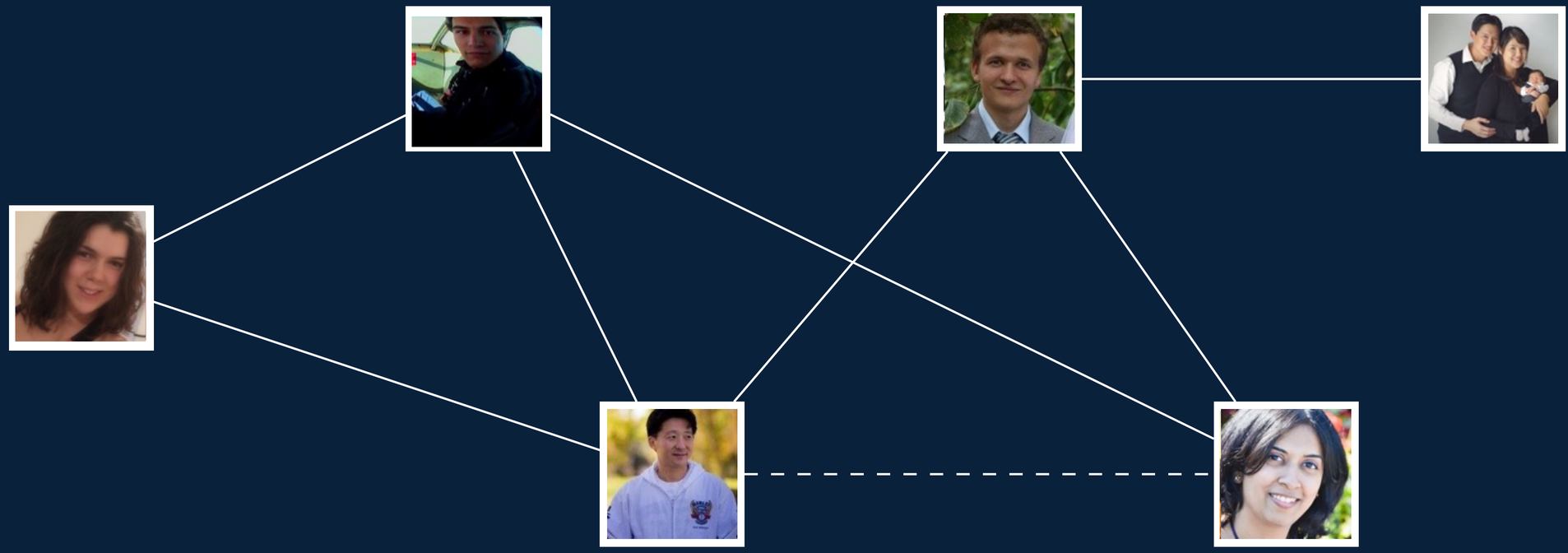
Dionysios Logothetis
Facebook

Sambavi Muthukrishnan
Facebook

Social Graph



Social Graph



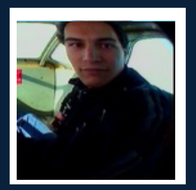
Вопрос:

Могут ли Jay и Sambavi быть друзьями?

