COMP 430
Intro. to Database Systems
Apache Spark

Slides use ideas from Matei Zaharia, et al.
What is Spark?

Data-flow engine to support data analysis in clusters
What is Spark?

Data-flow engine to support data analysis in clusters

Computation model that views data moving from computation unit to computation unit.

E.g., MapReduce
What is Spark?

Data-flow engine to support data analysis in clusters

Emphasis on keeping data in memory
- Esp. for data reused in iterative algorithms, e.g., PageRank
- Unlike MapReduce
What is Spark?

Data-flow engine to support data analysis in clusters

Numerous libraries
- Machine learning
- Graph processing
- Time-series
- SQL
- ...

Many parallel primitives
- Map, filter, reduce, groupby, join, ...

Not a database!

Underlying data is considered mostly static.
What is Spark?

Data-flow engine to support data analysis in clusters

Generally built on top of Hadoop File System (HDFS)

• Write-once read-many
• Large file – distributed
• Fault-tolerant

Primarily for large-scale computing.
Hides implementation details.
Resilient Distributed Datasets (RDDs)

Key data abstraction

• Immutable collection spread across cluster
• *Transformations* build RDDs from other RDDs – map, filter, ...
  • Lazily built in parallel
  • Automatically rebuilt on failure
• *Actions* do things with RDDs – aggregate, save, ...
• Controllable persistence – e.g., caching in RAM
RDDs – Mining console logs

```python
sc = new SparkContext()

messages = sc.textFile("hdfs://.../log.txt")
    .filter(lambda entry: entry.startsWith("Error"))
    .map(lambda entry: entry.split('t')[2])
messages.cache()

foo_count = messages.filter(lambda s: "foo" in s).count()
bar_count = messages.filter(lambda s: "bar" in s).count()
```

Each has type RDD[String].
RDD fault-tolerance

```
messages = sc.textFile("hdfs://.../log.txt")
    .filter(lambda entry: entry.startswith("Error"))
    .map(lambda entry: entry.split('t')[2])
```

**Lineage:** Each RDD knows transformation used to (re)compute it. By default, only store the lineage, not the data.
RDDs & parallelism

• RDDs side-effect-free
  • Scheduling simplified, including speculative execution
• RDDs immutable
  • Consistency trivial

• Computation location based upon data location to minimize communication.
Examples
MapReduce

result = data.flatMap(map_fn)
    .groupByKey()
    .map(lambda (k,vs): reduce_fn(k,vs))

result = data.flatMap(map_fn)
    .reduceByKey(combiner_fn)
    .map(lambda (k,vs): reduce_fn(k,vs))
Word count

counts = sc.textfile("hdfs://...")
    .flatMap(lambda line: line.split('s'))
    .map(lambda word: (word, 1))
    .reduceByKey(operator.add)
counts.save("hdfs://...")
Logistic regression

Goal: find best line separating two sets of points
Logistic regression

```python
sc = new SparkContext()

points = sc.textFile(...).map(readPoint).cache()
plane = numpy.random.ranf(size = NUM_DIMENSIONS)

for _ in range(NUM_ITERATIONS):
    gradient = points.map(lambda p: (1 / (1 + exp(-p.y * plane.dot(p.x))) - 1) * p.y * p.x)
    .reduce(operator.add)

    plane -= gradient
```

- p.x : D-1 dimensions
- p.y : 1 dimension
Logistic regression performance

- Hadoop: 110 s / iteration
- Spark: first iteration 80 s, further iterations 1 s
PageRank

Algorithm:
1. Start each page at rank = 1.
2. For each iteration,
   a. Page \( p \) contributes \( \frac{\text{rank}_p}{|\text{neighbors}_p|} \) to each neighbor.
   b. Set each page’s rank to \( 0.15 + 0.85 \times \text{contribution} \).
PageRank

links = ...  # RDD of (url, neighbors) pairs
ranks = ...  # RDD of (url, rank) pairs

def compute_contribs(pair):
    (url, (links, rank)) = pair
    return [(dest, rank / len(links)) for dest in links]

for _ in range(NUM_ITERATIONS):
    contributions = links.join(ranks)
    .flatMap(compute_contribs)
    ranks = contributions.reduceByKey(operator.add)
    .mapValues(lambda total_contribution: 0.15 + 0.85 * total_contribution)

ranks.saveAsTextFile(...)