阿里云
奥运会全球指定云服务商

SparkSQL在ETL中的应用

嵩林

2017.05.19

About Me

淘宝无线事业部

无线客户端数据采集开发(SDK+服务端)
数据开发(离线/实时)

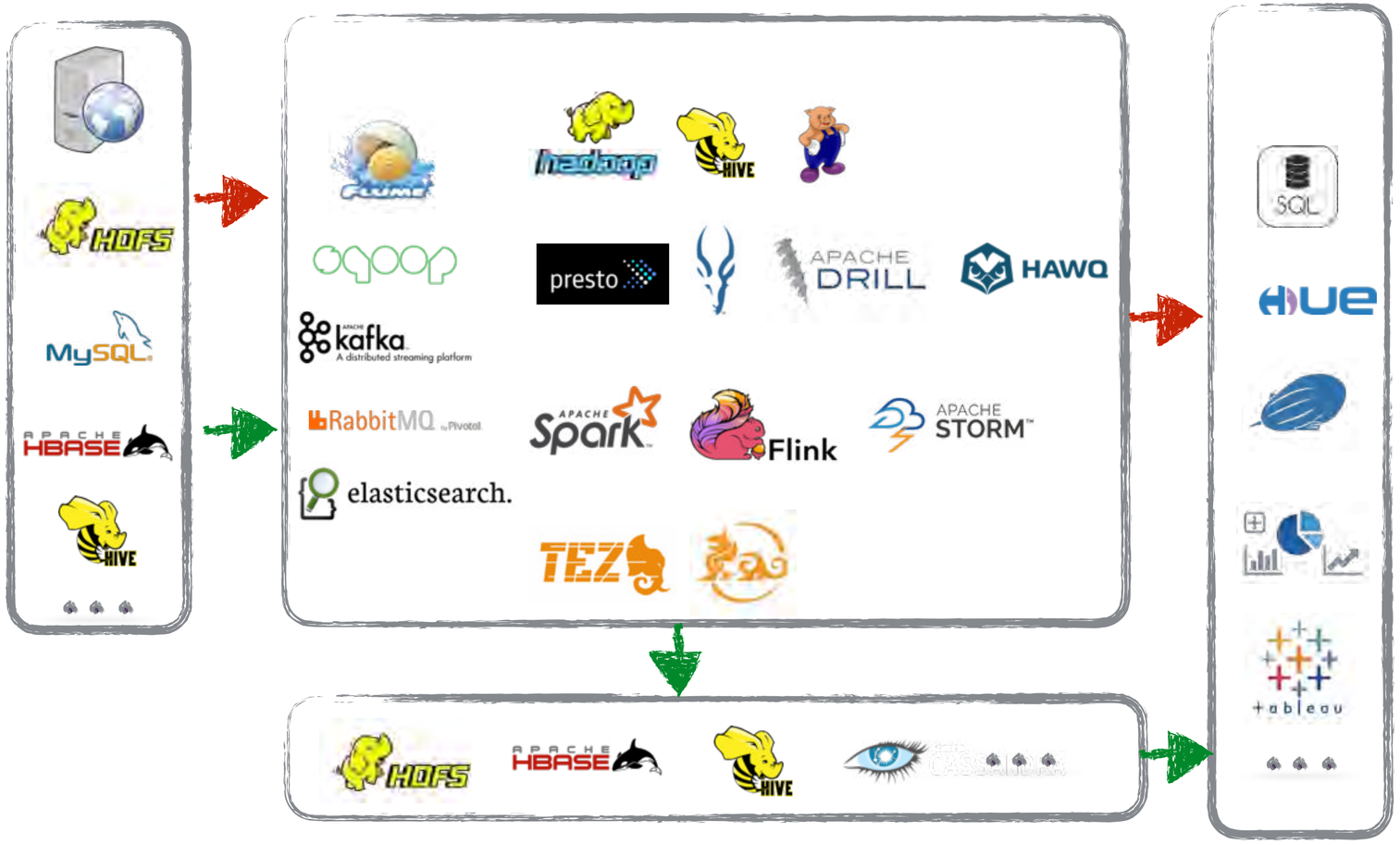
阿里云E-MapReduce团队

Spark/HBase相关开发
参与Spark社区贡献

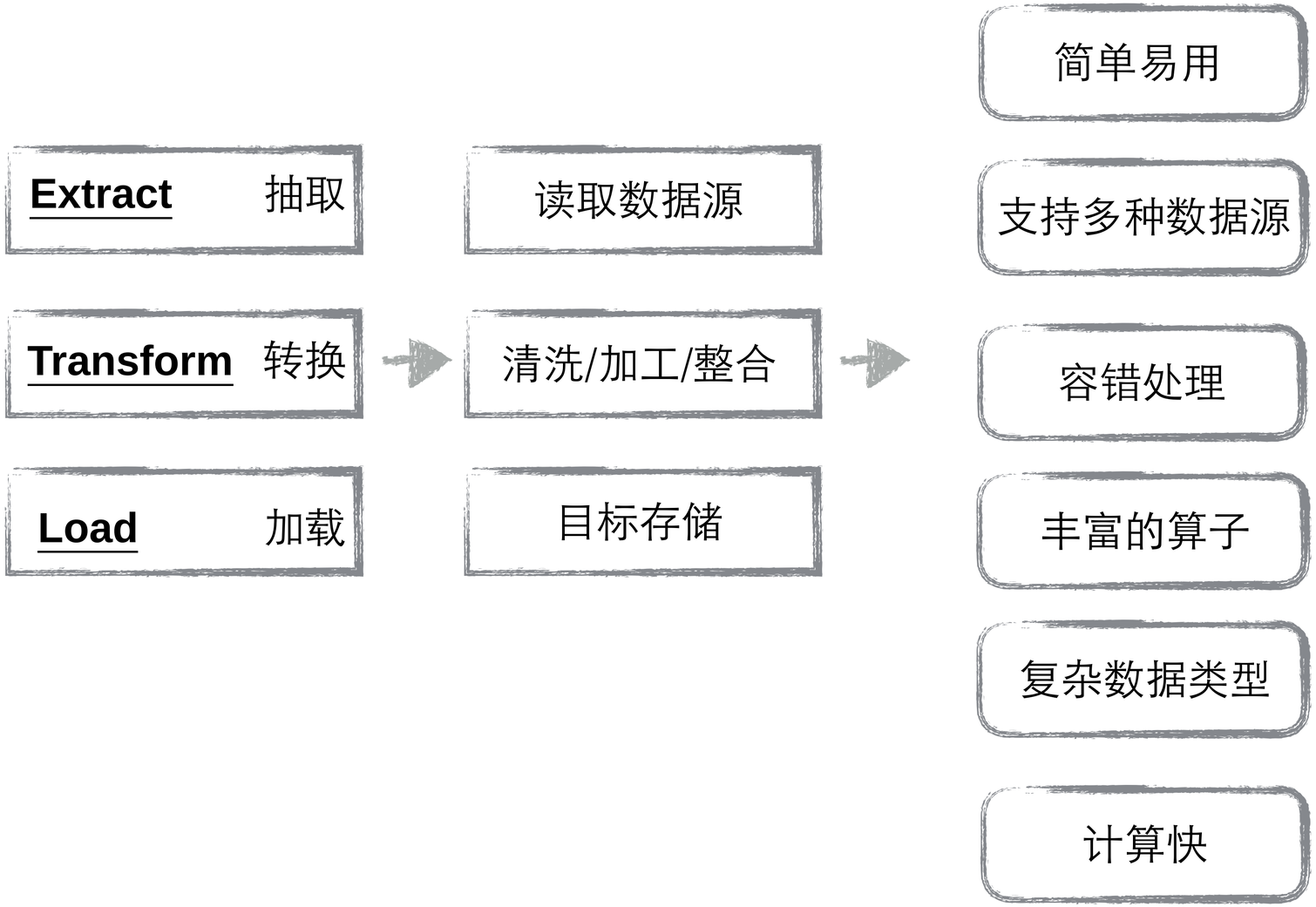
Agenda

- Data Pipeline
- ETL
- SparkSQL
  - DataSource
  - 丰富的算子
  - Hive兼容
  - 性能
  - 云上ETL