Ranking Features

7.6

9.3



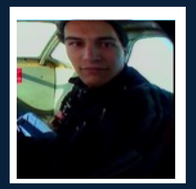
6.4

8.2

Ranking Features

7.6

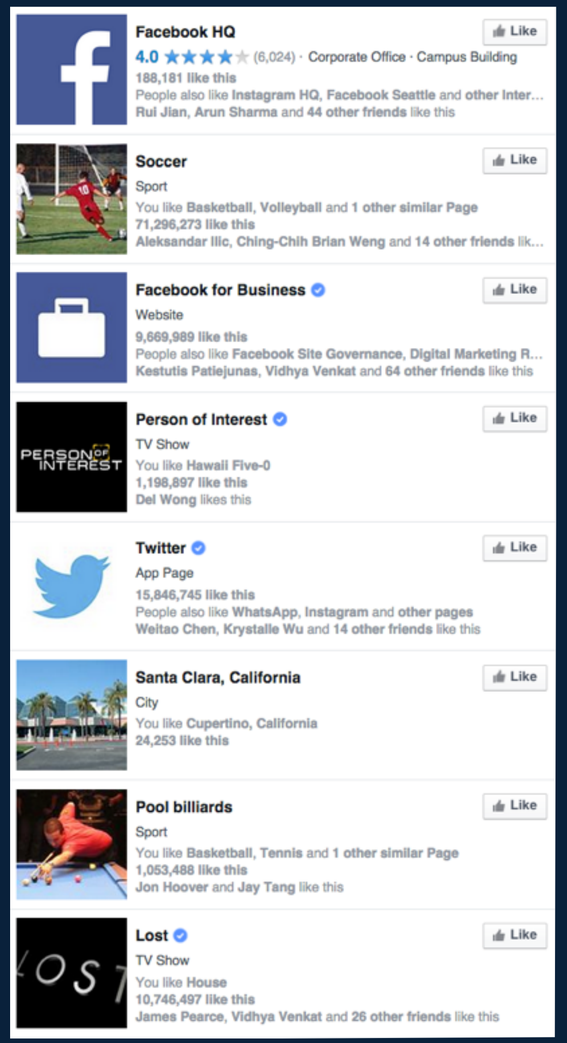
9.3



6.4

8.2

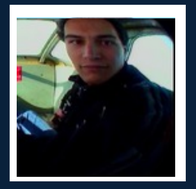
Recommendations



Ranking Features

7.6

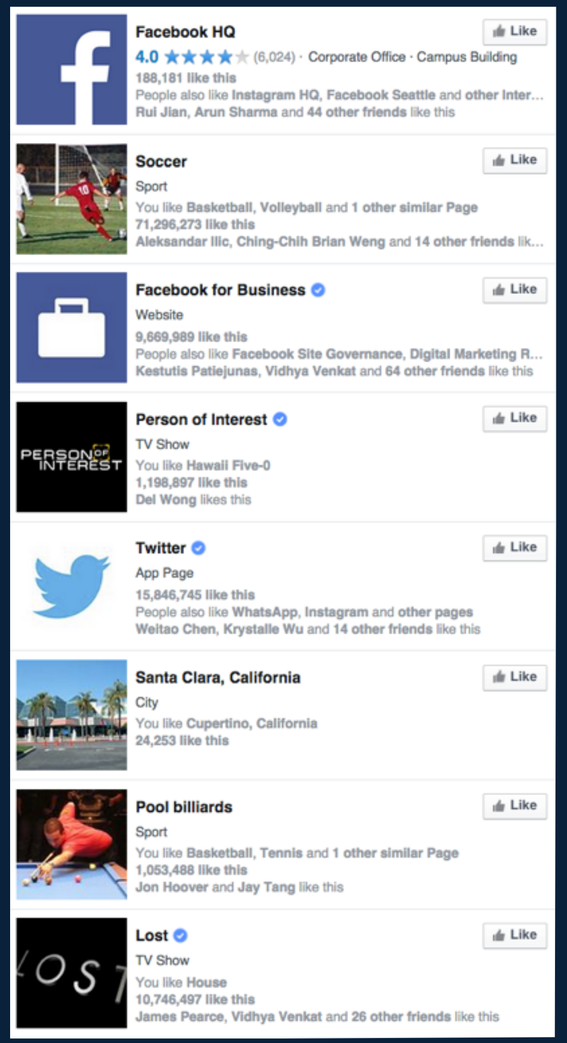
9.3



6.4

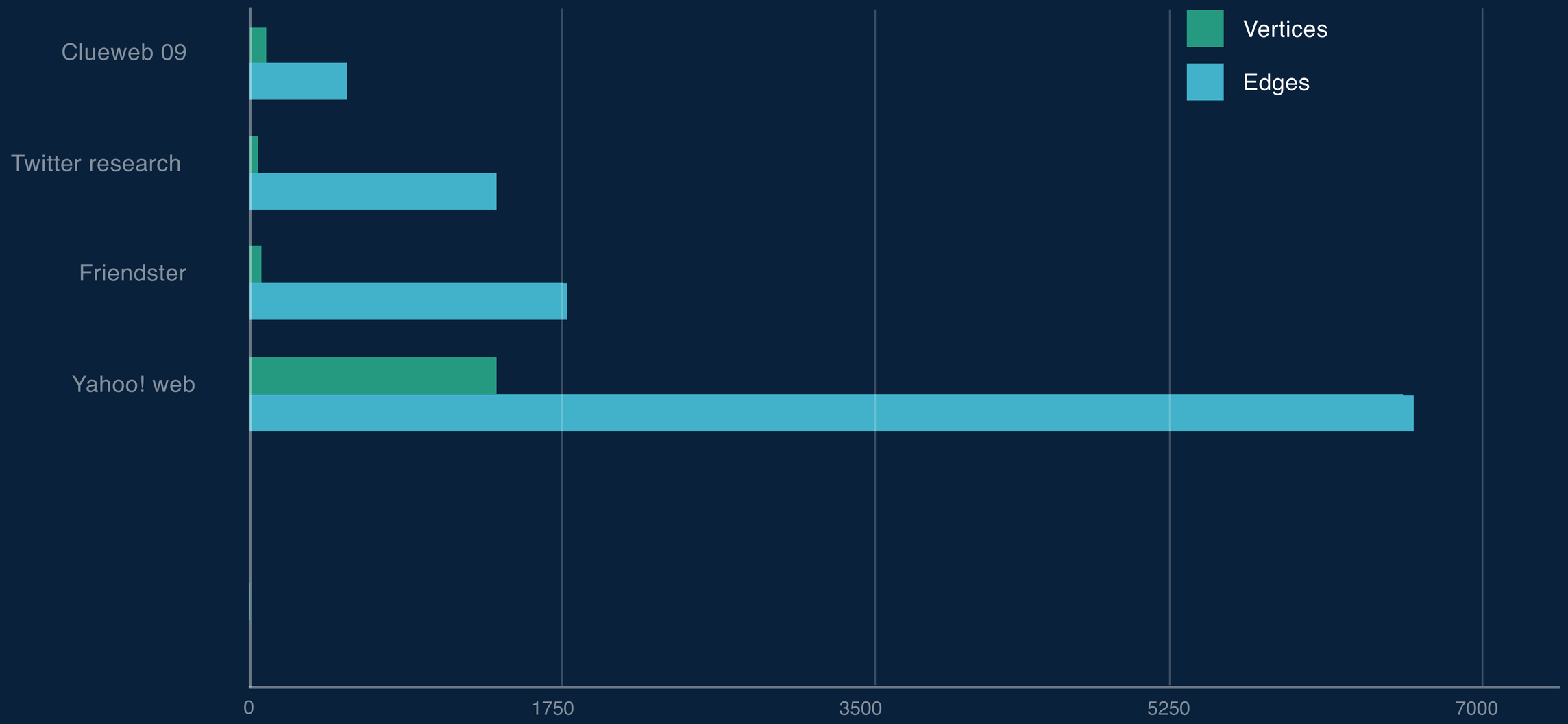
8.2

Recommendations

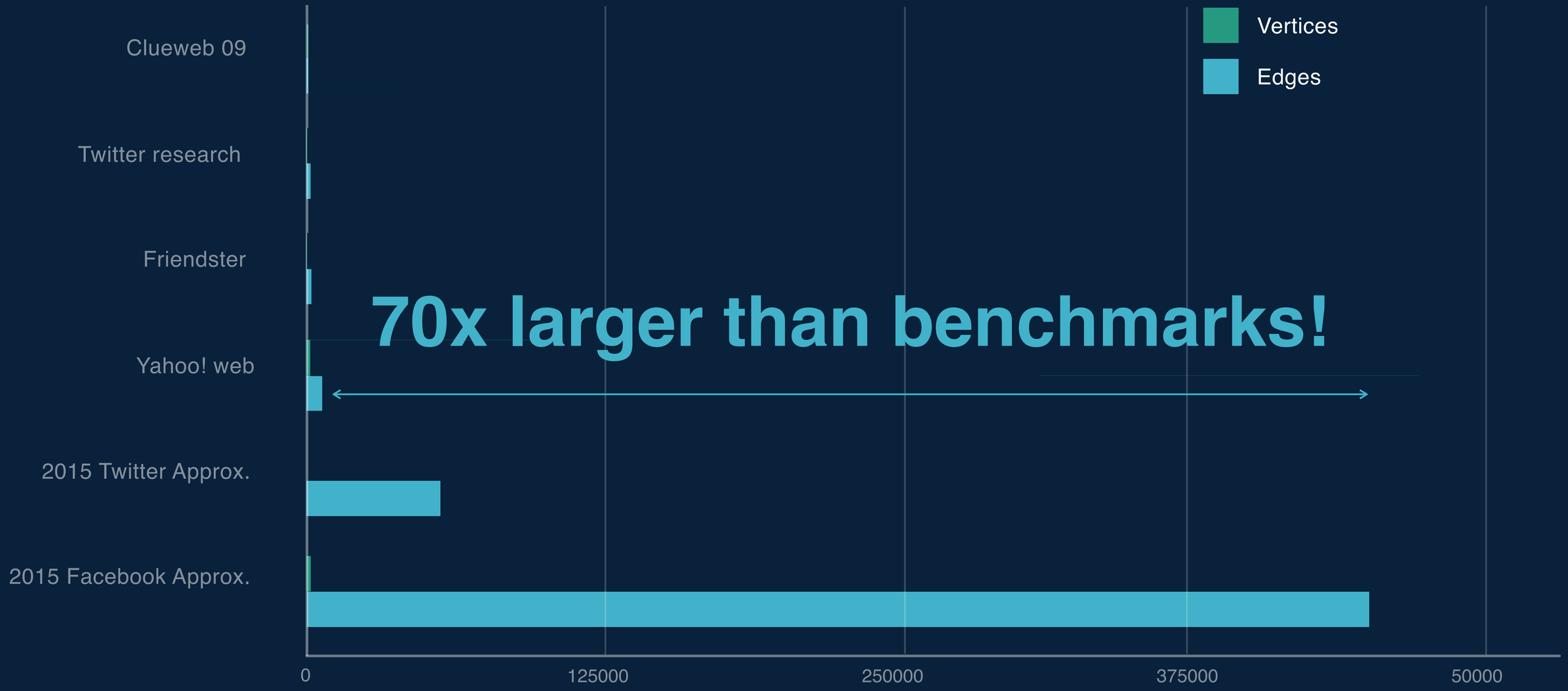


Data Partitioning

