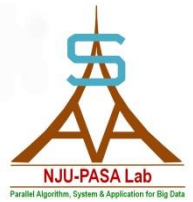
TACHYON

Tachyon的最新进展与用例介绍

顾 荣
南京大学 PASA大数据实验室
2015-06-27@Spark Meetup(Beijing)

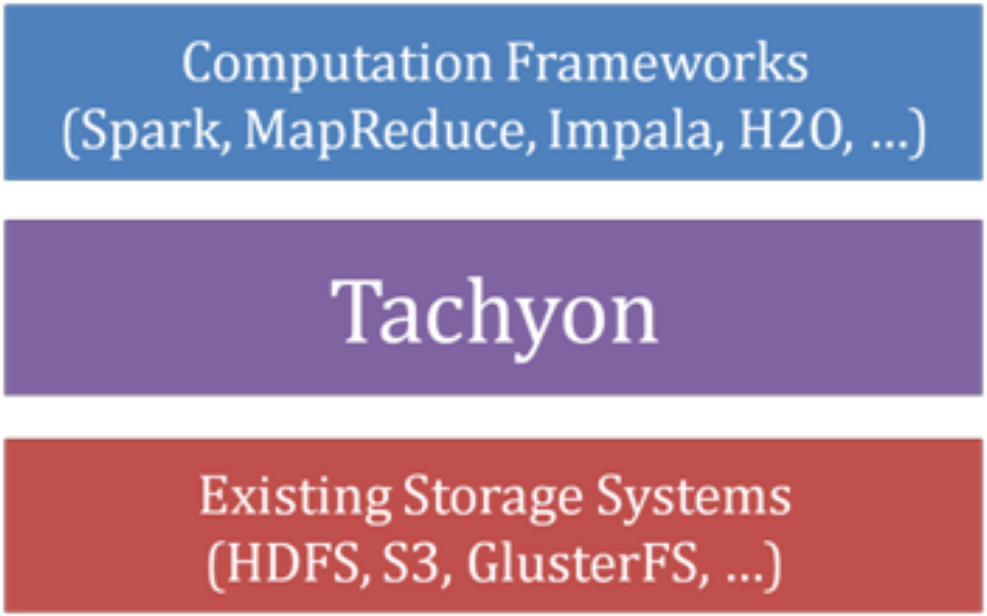
内容



- Tachyon社区的最新进展
- Tachyon的新特性(0.5版本之后)
- 应用案例分析

关于Tachyon

- Tachyon是以内存为中心的分布式文件系统，拥有高性能和容错能力，能够为集群框架（如Spark, MapReduce, Flink, Impala等）提供可靠的内存级速度的文件共享服务，目前最新发布版本是0.6.4



成立了一家商业公司Tachyon Nexus

- Founded by Tachyon creators and top committers, led by Haoyuan Li.
- $7.5 million Series A by Andreessen Horowitz
- Dedicated to Tachyon
- Committed to Open Source
- www.tachyonnexus.com

TACHYON

A Memory-Centric Distributed Storage System

NJU-PASA Lab
Parallel Algorithm, System & Application for Big Data

Enable New Workloads in Any Storage System

Generate value from your data in S3, GlusterFS, Swift, NFS, HDFS, etc.

Accelerate Access to Your Data

Tachyon helps companies improve their big data analytics speed by orders of magnitude.

Join a Fast-Growing, Open Source Community

One of the fastest-growing projects in the big data ecosystem. Amassed 100+ contributors from 30+ companies in 2 short years.

Work with the Compute Framework of Your Choice

Tachyon supports many different frameworks, such as Apache Spark, Apache MapReduce, Apache Flink, and Impala.

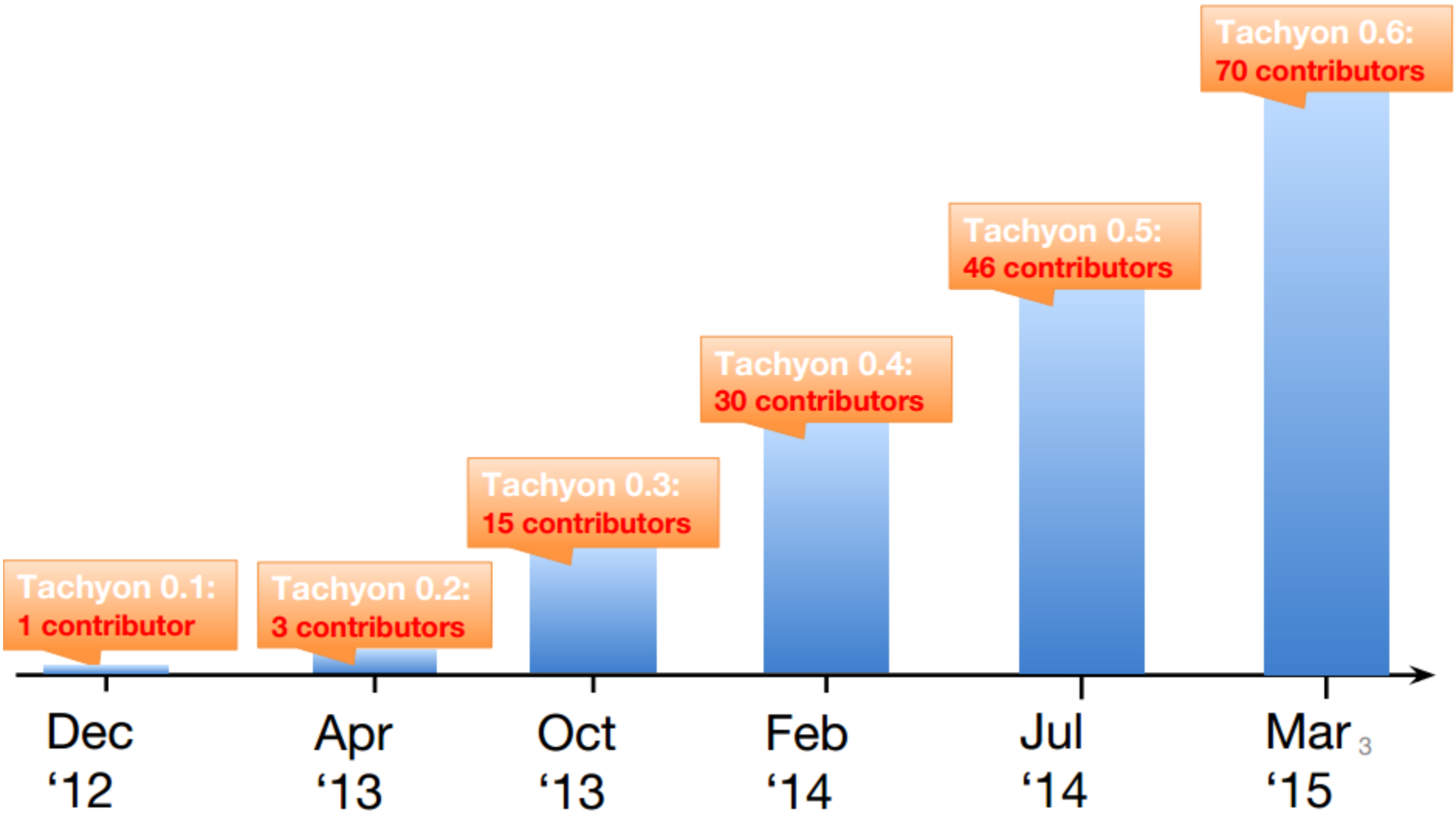
Build on Solid Foundation

Production Tachyon deployments scale out to more than 100 nodes, and manage PBs of data.

