DTCC
2017第八届中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2017

如何基于Alluxio提升Spark和Hadoop HDFS的数据访问性能与系统稳定性

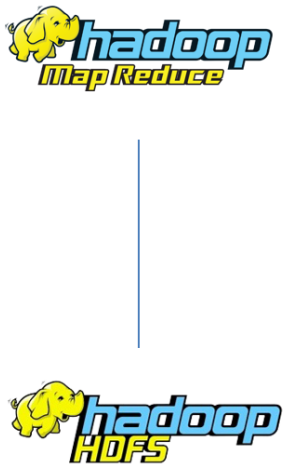
顾 荣 博士
南京大学 计算机系 助理研究员,
Alluxio项目PMC, Maintainer

2017/05/13@DTCC 2017(北京)

内容

- **Alluxio应用场景分析与1.4版本新特性介绍**
- 基于Alluxio的Spark DataFrame/RDD性能调优
- 基于Alluxio提升HDFS集群的性能和SLA稳定性
- 总结

BIG DATA ECOSYSTEM Yesterday

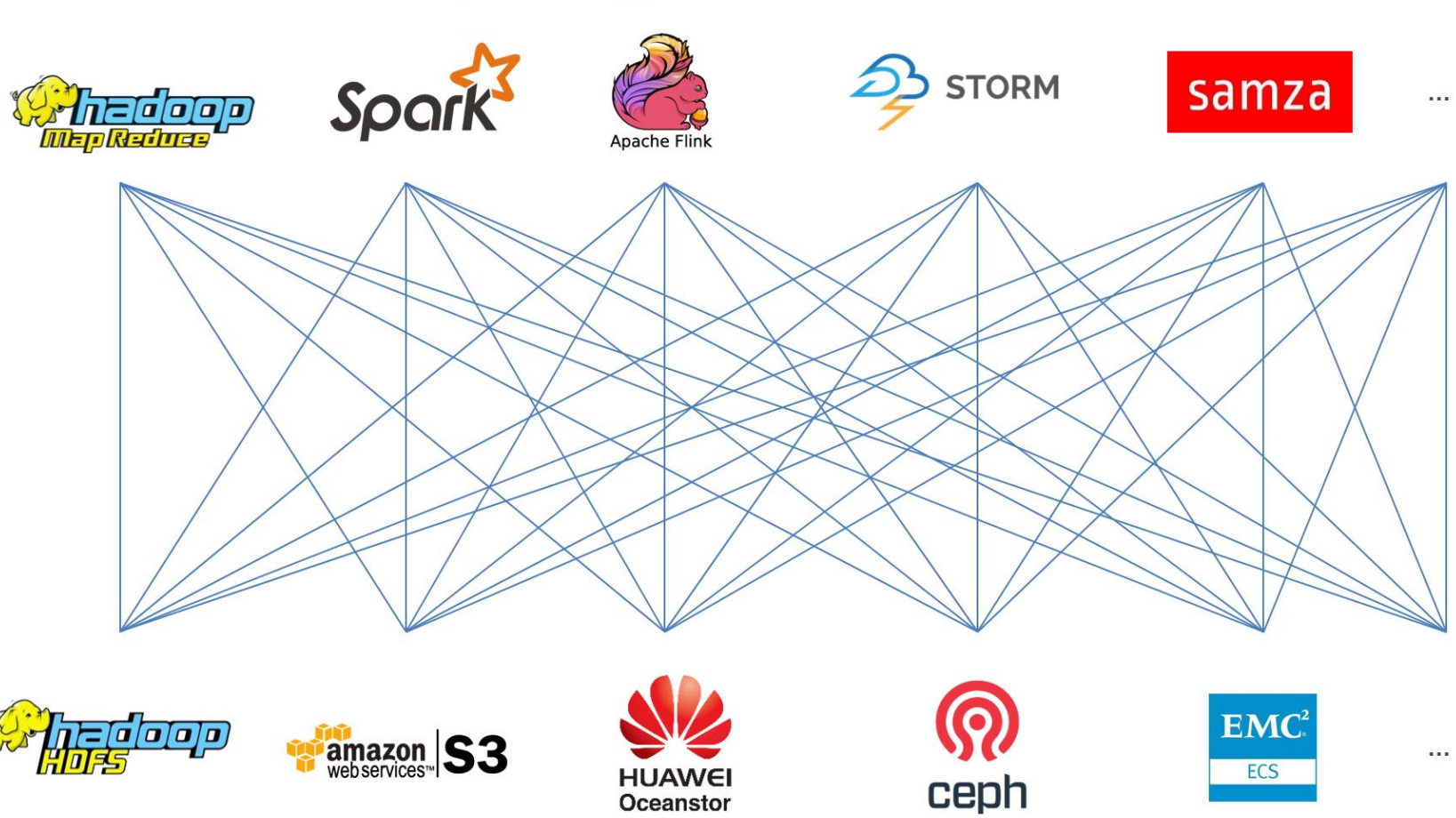


BIG DATA ECOSYSTEM Today

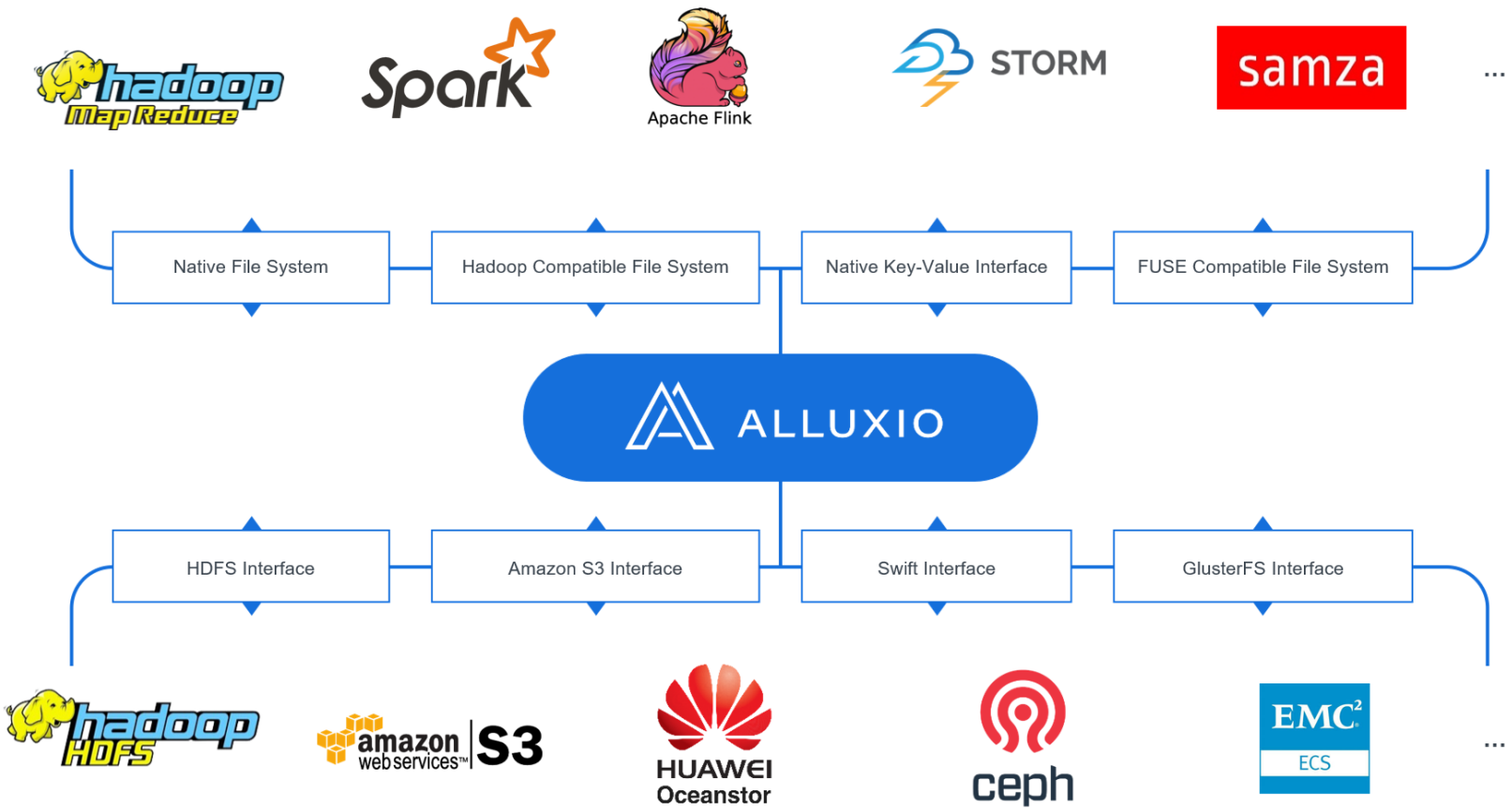
...

