



Apache Giraph

start analyzing graph relationships in your bigdata in 45 minutes

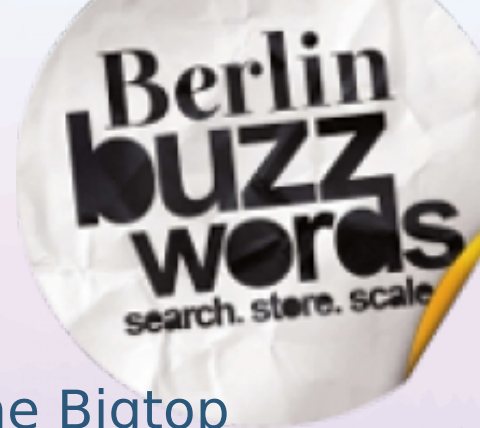
(or your money back)!



Who's this guy?

Roman Shaposhnik

- ASF junkie
 - VP of Apache Incubator, former VP of Apache Bigtop
 - Hadoop/Sqoop/Giraph committer
 - contributor across the Hadoop ecosystem)
- Used to be root@Cloudera
- Used to be a PHB at Yahoo!
- Used to be a UNIX hacker at Sun microsystems



Giraph in action (MEAP)



<http://manning.com/martella/>

I am hiring!

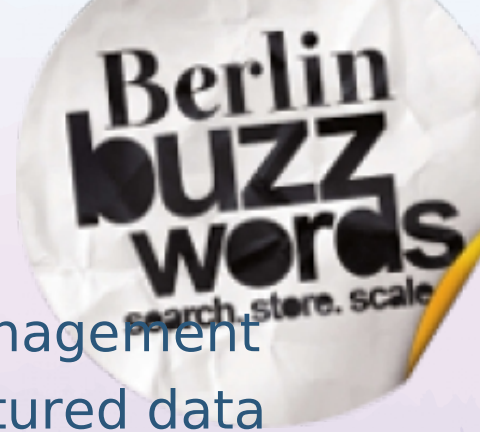




What's this all about?

Agenda

- A brief history of Hadoop-based bigdata management
- Extracting graph relationships from unstructured data
- A case for iterative and explorative workloads
- Bulk Synchronous Parallel to the rescue!
- Apache Giraph: a Hadoop-based BSP graph analysis framework
- Giraph application development
- Demos! Code! Lots of it!





On day one Doug created
HDFS/MR

Google papers

- GFS (file system)
 - distributed
 - replicated
 - non-POSIX
- MapReduce (computational framework)
 - distributed
 - batch-oriented (long jobs; final results)
 - data-gravity aware
 - designed for “embarrassingly parallel” algorithms



One size doesn't fit all

- Key-value approach
 - map is how we get the keys
 - shuffle is how we sort the keys
 - reduce is how we get to see all the values for a key
- Pipeline approach
- Intermediate results in a pipeline need to be flushed to HDFS
- A very particular “API” for working with your data





It's not about the size of your
data;
it's about what you do with it!

Graph relationships

- Entities in your data: tuples
 - customer data
 - product data
 - interaction data
- Connection between entities: graphs
 - social network or my customers
 - clustering of customers vs. products



Challenges

- Data is dynamic
 - No way of doing “schema on write”
- Combinatorial explosion of datasets
 - Relationships grow exponentially
- Algorithms become
 - explorative
 - iterative



Graph databases

- Plenty available
 - Neo4J, Titan, etc.
- Benefits
 - Tightly integrate systems with few moving parts
 - High performance on known data sets
- Shortcomings
 - Don't integrate with HDFS
 - Combine storage and computational layers
 - A sea of APIs





