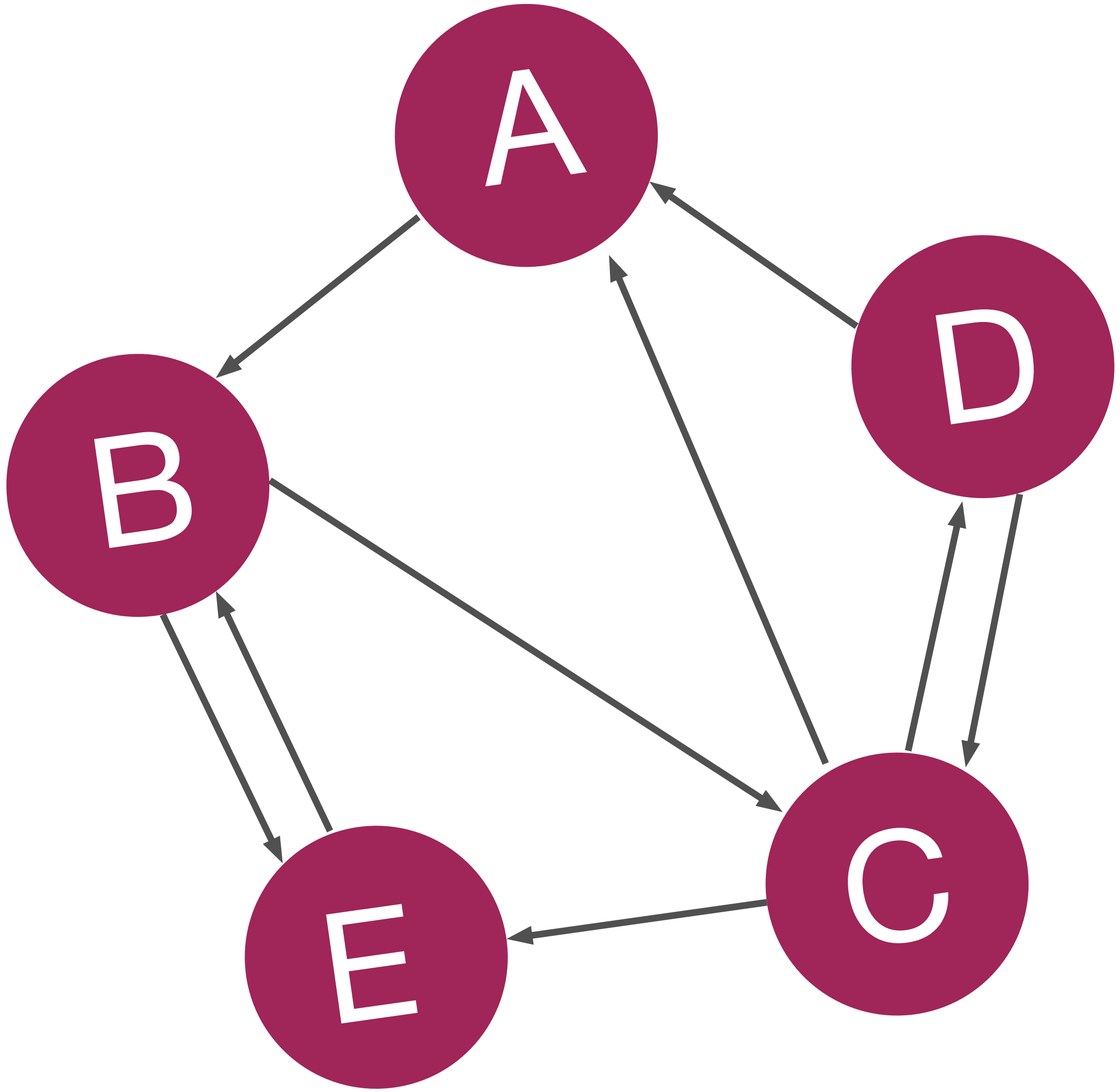
PageRank optimization

```
lines = sc.textFile("links.txt")
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add)
                    .mapValues(lambda rank: rank * 0.85 + 0.15)

ranks.collect()
```

```
lines = sc.textFile("links.txt")
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add)
                    .mapValues(lambda rank: rank * 0.85 + 0.15)

ranks.collect()
```