Benchmark Graphs

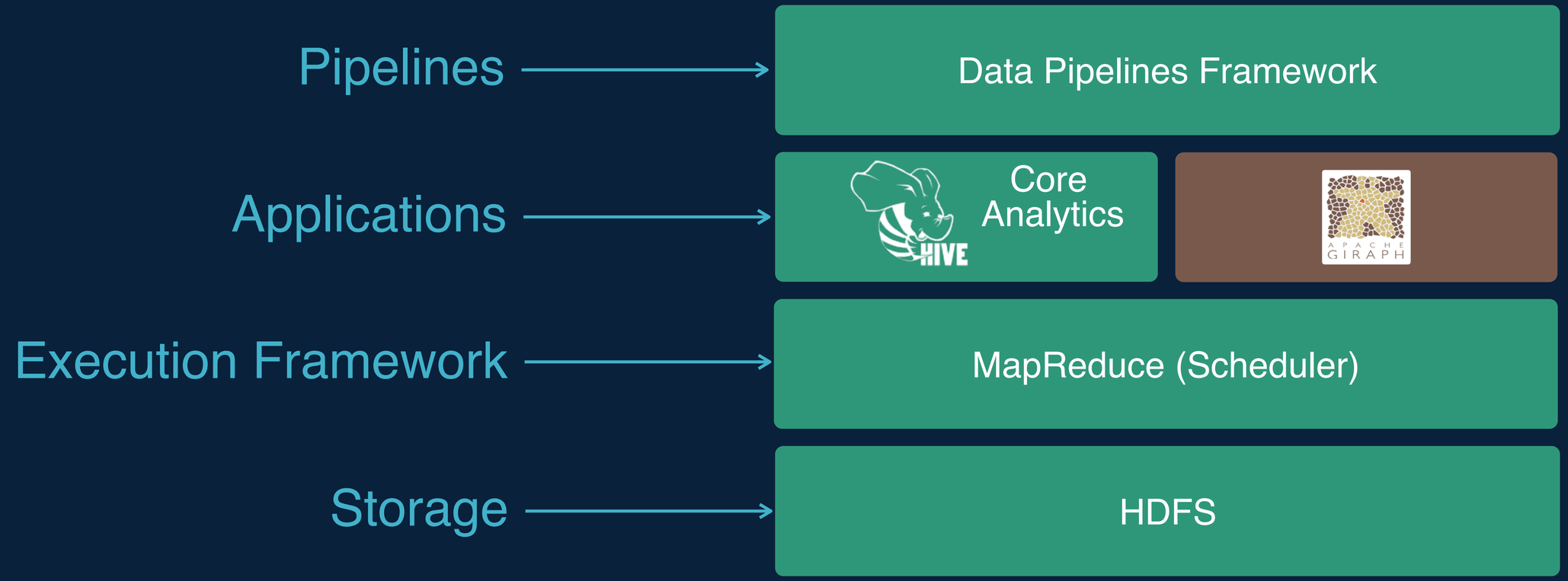


Benchmark to Social Graphs



Высоко-масштабируемая система анализа графов с открытым исходным кодом

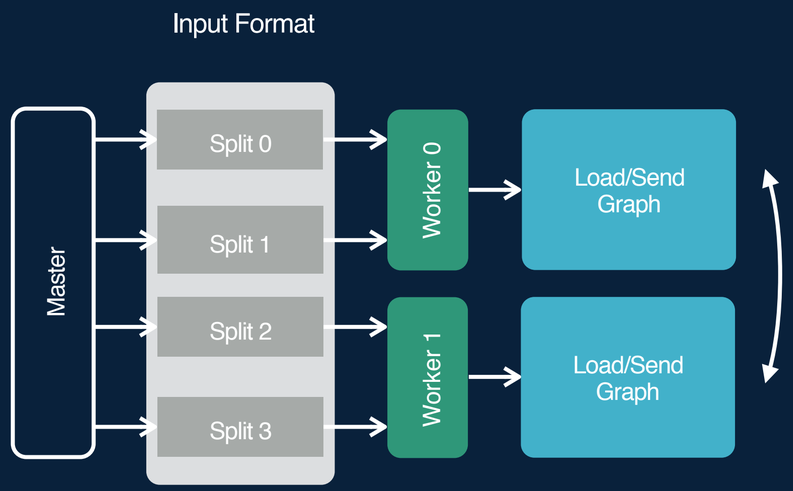
“Думай как вершина”, которая отправляет сообщения другим вершинам


Pipelines
Data Pipelines Framework
Applications
HIVE
Core
Analytics
APACHE
GIRAPH
Execution Framework
MapReduce (Scheduler)
Storage
HDFS