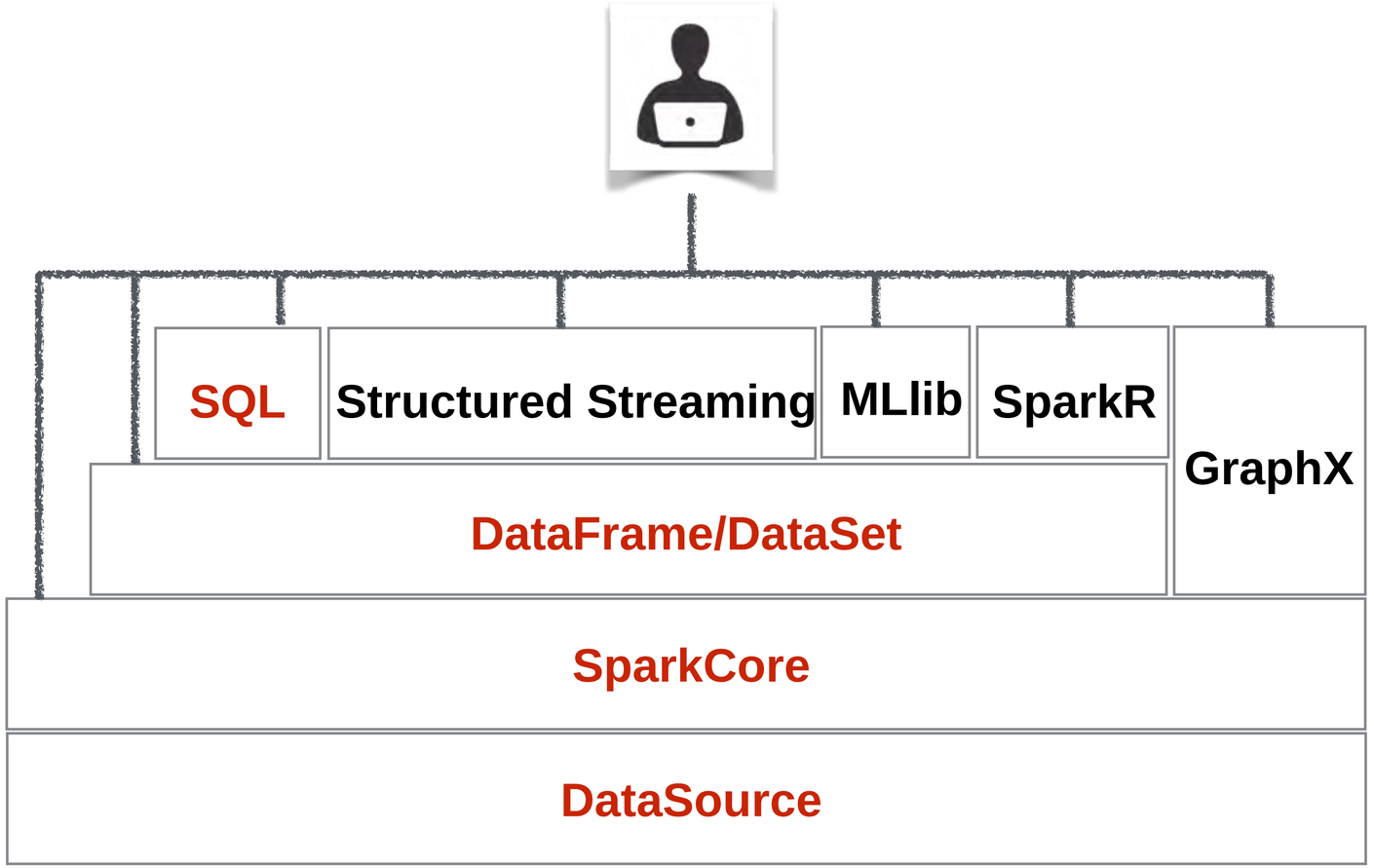
Data Pipeline



ETL

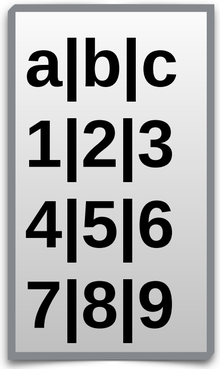


SparkSQL



SparkSQL

Simple ETL



a	b	c
1	2	3
4	5	6
7	8	9

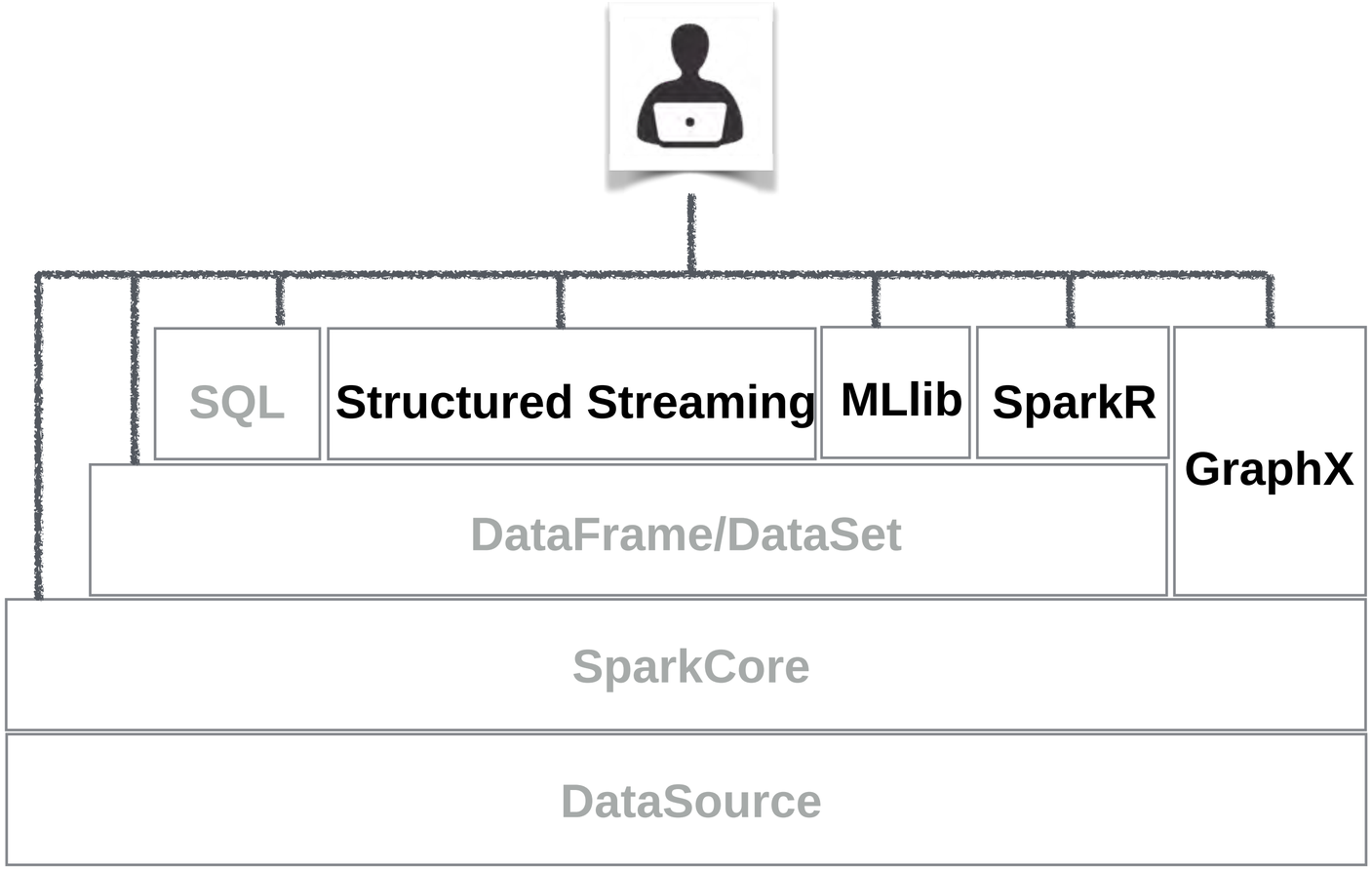
Extract

Transform

Load

```
spark.read
    .option("sep","|").option("header","true").csv("/csv_path")
    .filter(...)
    .map(...)
    .saveAsTable("t")
```