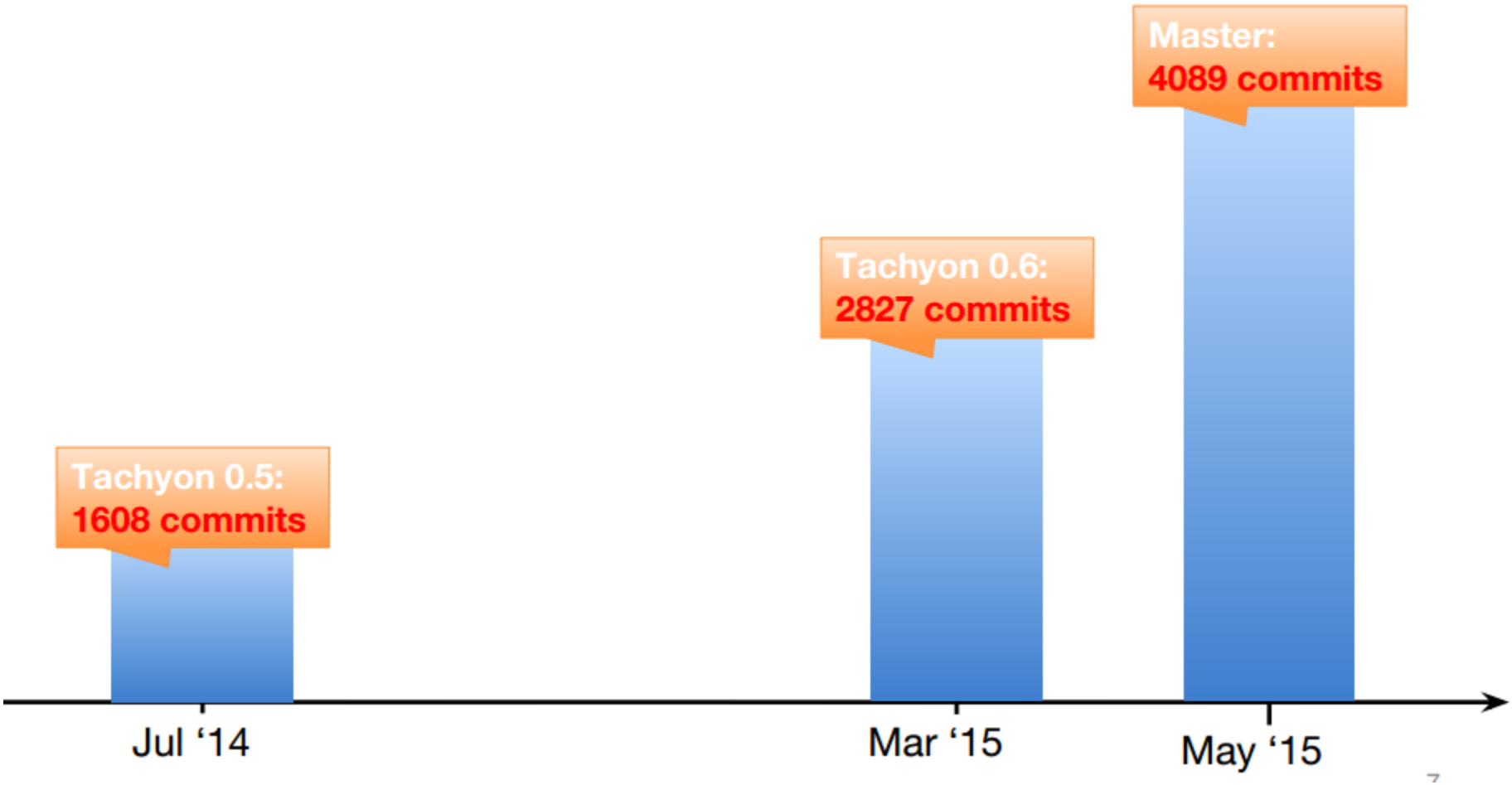
Scale up in Any Industry

Leaders from Internet, Finance, Energy, and Telecom are using Tachyon to meet their ever-demanding needs.

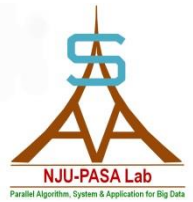
Contributors的情况



Commits数量的情况

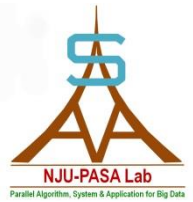


内容



- Tachyon社区的最新进展
- Tachyon的新特性(0.5版本之后)
- 应用案例分析

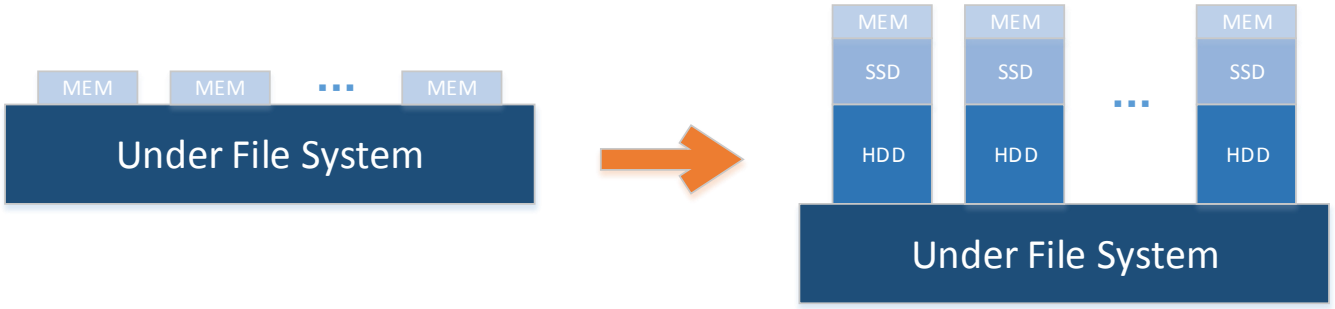
Tachyon新特性 (0.5.0之后)



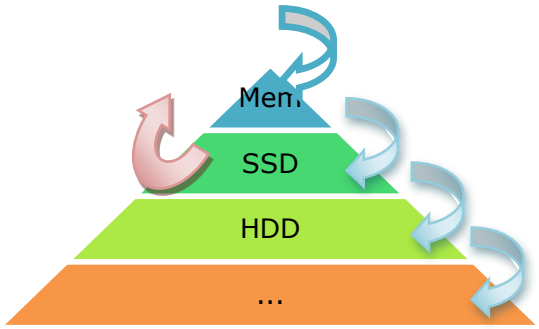
- Tiered Storage 层次化存储
  - 支持MEM、SSD、HDD的多层次存储
- 更方便的部署方式
  - 一键部署在 VirtualBox 或 Amazon Web Services 上
- 更多的底层存储系统
  - OpenStack Swift , IBM SoftLayer object store, Native S3
- 更清晰的代码结构
  - 模块化，方便并发开发和维护
- 更强大的性能测试模块
  - tachyon-perf，为用户和开发者提供便利
- 更多性能优化和新功能...

