OAP – SQL INDEX & DATA SOURCE CACHE

Shen Xiangxiang - Intel Corporation

2020/08/27

Agenda

- Overview
- SQL Data Source Cache Introduction
- SQL Index Introduction

What’s challenge

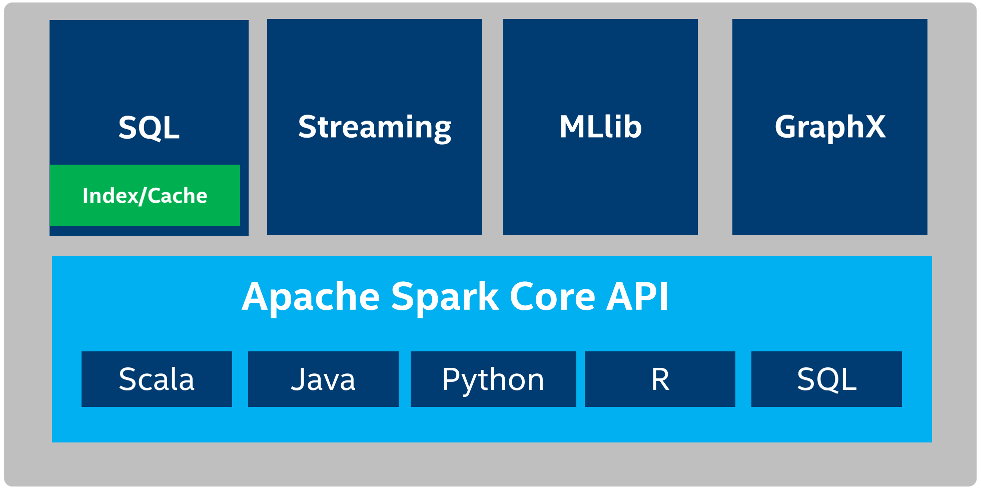
Interactive queries usually processes on a large data set but return a small portion of data filtering out with a specific condition. Customers are facing big challenges in meeting the performance requirement of interactive queries as we wants the result returned in seconds instead of tens of minutes or even hours.

```
select ss_sold_date_sk, ss_sold_time_sk, ss_item_sk, ss_cdemo_sk, ss_store_sk, ss_ticket_number,
        ss_ext_discount_amt, ss_ext_wholesale_cost, ss_ext_tax
        from fact.ss_sales
        where (date='20200801' and ss_customer='xxx' and ss_item_sk='806486')
        limit 10
```

For disaggregated compute and storage architecture, it is easy to introduce some performance challenges to the computing platform, some of which include, data locality, increased serialization/deserialization and sharing of hardware resources.

