

# Smarter Agent TechTalks



Apache Spark Pipelines

# The Problem: How do we handle big data?

- 2003 Google File System Paper
  - Outlined the theory and algorithms behind the building blocks of Google Bigtable (2006) and map-reduce computation (2004)
  - Basis for several current Google technologies including cloud, maps, earth, gmail, youtube...

## The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung  
Google\*

### ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

### Categories and Subject Descriptors

D [4]: 3—Distributed file systems

### 1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies. Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system.

Second, files are huge by traditional standards. Multi-GB files are common. Each file typically contains many application objects such as web documents. When we are regularly working with fast-growing data sets of many TBs comprising

# The Problem: How do we handle big data?

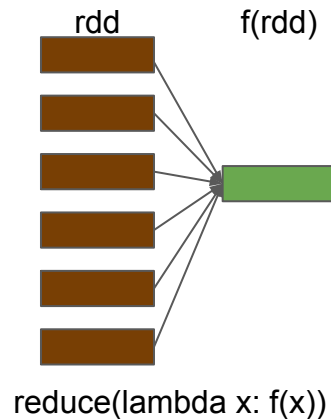
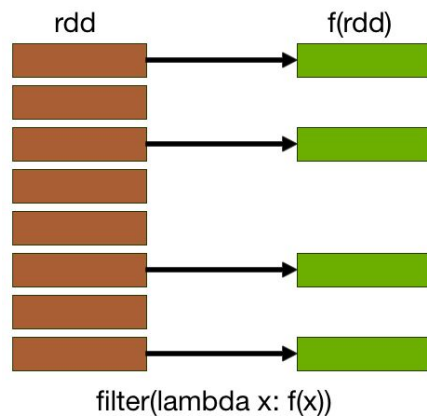
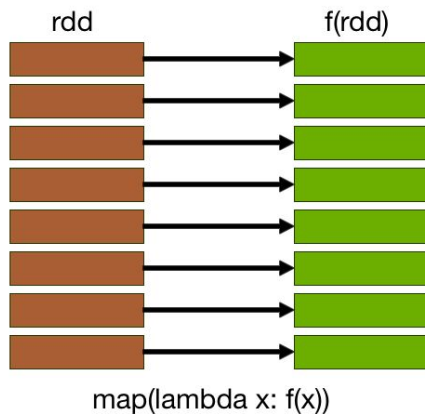
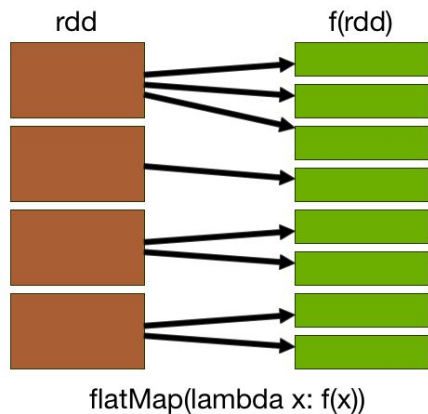
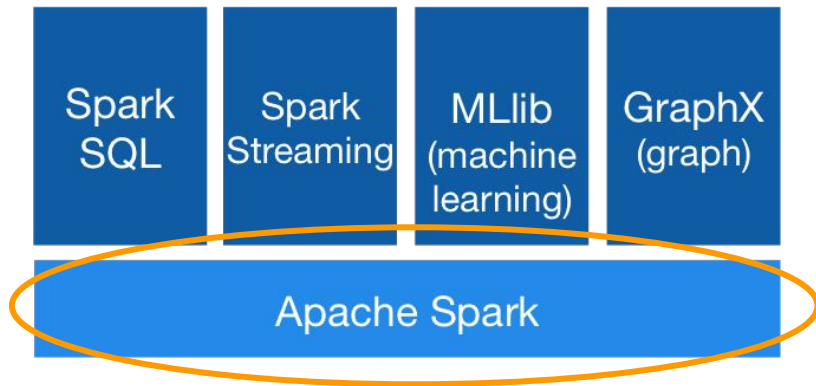
- 2006 Open Source Hadoop
  - Hadoop Distributed File System (HDFS) datastore
  - Map-reduce functionality
  - No in-memory computation, lots of disc i/o
- 2010 Open Source Spark
  - Builds on Hadoop functionality
  - Map-reduce functionality in memory when possible