Tiered Storage 层次化存储

- Tiered Storage使得Tachyon的缓存空间更大，且能更好地支持非易失性存储设备



- 高效的分配/替换策略以获得更好的性能



更多细节参考 http://tachyon-project.org/master/Tiered-Storage-on-Tachyon.html

Tiered Storage 层次化存储

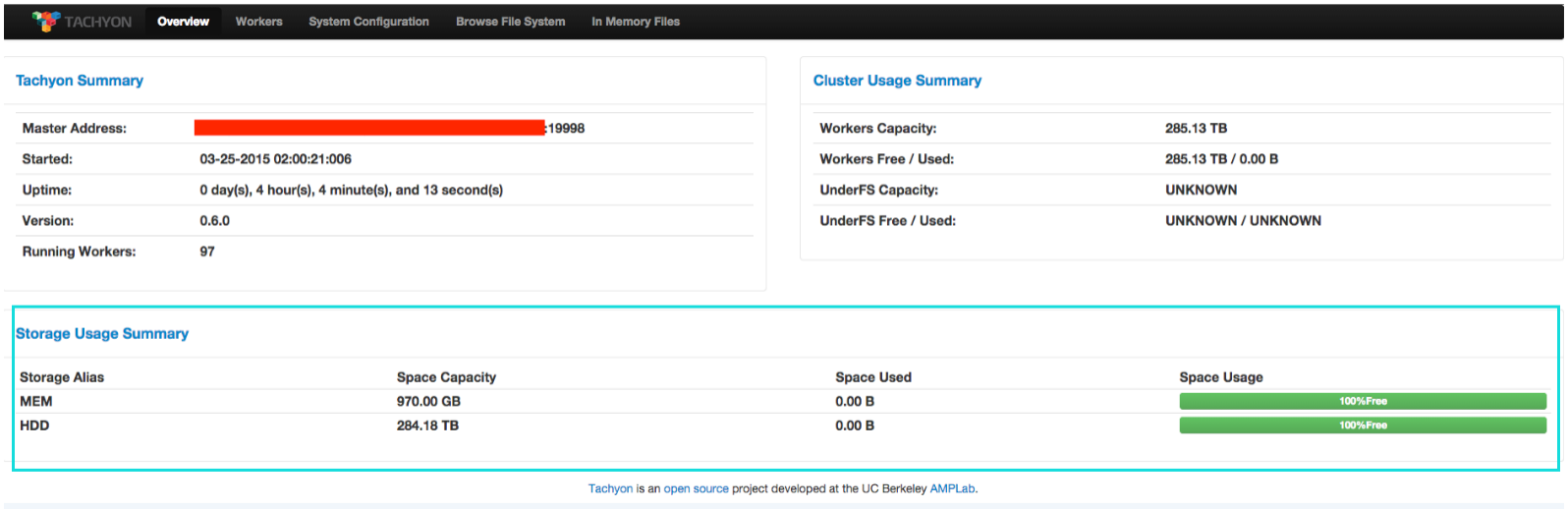
- 简单的配置方式

```
data.level1.dirs=”dir1, dir2…”; #high speed layer / small capacity
data.level2.dirs=”dir1, dir2…”; #low speed layer / large capacity

data.level1.dir.quota=”100G, 200G…”;
data.level2.dir.quota=”10T, 20T…”;

data.level1.dir.alias=”mem”;
data.level2.dir.alias=”hdd”;
```

- 方便的监控和管理



更方便的多环境部署方式

- 运行一个脚本就能将Tachyon部署在对应环境中
  - 运行*run_aws.sh*，部署到Amazon AWS
  - 运行*run_openstack.sh*，部署到OpenStack
  - 运行*run_docker.sh*，部署到Linux Container
  - 运行*run_vb.sh*，部署到VirtualBox

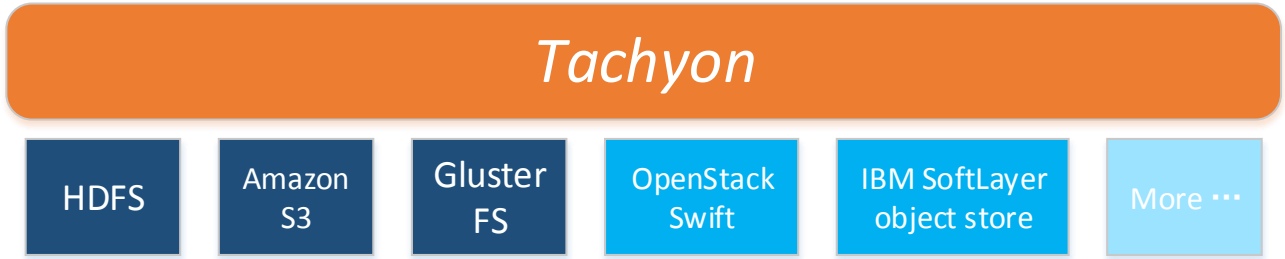
- 详细的个性化配置
  - tachyon版本
  - spark版本
  - UFS(hdfs,glusterfs…)版本
  - aws,docker…环境

```
vagrant
├── conf
│   ├── ec2-config.yml
│   ├── init.yml
│   └── openstack-config.yml
│   └── tachyon_version.yml
│   └── spark_version.yml
│   └── ufs_version.yml
├── README.md
├── run_xxx.sh
└── ...
```

更多细节参考 http://tachyon-project.org/master/Deploy-Module.html

更多的底层存储系统

- 新增了3个底层存储系统的支持
  - OpenStack Swift
  - IBM SoftLayer object store
  - Native S3
- 易于安装，不用修改Tachyon现有的外部接口



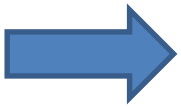
- Tachyon仍在不断支持更多的底层文件系统

更多细节参考 http://tachyon-project.org/master/Setup-UFS.html

更清晰的代码结构

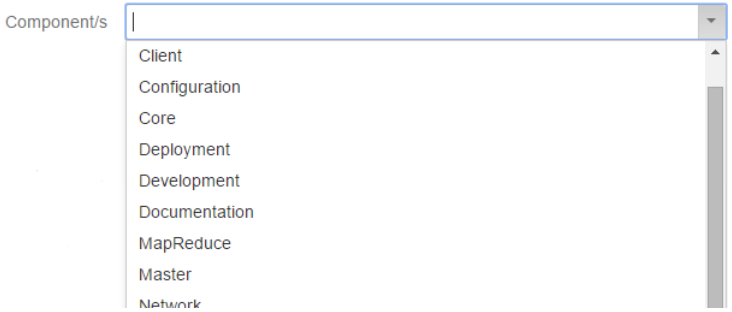
- 对代码结构进行了调整和模块化

```
Tachyon Project Parent
Tachyon Project Core
Tachyon Project Client
```



```
Tachyon Parent
Tachyon Common
Tachyon Under File System
Tachyon Under File System - Local FS
Tachyon Under File System - HDFS
Tachyon Under File System - Gluster FS
Tachyon Under File System - Swift
Tachyon Under File System - S3
Tachyon Clients
Tachyon Clients - Implementation
Tachyon Clients - Distribution
Tachyon Servers
Tachyon Integration Tests
Tachyon Shell
Tachyon Examples
Tachyon Assemblies
```