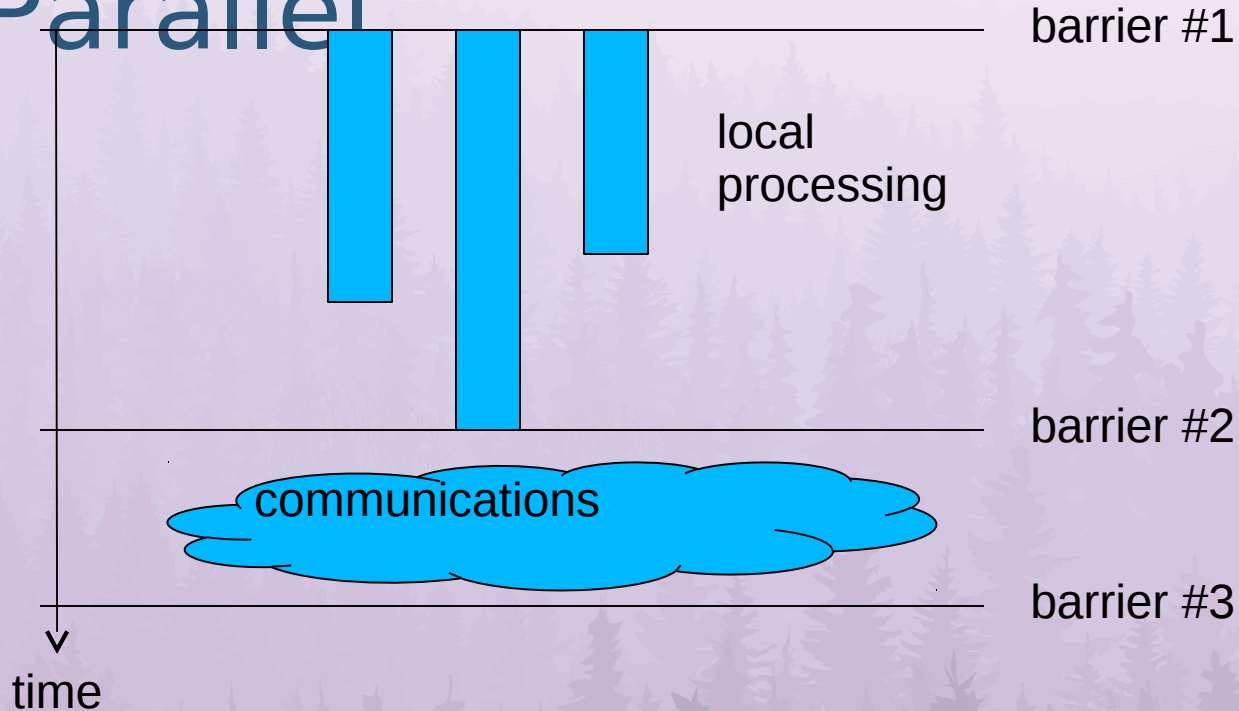
Enter Apache Giraph

Key insights

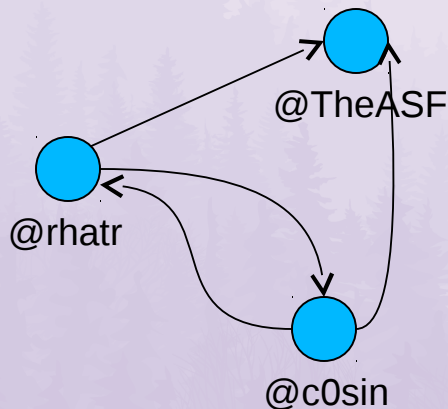
- Keep state in memory for as long as needed
- Leverage HDFS as a repository for unstructured data
- Allow for maximum parallelism (shared nothing)
- Allow for arbitrary communications
- Leverage BSP approach