2014 big data sort

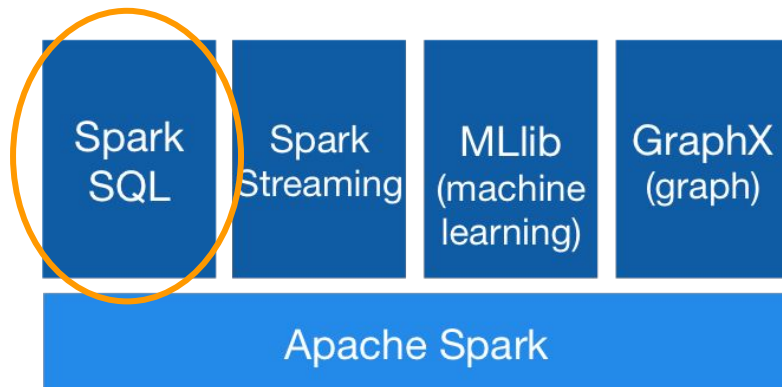
	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min

# Core Spark Basics

- Basic data type - resilient distributed dataset (rdd)
- Array like distributed data structure



# Spark SQL Basics



## Ways to Create DataFrame in Spark

Hive Data  
Csv Data  
Json Data  
RDBMS Data  
XML Data  
Parquet Data  
Cassandra Data  
RDDs

**Spark SQL**

### DataFrame

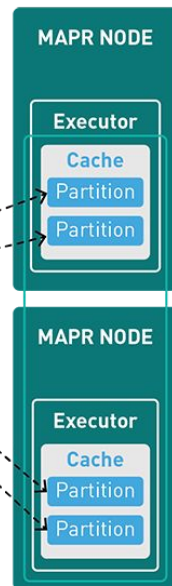
	Col1	Col2	Col3	.....
Row 1				
Row 2				
Row 3				
...				