Архитектура

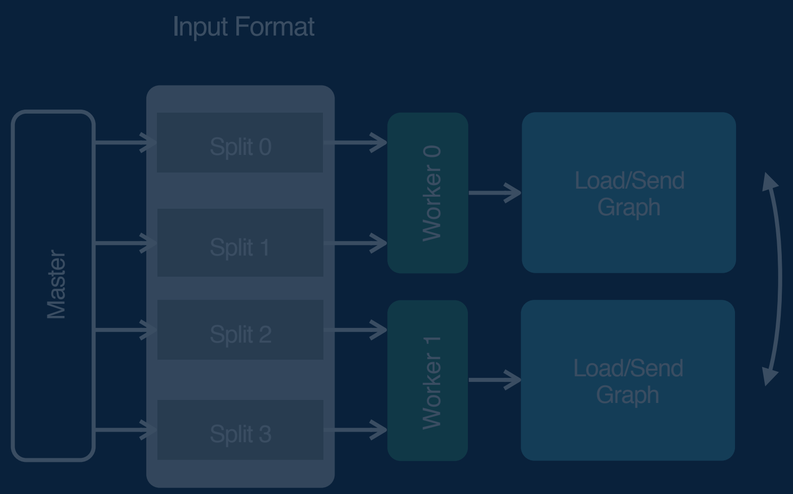
Архитектура

Loading the graph

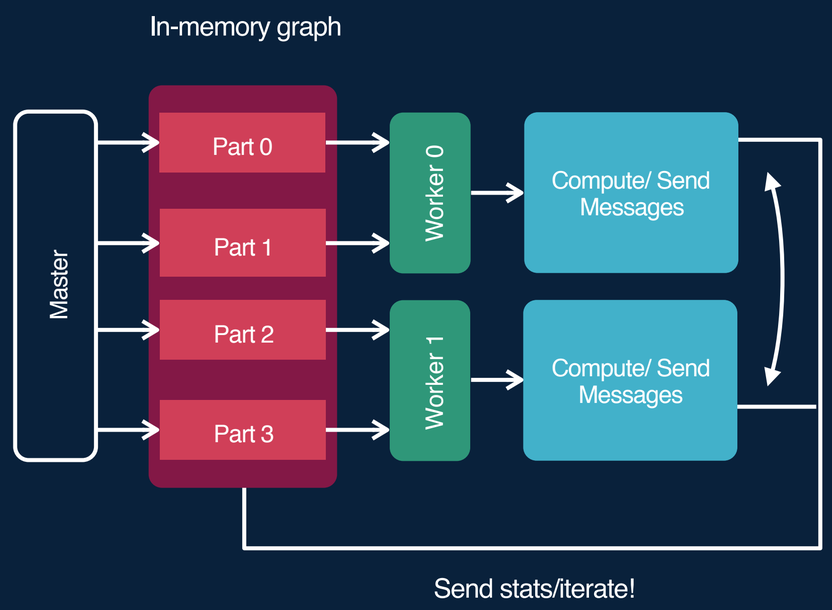


Архитектура

Loading the graph

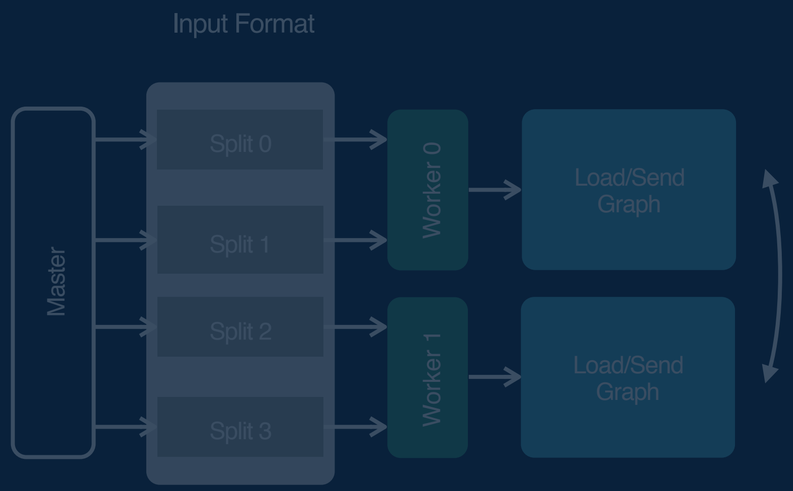


Compute / Iterate

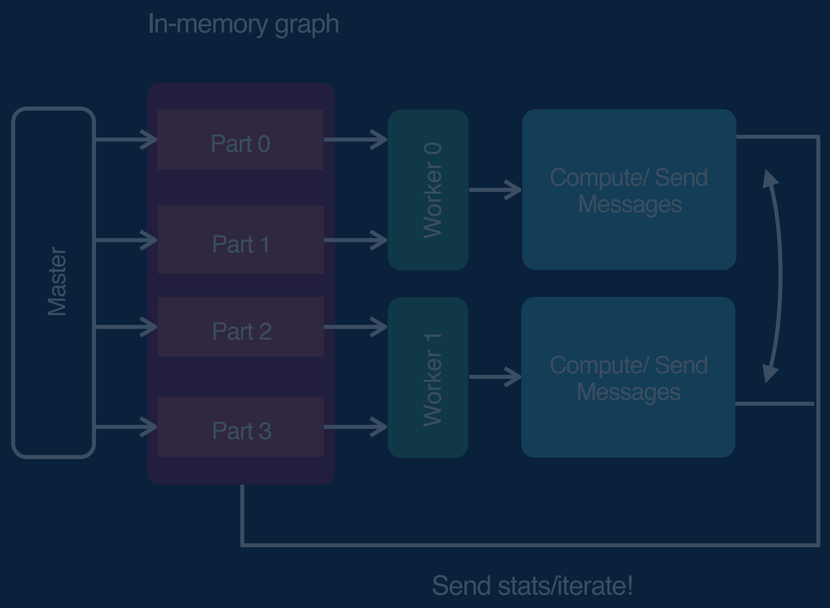


Архитектура

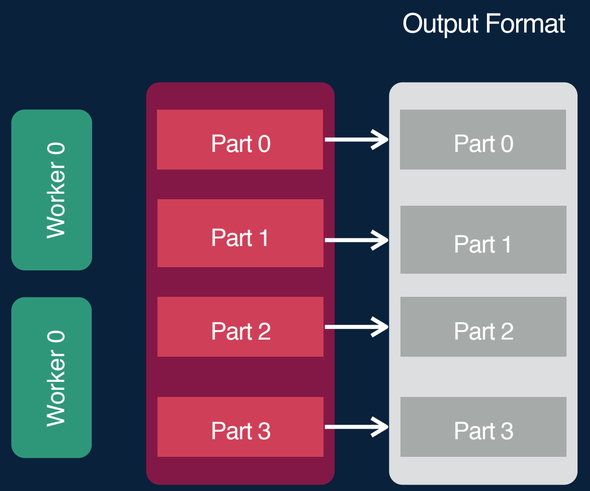
Loading the graph



Compute / Iterate

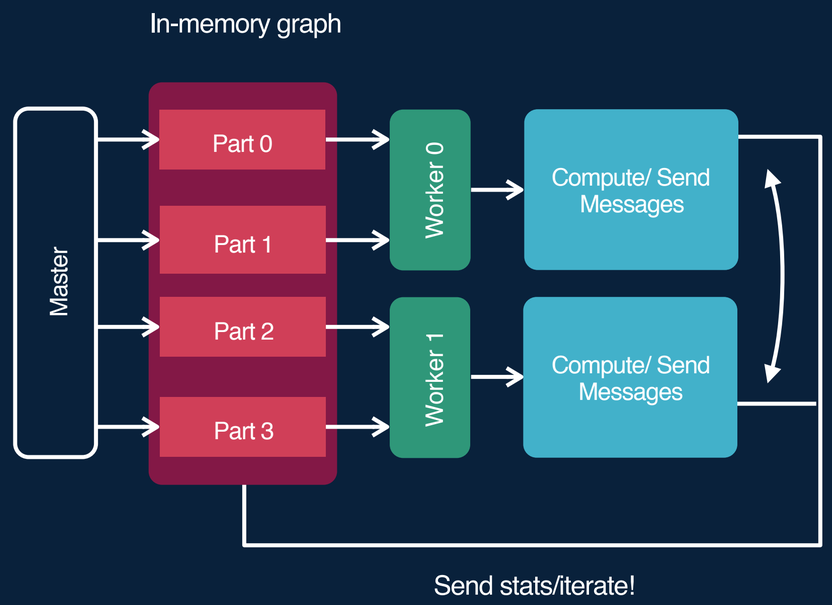


Storing the graph

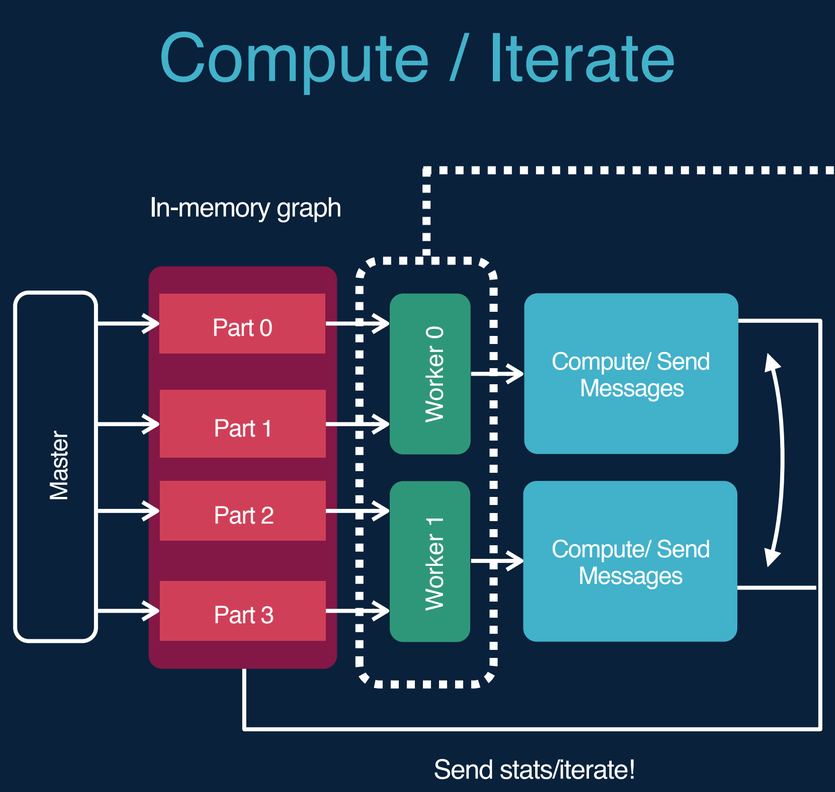


Parallelization Model

Compute / Iterate



Parallelization Model



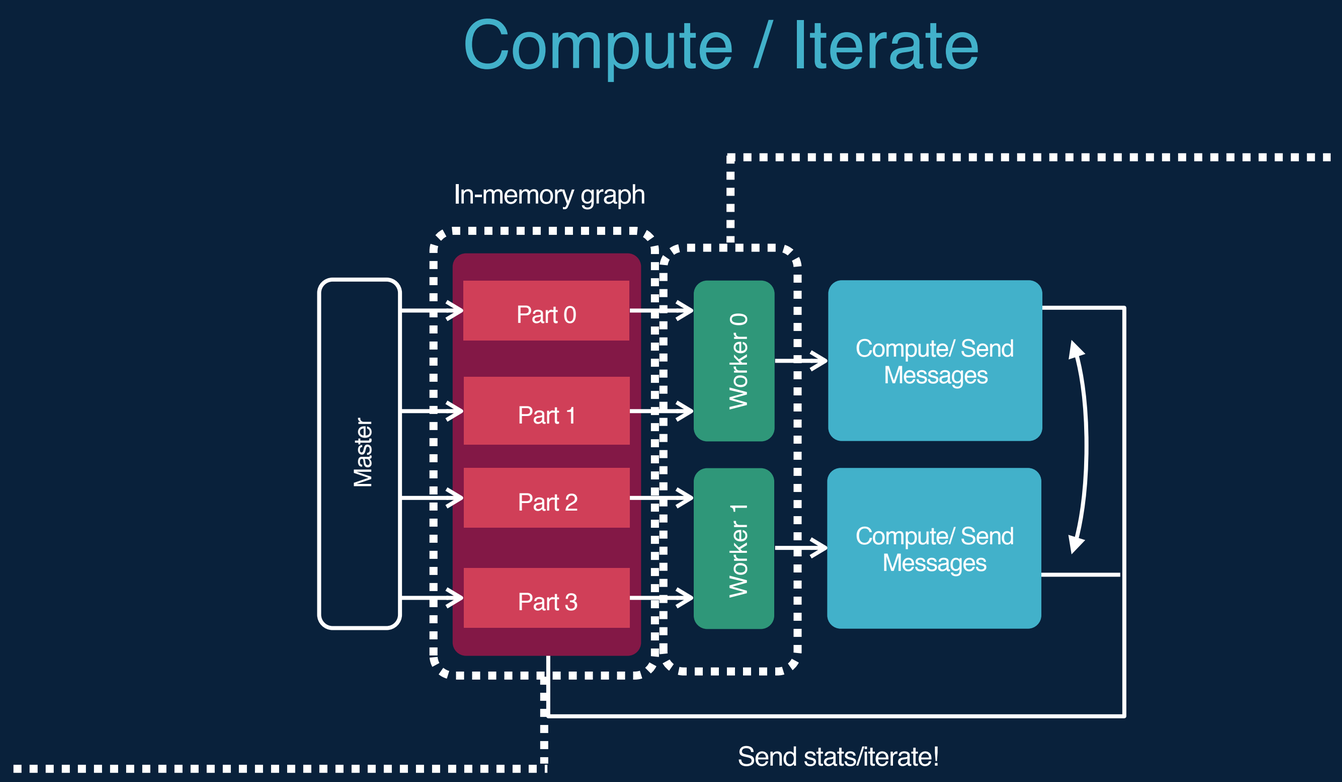
Worker parallelization: используется hadoop чтобы распределить задачи по нескольким машинам

Плюс: простота

Parallelization Model

Плюсы: меньше сетевых соединений, лучшая утилизация памяти (за счет совместного использования ресурсов)

Многопоточность: Каждая машина использует доступные ядра



Worker parallelization: используется hadoop чтобы распределить задачи по нескольким машинам

Плюс: простота

Полноценная поддержка ООП

- Ребер >> вершин (> 2 порядка)
- Реализации многих блоков системы можно заменить
  - Наприме: хранилище ребер с использованием примитивных массивов или библиотеки FastUtil
- Сериализация объектов в большие массивы байтов

```java
/**
 * Interface for data structures that store out-edges for a vertex.
 *
 * @param <I> Vertex id
 * @param <E> Edge value
 */
public interface OutEdges<I extends WritableComparable,
    E extends Writable> extends Iterable<Edge<I, E>>, Writable {

  void initialize(Iterable<Edge<I, E>> edges);

  void initialize(int capacity);

  void initialize();

  void add(Edge<I, E> edge);

  void remove(I targetVertexId);

  int size();
}
```

Page Rank

Map Reduce (Hadoop)

```
Index: core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
===================================================================
--- core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
+++ core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
@@ -0,0 +1,166 @@
+/**
+ * Licensed to the Apache Software Foundation (ASF) under one or more
+ * contributor license agreements.  See the NOTICE file distributed
with
