Performance Improvements in Apache Spark 2.0

Whole stage code generation and vectorization

Markus Dale, Databricks

June 2016

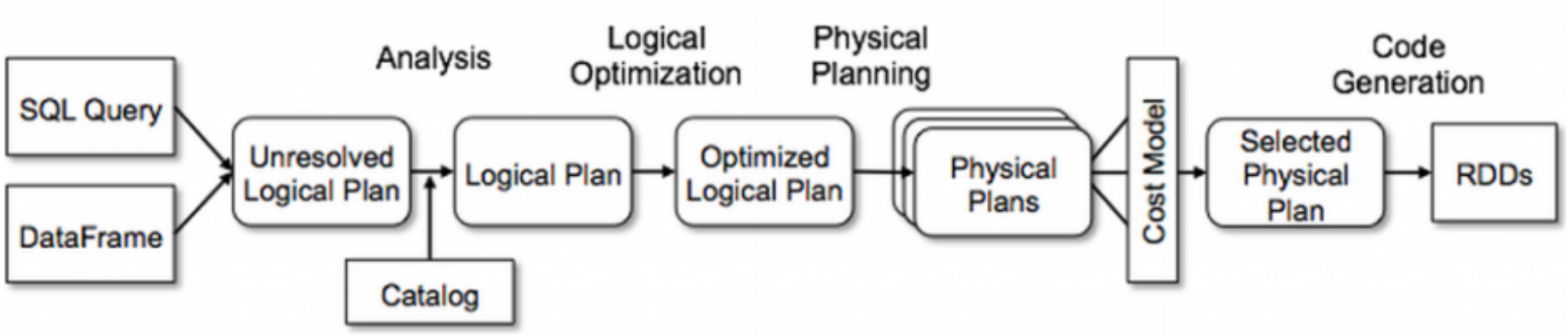
What's in Apache Spark 2.0.0?

- Over 2,000 JIRA tickets
- Some ticket titles:
  - (Dataset-oriented) API evolution in Spark 2.0
  - Create a full-fledged built-in SQL parser
  - Add support for off-heap caching
  - Model export/import for Pipeline API
  - Whole stage codegen

Project Tungsten - Closer to bare metal

- Apache Spark 1.3 introduced DataFrames/Catalyst Optimizer
- Tungsten 1.0:
  - Memory management and binary processing
  - Code generation for expression evaluation
  - See Project Tungsten: Bringing Apache Spark Closer to Bare Metal, Xin and Rosen (2015)

Catalyst Optimizer



See Databricks Catalyst Optimizer blog entry, Armbrust et al. (2015)

Project Tungsten 2.0 - reduce CPU bottlenecks

- virtual function calls
- reading or writing intermediate data to CPU cache or memory

Note: all following graphs from Agarwal, Liu, and Xin (2016)

Simple aggregate query with filter

```
select count(*) from store_sales
where ss_item_sk = 1000
```

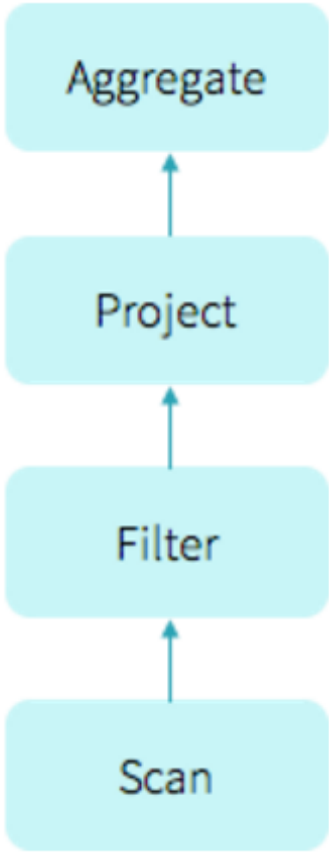


Figure : Filtered count query

Pre-2.0 Apache Spark: Volcano Iterator Model

```
class Filter(child: Operator, predicate: (Row => Boolean))
  extends Operator {
  def next(): Row = {
    var current = child.next()
    while (current == null || predicate(current)) {
      current = child.next()
    }
    return current
  }
}
```