SparkSQL



- DataSource
- 丰富的算子
- Hive兼容
- 性能
- 云上ETL

SparkSQL

1 **DataSource**

2 丰富的算子

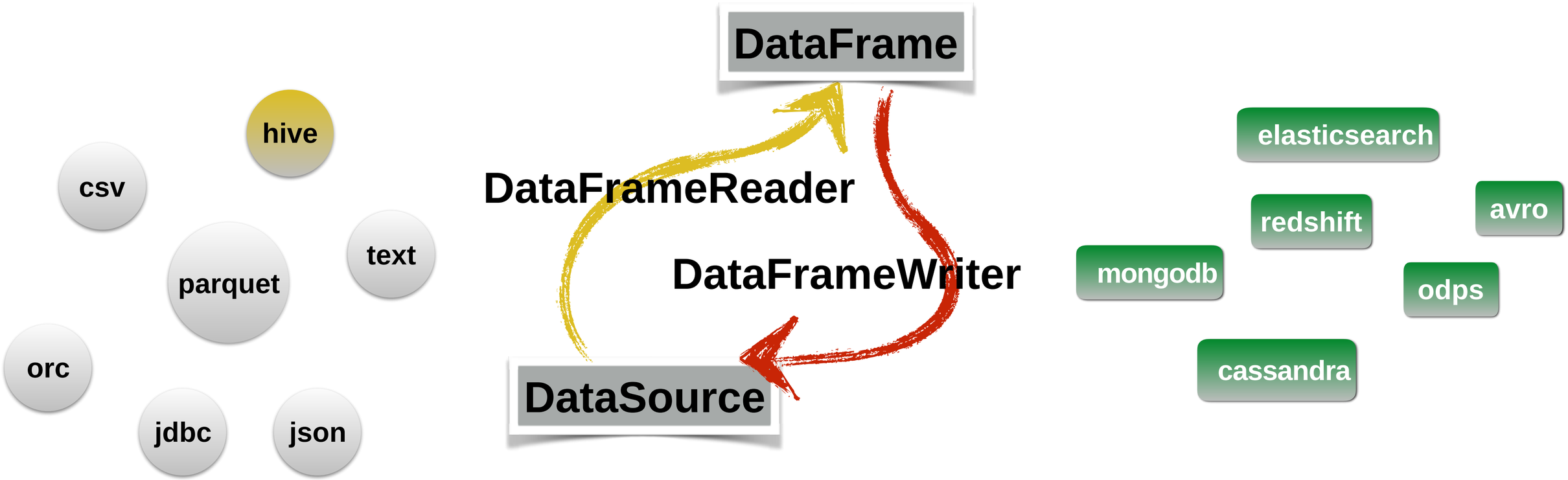
3 Hive兼容

4 性能

5 云上ETL

SparkSQL

DataSource



Build-in Support

Custom Implement

https://spark-packages.org/

SparkSQL

DataSource-schema

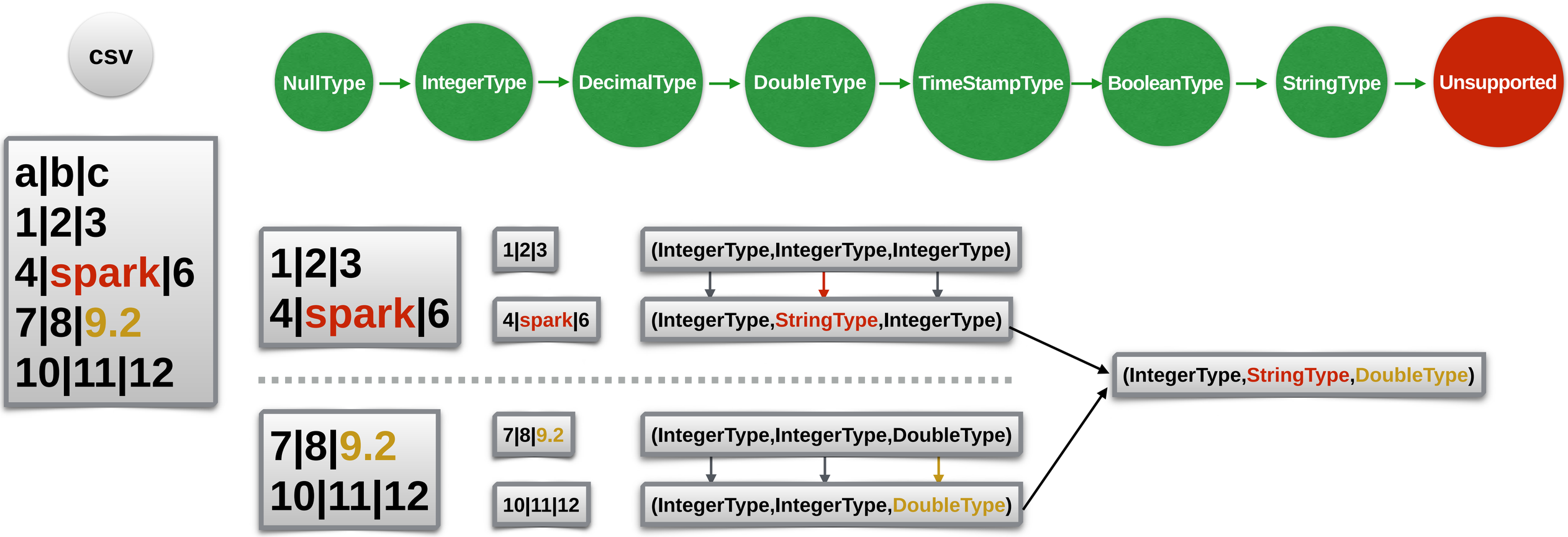
csv

```
a|b|c
1|2|3
4|5|6
7|8|9
```

spark.read .option("sep","|").option("header","true") .csv("/csv_path")	root |-- a: string (nullable = true) |-- b: string (nullable = true) |-- c: string (nullable = true)
spark.read .option("sep","|").option("header","true") .option("inferSchema","true").csv("/csv_path")	root |-- a: integer (nullable = true) |-- b: integer (nullable = true) |-- c: integer (nullable = true)
val st = StructType(StructField("e", IntegerType), StructField("f", IntegerType), StructField("g", IntegerType)) spark.read.schema(st).option("sep","|") .option("header","true").csv("/csv_path")	root |-- e: integer (nullable = true) |-- f: integer (nullable = true) |-- g: integer (nullable = true)

SparkSQL

DataSource-inferSchema



SparkSQL

DataSource-custom

trait

- RelationProvider
- 读 SchemaRelationProvider
- 写 CreateableRelationProvider
- DataSourceRegister

odps

- RelationProvider
- CreateableRelationProvider
- DataSourceRegister

```
class DefaultSource extends RelationProvider
  with CreatableRelationProvider
  with DataSourceRegister {
  private val log = LoggerFactory.getLogger(getClass)

  override def shortName(): String = "odps"

  // for read
  override def createRelation(
    sqlContext: SQLContext,
    parameters: Map[String, String]): BaseRelation = {
        ...
  }

  // for write
  override def createRelation(
    sqlContext: SQLContext,
    saveMode: SaveMode,
    parameters: Map[String, String],
    data: DataFrame): BaseRelation = {
        ...
  }
}
```