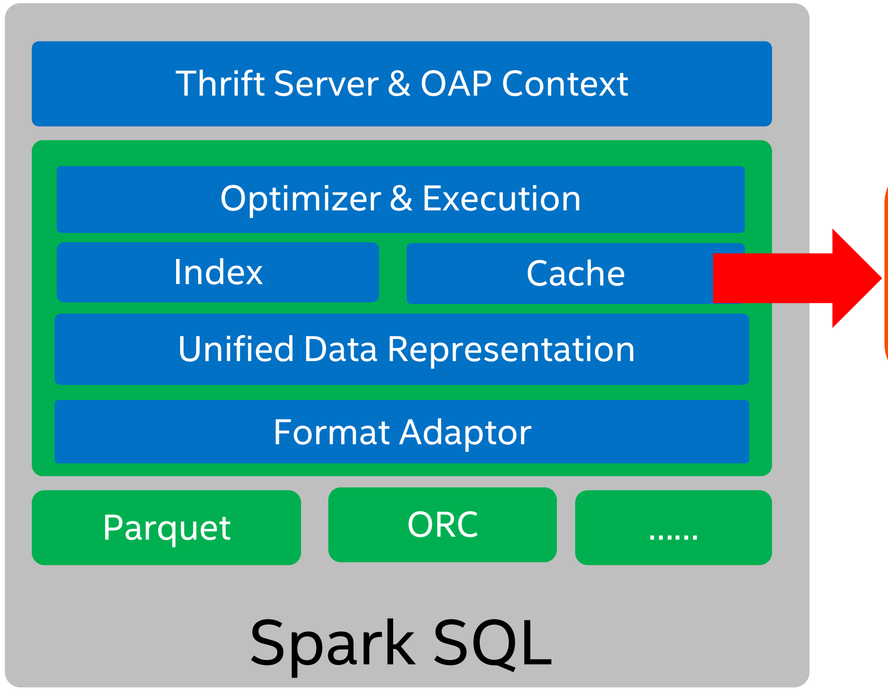
SQL Index & Data Source Cache Overview

SQL Index & Data Source Cache is a package for Spark to speed up interactive queries (ad-hoc queries) and improve the performance on the computing platform by utilizing cache and index technologies. By properly using index and cache (PMem), the performance of some interactive queries can possible be improved by order of magnitude.



SQL Index & Data Source Cache Architecture

SQL Index & Data Source Cache are designed to leverage the user defined indices and smart fine-grained in-memory data caching strategy for boosting Spark SQL performance.



By using PMem (AEP) as index and data cache, we can provide a more cost-effective solutions for high performance environment requirement

SQL Index & Data Source Cache – File Formats

- Supported Columnar based format Parquet and ORC
- Unified data representation supports different File Format transparently
- Modified data source reader with index & cache support

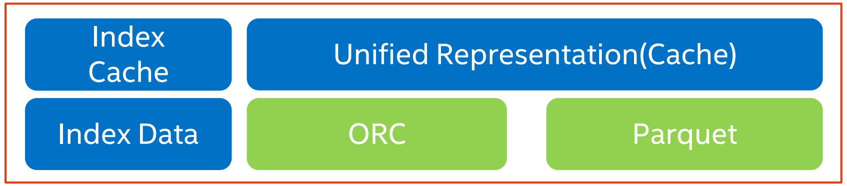
SQL DATA SOURCE CACHE INTRODUCTION

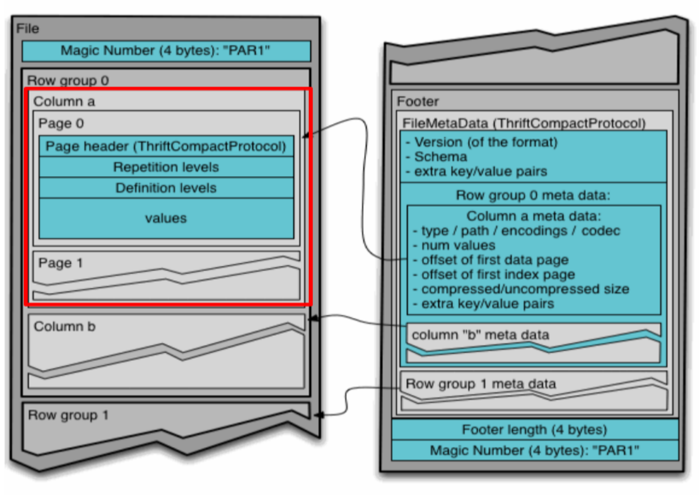
Fine-Grained Cache

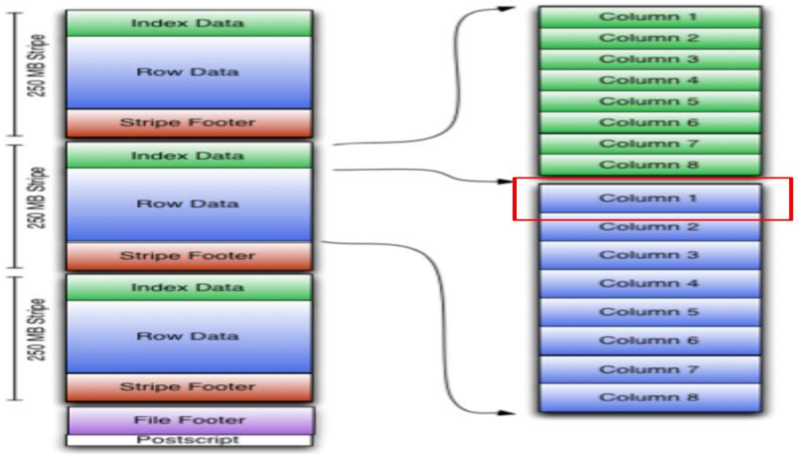
Fine-grained Unified Memory Representation Can Support typical data sources.