+ * this work for additional information regarding copyright ownership.
+ * The ASF licenses this file to You under the Apache License, Version
2.0
+ * (the "License"); you may not use this file except in compliance with
+ * the License.  You may obtain a copy of the License at
+ *
+ *     http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.mahout.graph.pagerank;
+
+import java.io.IOException;
```

Page Rank

Map Reduce (Hadoop)

```
Index: core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
===================================================================
--- core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
+++ core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
@@ -0,0 +1,166 @@
+/**
+ * Licensed to the Apache Software Foundation (ASF) under one or more
+ * contributor license agreements.  See the NOTICE file distributed
with
+ * this work for additional information regarding copyright ownership.
+ * The ASF licenses this file to You under the Apache License, Version
2.0
+ * (the "License"); you may not use this file except in compliance with
+ * the License.  You may obtain a copy of the License at
+ *
+ *     http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.mahout.graph.pagerank;
+
+import java.io.IOException;
```

Page Rank

Map Reduce (Hadoop)

```
Index: core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
===================================================================
--- core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
+++ core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
@@ -0,0 +1,166 @@
+/**
+ * Licensed to the Apache Software Foundation (ASF) under one or more
+ * contributor license agreements.  See the NOTICE file distributed
with
+ * this work for additional information regarding copyright ownership.
+ * The ASF licenses this file to You under the Apache License, Version
2.0
+ * (the "License"); you may not use this file except in compliance with
+ * the License.  You may obtain a copy of the License at
+ *
+ *     http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.mahout.graph.pagerank;
+
+import java.io.IOException;
```