ROW		COLUMN	
function	id	symbol	price
BUY	1	GOOG	28.0
BUY	2	GOOG	28.0
BUY	3	GOOG	28.0
BUY	4	GOOG	28.0
BUY	5	GOOG	28.0
BUY	6	GOOG	28.0
BUY	7	GOOG	28.0
BUY	8	GOOG	28.0
BUY	9	GOOG	28.0
BUY	10	GOOG	28.0
BUY	11	GOOG	28.0
BUY	12	GOOG	28.0
BUY	13	GOOG	28.0
BUY	14	GOOG	28.0
BUY	15	GOOG	28.0
BUY	16	GOOG	28.0
BUY	17	GOOG	28.0
BUY	18	GOOG	28.0
BUY	19	GOOG	28.0
BUY	20	GOOG	28.0
BUY	21	GOOG	28.0
BUY	22	GOOG	28.0
BUY	23	GOOG	28.0
BUY	24	GOOG	28.0
BUY	25	GOOG	28.0
BUY	26	GOOG	28.0
BUY	27	GOOG	28.0
BUY	28	GOOG	28.0
BUY	29	GOOG	28.0
BUY	30	GOOG	28.0
BUY	31	GOOG	28.0
BUY	32	GOOG	28.0
BUY	33	GOOG	28.0
BUY	34	GOOG	28.0
BUY	35	GOOG	28.0
BUY	36	GOOG	28.0
BUY	37	GOOG	28.0
BUY	38	GOOG	28.0
BUY	39	GOOG	28.0
BUY	40	GOOG	28.0
BUY	41	GOOG	28.0
BUY	42	GOOG	28.0
BUY	43	GOOG	28.0
BUY	44	GOOG	28.0
BUY	45	GOOG	28.0
BUY	46	GOOG	28.0
BUY	47	GOOG	28.0
BUY	48	GOOG	28.0
BUY	49	GOOG	28.0
BUY	50	GOOG	28.0
BUY	51	GOOG	28.0
BUY	52	GOOG	28.0
BUY	53	GOOG	28.0
BUY	54	GOOG	28.0
BUY	55	GOOG	28.0
BUY	56	GOOG	28.0
BUY	57	GOOG	28.0
BUY	58	GOOG	28.0
BUY	59	GOOG	28.0
BUY	60	GOOG	28.0
BUY	61	GOOG	28.0
BUY	62	GOOG	28.0
BUY	63	GOOG	28.0
BUY	64	GOOG	28.0
BUY	65	GOOG	28.0
BUY	66	GOOG	28.0
BUY	67	GOOG	28.0
BUY	68	GOOG	28.0
BUY	69	GOOG	28.0
BUY	70	GOOG	28.0
BUY	71	GOOG	28.0
BUY	72	GOOG	28.0
BUY	73	GOOG	28.0
BUY	74	GOOG	28.0
BUY	75	GOOG	28.0
BUY	76	GOOG	28.0
BUY	77	GOOG	28.0
BUY	78	GOOG	28.0
BUY	79	GOOG	28.0
BUY	80	GOOG	28.0
BUY	81	GOOG	28.0
BUY	82	GOOG	28.0
BUY	83	GOOG	28.0
BUY	84	GOOG	28.0
BUY	85	GOOG	28.0
BUY	86	GOOG	28.0
BUY	87	GOOG	28.0
BUY	88	GOOG	28.0
BUY	89	GOOG	28.0
BUY	90	GOOG	28.0
BUY	91	GOOG	28.0
BUY	92	GOOG	28.0
BUY	93	GOOG	28.0
BUY	94	GOOG	28.0
BUY	95	GOOG	28.0
BUY	96	GOOG	28.0
BUY	97	GOOG	28.0
BUY	98	GOOG	28.0
BUY	99	GOOG	28.0
BUY	100	GOOG	28.0