A
B
D
E
C

- 10 executors, 2 cores & 4GB per executor
- 9 lines
- 30 iterations

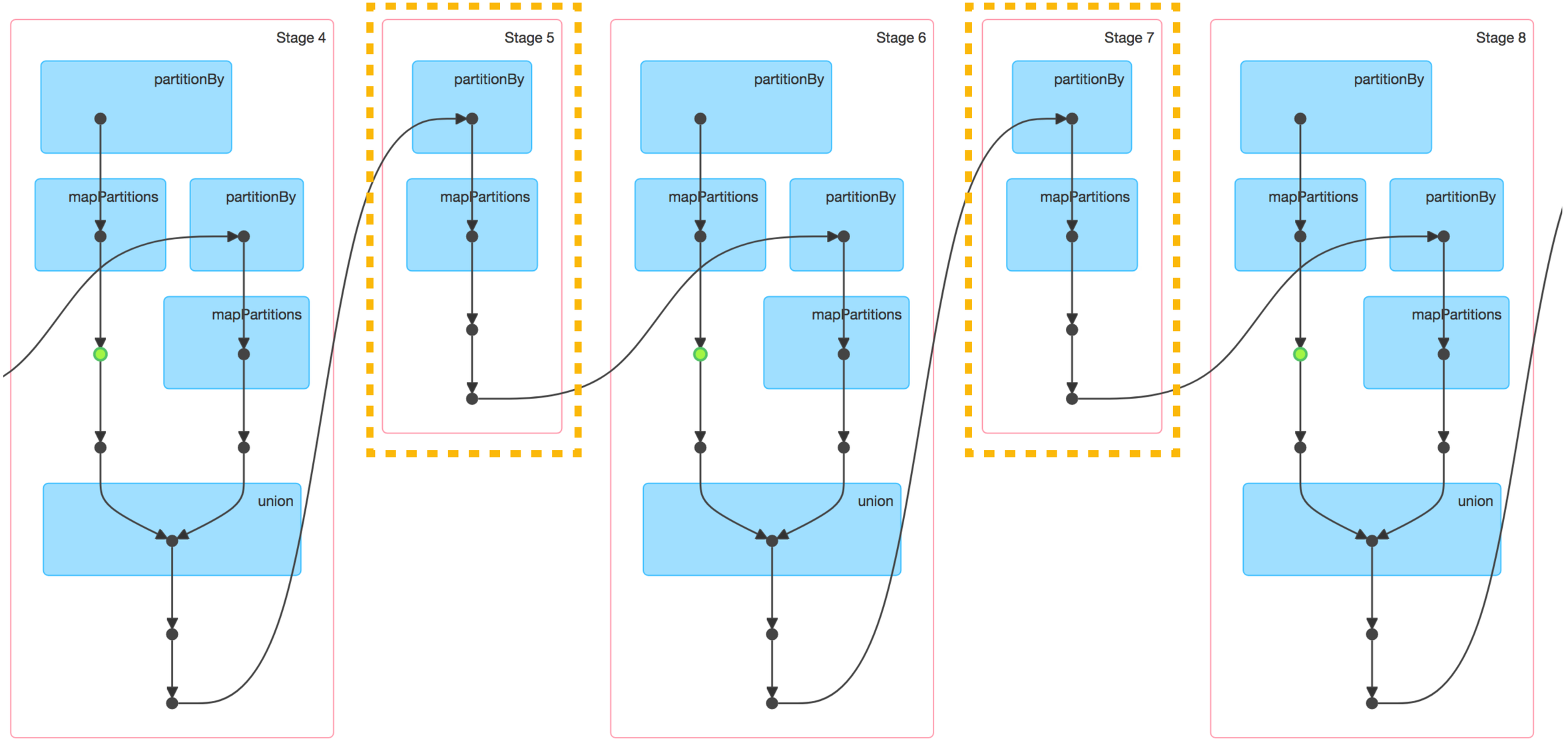
Spark Jobs (?)