Bulk Synchronous Parallel



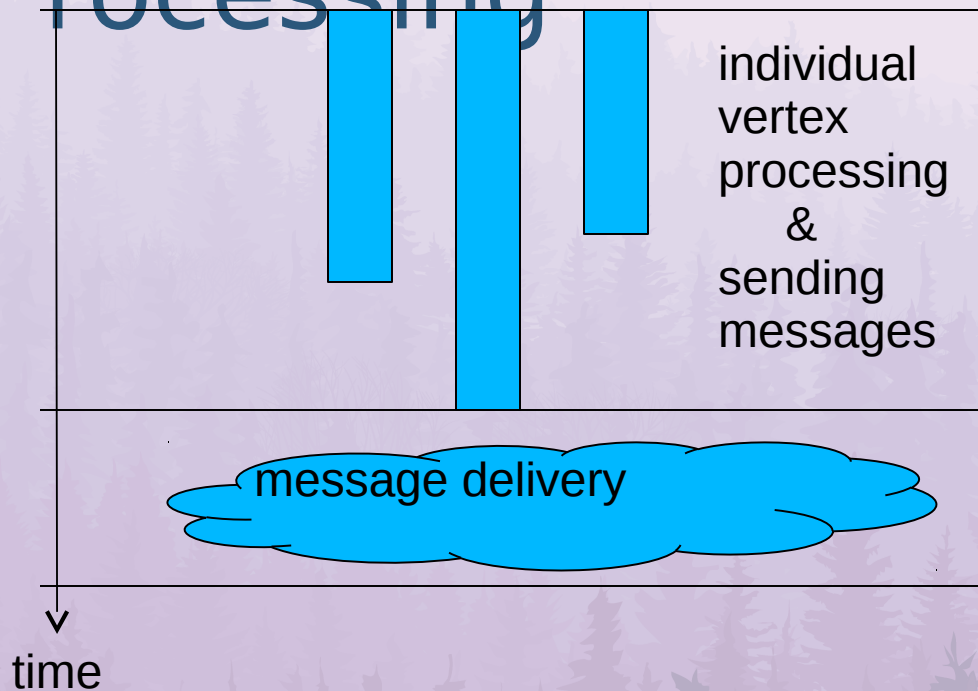
BSP applied to graphs



Think like a vertex:

- I know my local state
- I know my neighbours
- I can send messages to vertices
- I can declare that I am done
- I can mutate graph topology

Bulk Sequential Processing



superstep #1

individual
vertex
processing
&
sending
messages

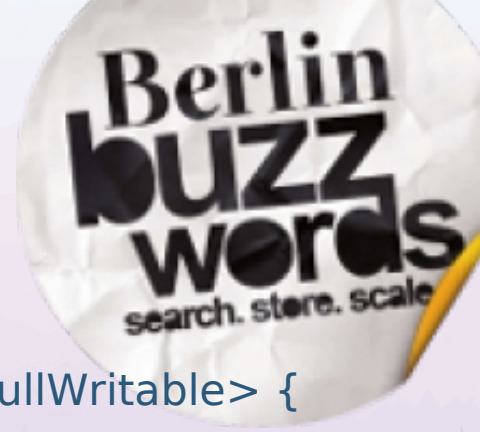
all vertices are "done"

message delivery

superstep #2

time

Giraph “Hello World”



```
public class GiraphHelloWorld extends
    BasicComputation<IntWritable, IntWritable, NullWritable, NullWritable> {

    public void compute(Vertex<IntWritable, IntWritable, NullWritable> vertex,
        Iterable<NullWritable> messages) {
        System.out.println("Hello world from the: " + vertex.getId() + " : ");
        for (Edge<IntWritable, NullWritable> e : vertex.getEdges()) {
            System.out.println(" " + e.getTargetVertexId());
        }
        System.out.println("");
    }
}
```


Mighty four of Giraph API



```
BasicComputation<IntWritable, // VertexID -- vertex ref
                    IntWritable, // VertexData -- a vertex
                    datum
                    NullWritable, // EdgeData -- an edge
                    label datum
                    NullWritable>// MessageData -- message
                    payload
```

On circles and arrows

- You don't even need a graph to begin with!
 - Well, ok you need at least one node
- Dynamic extraction of relationships
 - EdgelineInputFormat
 - VetexInputFormat
- Full integration with Hadoop ecosystem
 - HBase/Accumulo, Gora, Hive/HCatalog