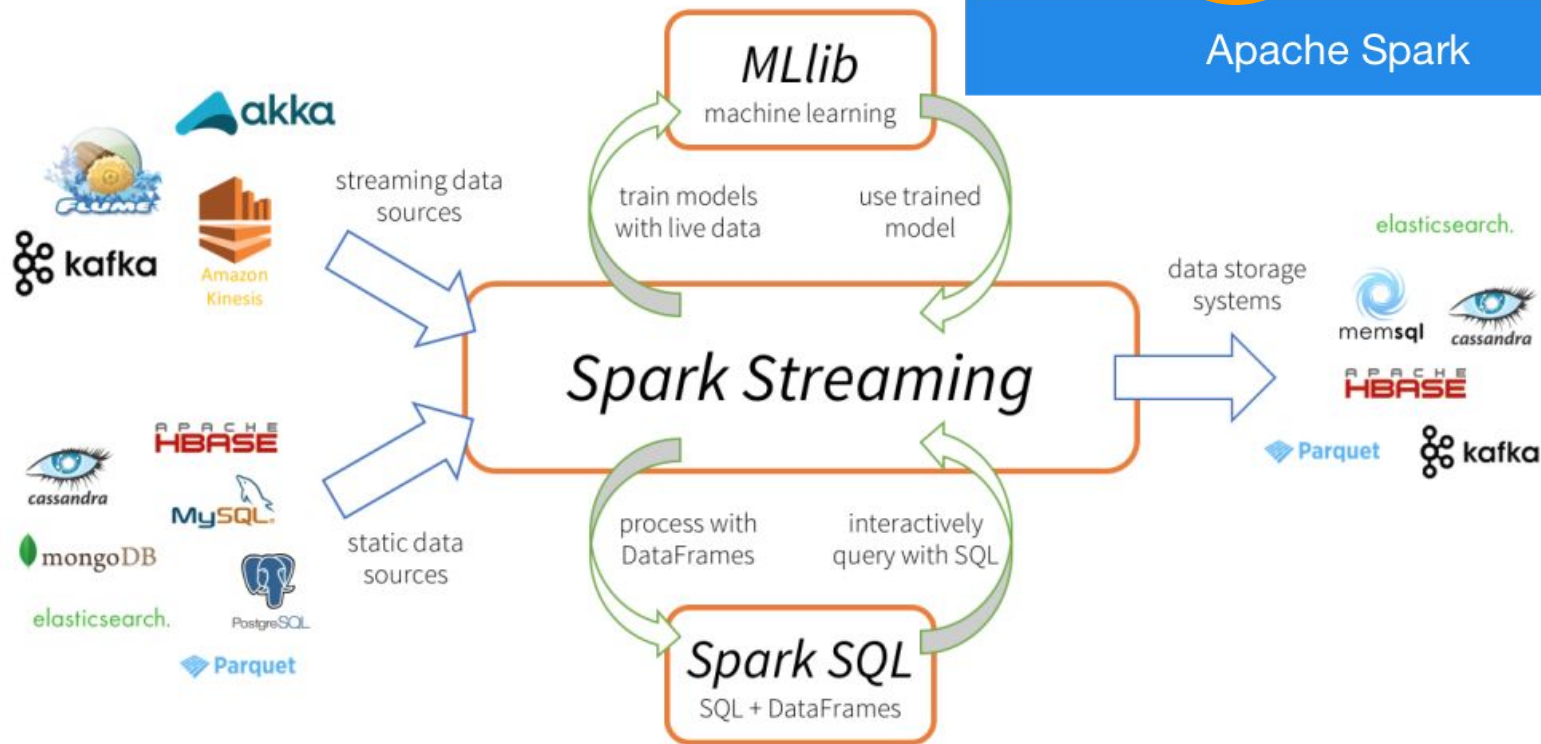
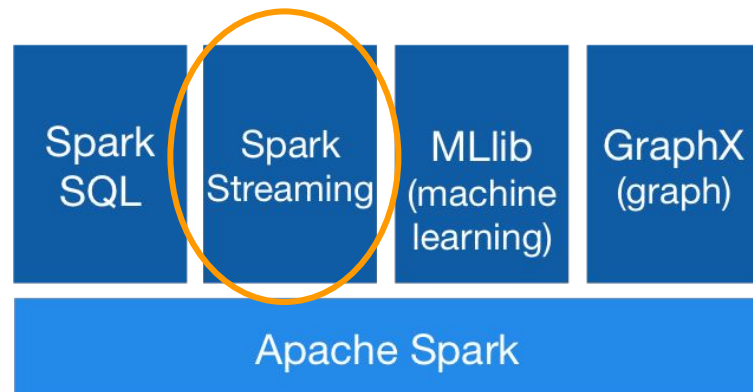
DataFrame is like a partitioned table.

Dataset[Row]

