SEISMIC 3D VOLUME TRANSPOSE ON APACHE SPARK

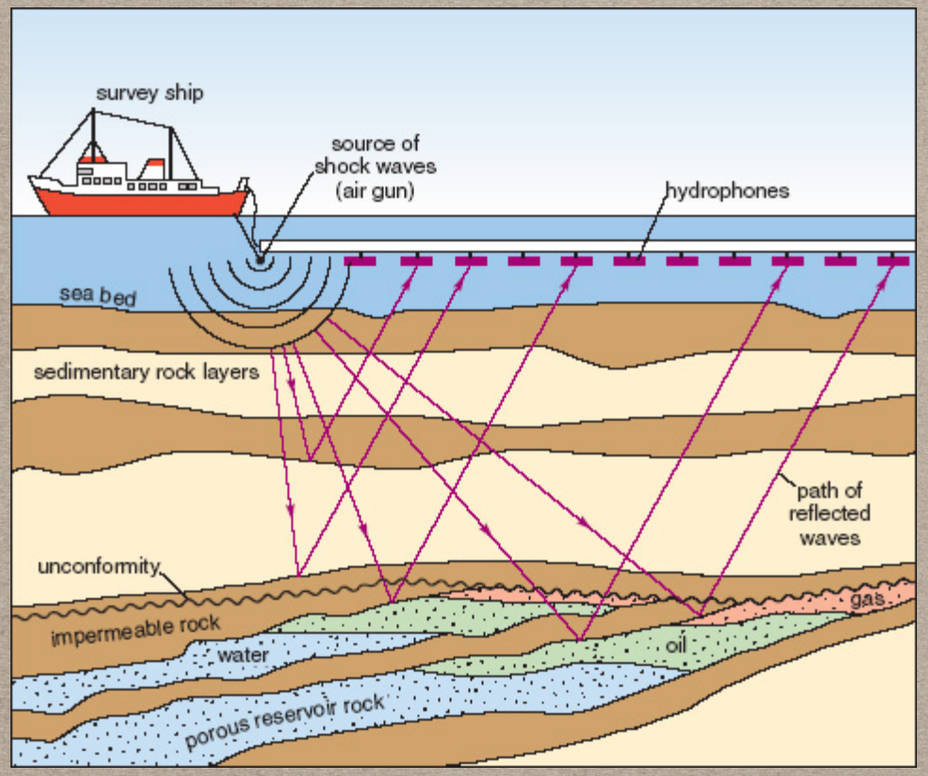
PARALLEL COMPUTING FINAL PROJECT

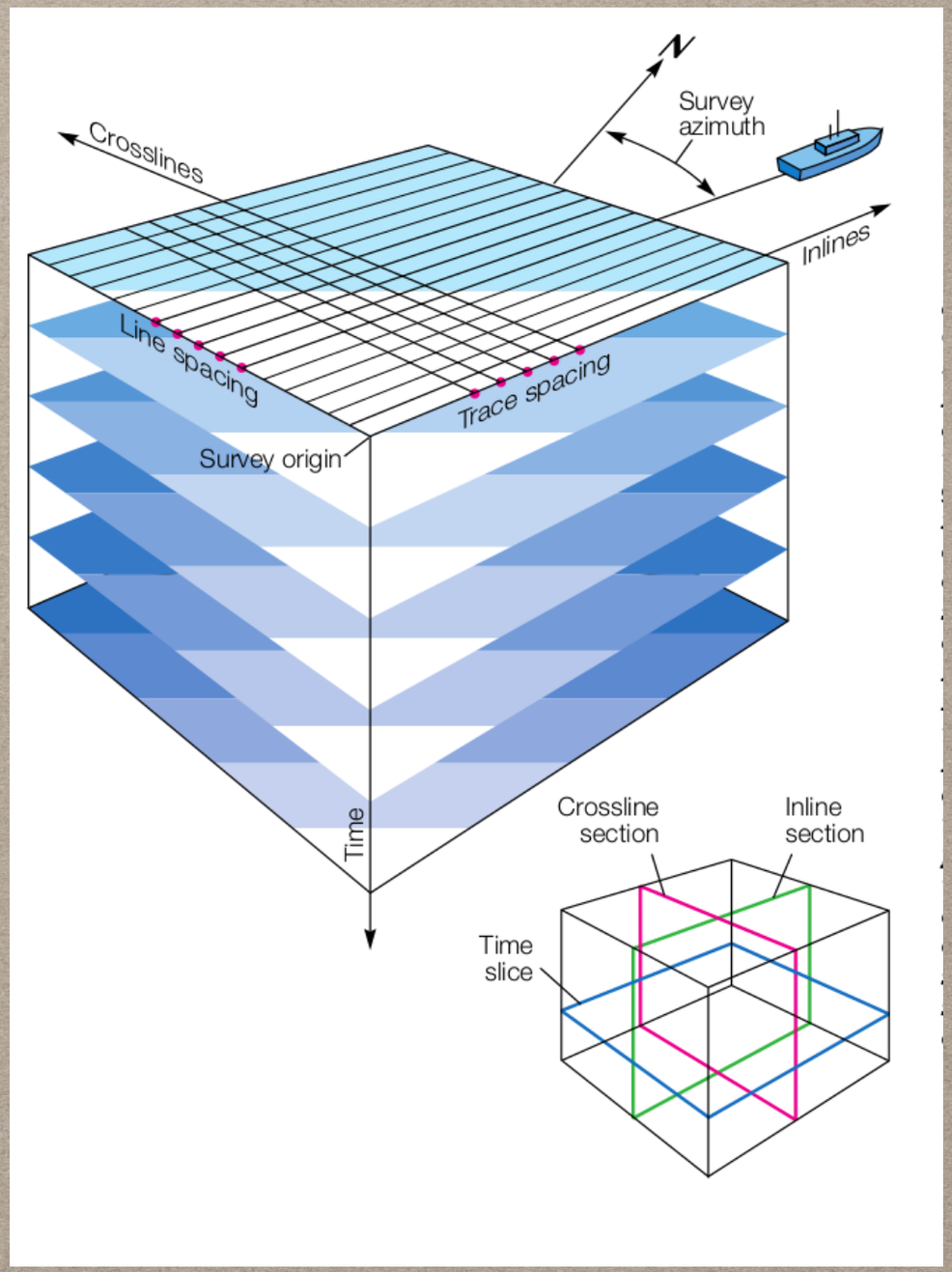
ADVISOR: DR. LEI HUANG
STUDENT: CHAO CHEN

OUTLINE

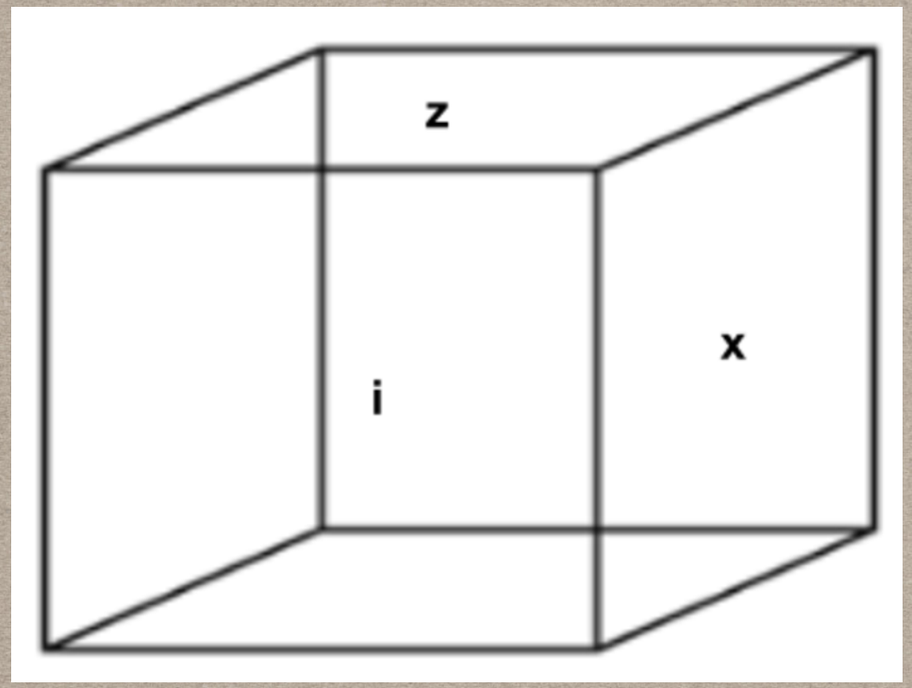
- Introduction
  - Seismic 3D Volume Data & Transpose
  - Spark & Spark RDD programming model
- Implementation
  - Distributed 3D Volume Data
  - Traces-Indexed flatMap
  - Group
  - Sort
- Performance Analysis
  - NMON & Spark Web UI
  - Timeline
  - Workers(Nodes) Activities
- Conclusion