- Parquet: One column data in one RowGroup
- ORC: One column data in one Stripe

Index Data Cache



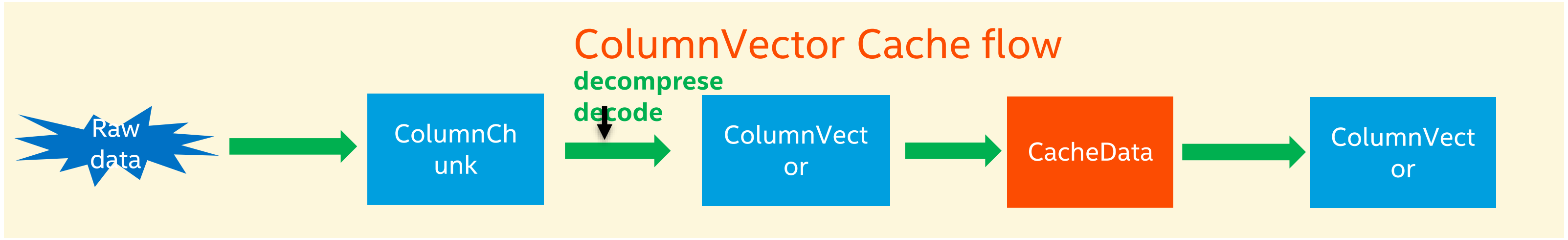

File
Magic Number (4 bytes): "PAR1"
Row group 0
Column a
Page 0
Page header (ThriftCompactProtocol)
Repetition levels
Definition levels
values
Page 1
Column b
Row group 1
Footer
FileMetaData (ThriftCompactProtocol)
- Version (of the format)
- Schema
- extra key/value pairs
Row group 0 meta data:
Column a meta data:
- type / path / encodings / codec
- num values
- offset of first data page
- offset of first index page
- compressed/uncompressed size
- extra key/value pairs
column "b" meta data
Row group 1 meta data
Footer length (4 bytes)
Magic Number (4 bytes): "PAR1"

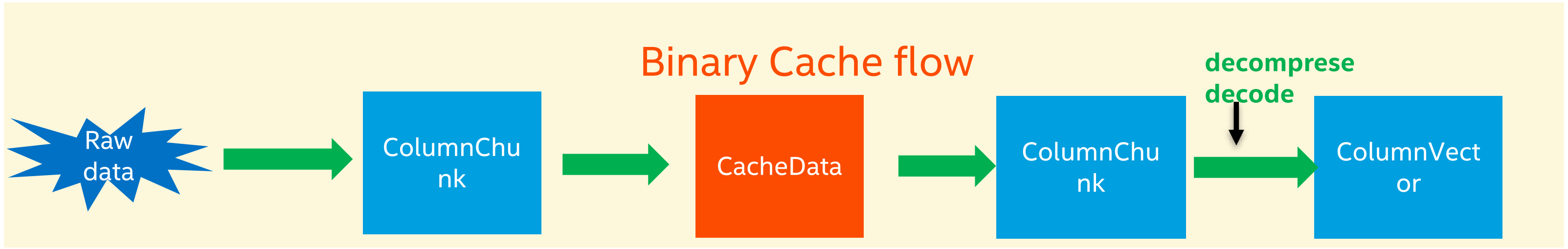


Cache Format

- Vectorized data (ColumnVector)
- Binary raw data (ColumnChunk)

Take parquet to illustrate





Cache Strategy

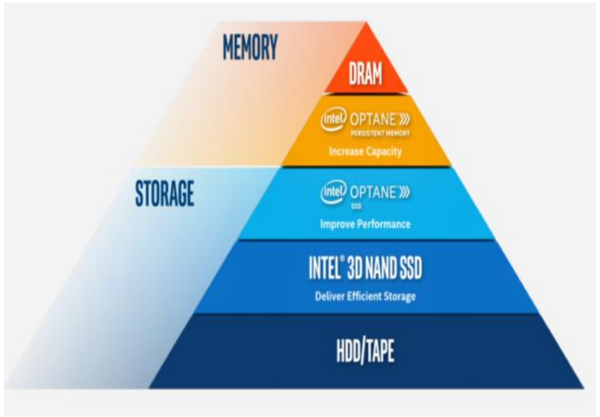
- Smart Policy for Cache Eviction
  - LRU cache policy
  - Automatic caching and eviction transparently to end user
- Cache Hot tables
  - User can specify tables to cache. For example, cache dimension tables in the data warehouse.