Giraph

```
public class PageRankComputation extends BasicComputation<LongWritable,
DoubleWritable, FloatWritable, DoubleWritable> {
  public void compute(
      Vertex<LongWritable, DoubleWritable, FloatWritable> vertex,
      Iterable<DoubleWritable> messages) {
    // Calculate new page rank value
    if (getSuperstep() >= 1) {
      double sum = 0;
      for (DoubleWritable message : messages) {
        sum += message.get();
      }
      vertex.getValue().set(0.15d / getTotalNumVertices() +0.85d * sum);
    }
    // Send page rank value to neighbors
    if (getSuperstep() < 30) {
      sendMsgToAllEdges(new DoubleWritable(getVertexValue().get() /
                        getNumOutEdges()));
    } else {
      voteToHalt();
    }
  }
}
```

Page Rank

Map Reduce (Hadoop)

```
Index: core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
===================================================================
--- core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
+++ core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
@@ -0,0 +1,166 @@
+/**
+ * Licensed to the Apache Software Foundation (ASF) under one or more
+ * contributor license agreements.  See the NOTICE file distributed
with
+ * this work for additional information regarding copyright ownership.
+ * The ASF licenses this file to You under the Apache License, Version
2.0
+ * (the "License"); you may not use this file except in compliance with
+ * the License.  You may obtain a copy of the License at
+ *
+ *     http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.mahout.graph.pagerank;
+
+import java.io.IOException;
```

Giraph

```
public class PageRankComputation extends BasicComputation<LongWritable,
DoubleWritable, FloatWritable, DoubleWritable> {
  public void compute(
      Vertex<LongWritable, DoubleWritable, FloatWritable> vertex,
      Iterable<DoubleWritable> messages) {
    // Calculate new page rank value
    if (getSuperstep() >= 1) {
      double sum = 0;
      for (DoubleWritable message : messages) {
        sum += message.get();
      }
      vertex.getValue().set(0.15d / getTotalNumVertices() +0.85d * sum);
    }
    // Send page rank value to neighbors
    if (getSuperstep() < 30) {
      sendMsgToAllEdges(new DoubleWritable(getVertexValue().get() /
                        getNumOutEdges()));
    } else {
      voteToHalt();
    }
  }
}
```

Page Rank

Map Reduce (Hadoop)

```
Index: core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
===================================================================
--- core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
+++ core/src/main/java/org/apache/mahout/graph/pagerank/Pagerank.java
(revision )
@@ -0,0 +1,166 @@
+/**
+ * Licensed to the Apache Software Foundation (ASF) under one or more
+ * contributor license agreements.  See the NOTICE file distributed
with
+ * this work for additional information regarding copyright ownership.
+ * The ASF licenses this file to You under the Apache License, Version
2.0
+ * (the "License"); you may not use this file except in compliance with
+ * the License.  You may obtain a copy of the License at
+ *
+ *     http://www.apache.org/licenses/LICENSE-2.0
+ *
+ * Unless required by applicable law or agreed to in writing, software
+ * distributed under the License is distributed on an "AS IS" BASIS,
+ * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
+ * See the License for the specific language governing permissions and
+ * limitations under the License.
+ */
+
+package org.apache.mahout.graph.pagerank;
+
+import java.io.IOException;
```

Giraph

```
public class PageRankComputation extends BasicComputation<LongWritable,
DoubleWritable, FloatWritable, DoubleWritable> {
  public void compute(
      Vertex<LongWritable, DoubleWritable, FloatWritable> vertex,
      Iterable<DoubleWritable> messages) {
    // Calculate new page rank value
    if (getSuperstep() >= 1) {
      double sum = 0;
      for (DoubleWritable message : messages) {
        sum += message.get();
      }
      vertex.getValue().set(0.15d / getTotalNumVertices() +0.85d * sum);
    }
    // Send page rank value to neighbors
    if (getSuperstep() < 30) {
      sendMsgToAllEdges(new DoubleWritable(getVertexValue().get() /
                        getNumOutEdges()));
    } else {
      voteToHalt();
    }
  }
}
```

Pregel Extensions

Pregel model

- Каждая вершина реализует функцию compute т.е. “думай как вершина”
- Все вершины в активном состоянии выполняют функцию compute в течении супер шага (BSP)
- Combiners используются чтобы агрегировать сообщения
- Aggregators используются чтобы считать глобальную статистику каждый супер шаг

Расширения модели, добавленные в Giraph

- Computation стал полноценным объектом
- Computation определяется для master, worker, и вершин
- Master computation выполняется централизовано, и имеет возможность менять computations для вершин и workers
- Computations можно объединять вместе и использовать повторно

First Class Computation

```
class Vertex {
  public:
    virtual void Compute(MessageIterator* msgs) = 0;
 …
};
```

```
public interface Computation {
  void compute(Vertex<I, V, E> vertex, Iterable<M1> messages);
…
}
```

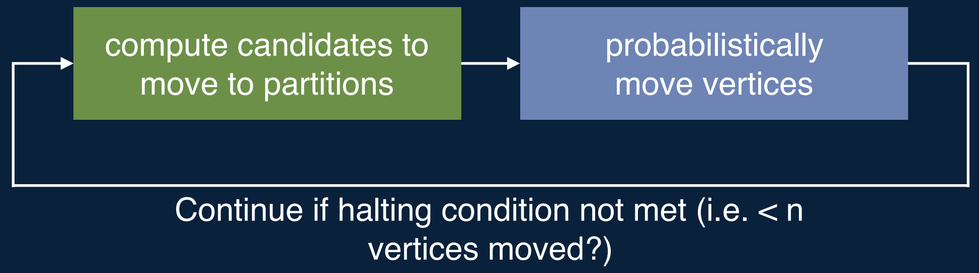
Defining Computation for Master/Worker

```
public abstract class MasterCompute {

  public abstract void compute();

  public abstract void haltComputation();

  public final void setComputation(
      Class<? extends Computation> computationClass);

  public final void setMessageCombiner(
      Class<? extends MessageCombiner> combinerClass);

  public final void setIncomingMessage(
      Class<? extends Writable> incomingMessageClass);

  public final void setOutgoingMessage(
      Class<? extends Writable> outgoingMessageClass);

  public final void setOutgoingMessageClasses(
      MessageClasses<? extends WritableComparable,
          ? extends Writable> outgoingMessageClasses);

  public final <A extends Writable> void setAggregatedValue(
      String name, A value);
…
}
```