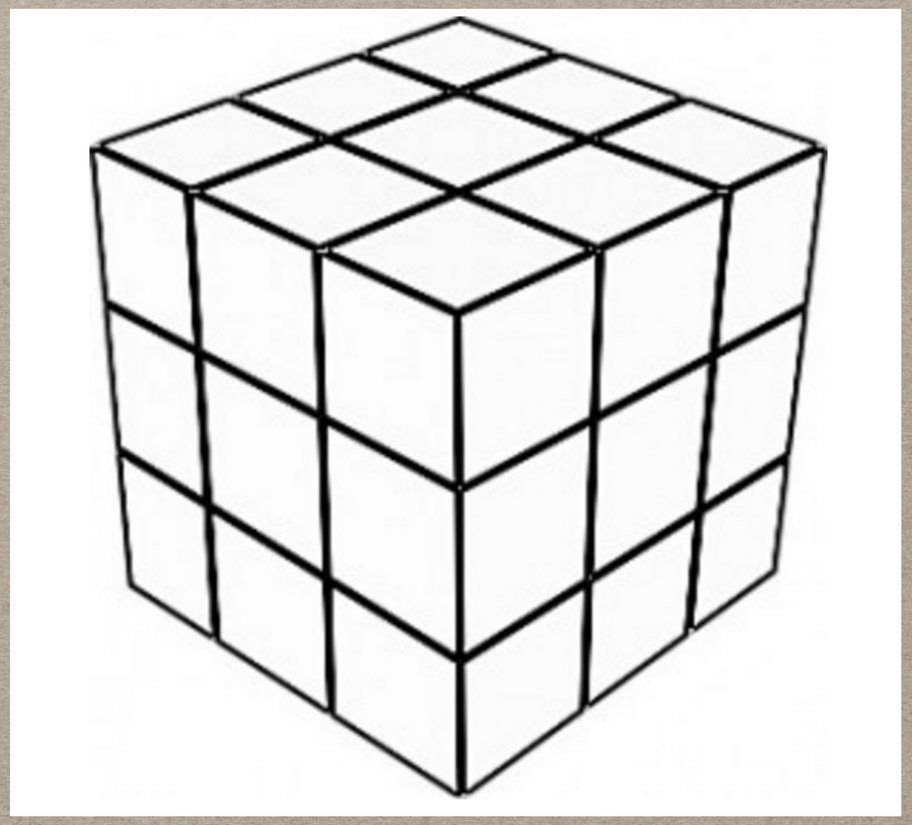
SEISMIC 3D VOLUME DATA





SEISMIC 3D VOLUME DATA





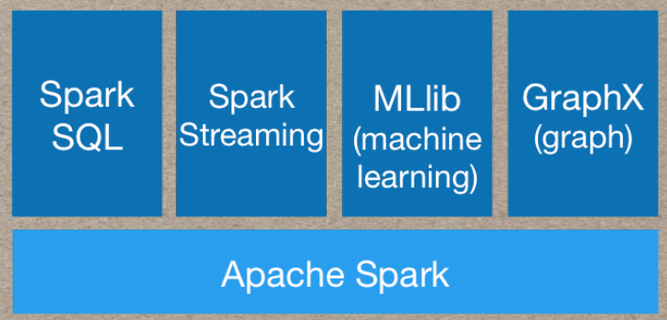
APACHE SPARK

- Speed

  Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

- Easy of Use

- Fault tolerance



SPARK RDD PROGRAMMING MODEL

```
val textFile = spark.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