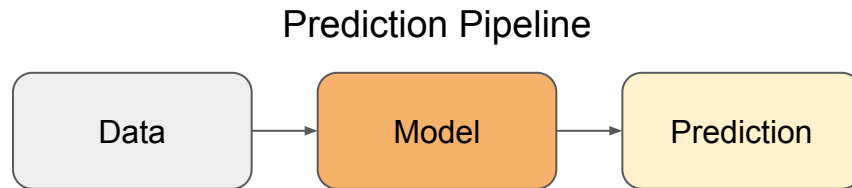
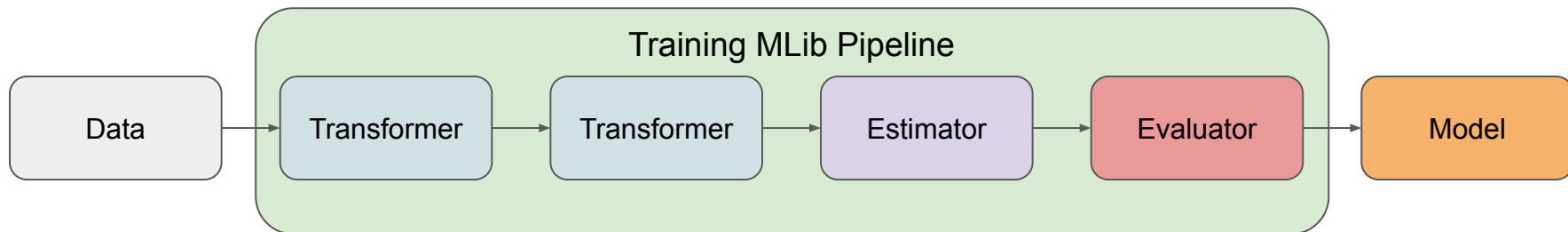
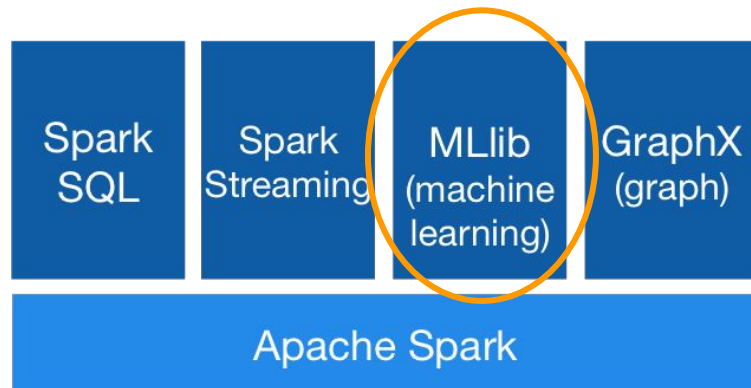
id	symbol	price
1	GOOG	28.0
2	GOOG	28.0
3	GOOG	28.0
4	GOOG	28.0
5	GOOG	28.0
6	GOOG	28.0
7	GOOG	28.0
8	GOOG	28.0
9	GOOG	28.0
10	GOOG	28.0
11	GOOG	28.0
12	GOOG	28.0
13	GOOG	28.0
14	GOOG	28.0
15	GOOG	28.0
16	GOOG	28.0
17	GOOG	28.0
18	GOOG	28.0
19	GOOG	28.0
20	GOOG	28.0
21	GOOG	28.0
22	GOOG	28.0
23	GOOG	28.0
24	GOOG	28.0
25	GOOG	28.0
26	GOOG	28.0
27	GOOG	28.0
28	GOOG	28.0
29	GOOG	28.0
30	GOOG	28.0
31	GOOG	28.0
32	GOOG	28.0
33	GOOG	28.0
34	GOOG	28.0
35	GOOG	28.0
36	GOOG	28.0
37	GOOG	28.0
38	GOOG	28.0
39	GOOG	28.0
40	GOOG	28.0
41	GOOG	28.0
42	GOOG	28.0
43	GOOG	28.0
44	GOOG	28.0
45	GOOG	28.0
46	GOOG	28.0
47	GOOG	28.0
48	GOOG	28.0
49	GOOG	28.0
50	GOOG	28.0
51	GOOG	28.0
52	GOOG	28.0
53	GOOG	28.0
54	GOOG	28.0
55	GOOG	28.0
56	GOOG	28.0
57	GOOG	28.0
58	GOOG	28.0
59	GOOG	28.0
60	GOOG	28.0
61	GOOG	28.0
62	GOOG	28.0
63	GOOG	28.0
64	GOOG	28.0
65	GOOG	28.0
66	GOOG	28.0
67	GOOG	28.0
68	GOOG	28.0
69	GOOG	28.0
70	GOOG	28.0
71	GOOG	28.0
72	GOOG	28.0
73	GOOG	28.0
74	GOOG	28.0
75	GOOG	28.0
76	GOOG	28.0
77	GOOG	28.0
78	GOOG	28.0
79	GOOG	28.0
80	GOOG	28.0
81	GOOG	28.0
82	GOOG	28.0
83	GOOG	28.0
84	GOOG	28.0
85	GOOG	28.0
86	GOOG	28.0
87	GOOG	28.0
88	GOOG	28.0
89	GOOG	28.0
90	GOOG	28.0
91	GOOG	28.0
92	GOOG	28.0
93	GOOG	28.0
94	GOOG	28.0
95	GOOG	28.0
96	GOOG	28.0
97	GOOG	28.0
98	GOOG	28.0
99	GOOG	28.0
100	GOOG	28.0