Cache Storage

- DRAM
  - Off Heap Memory
    - ❑ Stay out of JVM GC
- Persistent Memory Module (PMem)
  - Offer a large and persistent memory tier at an affordable cost.

Intel® Optane™ Persistent Memory 200 Series (128GB PMEM) Module

- 128 GB Capacity
- Persistent Memory Module (PMem) Form Factor
- DDR-T Interface

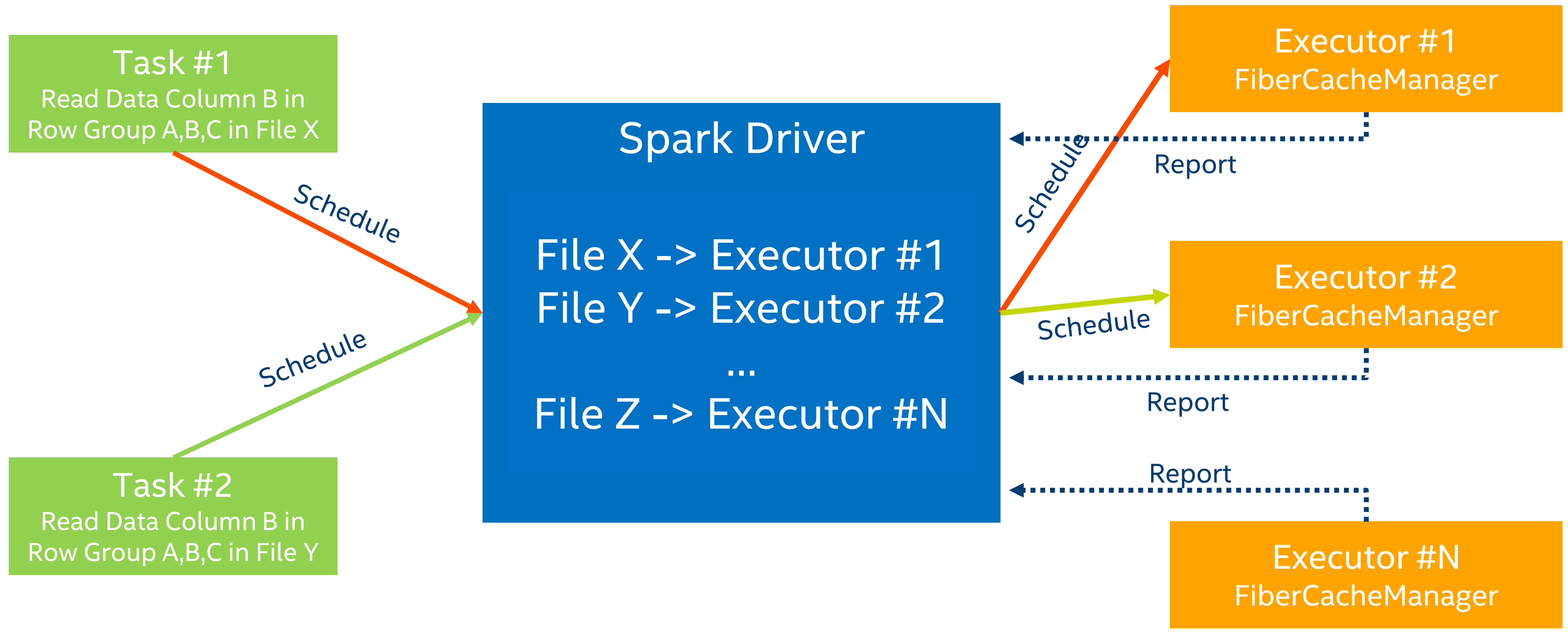


Data Source Cache backend

- DRAM
  - Google Guava Cache library
- PMem
  - Vmemcache
    - ❑ An embeddable and lightweight in-memory caching solution.
  - Memkind with Guava
    - ❑ The memkind library is a user extensible heap manager built on top of jemalloc
  - Plasma
    - ❑ A high-performance shared-memory object store of Apache Arrow
    - ❑ Provide node-level external cache service