- Input from Sequential or Distributed FileSystem
- Map function: Api to perform operation on each split.
- RDD: Interface to manage your distribution.

SPARK RDD MODEL

Scientific Answer: RDD is an Interface!

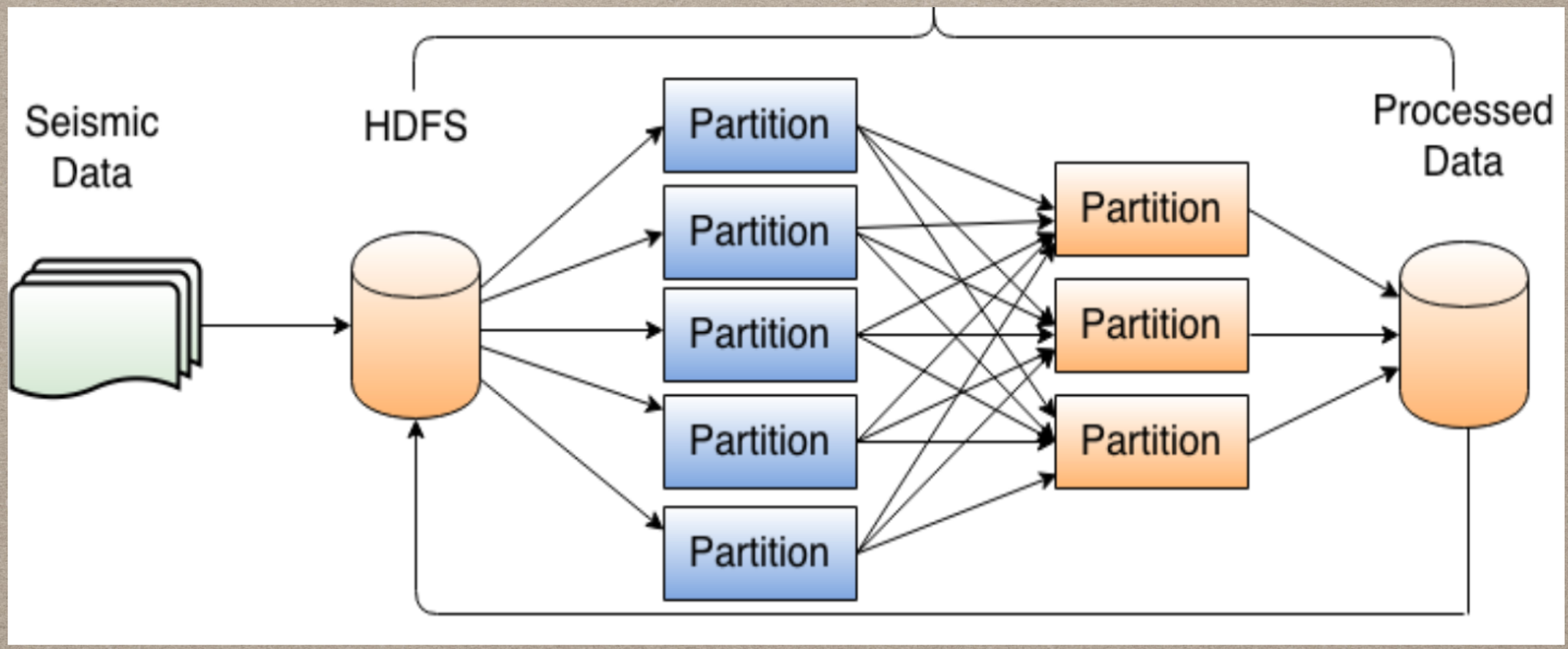
1. Set of *partitions* ("splits" in Hadoop)
2. List of *dependencies* on parent RDDs