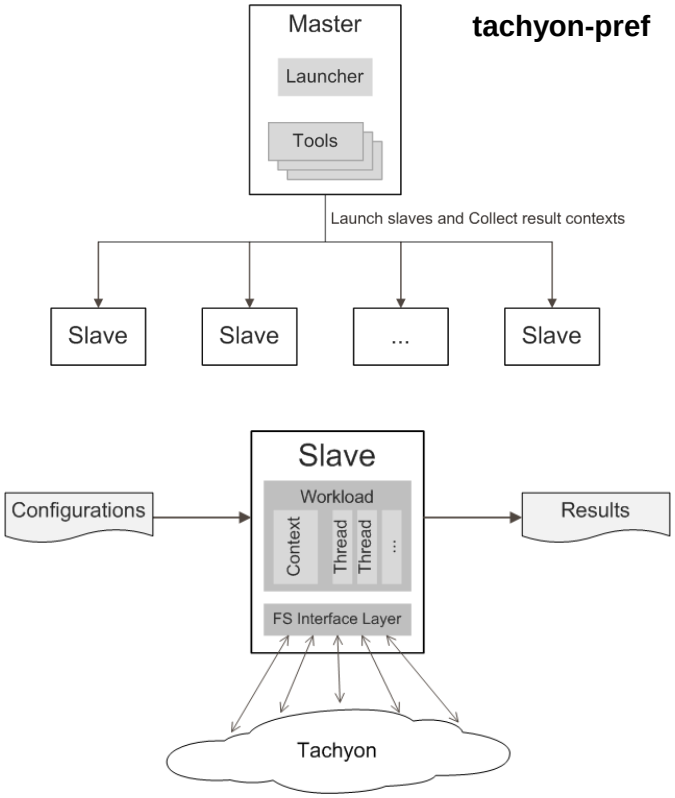
- 专注于各自的模块，提高了开发和维护的效率



更多细节参考 https://tachyon.atlassian.net/browse/TACHYON-49

更强大的性能测试模块

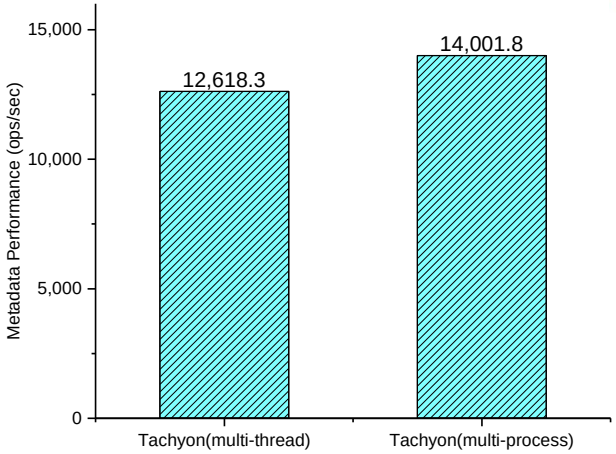
- tachyon-perf：面向Tachyon的性能测试工具
  - 便于用户根据自己的环境调试性能
  - 为开发者提供性能评测的手段
- 与Tachyon完美契合
  - 支持多节点/多进程/多线程测试
  - 提供多种测试用例
  - 能够根据需要定制或添加新的测试用例

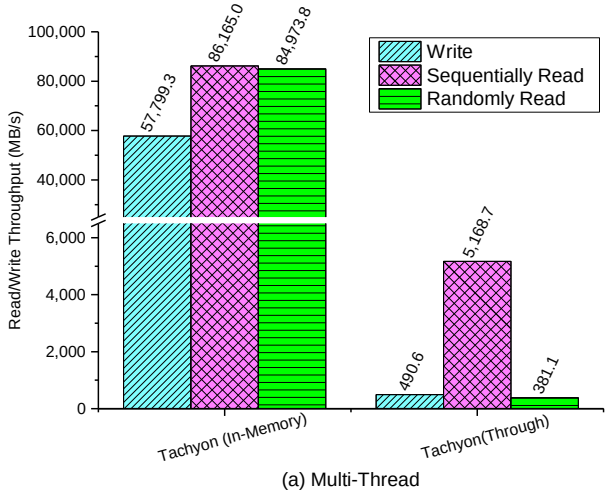


更多细节参考 https://github.com/amplab/tachyon/tree/perf

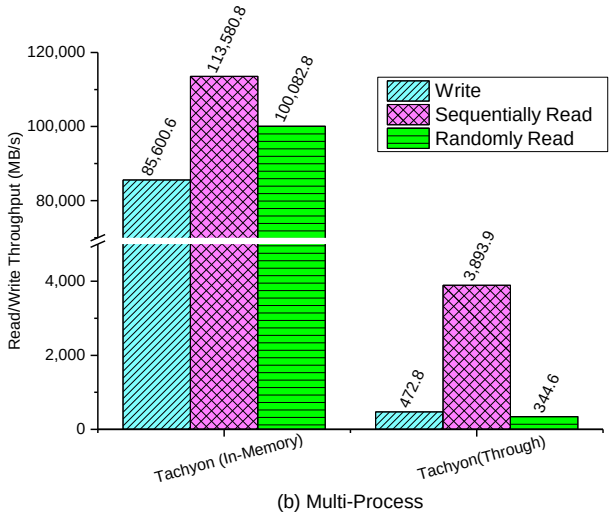
更强大的性能测试模块

- 元数据操作性能 →
- 基本读写性能 ↓





(a) Multi-Thread



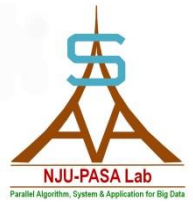
(b) Multi-Process

更多细节参考 https://github.com/amplab/tachyon/tree/perf

更多性能优化和新功能

- 优化节点之间数据传输的性能
  - 使用 Netty 代替原先的 java.nio
- 更方便的配置机制
  - 用户能够选择更熟悉的配置方式
    - 命令行环境：*-Dtachyon.foo="bar"*
    - Java环境：*tachyonConf.set("tachyon.foo", "bar");*
- [进行中]新的分配/替换策略
  - 更好地管理Tiered Storage的空间
  - 提升Tachyon作为大数据应用存储层的性能
- More…

内容



- Tachyon社区的最新进展
- Tachyon的新特性(0.5版本之后)
- 应用案例分析

**感谢由百度US工程师Shaoshan Liu为本节提供的一些数据和相关slides;*

百度北美的TB级别日常查询系统

30X Acceleration of Baidu's Big Data Analytics Workload

Tachyon Summary

Master Address:	
Started:	05-11-2015 10:30:40:438
Uptime:	4 day(s), 16 hour(s), 3 minute(s), and 50 second(s)
Version:	0.6.0-baidu
Running Workers:	104