https://github.com/aliyun/aliyun-emapreduce-sdk.git

SparkSQL

DataSource-custom

odps

RelationProvider

CreateableRelationProvider

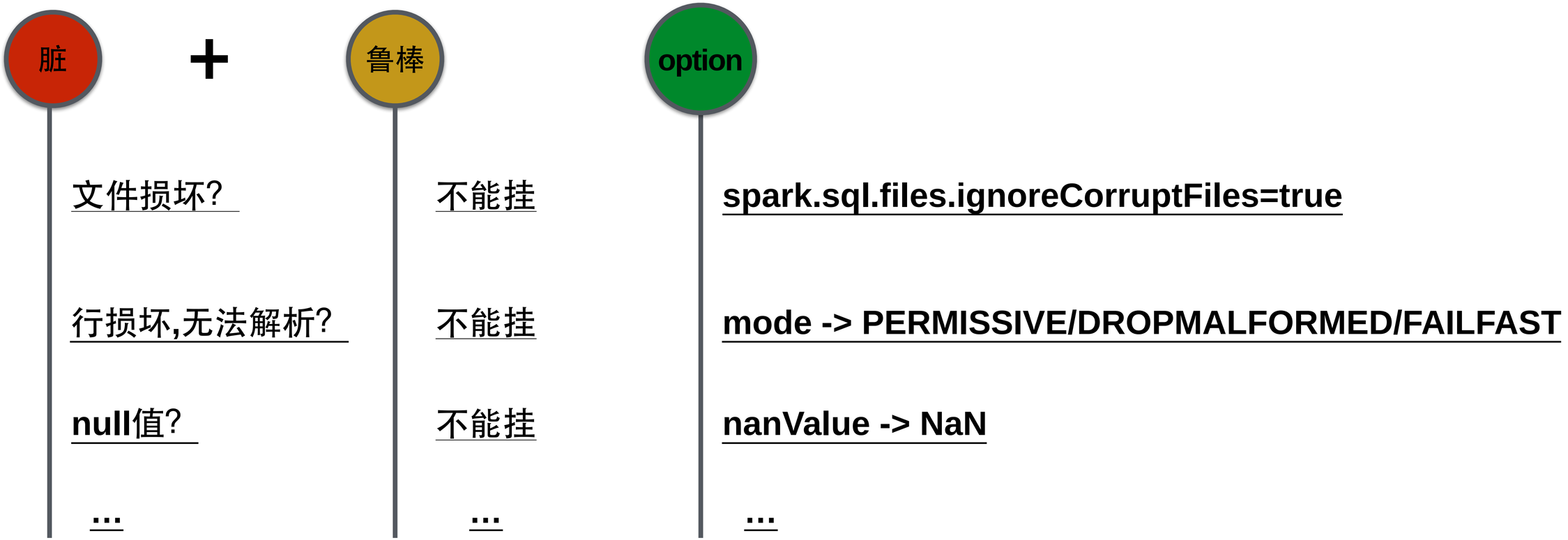
DataSourceRegister

```
val readDF = spark.read
    .format("org.apache.spark.aliyun.odps.datasource")
    .option("odpsUrl", odpsUrl)
    .option("tunnelUrl", tunnelUrl)
    .option("table", table)
    .option("project", project)
    .option("accessKeySecret", accessKeySecret)
    .option("accessKeyId", accessKeyId).load()
```

```
df.write.format("odps")
    .option("odpsUrl", odpsUrl)
    .option("tunnelUrl", tunnelUrl)
    .option("table", table)
    .option("project", project)
    .option("accessKeySecret", accessKeySecret)
    .option("accessKeyId", accessKeyId).mode("overwrite").save()
```

SparkSQL

DataSource-容错处理



SparkSQL

DataSource-容错处理

```
{"a":1,"b":2,"c":3}
{"a":4,:5,"c":6}
{"a":7,"b":8,"c":9}
```

spark.read.option("mode","PERMISSIVE") .json("/json_path")	+----------------------+------+------+------+ | _corrupt_record | a | b | c | +----------------------+------+------+------+ | null | 1 | 2 | 3 | |{"a":4,:5,"c":6} | null | null | null | | null | 7 | 8 | 9 | +----------------------+------+------+------+
spark.read.option("mode","DROPMALFORMED") .json("/json_path")	+---+---+---+ | a | b | c | +—+---+---+ | 1 | 2 | 3 | | 7 | 8 | 9 | +---+---+---+
spark.read.option("mode","FAILFAST") .json("/json_path")	Caused by: com.fasterxml.jackson.core.JsonParseException: Unexpected character (':' (code 58)): was expecting double-quote to start field name at [Source: UNKNOWN; line: 1, column: 9]