# Spark Streaming Basics



# Spark MLib Basics



# Spark GraphX Basics

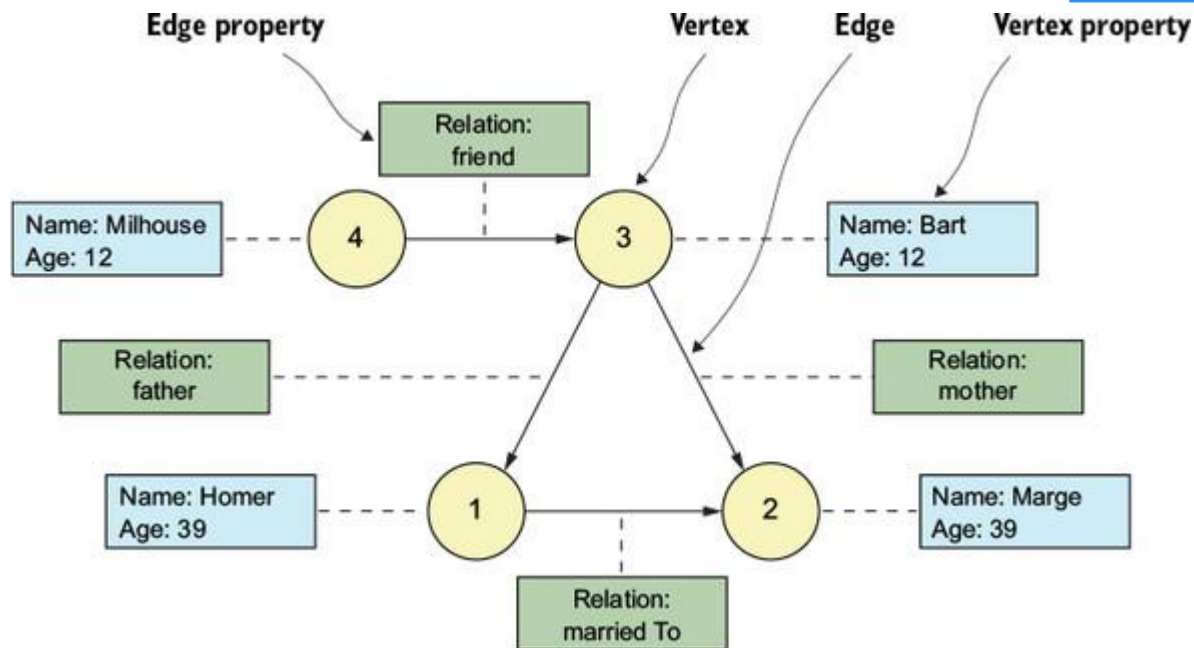
Spark  
SQL

Spark  
Streaming

MLlib  
(machine  
learning)

GraphX  
(graph)

Apache Spark



- Analyze relationships between elements of a network in parallel
- Pagerank algorithms for connections between vertices, edge analytics, etc