Cluster Usage Summary

Workers Capacity:	1247.63 TB
Workers Free / Used:	1243.32 TB / 4407.93 GB
UnderFS Capacity:	UNKNOWN
UnderFS Free / Used:	UNKNOWN / UNKNOWN

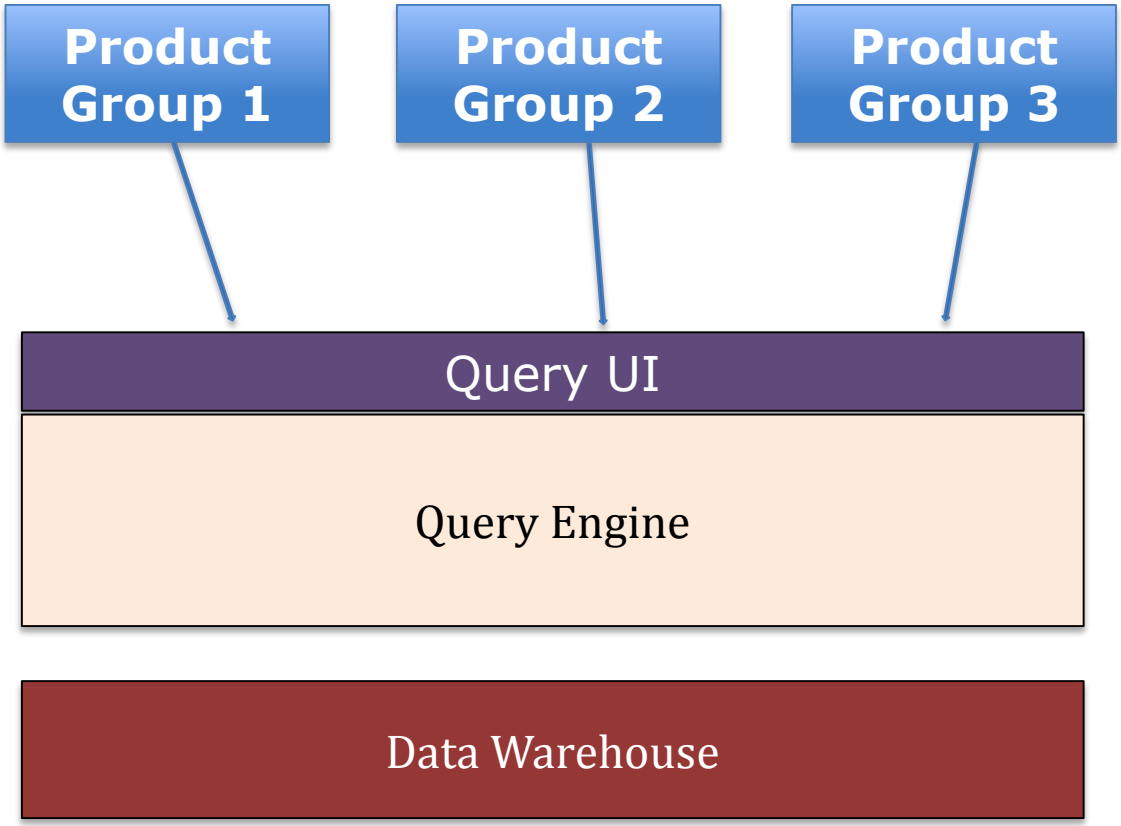
Storage Usage Summary

Storage Alias	Space Capacity	Space Used	Space Usage
MEM	1668.00 GB	1619.06 GB	97%Used
HDD	1246.00 TB	2788.87 GB	100%Free

~ 100 nodes in deployment, > 1 PB storage space

**感谢由百度US工程师Shaoshan Liu提供的一些数据和相关slides;*

Baidu US 即席查询系统简要架构

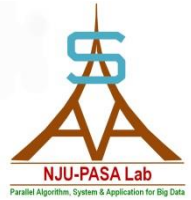


Sample Query Sequence:

```
SELECT  event_query,
COUNT(event_query) as cnt
FROM data_warehouse
WHERE event_day="20150528”
AND event_action="query_click"
GROUP BY event_query
ORDER BY cnt DESC
```

```
SELECT  event_province,
COUNT(event_query) as cnt
FROM data_warehouse
WHERE event_day="20150528”
AND event_action=“query_click”
AND event_query=“baidu stock"
GROUP BY event_province
ORDER BY cnt DESC
```

从Hive迁移至Spark SQL



Hive on MR

Hive

Map Reduce

4X Improvement but not good enough!

Spark SQL

Meta Store

HiveQL

UDFs

SerDes

Spark SQL

Apache Spark

Compute Center

Data Center

Data Warehouse

BFS

如何进一步提升性能？

- 问题：
  - 数据节点和计算节点可能不在同一个数据中心，因此数据存取的延时比较长
  - 这会导致即席查询的应用的性能急剧下降
- 解决方案：
  - 采用Tachyon作为一个对用户透明的缓存层
  - Cold query：从远处数据节点读取数据
  - Warm\hot query:直接从Tachyon上读取数据
  - 在百度最初有50个节点部署了Spark和Tachyon
    - 主要提供Spark SQL ad-hoc queries服务
    - 采用Tachyon作为透明缓存层

具有Cache层的交互式查询引擎

- 管理view的metadata
- 接受来自operation manager的请求:如果cache miss, 那么就通过从data warehouse读取数据并写入到Tachyon来建立新的view.

Query UI

Operation Manager

View Manager

Cache Meta

- 接受来自UI的查询；
- 采用Spark SQL进行查询解析和优化；
- 检测需要的数据是否在cache中，如果是则直接从Tachyon中读，否则启动一个Spark作业从Data Warehouse中读取数据；