3. Function to *compute* a partition (as an Iterator) given its parent(s)

"lineage"

4. (Optional) *partitioner* (hash, range)
5. (Optional) *preferred location*(s) for each partition

optimized execution

VOLUME DATA DISTRIBUTION



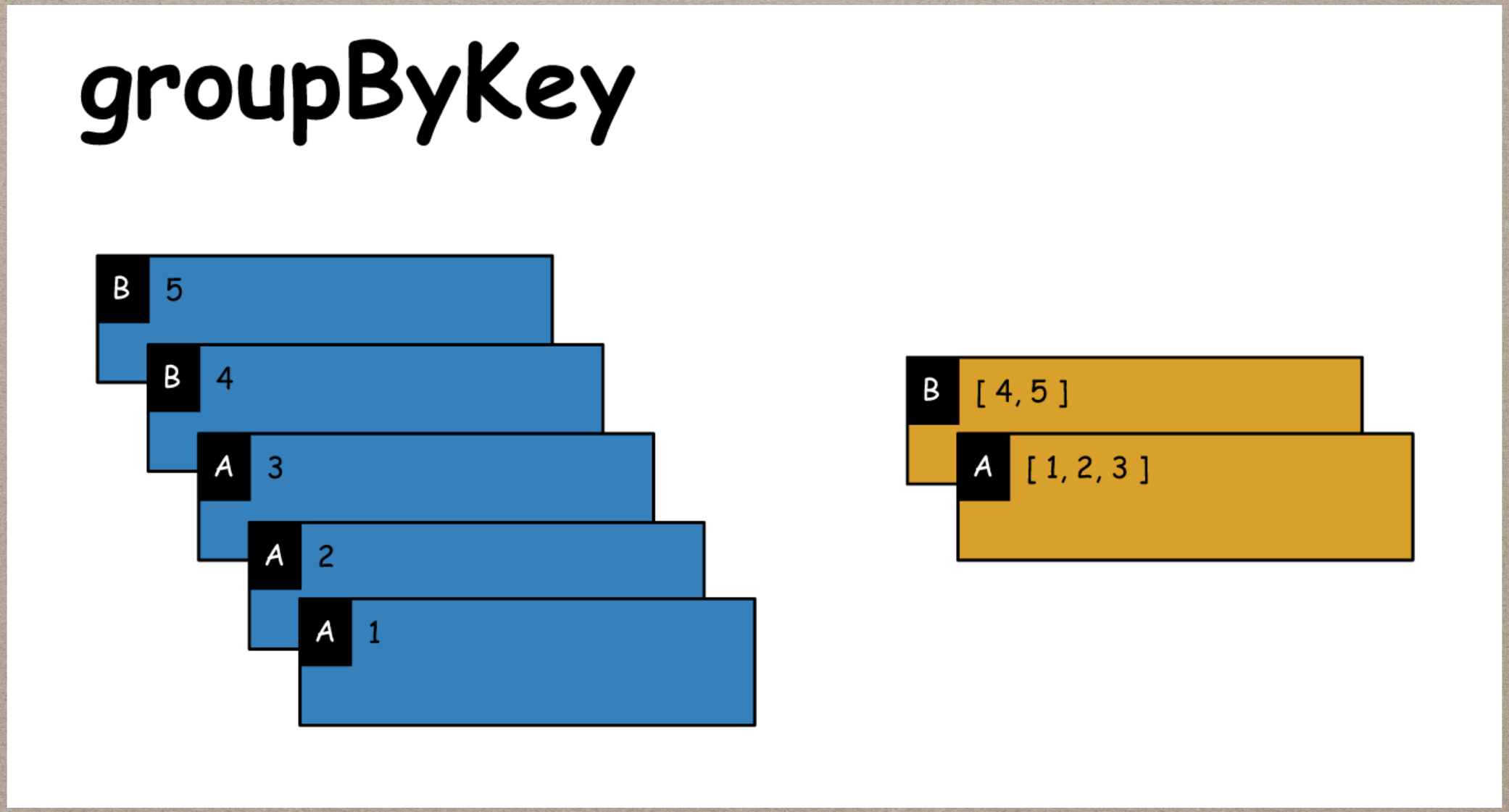
INDEXED FLATMAP
INDEX: (IDX(TRACE)-IDX(SLICE))