User: pklemenkov
Total Uptime: 1.1 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <magic-timeit>:257	2017/07/25 04:50:52	24 s	63/63	2046/2046


Stage 4
partitionBy
mapPartitions
partitionBy
mapPartitions
union
Stage 5
partitionBy
mapPartitions
Stage 6
partitionBy
mapPartitions
partitionBy
mapPartitions
union
Stage 7
partitionBy
mapPartitions
Stage 8
partitionBy
mapPartitions
partitionBy
mapPartitions
union

```
for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add)
                    .mapValues(lambda rank: rank * 0.85 + 0.15)
```

```
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

>> links.getNumPartitions(), links.partitioner
(2, <pyspark.rdd.Partitioner at 0x7f1ffdb65050>)

>> ranks.getNumPartitions(), ranks.partitioner
(2, None)

>> links.partitioner == ranks.partitioner
False
```

```
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
             .partitionBy(2)
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

>> links.getNumPartitions(), links.partitioner
(2, <pyspark.rdd.Partitioner at 0x7f1ffdb65050>)

>> ranks.getNumPartitions(), ranks.partitioner
(2, None)

>> links.partitioner == ranks.partitioner
False
```

```
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
              .partitionBy(2)
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

>> links.getNumPartitions(), links.partitioner
(2, <pyspark.rdd.Partitioner at 0x7f1ffdb65050>)

>> ranks.getNumPartitions(), ranks.partitioner
(2, None)

>> links.partitioner == ranks.partitioner
False
```

```
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
              .partitionBy(2)
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0),
                  preservePartitioning=True)

>> links.getNumPartitions(), links.partitioner
(2, <pyspark.rdd.Partitioner at 0x7f1ffdb65050>)

>> ranks.getNumPartitions(), ranks.partitioner
(2, <pyspark.rdd.Partitioner at 0x7f1ffdb65050>)

>> links.partitioner == ranks.partitioner
True
```

```
for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add,
                                 numPartitions=links.getNumPartitions())
                    .mapValues(lambda rank: rank * 0.85 + 0.15)
```

```
for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add,
                                 numPartitions=links.getNumPartitions())
                    .mapValues(lambda rank: rank * 0.85 + 0.15)
```

```
links = lines.map(parseNeighbors)
             .distinct()
             .groupByKey()
             .partitionBy(2)
             .cache()
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0)),
                  preservePartitioning=True)

for iteration in xrange(n):
    contribs = links.join(ranks).flatMap(
                        lambda u: computeContribs(u[1][0], u[1][1]))
    ranks = contribs.reduceByKey(add,
                                 numPartitions=links.getNumPartitions())
                    .mapValues(lambda rank: rank * 0.85 + 0.15)
ranks.collect()
```