Spark

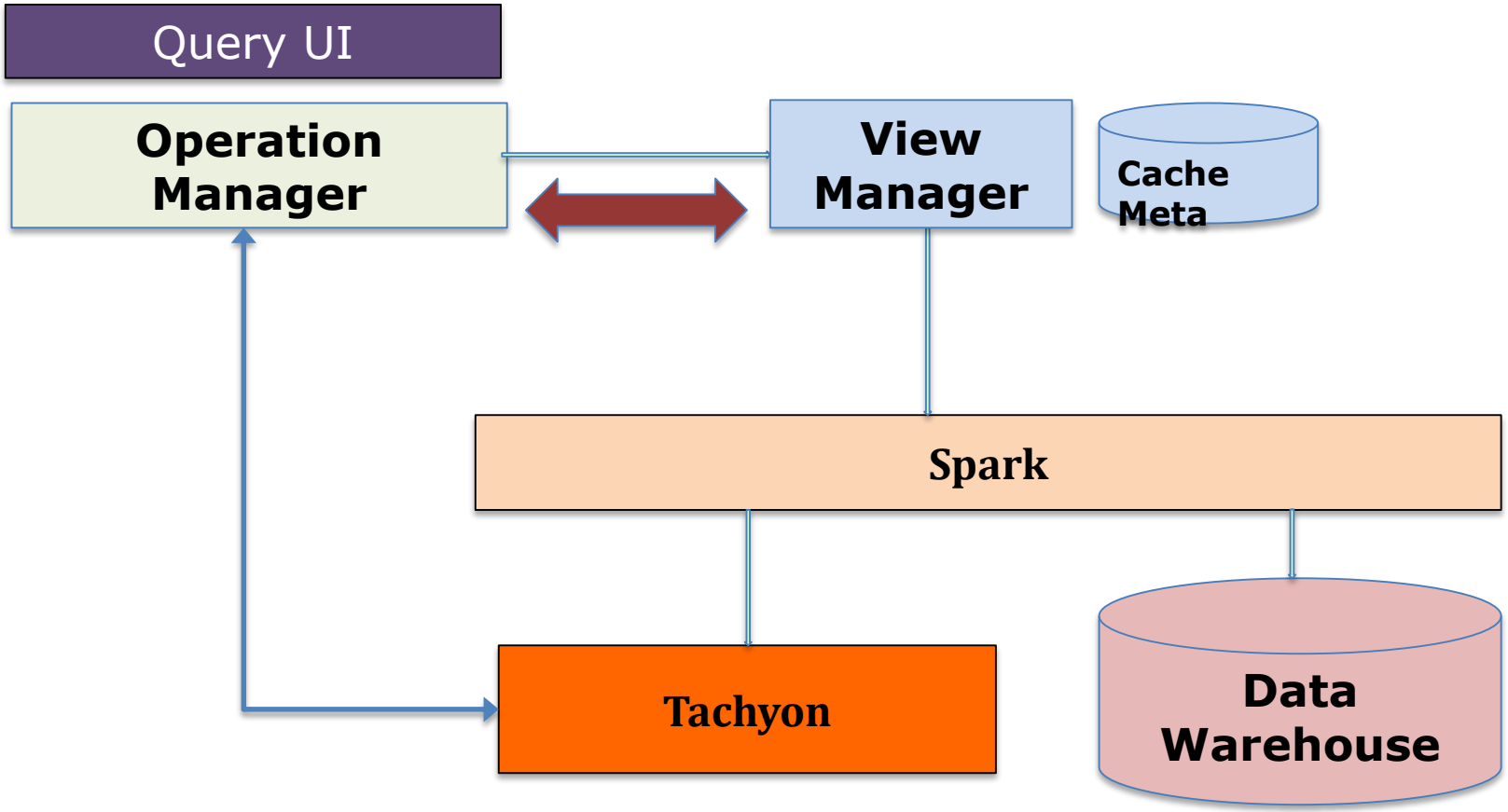
Tachyon

Data Warehouse

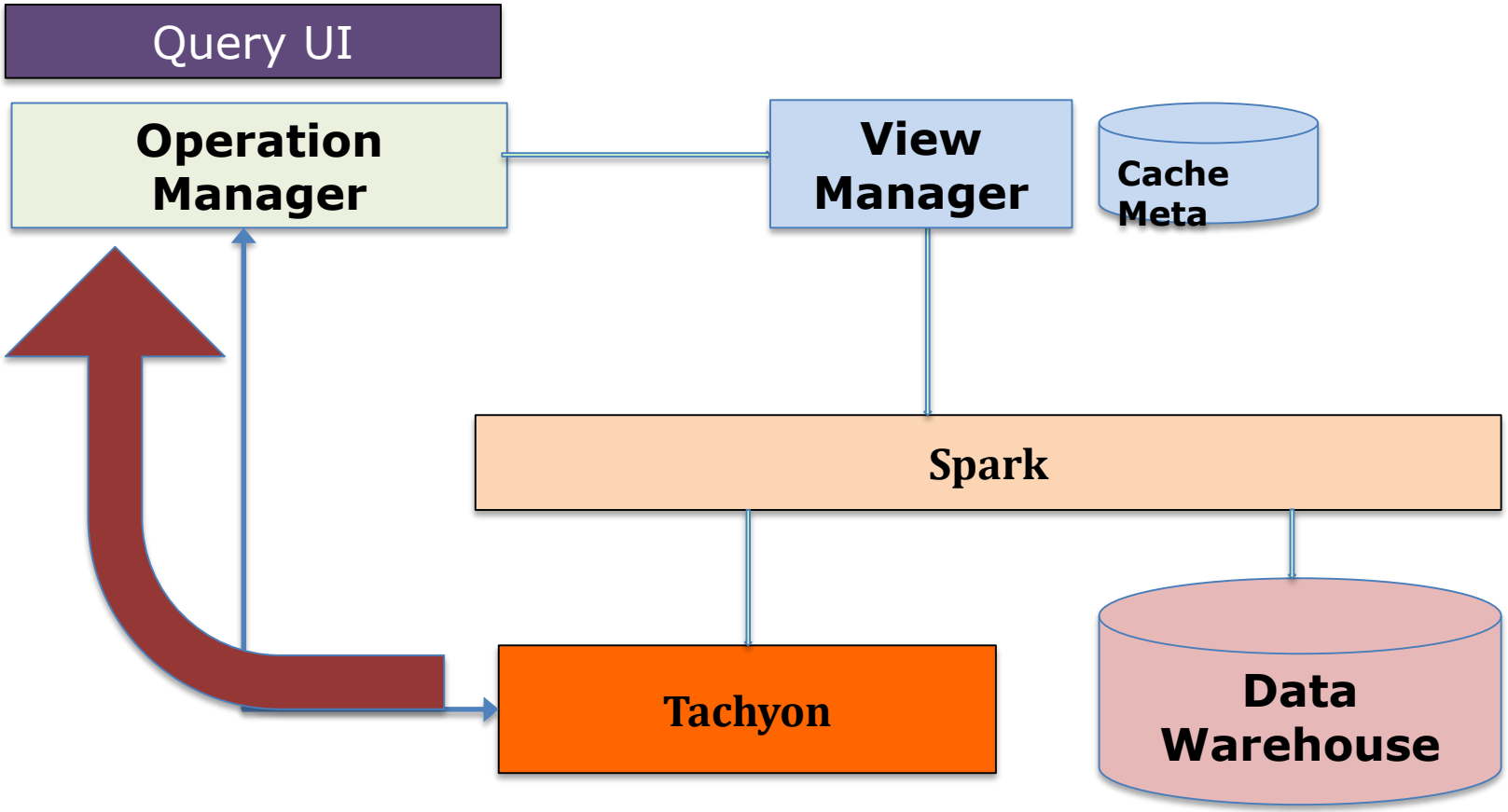
- View cache: 以视图为单位进行缓存，而不以原始数据块为单位缓存
- View结构：<table name, partition key, attributes, data>