Anatomy of Giraph run

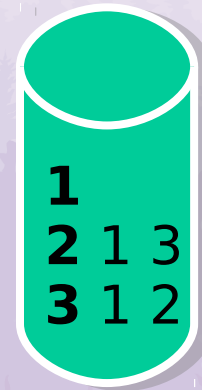
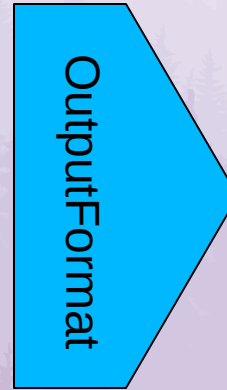
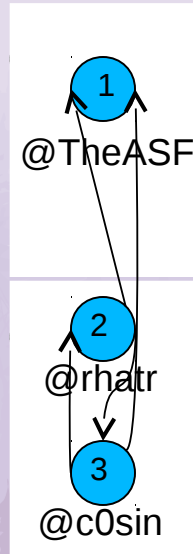
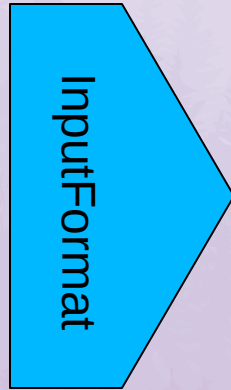