...

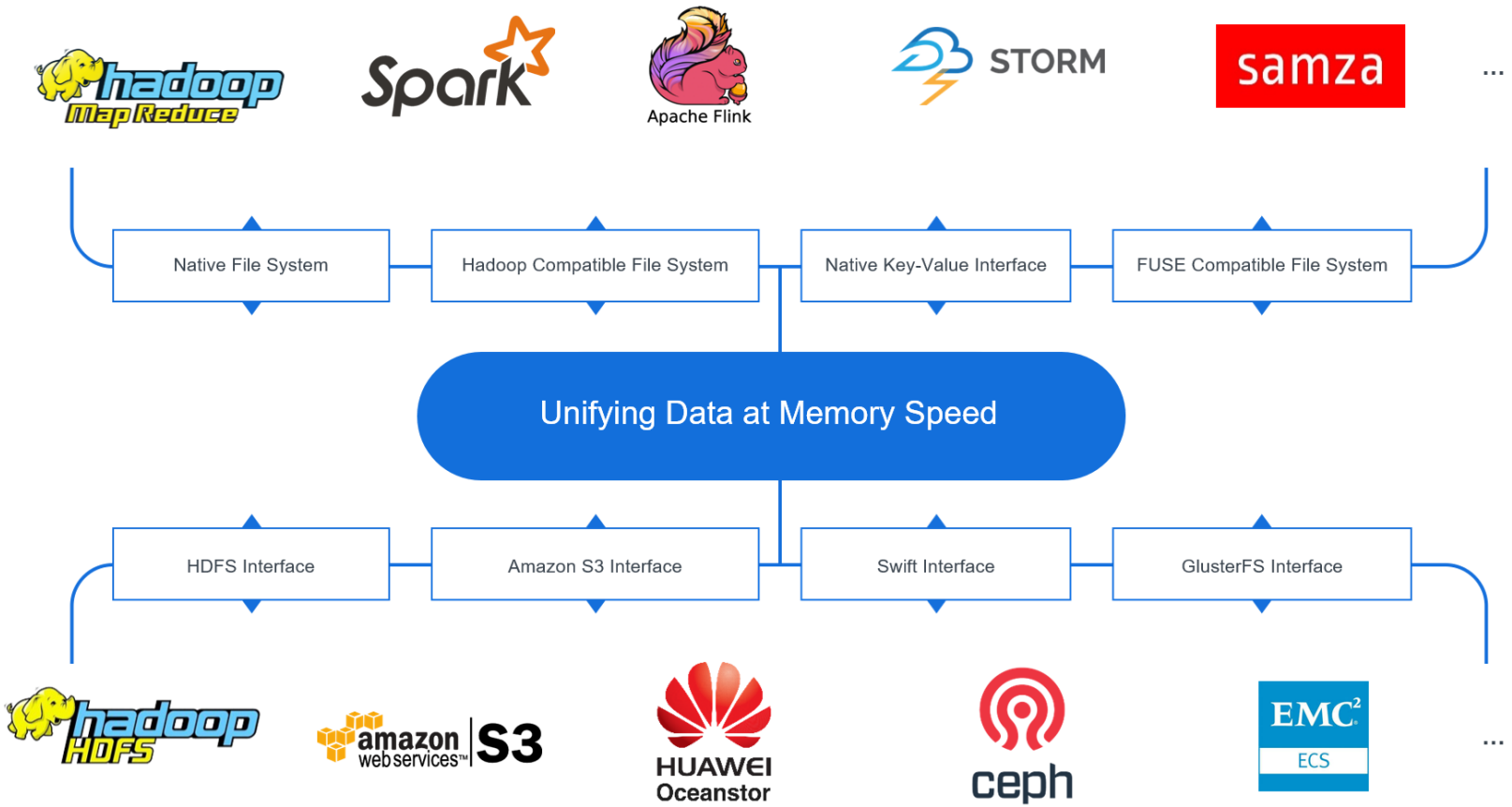
BIG DATA ECOSYSTEM Issue



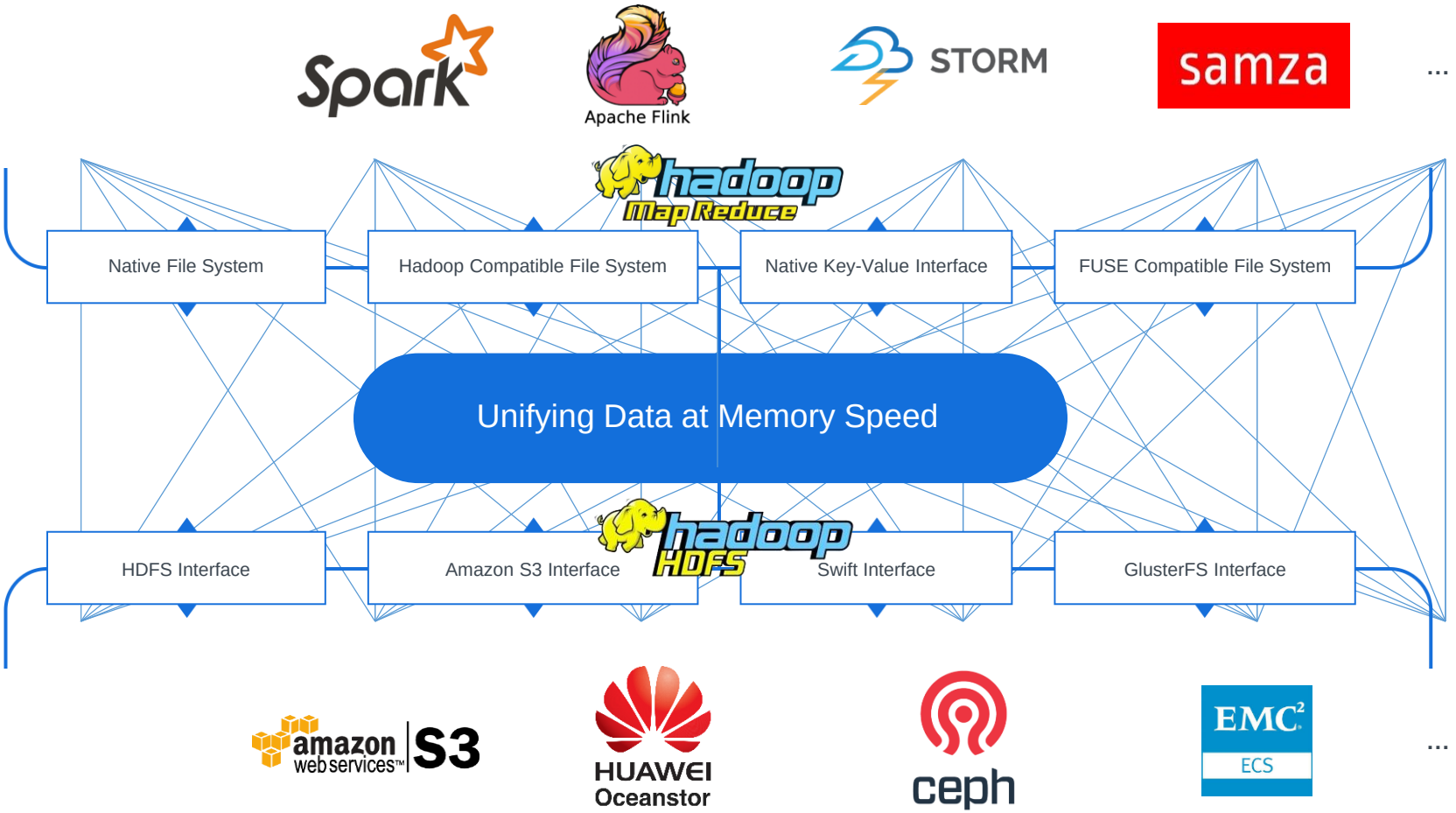
BIG DATA ECOSYSTEM With Alluxio