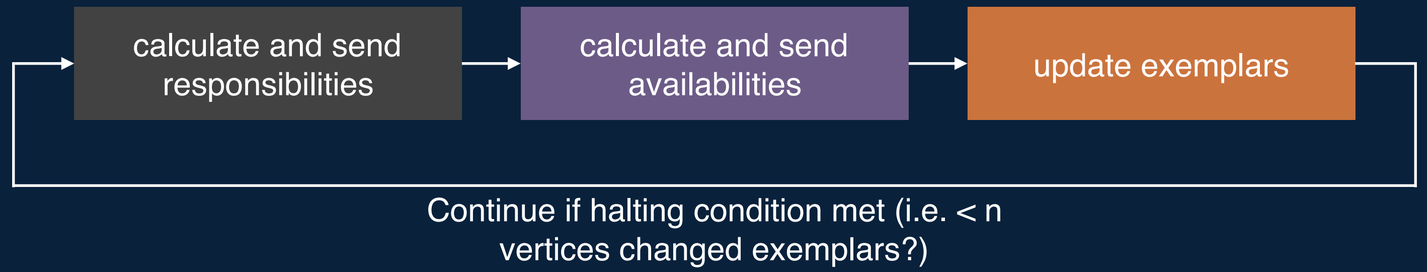
```
public abstract class WorkerContext {

  public abstract void preApplication();

  public abstract void postApplication();

  public abstract void preSuperstep();

  public abstract void postSuperstep();
…
}
```

Пример сложного приложения

Balanced Label Propagation

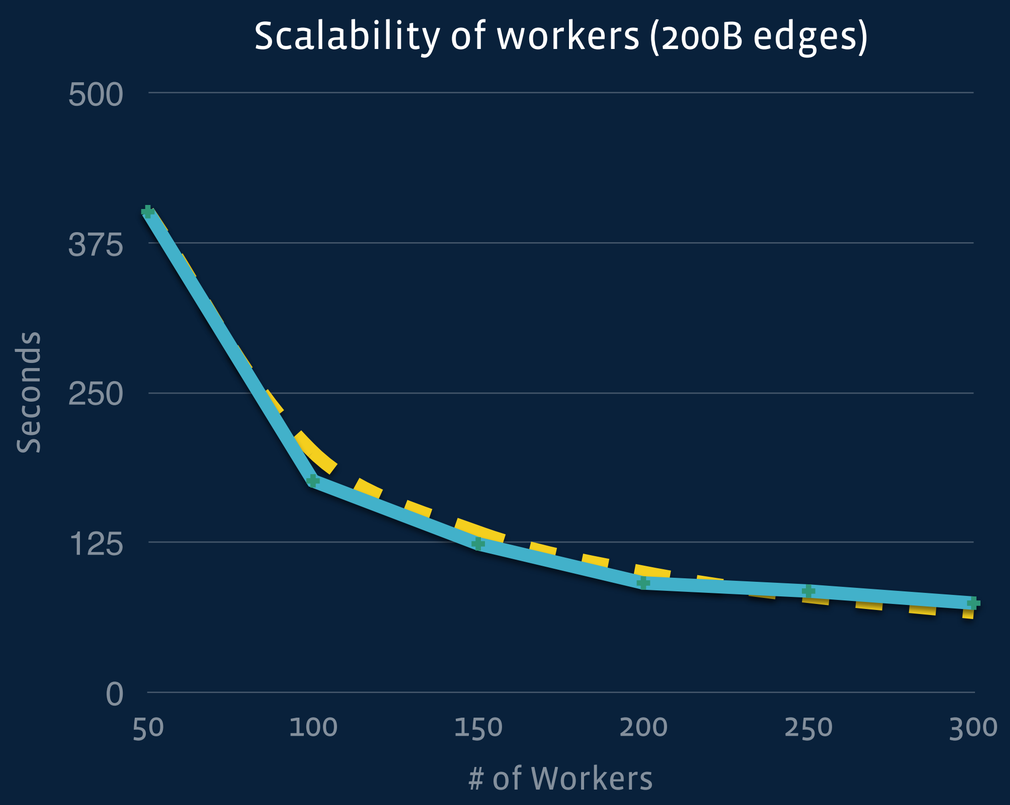


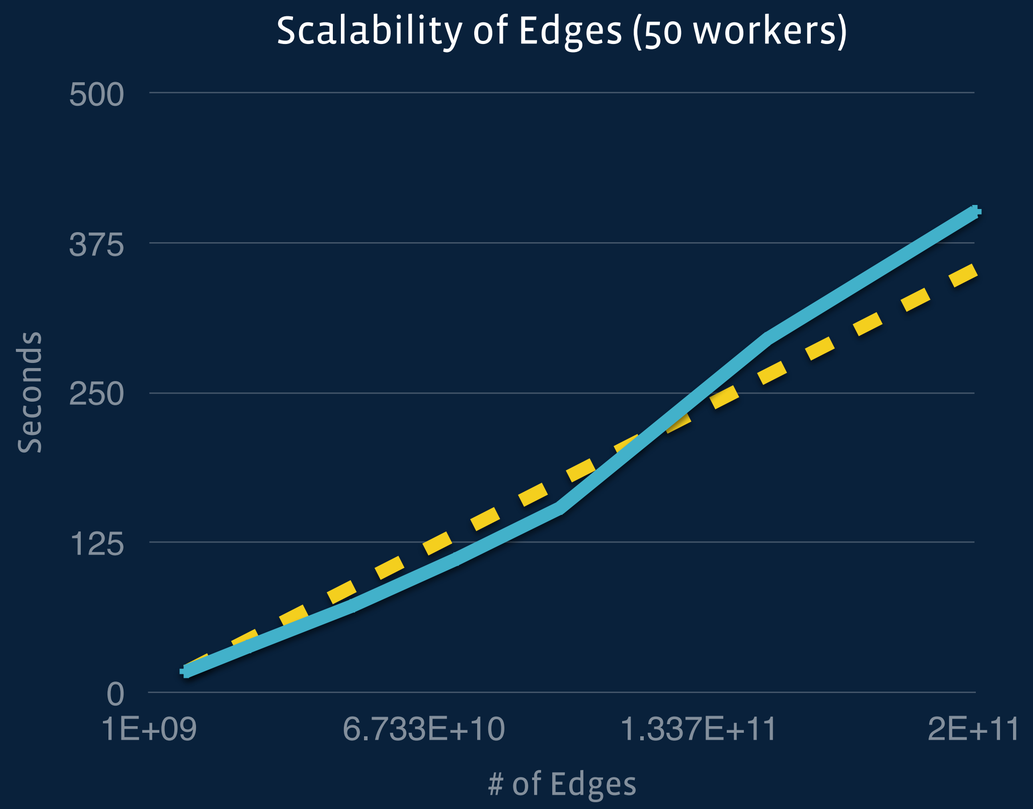
Affinity Propagation



Benchmarks/Performance

Scalability





Giraph Ideal

“ Миллиард ребер это не круто….
знаете, что круто? ”

“ Миллиард ребер это не круто….
знаете, что круто? ”

ТРИЛЛИОН РЕБЕР.