HDFS

mappers

reducers

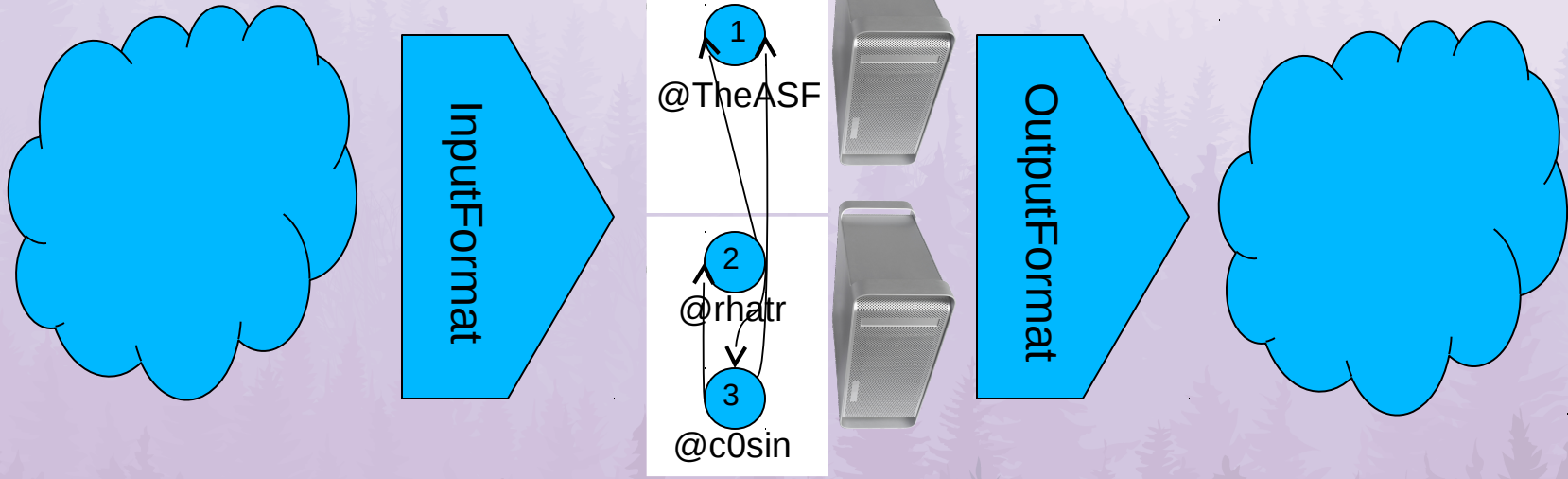
HDFS



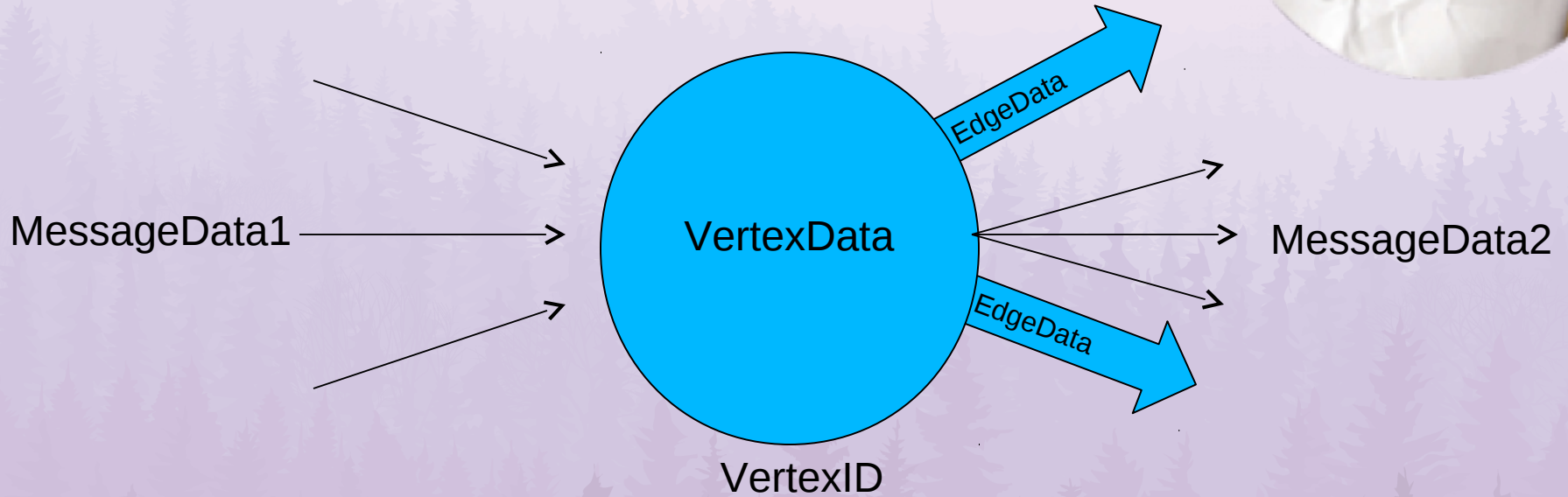
Anatomy of Giraph run



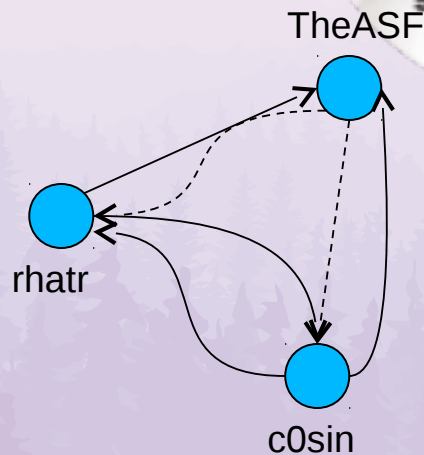
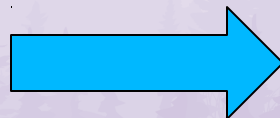
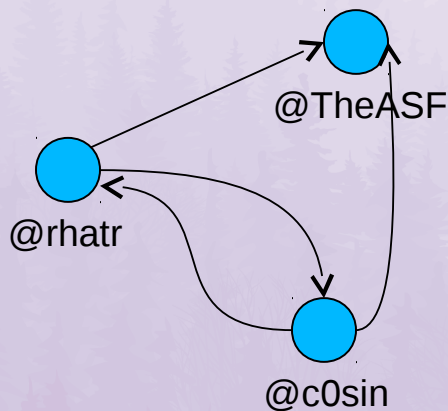
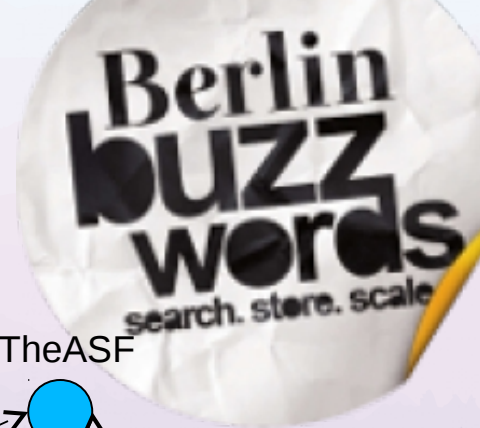
mappers or YARN containers