BIG DATA ECOSYSTEM With Alluxio



BIG DATA ECOSYSTEM [redacted]



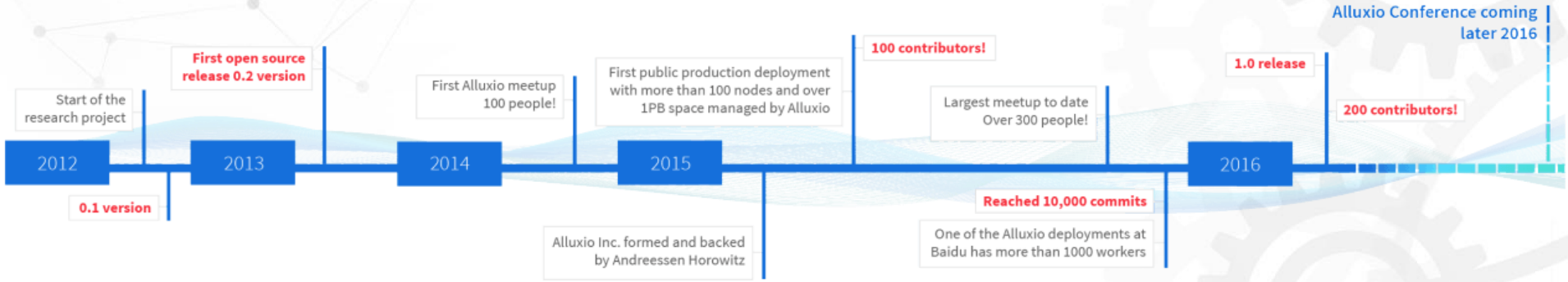
Alluxio是什么