```
def transposeI2X() = {

    def i2x(rdd:(BytesWritable,BytesWritable), dimK:Int, dimJ:Int, dimI:Int) = {
        val idxI = SeismicData.getIndex(rdd._1);
        val cnt = SeismicData.getNum(rdd._1);
        val lines = SeismicData.getFloatsPair(rdd._2, dimJ, dimK, cnt, idxI);
        var traces = new Array[(Int, Array[Float])](dimJ * cnt);
        for (i <- 0 until cnt) {
            val iline = lines(i)._2;
            val ikey = lines(i)._1;
            for (j <- 0 until dimJ) {
                val trace = iline.slice(j * dimK, (j + 1) * dimK);
                traces(i * dimJ + j) = ((j << 16) + ikey, trace);
            }
        }
        traces
    }

    println("transposeI2X");
    val ktraces = inlineBytesRDD.flatMap( {case (k,v) => i2x((k,v),sizeK,sizeJ,sizeI)} )
    xlineArrayRDD = ktraces.groupBy(grpFunc).map(sortFunc).sortByKey(true, sizeJ + 1);
    xlineArrayRDD
}
```

GROUP BY TRACE INDEX



SORT BY TRACE & SLICE INDEX

- Slice index: Sort the traces in slice after transposing.
- Trace index: Sort the slices after transposing.

PERFORMANCE ANALYSIS

- Spark Web UI: Task Monitoring & Timeline
- NMON: Resource & activity monitoring

TIMELINE

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20151202150525-0028	org.pvamu.sac.sdk.TransTest	224	15.6 GB	2015/12/02 15:05:25	root	FINISHED	17 s
app-20151202150202-0027	org.pvamu.sac.sdk.TransTest	224	15.6 GB	2015/12/02 15:02:02	root	FINISHED	10 s

Spark Jobs (?)

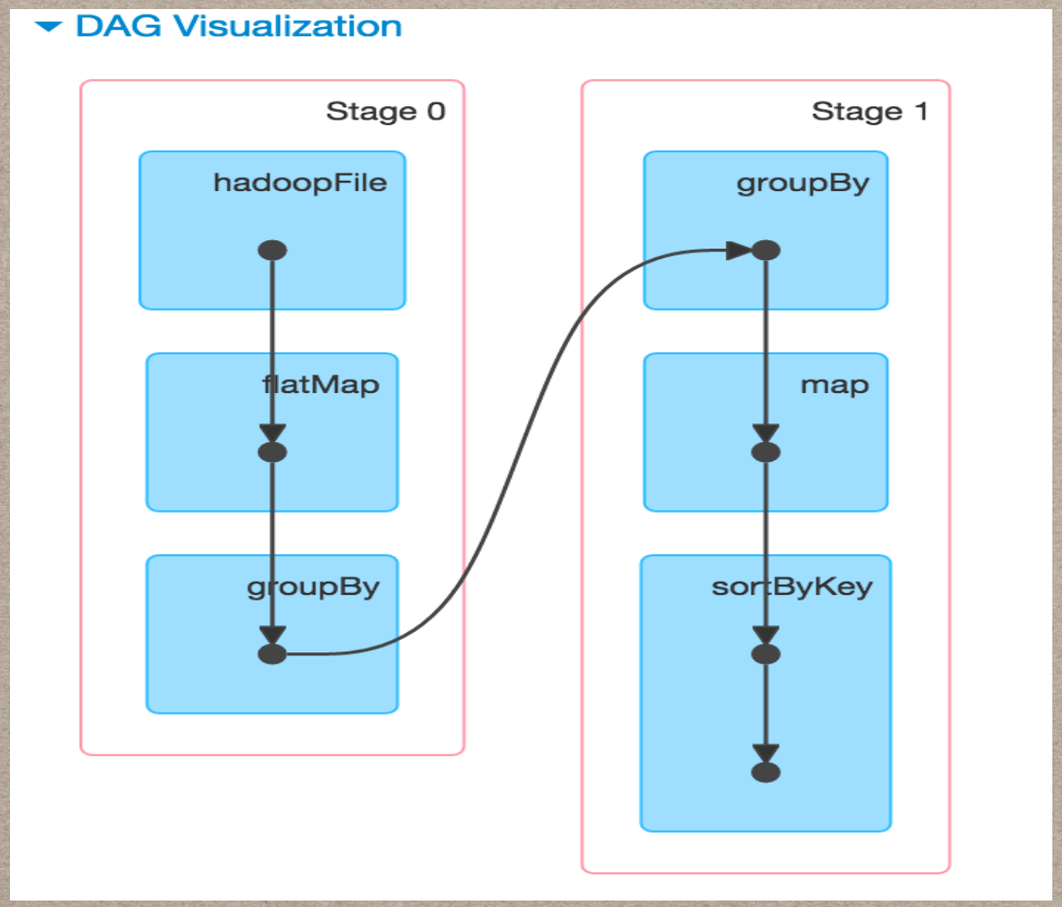
Scheduling Mode: FIFO
Completed Jobs: 2

▸ Event Timeline

Completed Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	saveAsHadoopFile at SeismicData.scala:776	2015/12/02 15:05:34	5 s	2/2 (1 skipped)	1082/1082 (600 skipped)
0	sortByKey at SeismicVolume.scala:264	2015/12/02 15:05:26	8 s	2/2	1200/1200