A vertex view



Turning Twitter into Facebook



Ping thy neighbours



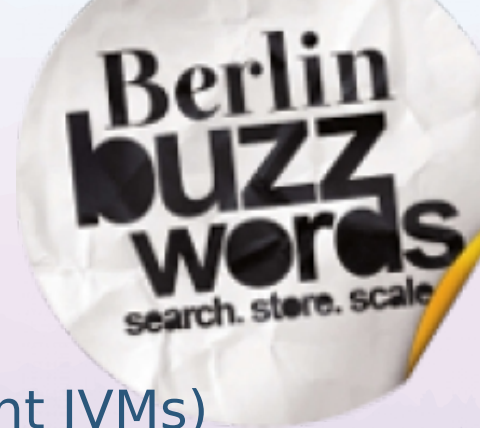
```
public void compute(Vertex<Text, DoubleWritable, DoubleWritable> vertex,
    Iterable<Text> ms ){
    if (getSuperstep() == 0) {
        sendMessageToAllEdges(vertex, vertex.getId());
    } else {
        for (Text m : ms) {
            if (vertex.getEdgeValue(m) == null) {
                vertex.addEdge(EdgeFactory.create(m, SYNTHETIC_EDGE));
            }
        }
    }
    vertex.voteToHalt();
}
```



Demo time!

But I don't have a cluster!

- Hadoop in pseudo-distributed mode
 - All Hadoop services on the same host (different JVMs)
- Hadoop-as-a-Service
 - Amazon's EMR, etc.
- Hadoop in local mode



Prerequisites

- Apache Hadoop 1.2.1
- Apache Giraph 1.1.0-SNAPSHOT
- Apache Maven 3.x
- JDK 7+



Setting things up

```
$ curl hadoop.tar.gz | tar xzvf -
```

```
$ git clone git://git.apache.org/giraph.git ; cd giraph
```

```
$ mvn -Phadoop_1 package
```

```
$ tar xzvf *dist*/*.tar.gz
```

```
$ export HADOOP_HOME=/Users/shapor/dist/hadoop-1.2.1
```

```
$ export GIRAPH_HOME=/Users/shapor/dist/
```

```
$ export HADOOP_CONF_DIR=$GIRAPH_HOME/conf
```

```
$ PATH=$HADOOP_HOME/bin:$GIRAPH_HOME/bin:$PATH
```



Setting project up (maven)

```
<dependencies>
  <dependency>
    <groupId>org.apache.giraph</groupId>
    <artifactId>giraph-core</artifactId>
    <version>1.1.0-SNAPSHOT</version>
  </dependency>

  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-core</artifactId>
    <version>1.2.1</version>
  </dependency>
</dependencies>
```



Running it

```
$ mvn package
```

```
$ giraph target/*.jar giraph.GiraphHelloWorld \
```

```
-vip src/main/resources/1 \
```

```
-vif
```

```
org.apache.giraph.io.formats.IntIntNullTextInputFormat \
```

```
-w 1 \
```

```
-ca
```

```
giraph.SplitMasterWorker=false,giraph.logLevel=error
```