- 采用基于HDFS的解决方案存储所有的原始数据；

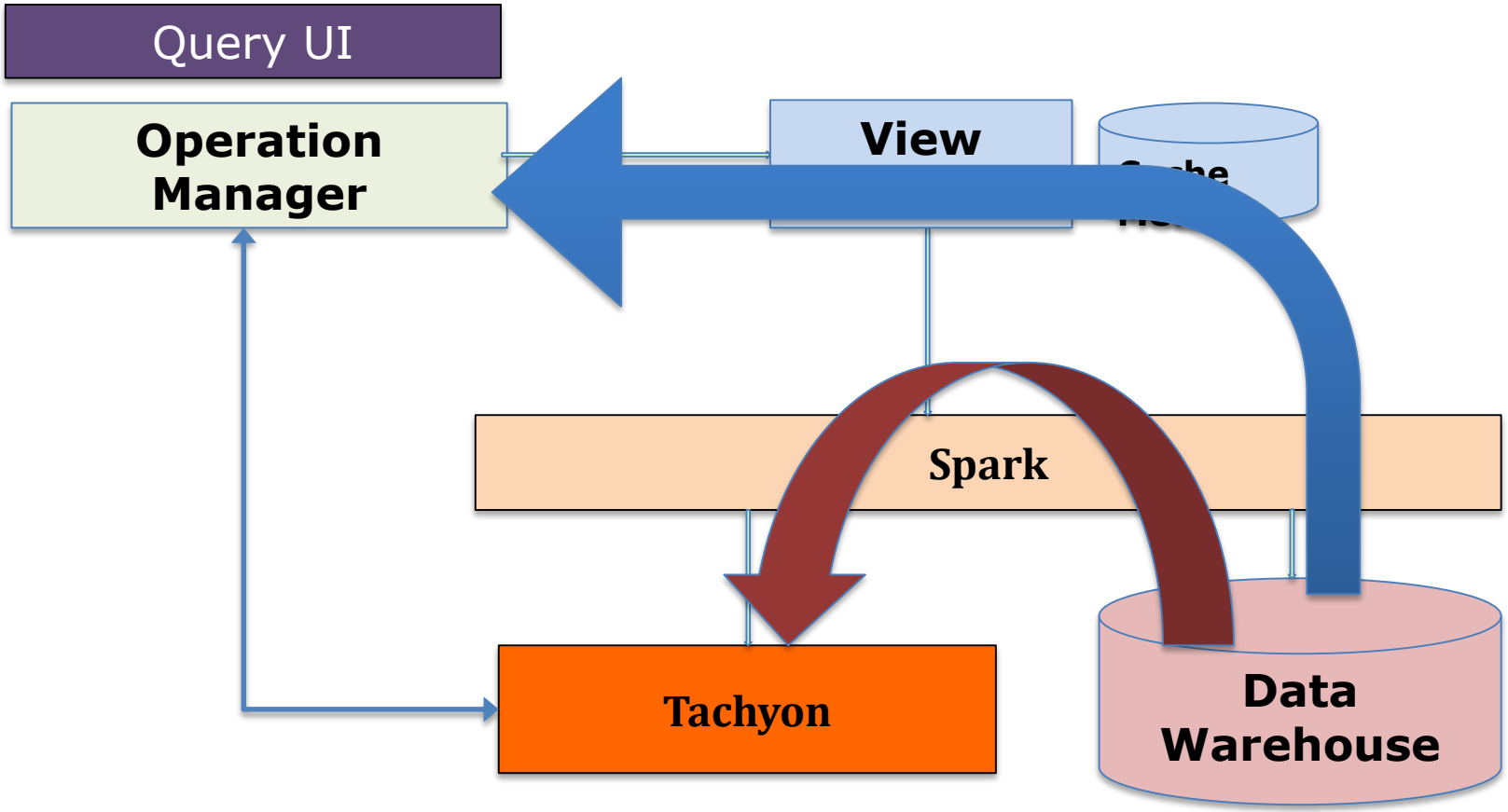
Query: Check Cache



Hot Query: Cache Hit



Cold Query: Cache Miss



示例

```
SELECT a.key * (2 + 3),
b.value
FROM T a JOIN T b
ON a.key=b.key AND a.key>3
```

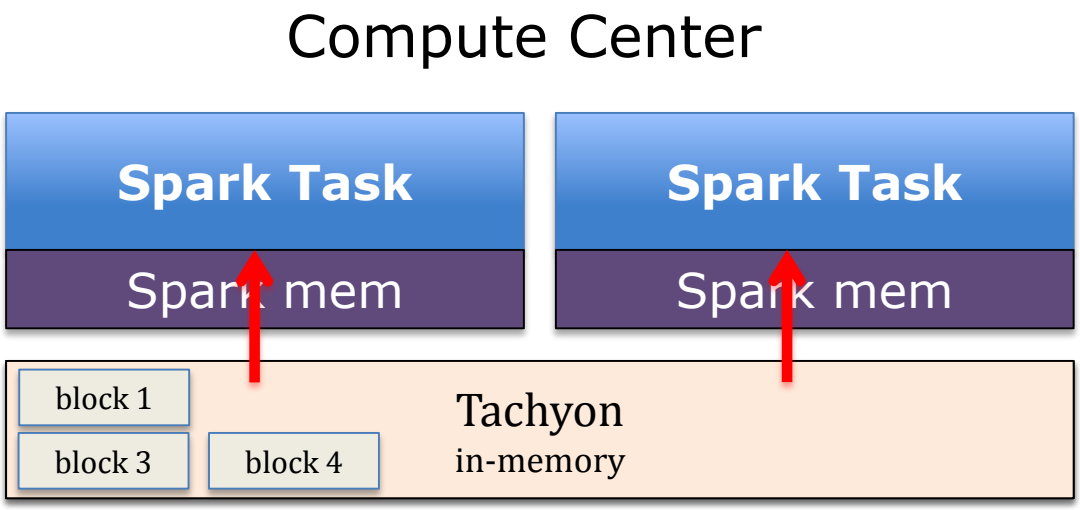
```
== Physical Plan ==
Project [(CAST(key#27, DoubleType) * 5.0) AS c_0#24,value#30]
 BroadcastHashJoin [key#27], [key#29], BuildLeft
  Filter (CAST(key#27, DoubleType) > 3.0)
   HiveTableScan [key#27], (MetastoreRelation default, T, Some(a)), None
  HiveTableScan [key#29,value#30], (MetastoreRelation default, T, Some(b)),
None
```

一旦我们得到了Spark SQL的physical plan, 我们解析其中的HiveTableScan部分，然而决定确定查询所需要的视图是否在Cache中；

Cache Hit: directly pull data from Tachyon

Cache Miss: get data from remote data storage

性能提升的情况



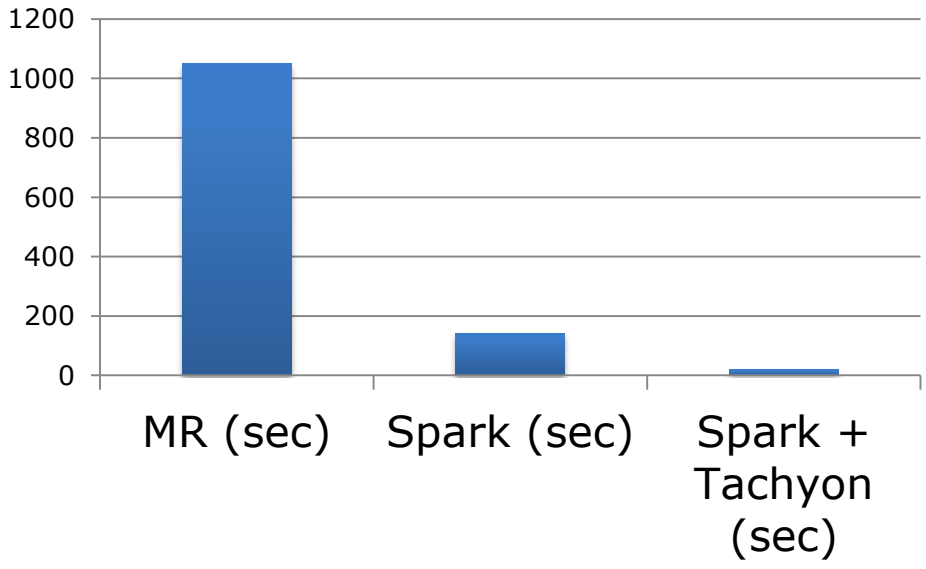
- Read from remote data center: ~ 100 ~ 150 seconds
- Read from Tachyon remote node: 10 ~ 15 sec
- Read from Tachyon local node: ~ 5 sec

Tachyon Brings 30X Speed-up !