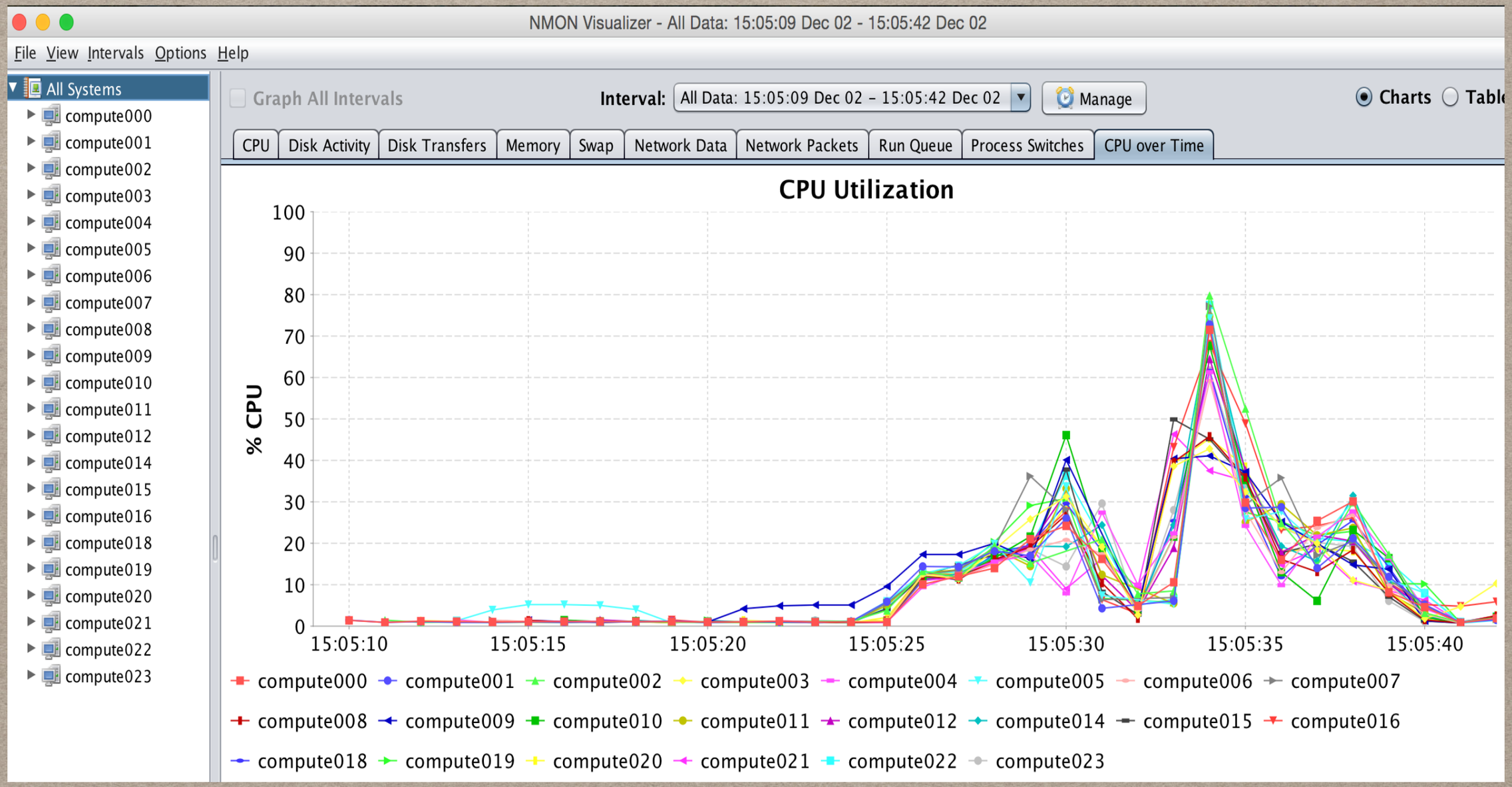
TIMELINE



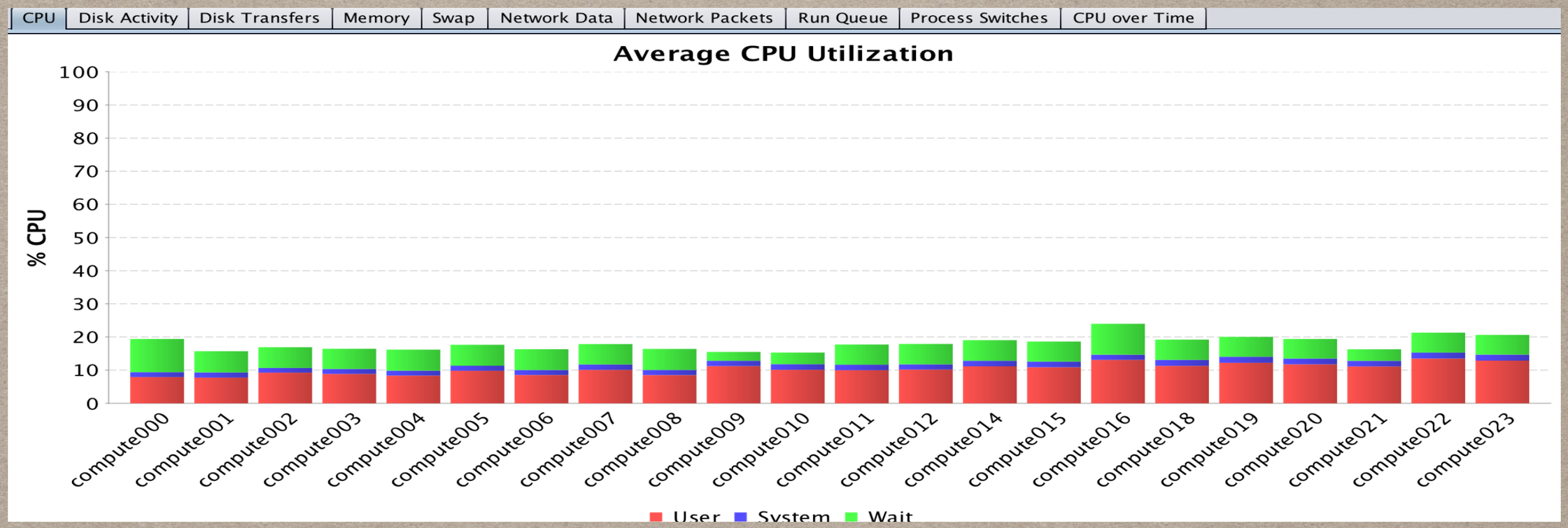
Completed Stages (2)

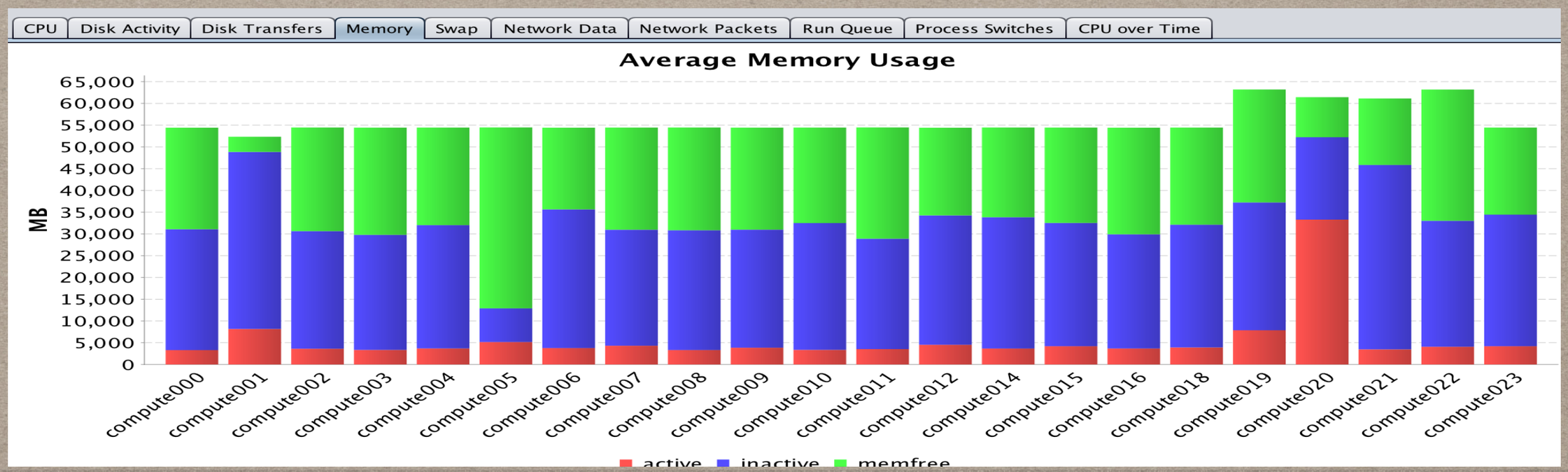
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total
1	sortByKey at SeismicVolume.scala:264	+details	2015/12/02 15:05:32	2 s	600/600
0	groupBy at SeismicVolume.scala:264	+details	2015/12/02 15:05:26	6 s	600/600

WORKERS(NODES) ACTIVITIES



WORKERS(NODES) ACTIVITIES





CONCLUSION

- Advantages
  - IO: Bigdata distributed to small local splits via Hadoop filesystem
  - RDD: All data & operations performed in RAM
  - Fault tolerance: Faults detection, rescheduling
- Bottleneck (When dependency happens)
  - Shuffle: Reforming partitions, shuffle distributed data via network and filesystem. (Group, Sort)
  - Communication: Collect data from distributions via network. (Count, Sort (without repartition))

Thanks.