Testing it

```
public void testNumberOfVertices() throws Exception {
    GiraphConfiguration conf = new GiraphConfiguration();
    conf.setComputationClass(GiraphHelloWorld.class);

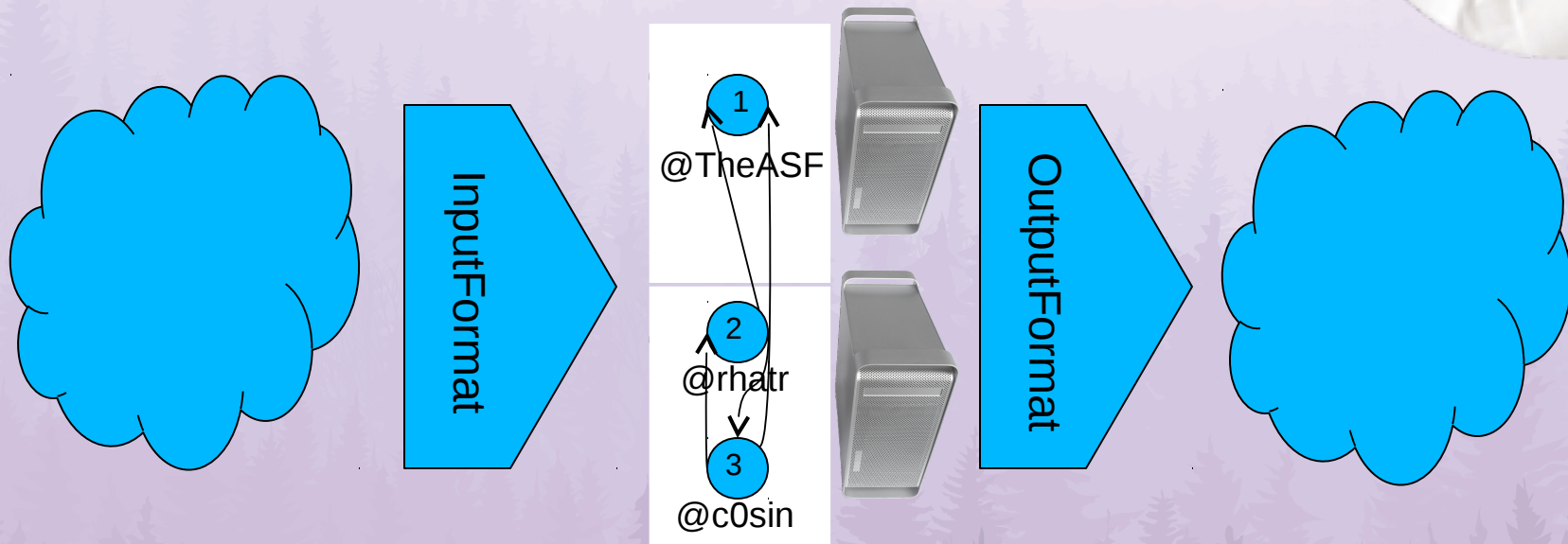
    conf.setVertexInputFormatClass(TextDoubleDoubleAdjacencyListVertexInputForma
t.class);

    ...
    Iterable<String> results =
        InternalVertexRunner.run(conf, graphSeed);
    ...
}
```