Giraph способен считать одну итерацию page rank на графе с **1,000,000,000,000+** ребер

< **3** минуты

используя кластер из 200 машин

Giraph vs Hive (Graph Applications)

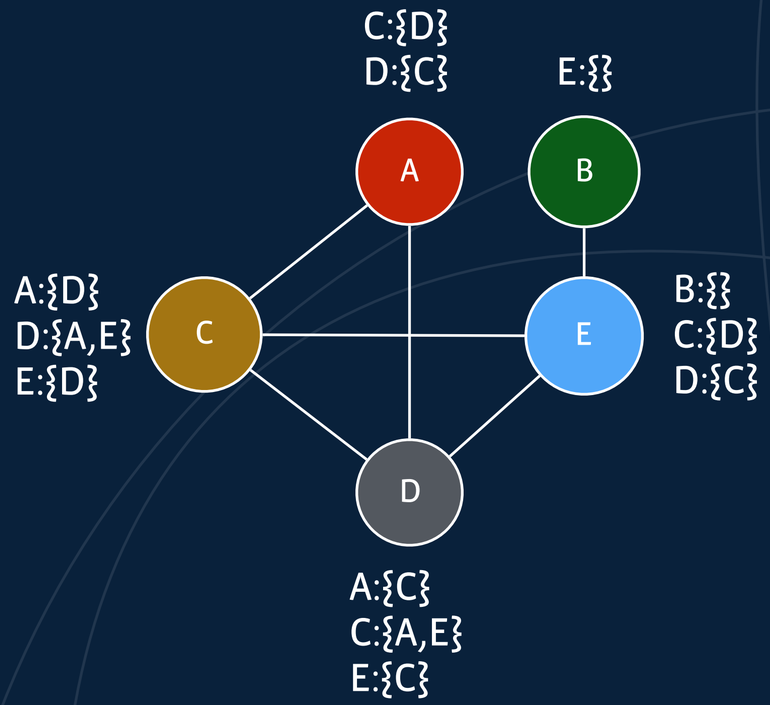
Application	Hive	Giraph	Speedup
Page rank (single iteration) 400B+ edges	Total CPU 16.5M secs	Total CPU 0.6M secs	26x
	Elapsed Time 600 mins	Elapsed Time 19 mins	120x
Friends-of-friends score 71B+ edges	Total CPU 255M secs	Total CPU 18M secs	14x
	Elapsed Time 7200 mins	Elapsed Time 110 mins	65x

Giraph vs Hive (Hive Queries)

Data Size	Hive	Giraph	Speedup
360B actions 40M objects	Total CPU 22 days	Total CPU 4 days	5.5x
	Elapsed Time 64 mins	Elapsed Time 7 mins	9.1x
162B actions 74M objects	Total CPU 92 days	Total CPU 18 days	5.1x
	Elapsed Time 129 mins	Elapsed Time 19 mins	6.8x
620B actions 110M objects	Total CPU 485 days	Total CPU 78 days	6.2x
	Elapsed Time 510 mins	Elapsed Time 45 mins	11.3x

Large/Imbalanced Supersteps

- Пример: вычисление количества общих друзей (локальный коэффициент кластеризации)
  - Отправить список друзей всем друзьям
  - Посчитать пересечение полученного списка со своим

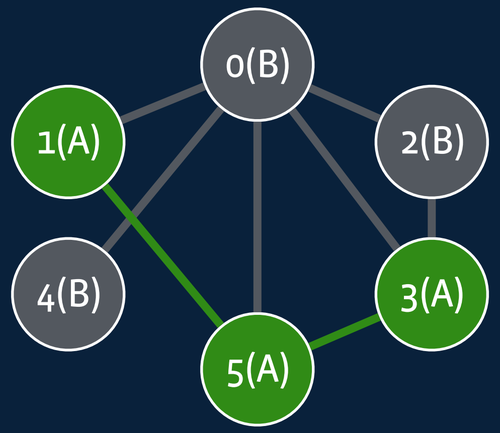


Messages memory = (1.23B MAP (as of 1/2014)) x (200+ average friends (2011 S1)) x (8-byte ids (longs))
= 394 TB of memory required for messages
= Assuming 100 GB machines, this is 3,940 machines (not including the graph)

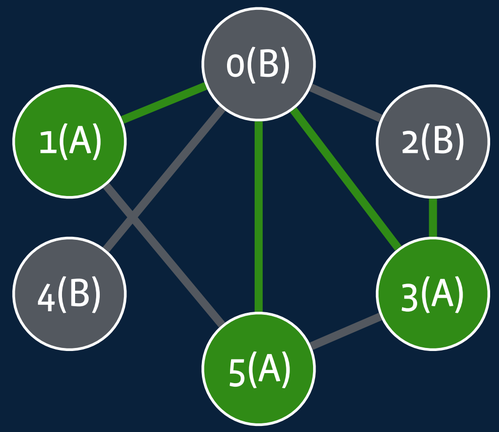
Superstep Splitting

- Разбить супер-шаг на несколько итераций
- Разделить сообщения по отправителям и получателям в несколько групп (например: hash(vertexId) % Ng)

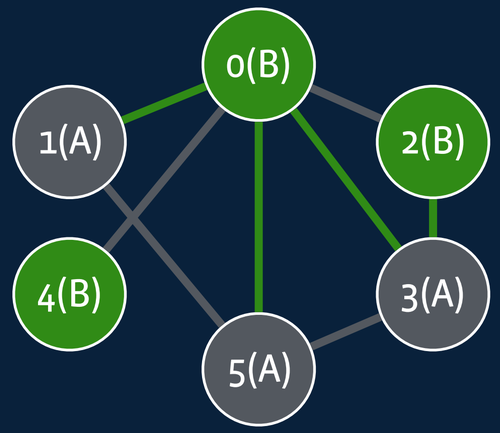
Sources: A (on), B (off)
Destinations: A (on), B (off)



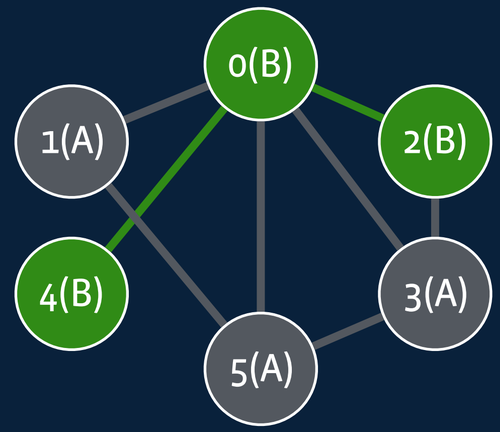
Sources: A (on), B (off)
Destinations: A (off), B (on)



Sources: A (off), B (on)
Destinations: A (on), B (off)



Sources: A (off), B (on)
Destinations: A (off), B (on)



Superstep Splitting Ограничения

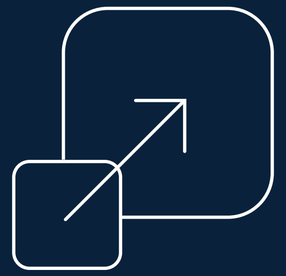
- Вычисления должны обладать свойствами коммутативность и ассоциативности
- Одно сообщение не должно превышать размера буфера
- В экстремальных случаях следует использовать специализированные алгоритмы деления графов для того чтобы сбалансировать нагрузку

Последние достижения

- Рекоммендации (https://code.facebook.com/posts/861999383875667/recommending-items-to-more-than-a-billion-people/)
- Blocks and Pieces (http://giraph.apache.org/blocks.html)
- Adaptive OOC

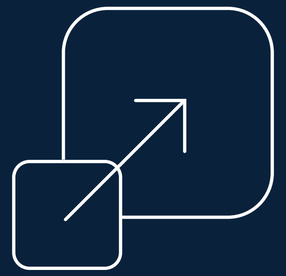
Future Work

Future Work



Scalability

Future Work



Scalability

Applications

Future Work

Performance

Future Work

Performance

Future Work

Thank You