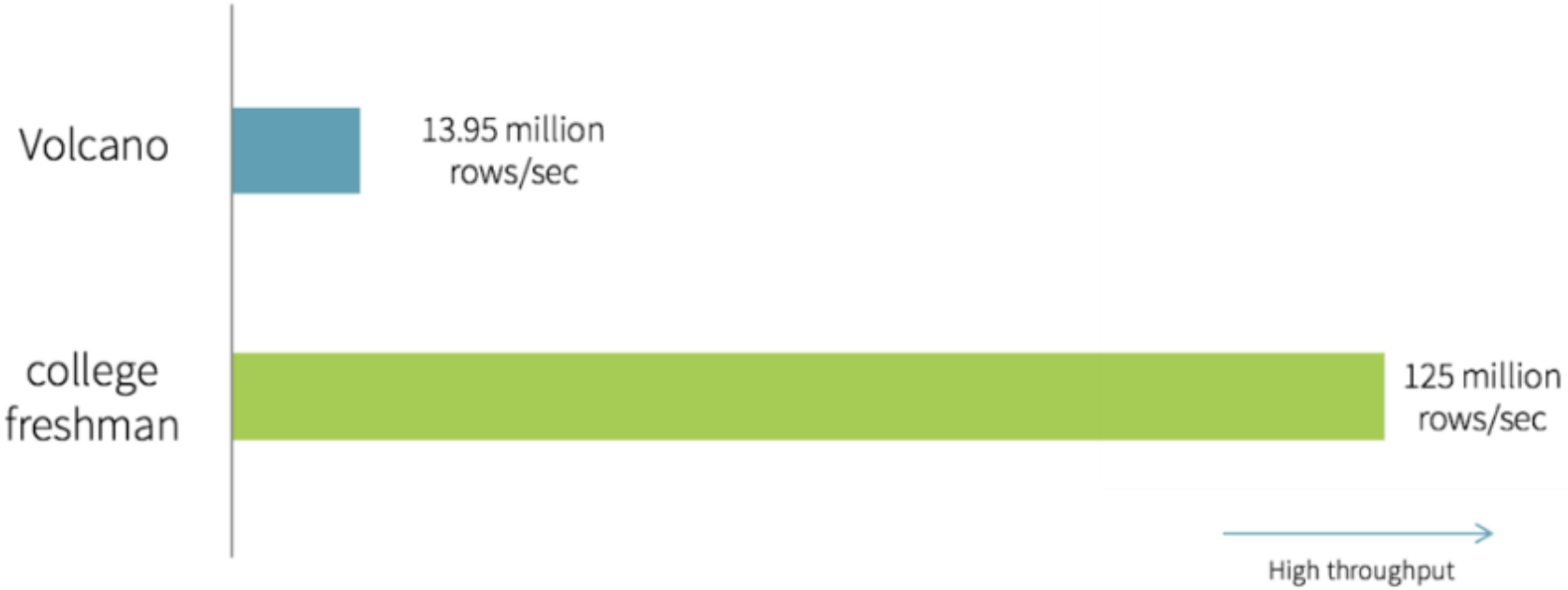
See Graefe (1994)

Handwritten Code

```
var count = 0
for (ss_item_sk in store_sales) {
    if (ss_item_sk == 1000) {
        count += 1
    }
}
```

Handwritten vs. Volcano

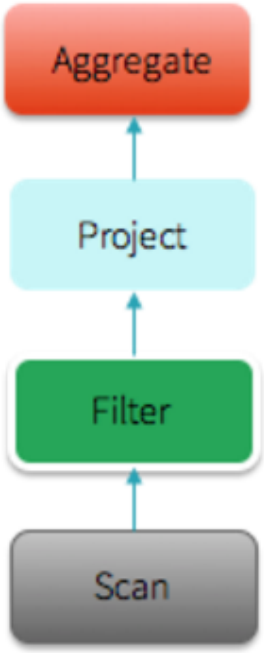
- Single threaded data from Parquet on disk



Whole Stage Code Generation

- No virtual function dispatches
- Intermediate data in CPU registers
- Loop unrolling and SIMD

Whole Stage Code Generation



```
long count = 0;
for (ss_item_sk in store_sales) {
  if (ss_item_sk == 1000) {
    count += 1;
  }
}
```

explain()

- Mark with *

Vectorization

Demo

org.apache.spark.sql.execution.vectorized.ColumnarBatch.Row

spark.sql("select count(a) from df").explain()

SparkSession SparkSession.builder.getOrCreate()

References I

Agarwal, Sameer, Davies Liu, and Reynold S. Xin. 2016. "Apache Spark as a Compiler." https://databricks.com/blog/2016/05/23/apache-spark-as-a-compiler-joining-a-billion-rows-per-secc html.

Armbrust, Michael, Yin Huai, Cheng Liang, Reynold S. Xin, and Matei Zaharia. 2015. "Deep Dive into Spark SQL's Catalyst Optimizer." https://databricks.com/blog/2015/04/13/deep-dive-into-spark-sqls-catalyst-optimizer.html.

Graefe, G. 1994. "Volcano-an Extensible and Parallel Query Evaluation System." *IEEE Transactions on Knowledge and Data Engineering* 6 (1) (February): 120–135. doi:10.1109/69.273032.

Xin, Reynold S., and Josh Rosen. 2015. "Project Tungsten - Bringing Apache Spark Closer to Bare Metal." https://databricks.com/blog/2015/04/28/

References II

project-tungsten-bringing-spark-closer-to-bare-metal.html.