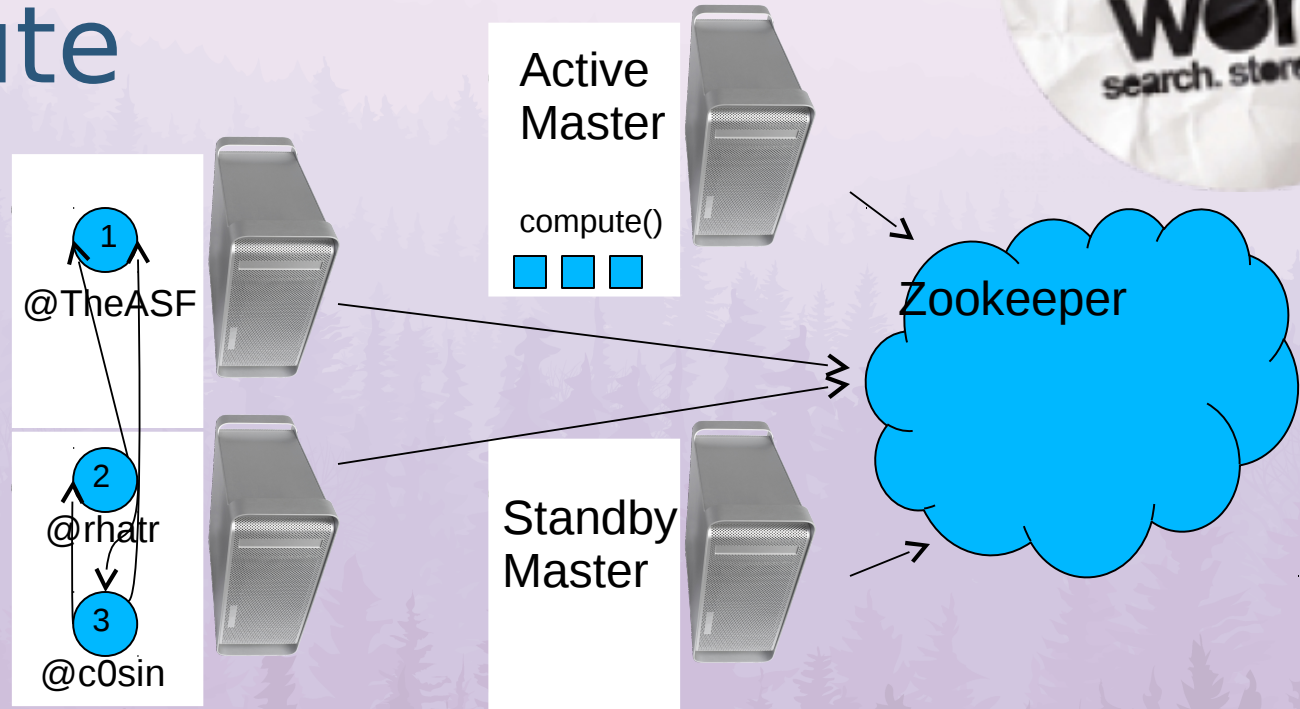


Simplified view

mappers or YARN containers



Master and master compute



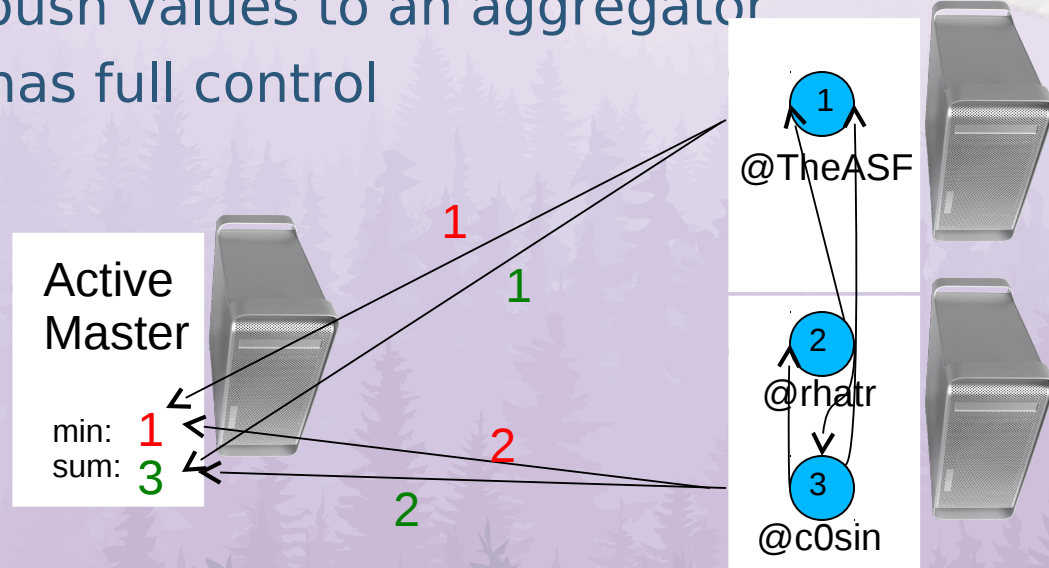
Master compute

- Runs before slave compute()
- Has a global view
- A place for aggregator manipulation



Aggregators

- “Shared variables”
- Each vertex can push values to an aggregator
- Master compute has full control





Questions?