Data Center

Baidu File System (BFS)

总体性能对比



Cases:

1. 采用 MR 查询6 TB的数据；
2. 采用 Spark 查询6 TB的数据；
3. 采用Spark + Tachyon 查询6 TB的数据；

Results:

1. Spark + Tachyon 相比于原先的 MR 提升了50-fold 的性能

Lessons Learned.

问题1: Failed to Cache Blocks

实验当中，我们发现**有些数据块无法缓存在Tachyon中，相同的查询不断地从远处的节点读取数据而不从Tachyon中读取**

```
@Override
public void close() throws IOException {
  if (mClosed) {
    return;
  }
  if (mRecache) {
    // We only finish re-caching if we've gotten to the end of the file
    if (mBlockPos == mBlockInfo.length) {
      mBlockOutStream.close();
    } else {
      mBlockOutStream.cancel();
    }
  }
  if (mCheckpointInputStream != null) {
    mCheckpointInputStream.close();
  }
  mClosed = true;
}
```

问题1：Failed to Cache Blocks

Before Change:

```
Time taken: 160.786 seconds
```

```
Time taken: 131.347 seconds
```

After Change:

```
Time taken: 5.9 seconds
```

```
Time taken: 5.891 seconds
```

原因： Tachyon 只会缓存那些整个block都被读过的。

解决方案： 如果想要cache某个block，就将其整个读下。

问题 2: Locality Problem

- DAGScheduler:
  - 当DAGScheduler安排tasks时，它会优先将任务分配给那些拥有待处理的数据的节点，从而避免网络传输数据以获得很好的执行性能。

- 此外, master 会认为其是local (no remote fetch needed)

Locality Level
NODE_LOCAL
NODE_LOCAL

问题 2: Locality Problem

- 然而, 我们还是观察到了大量的网络传输:

```
**** FileInStream: read: new block index is 5***** BlockInStream: file /app/tpcds/store_sales_small/store_sales
**** RemoteBlockInStream : block idx 5  mBlockInfo.blockId 149250113541
**** RemoteBlockInStream : block isCache() block idx 5  mBlockInfo.blockId 149250113541
```

- 效果:
  - We expect the Tachyon **cache hit rate is 100%**
  - We end up with **33% cache hit rate**

原因: 我们用的是一个过旧的InputFormat

解决方案: 更新InputFormat

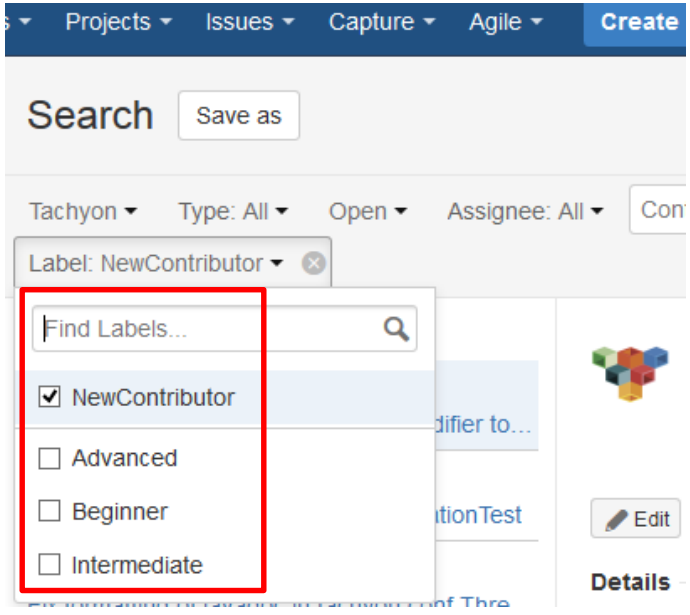
欢迎加入Tachyon社区！

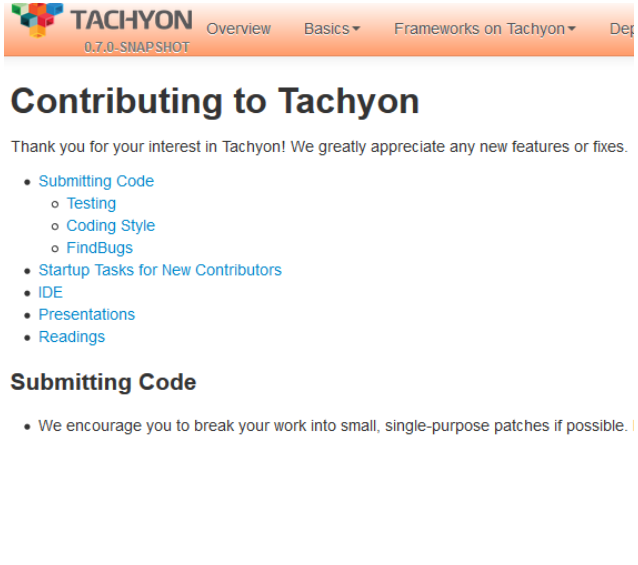
- For Users

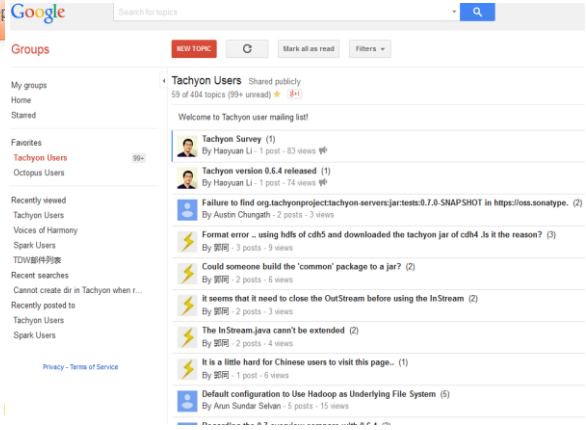
https://groups.google.com/forum/?fromgroups#!forum/tachyon-users

- For Developers

http://tachyon-project.org/master/Contributing-to-Tachyon.html
https://tachyon.atlassian.net/secure/Dashboard.jspa







欢迎关注我们的Tachyon中文博客！

Email:gurongwalker@gmail.com

微博：南京大学_顾荣

Thanks.