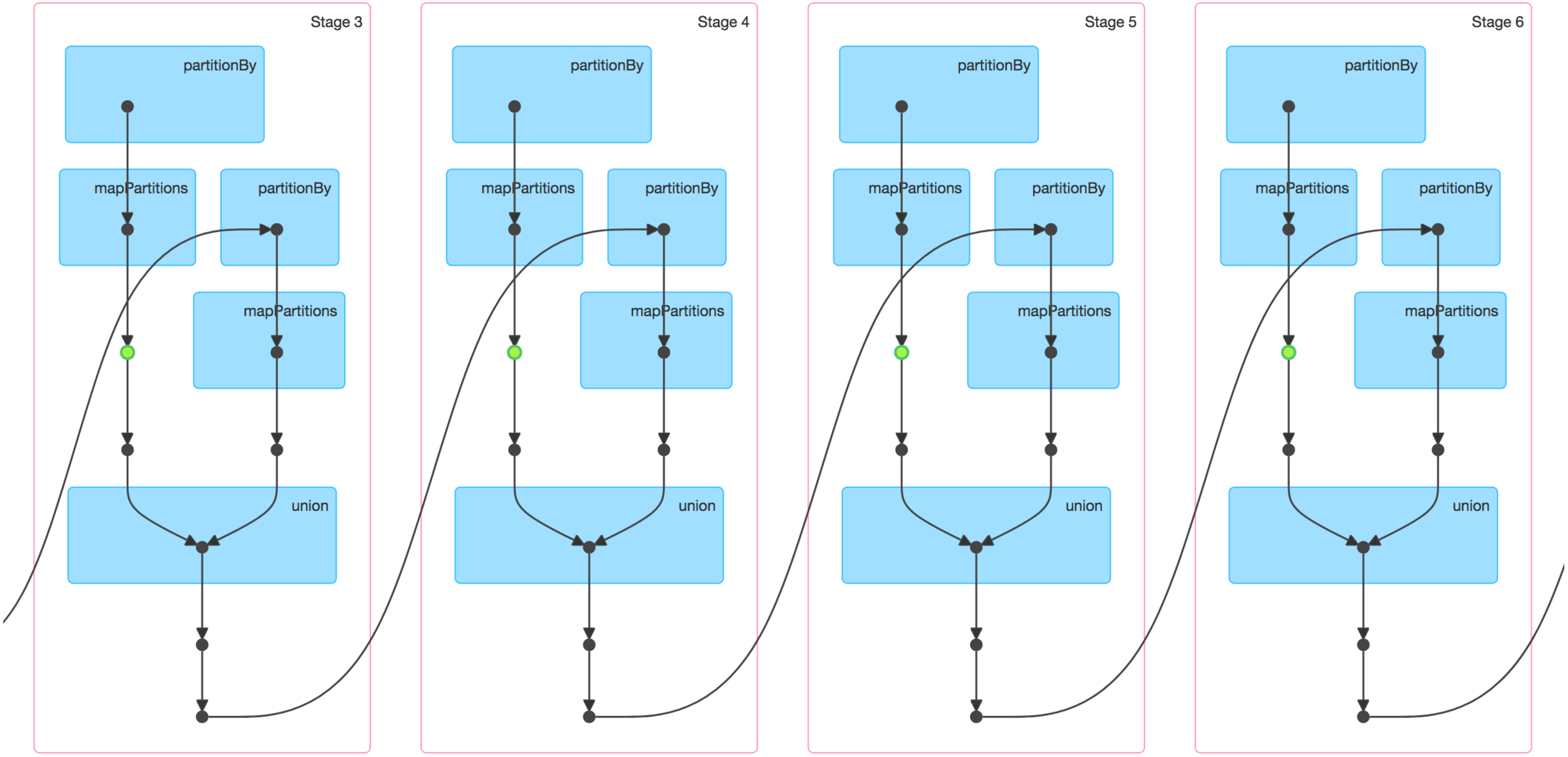
Spark Jobs (?)

User: pklemenkov
Total Uptime: 55 s
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <magic-timeit>:257	2017/07/25 05:16:55	12 s	33/33	66/66


Stage 3
partitionBy
mapPartitions
partitionBy
mapPartitions
union
Stage 4
partitionBy
mapPartitions
partitionBy
mapPartitions
union
Stage 5
partitionBy
mapPartitions
partitionBy
mapPartitions
union
Stage 6
partitionBy
mapPartitions
partitionBy
mapPartitions
union

- 10 executors, 2 cores & 4GB per executor
- Berkeley-Stanford web graph
  - 685230 nodes
  - 7600595 edges
  - 106 MB uncompressed
- 5 iterations

Spark Jobs (?)

User: pklemenkov
Total Uptime: 5.6 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <magic-timeit>:257	2017/07/25 05:30:57	4.1 min	13/13	192/192

Spark Jobs (?)

User: pklemenkov
Total Uptime: 5.6 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

~20% performance gain

Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <magic-timeit>:257	2017/07/25 05:30:57	4.1 min	13/13	192/192

Summary

- Reduce shuffles!
- Explicit (known) partitioner
- Preserve partitioner
- Reduce shuffle volume
  - reduceByKey vs groupByKey