SparkSQL

1 DataSource

2 丰富的算子

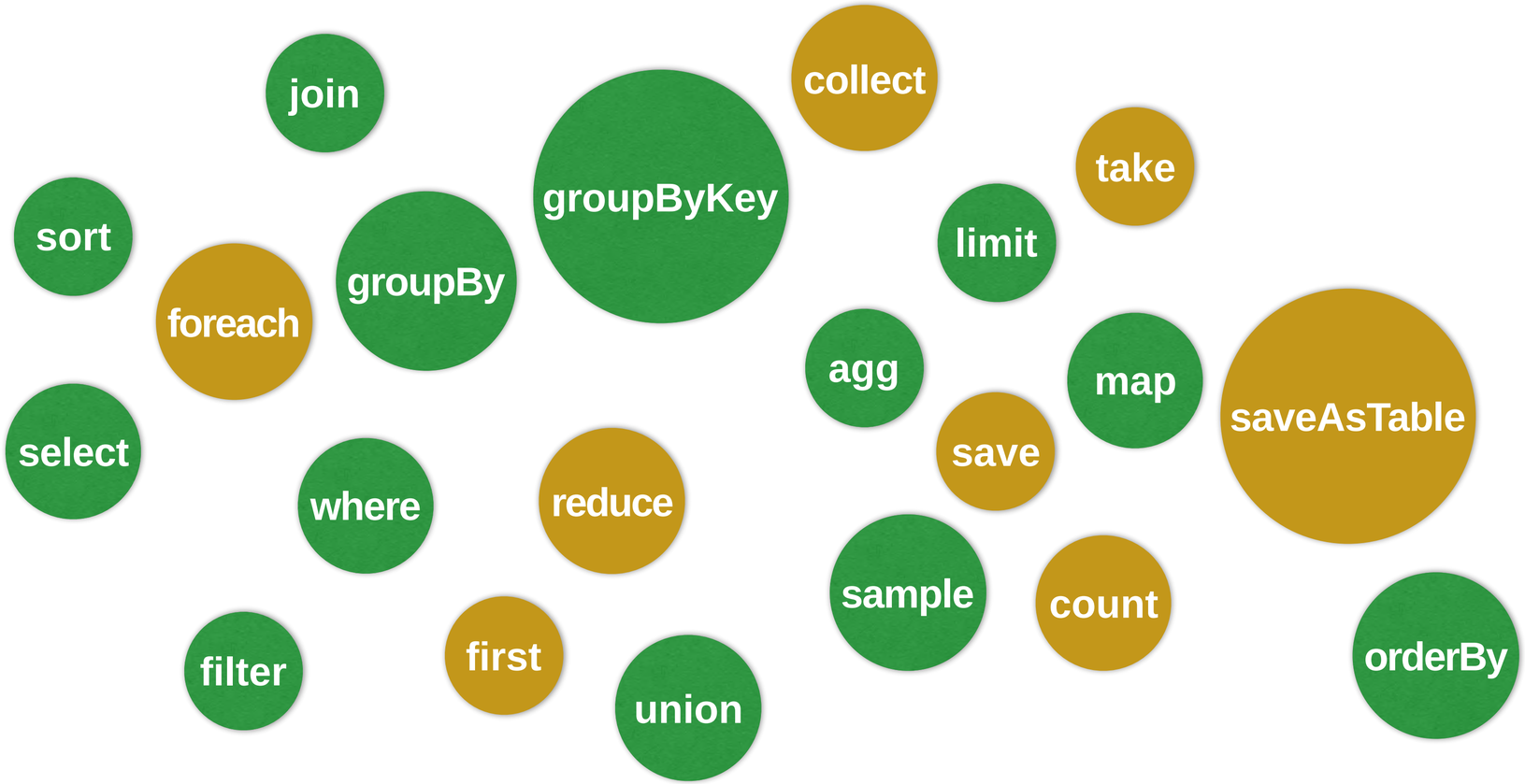
3 Hive兼容

4 性能

5 云上ETL

SparkSQL

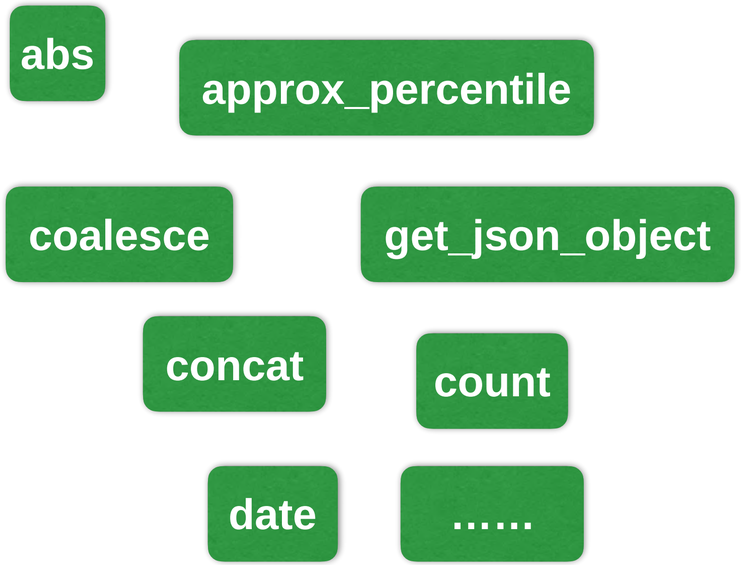
丰富的算子



SparkSQL

丰富的算子-UDF/UDAF/UDTF

```
spark.sql("show functions").show
```



UDF	spark.udf.register spark.sql("CREATE FUNCTION….")	val myudf = spark.udf.register("testudf",(n:Int) => {n + 1}) spark.sql("select testudf(a) from t").show or df.select(myudf("a")).show
UDTF	df.flatMap df.select(explode(a)) df.explode	import org.apache.spark.sql.functions.explode val df1 = df.withColumn("b", explode($"b")) or val df2 = df.as[(String ,Seq[Int])].flatMap { case (a:String ,b:Seq[Int]) => b.map(value => (a, value)) }.toDF("a" ,"b")

UDAF	abstract class UserDefinedAggregateFunction
	
	spark.udf.register
		public MySum extends UserDefinedAggregateFunction {
		…
		}
		
		spark.udf.register("mysum", new MySum)

a	b
hz	[1,2]
bj	[3,4]

udtf

a	b
hz	1
hz	2
bj	3
bj	4

SparkSQL

1 DataSource

2 丰富的算子

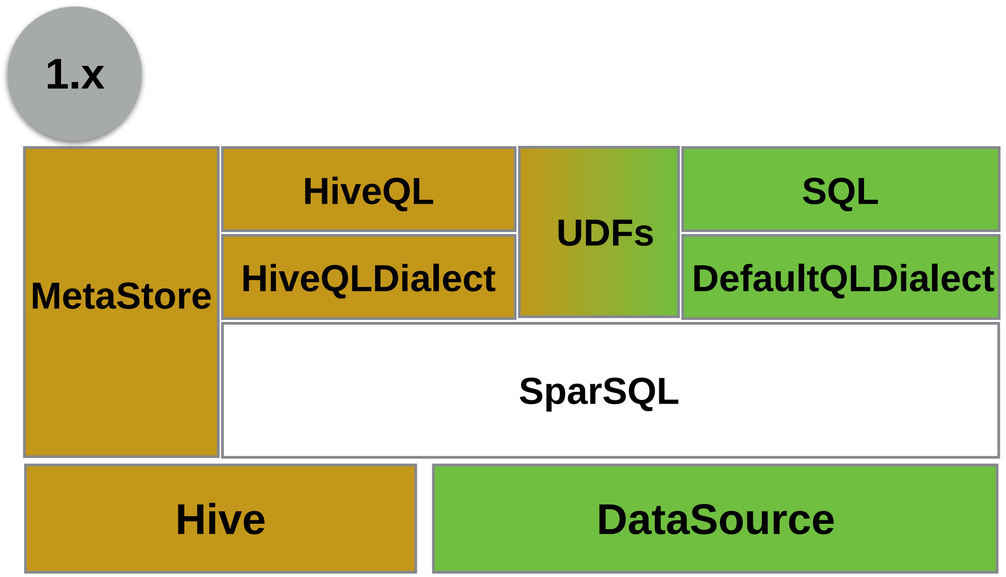
3 **Hive兼容**

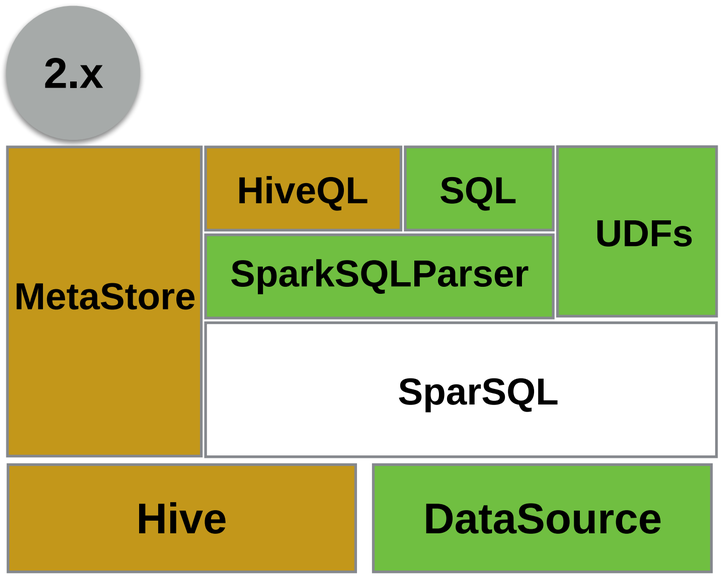
4 性能

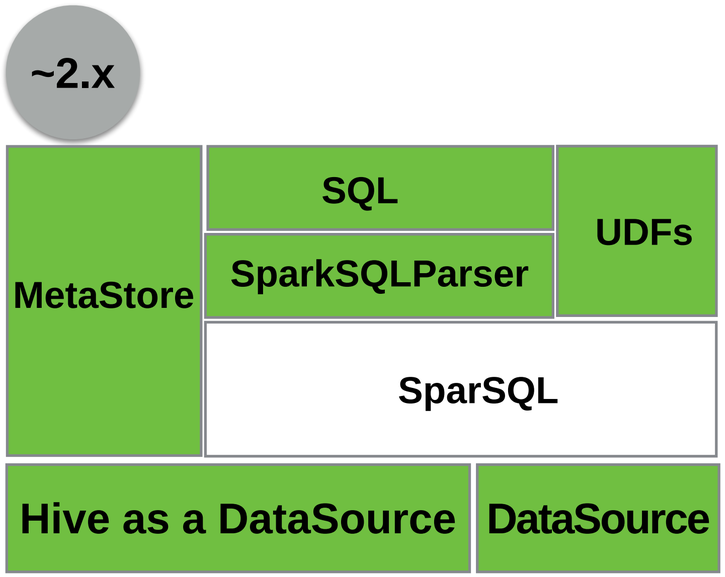
5 云上ETL

SparkSQL

Hive兼容







SparkSQL

Hive兼容

Hive as a DataSource

DDL

```
spark.sql("CREATE TABLE t(a string) USING hive")
```

```
spark.sql(
    """
      |CREATE TABLE t(a string, b string) USING hive PARTITIONED BY(b)
    """)
```

```
df.write.format("hive")
      .option("fileFormat","avro")
      .mode("append")
      .partitionedBy("a")
      .saveAsTable("t")
```

```
df.read.format("hive").table("t")
```