Data Source Cache (Report & Schedule)

Cache-locality aware Task Scheduling



Get Started - Data Source Cache

YRAN with client mode

Basic Configuration:

```
spark.sql.extensions          org.apache.spark.sql.OapExtensions
spark.files                   /home/oap/jars/oap-cache-<version>-with-spark-<version>.jar,/home/oap/jars/oap-
common-<version>-with-spark-<version>.jar
spark.executor.extraClassPath   ./oap-cache-<version>-with-spark-<version>.jar:./oap-common-<version>-with-
spark-<version>.jar
spark.driver.extraClassPath     /home/oap/jars/oap-cache-<version>-with-spark-
<version>.jar:/home/oap/jars/oap-common-<version>-with-spark-<version>.jar
```

Get Started – Data Source Cache (DRAM)

YRAN with client mode

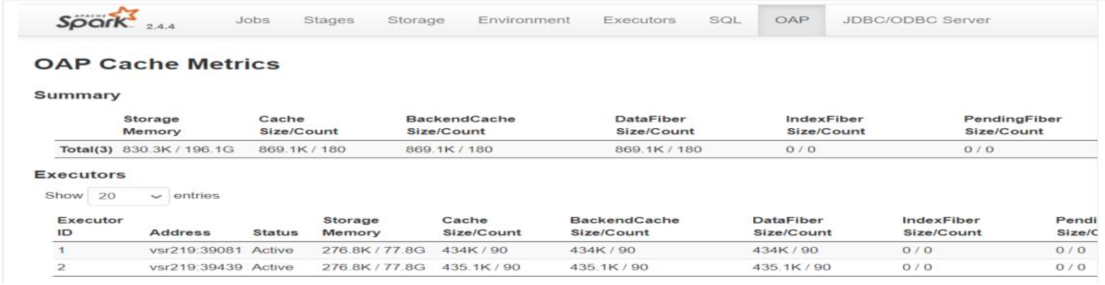
Configuration:

spark.oap.cache.strategy	guava
spark.sql.oap.fiberCache.memory.manager	offheap
spark.sql.oap.fiberCache.offheap.memory.size	50g
spark.executor.memoryOverhead	50g
spark.sql.oap.parquet.data.cache.enable	true

Use Cache

```
spark-sql> SELECT * FROM src WHERE a > 100 and a < 1000;
```

Cache UI



OAP Cache Metrics

Summary

	Storage Memory	Cache Size/Count	BackendCache Size/Count	DataFiber Size/Count	IndexFiber Size/Count	PendingFiber Size/Count
Total(3)	830.3K / 196.1G	869.1K / 180	869.1K / 180	869.1K / 180	0 / 0	0 / 0

Executors

Show 20 entries

Executor ID	Address	Status	Storage Memory	Cache Size/Count	BackendCache Size/Count	DataFiber Size/Count	IndexFiber Size/Count	Pendi Size/C
1	vsr219:39081	Active	276.8K / 77.8G	434K / 90	434K / 90	434K / 90	0 / 0	0 / 0
2	vsr219:39439	Active	276.8K / 77.8G	435.1K / 90	435.1K / 90	435.1K / 90	0 / 0	0 / 0

Get Started – Data Source Cache (PMem)

YRAN with client mode

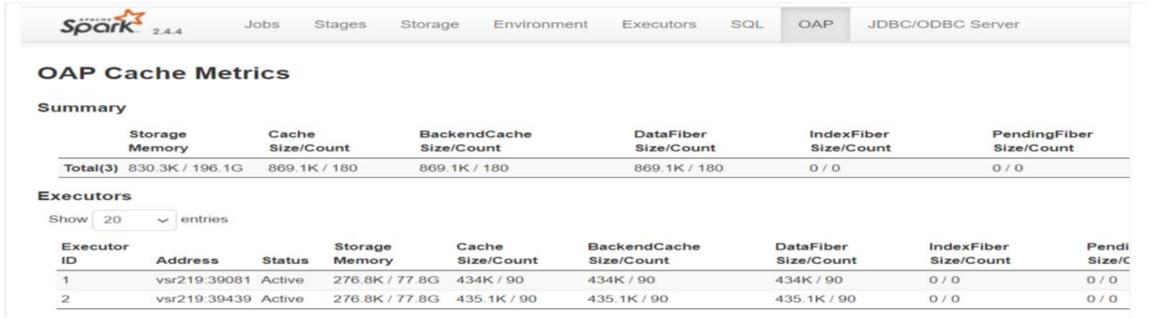
Configuration:

spark.oap.cache.strategy	vmem
spark.sql.oap.fiberCache.persistent.memory.initial.size	256g
spark.sql.oap.cache.guardian.memory.size	10g
spark.sql.oap.parquet.binary.cache.enabled	true