阿里云
奥运会全球指定云服务商

SparkSQL

1 DataSource

2 丰富的算子

3 Hive兼容

4 性能

5 云上ETL

SparkSQL

性能

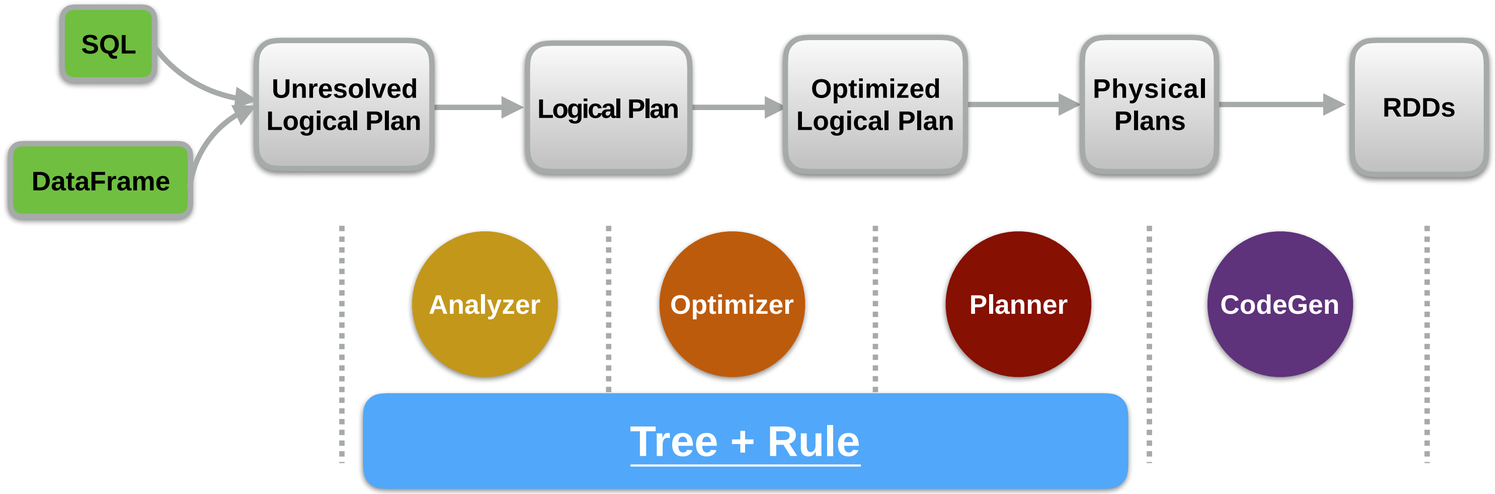
Catalyst

Project Tungsten

SparkSQL

Catalyst

DataFrame API	SQL
Catalyst	
SparkCore(RDD)	
DataSource	

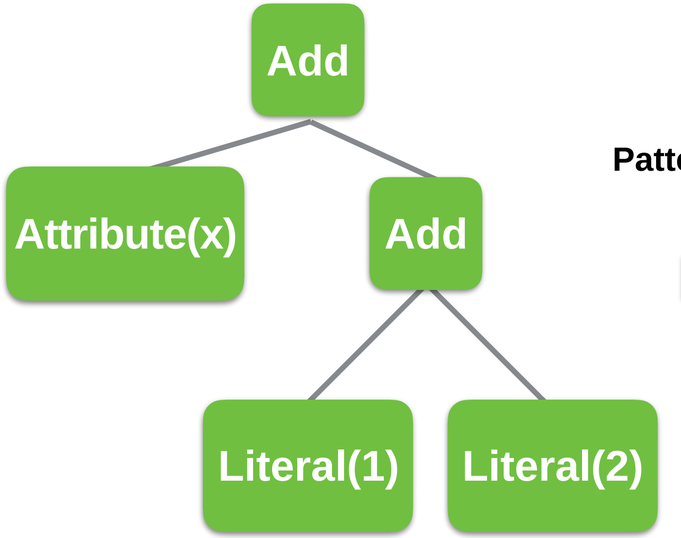


SparkSQL

Catalyst

Tree + Rule

x+(1+2)

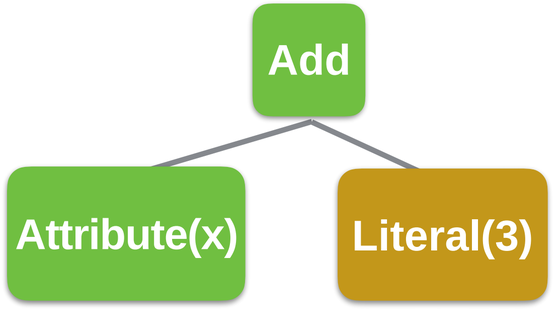


Tree-1

Pattern Matching

Rule

```
tree1.transform {
 case Add(Literal(c1), Literal(c2)) => Literal(c1+c2)
}
```



Tree-2

Pattern Matching

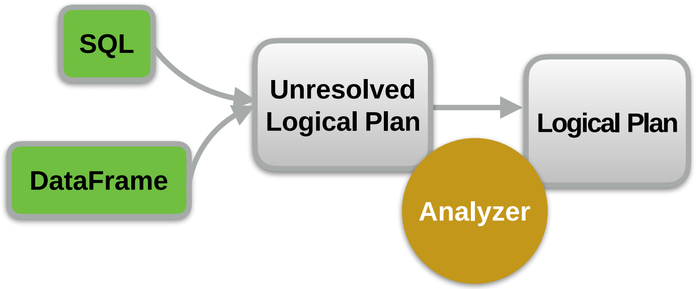
Rule

.....

Tree-N

SparkSQL

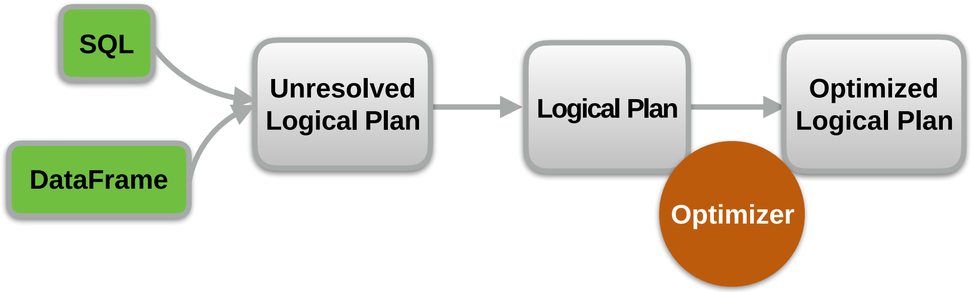
Catalyst



select a from t

SparkSQL

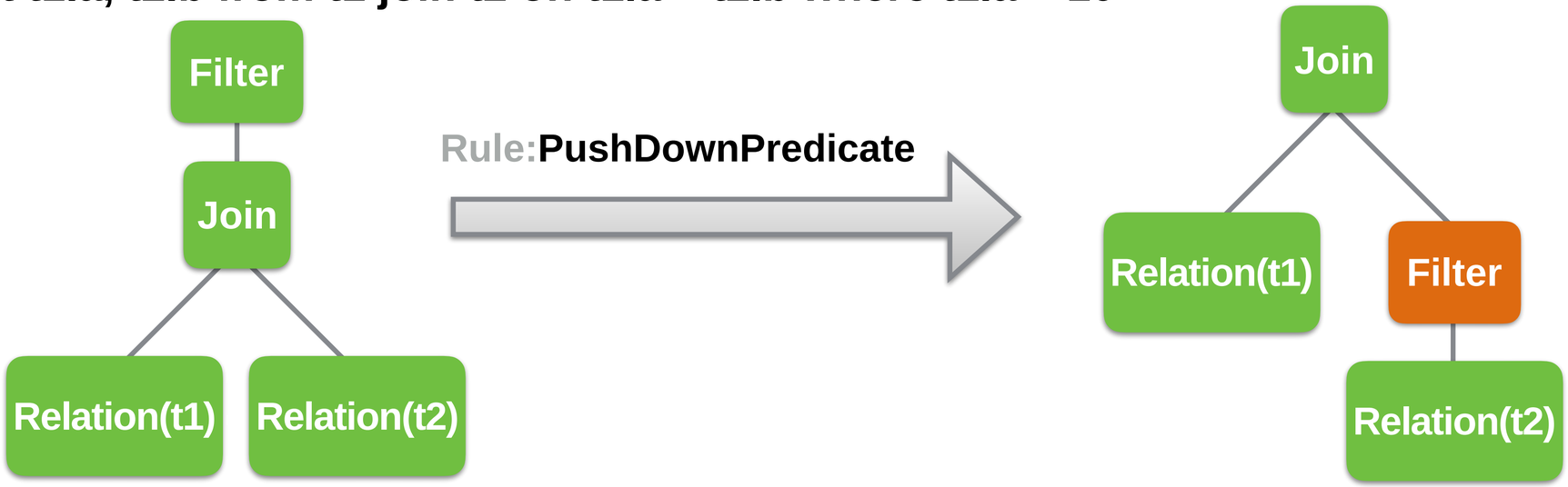
Catalyst



Optimizer Rules:

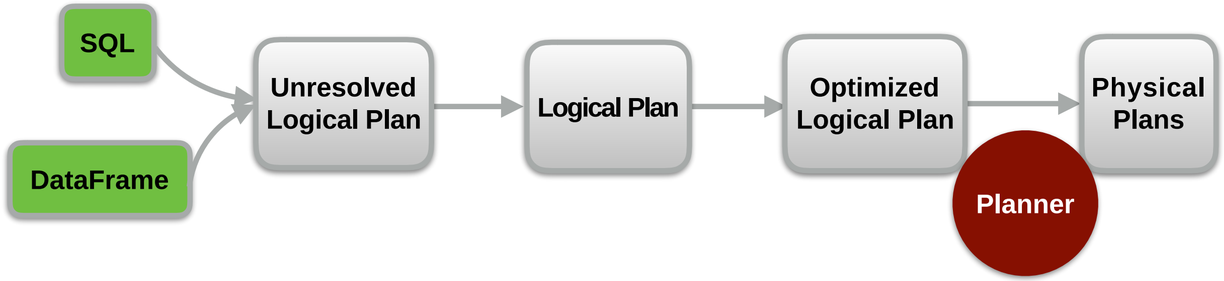
PushDownPredicate
ColumnPruning
ConstantFolding
EliminateOuterJoin
ReorderJoin
...

select t1.a, t2.b from t1 join t2 on t1.a = t2.b where t1.a = 10

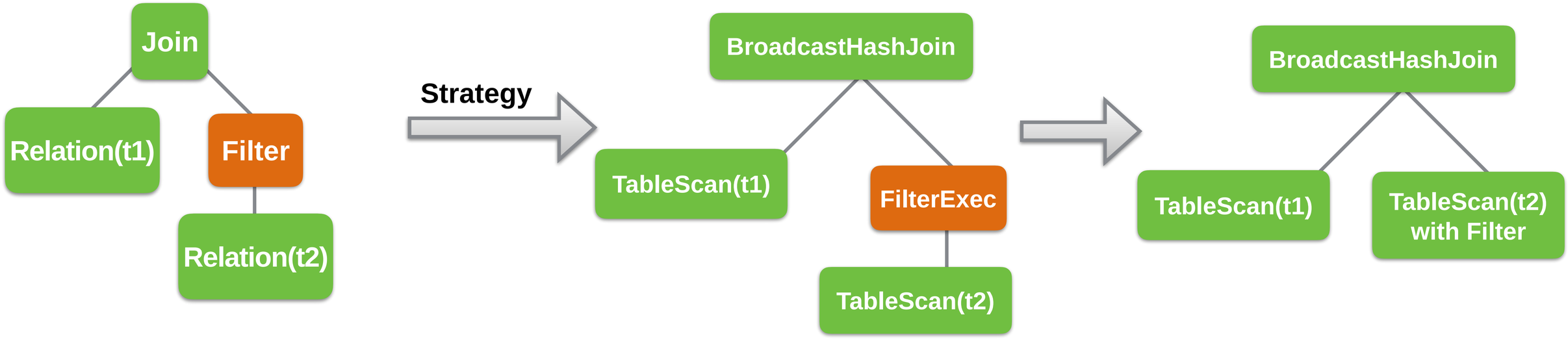


SparkSQL

Catalyst

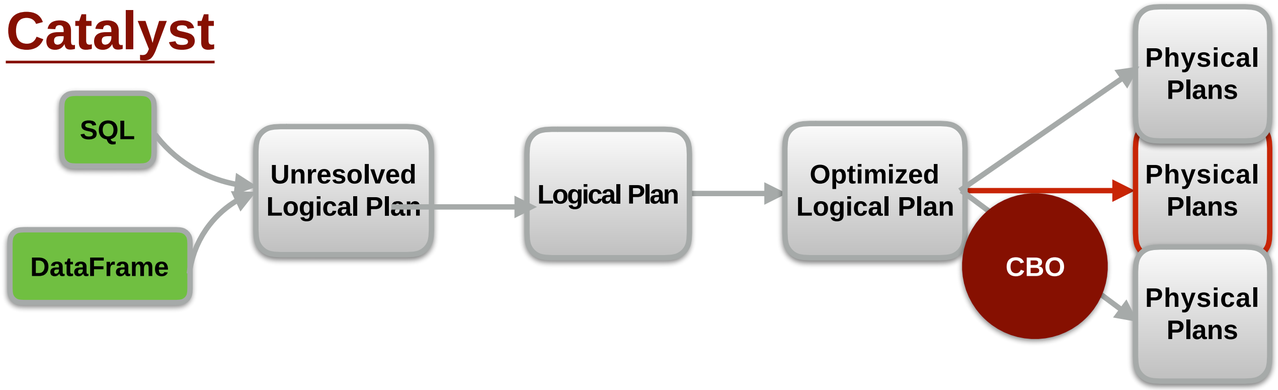


select t1.a, t2.b from t1 join t2 on t1.a = t2.b where t1.a = 10



SparkSQL

Catalyst



ANALYZE TABLE table COMPUTE STATISTICS

TableStatistics:rowCount/size/ColStatus

ColumnStatistics:distinct/min/max…

采集信息

CatalogTable

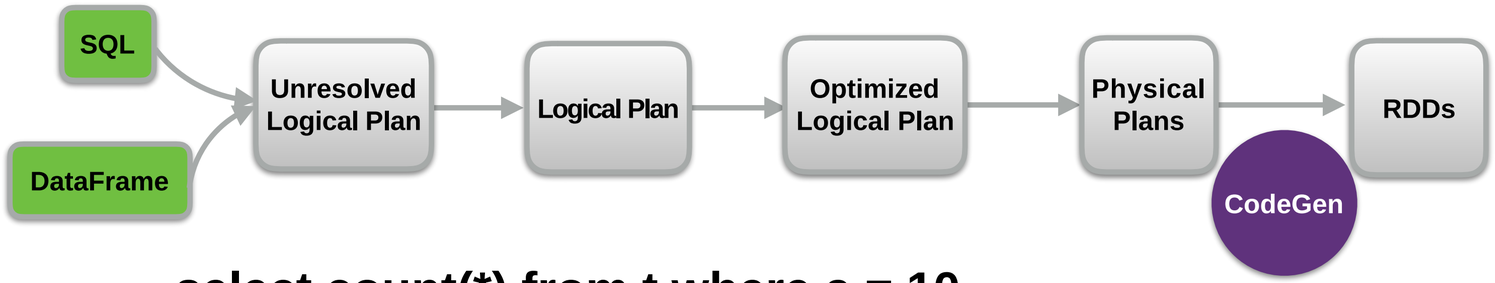
存储信息

Cost Function 计算代价
选择代价最小的PhysicalPlan

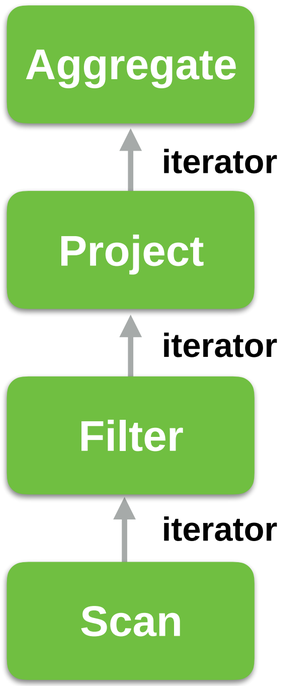
优化查询计划

SparkSQL

Catalyst



select count(*) from t where a = 10




```
class Filter
{
  def next(): Row = {
    var current = child.next()
    return current
  }
}
```

☺ 灵活组合

☹ 虚函数调用
中间临时变量 + trillion rows = 性能

WholeStageCodeGen

```
var count = 0
for (a in t) {
  if (a == 1000) {
    count += 1
  }
}
```

SparkSQL

Project Tungsten

Whole Stage CodeGen:

减少迭代中间变量/函数调用,pipeline融合进一个函数

Memory Management:

直接存取二进制, 减少Java对象开销/GC

Cache-aware computation:

寄存器/L1/L2/L3 CPU缓存操作,减少CPU读取内存操作

SparkSQL

1 DataSource

2 丰富的算子

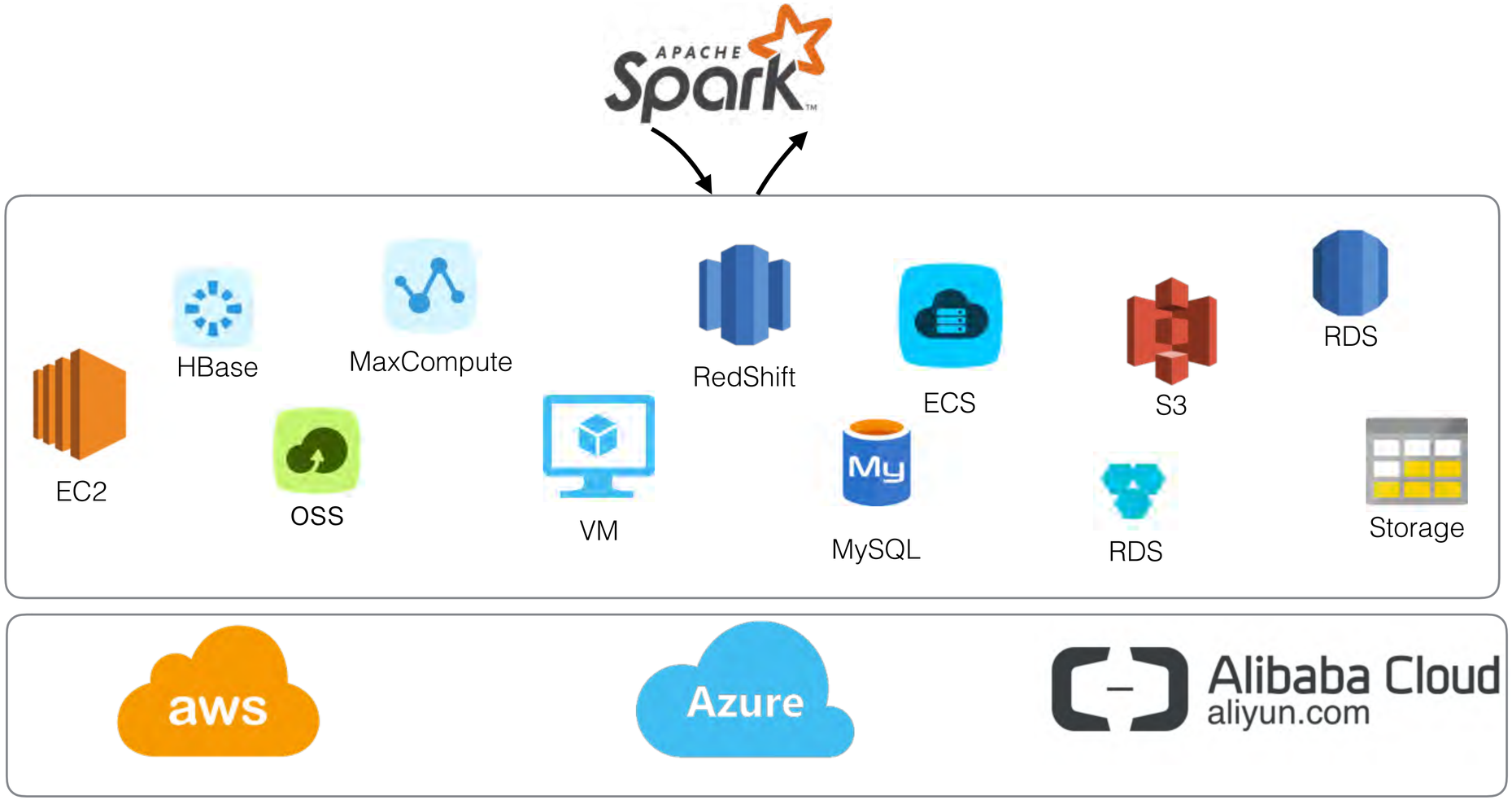
3 Hive兼容

4 性能

5 **云上ETL**

SparkSQL

云上ETL



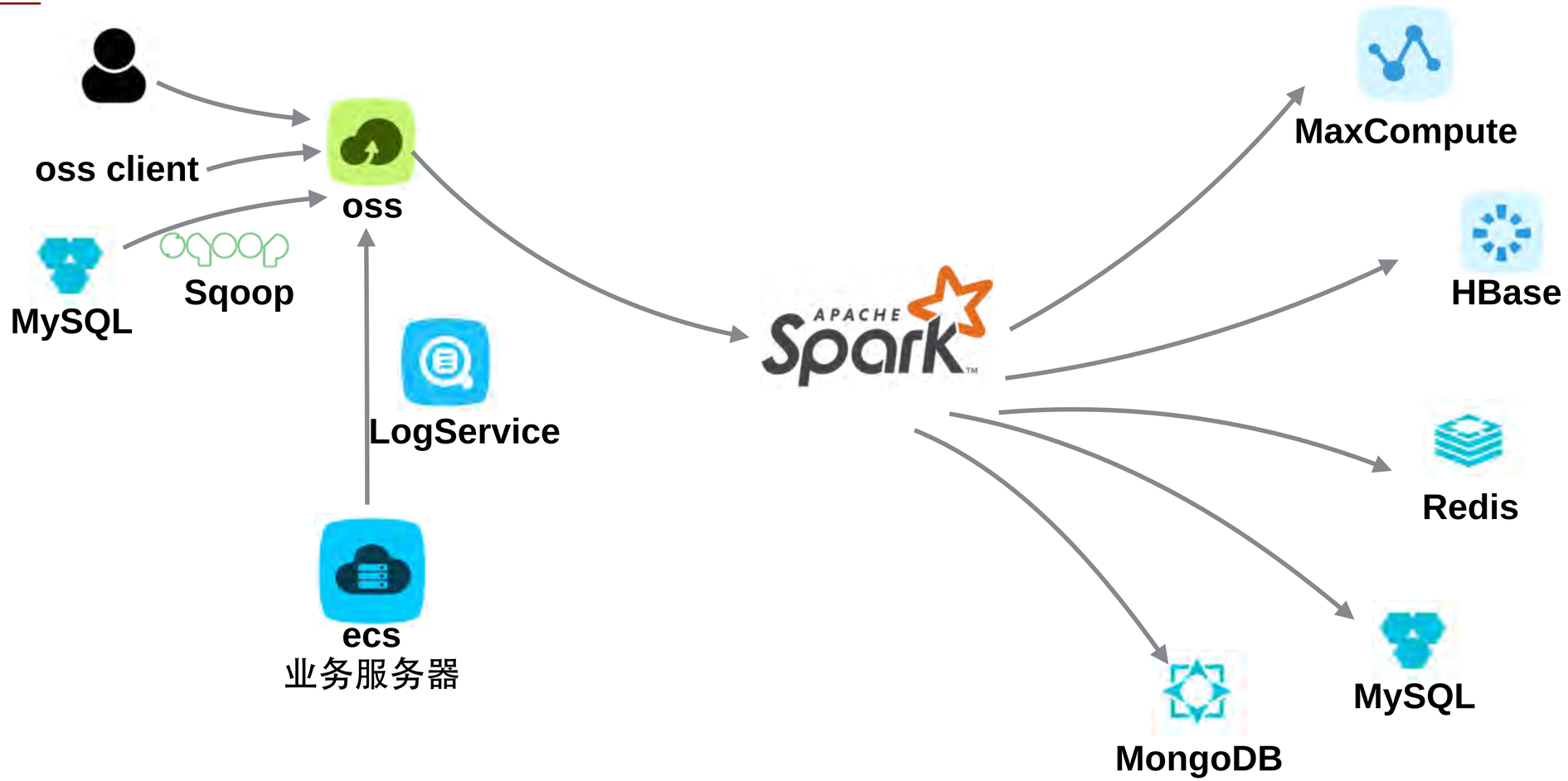
SparkSQL

云上ETL

- 数据源多
- 成本/性能
- 存储计算分离
- 资源动态扩容

SparkSQL

云上ETL



https://github.com/aliyun/aliyun-emapreduce-sdk

SparkSQL

云上ETL

OSS

成本低：

价格低廉
无需副本

存储计算分离

对象存储

性能：0.7:1.1

成本高：

价格高
2~3副本

BlockStorage

云盘

阿里云
奥运会全球指定云服务商

谢谢！