Use Cache

```
spark-sql> SELECT * FROM src WHERE a > 100 and a < 1000;
```

Cache UI



OAP Cache Metrics

Summary

	Storage Memory	Cache Size/Count	BackendCache Size/Count	DataFiber Size/Count	IndexFiber Size/Count	PendingFiber Size/Count
Total(3)	830.3K / 196.1G	869.1K / 180	869.1K / 180	869.1K / 180	0 / 0	0 / 0

Executors

Show 20 entries

Executor ID	Address	Status	Storage Memory	Cache Size/Count	BackendCache Size/Count	DataFiber Size/Count	IndexFiber Size/Count	Pendi Size/C
1	vsr219:39081	Active	276.8K / 77.8G	434K / 90	434K / 90	434K / 90	0 / 0	0 / 0
2	vsr219:39439	Active	276.8K / 77.8G	435.1K / 90	435.1K / 90	435.1K / 90	0 / 0	0 / 0

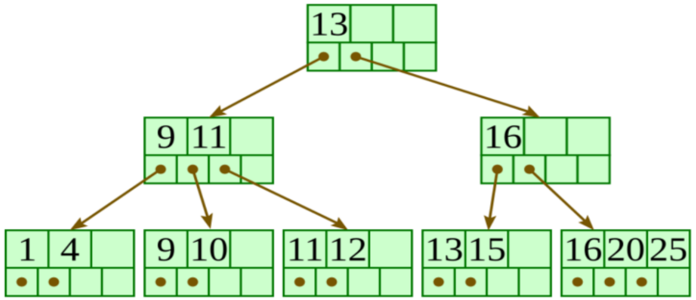
SQL INDEX INTRODUCTION

SQL Index Type

We support following Index types

- B+ Tree Index
  - Suite for range scan. Support prefix string search
- Bitmap Index
  - Suite for enumeration value

B+ Tree



SQL Index File Location

- Unified Index file format for Parquet and ORC
  - {datafile}_{column}.index

- Separated Index file dependent from raw data
  - Default directory : The directory where the data file is located
  - Specify directory : spark.sql.oap.index.directory hdfs://xx:9000/xx

SQL Index Statistics

- Index Statistics
  - MinMax
  - BloomFilter
  - Sample
  - PartbyValue
- Index Selection based on Statistics
  - Skip partition
  - Full scan
  - Use index

SQL Index DDL

- DDL - Create / Drop / Refresh / Check / Show Index
  - CREATE OINDEX index_name ON table_name (column_name) USING [BTREE, BITMAP]
  - BTREE is default Index Type

Spark Shell Example:

```
> spark.sql(s"""CREATE TABLE oap_test (a INT, b STRING)
     USING parquet
     OPTIONS (path 'hdfs:///user/oap/')""".stripMargin)
> val data = (1 to 30000).map { i => (i, s"this is test $i") }.toDF().createOrReplaceTempView("t")
> spark.sql("insert overwrite table oap_test select * from t")
> spark.sql("create oindex index1 on oap_test (a)")
> spark.sql("show oindex from oap_test").show()
> spark.sql("drop oindex index1 on oap_test")
```

Get Started - SQL Index

on YRAN with client mode

Configuration:

```
spark.sql.extensions             org.apache.spark.sql.OapExtensions
spark.files                      /home/oap/jars/oap-cache-<version>-with-spark-<version>.jar,/home/oap/jars/oap-common-<version>-with-spark-<version>.jar
spark.executor.extraClassPath    ./oap-cache-<version>-with-spark-<version>.jar:./oap-common-<version>-with-spark-<version>.jar
spark.driver.extraClassPath      /home/oap/jars/oap-cache-<version>-with-spark-<version>.jar:/home/oap/jars/oap-common-<version>-with-spark-<version>.jar
```

Use index:

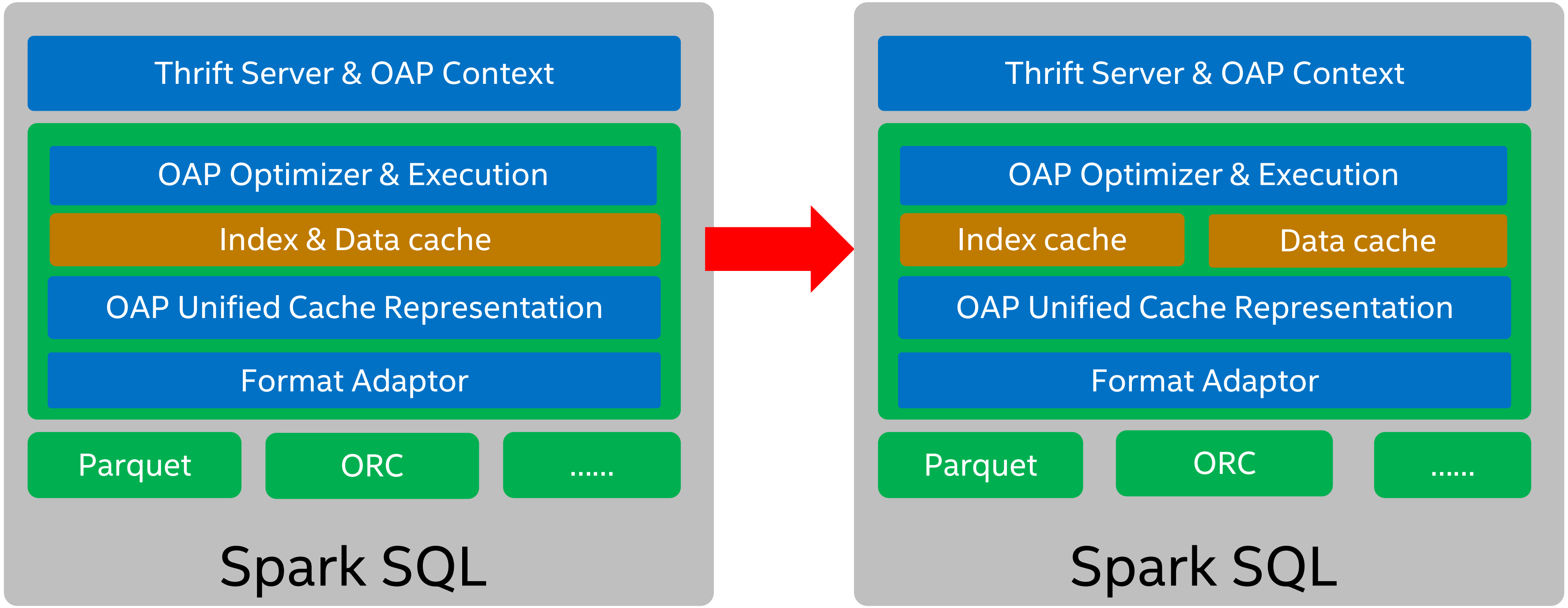
```
spark-sql> CREATE TABLE src(a: Int, b: String) USING Parquet;

spark-sql> INSERT INTO TABLE src SELECT key, value FROM xxx;

spark-sql> Create OINDEX on src (a) USING BTREE;

spark-sql> SELECT * FROM src WHERE a > 100 and a < 1000;
```

Optimization Notice

Index and data cache separation

Use independent data cache and index cache pool

By using DARM as index cache and PMem as data cache, we can provide a more cost-effective solutions for high performance environment requirement


Thrift Server & OAP Context
OAP Optimizer & Execution
Index & Data cache
OAP Unified Cache Representation
Format Adaptor
Parquet
ORC
……
Spark SQL
Thrift Server & OAP Context
OAP Optimizer & Execution
Index cache
Data cache
OAP Unified Cache Representation
Format Adaptor
Parquet
ORC
……
Spark SQL

加入Apache Spark中国技术社区钉钉群

关注Apache Spark技术交流社区微信公众号