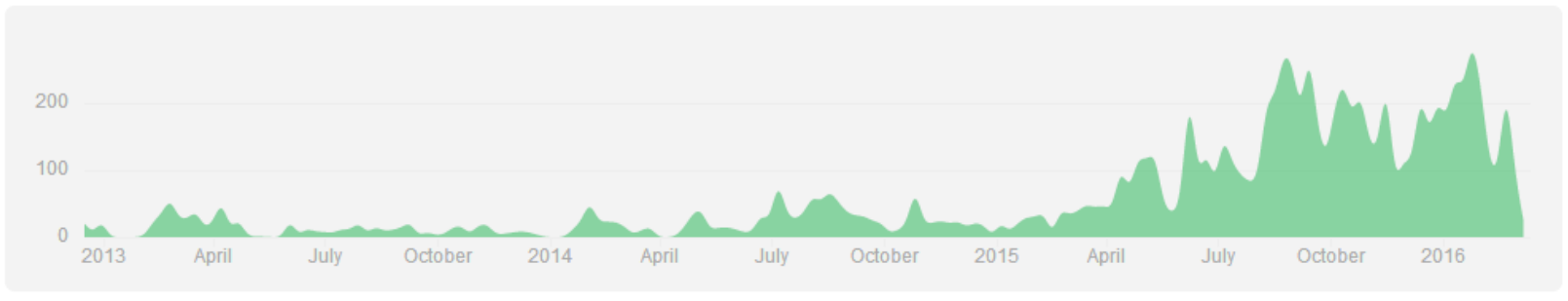
ALLUXIO

- Alluxio是世界上第一个以内存为中心(memory-centric)的虚拟的分布式存储系统。
- Alluxio介于计算框架和现有的存储系统之间，为大数据软件栈带来了显著的性能提升。

Alluxio的发展

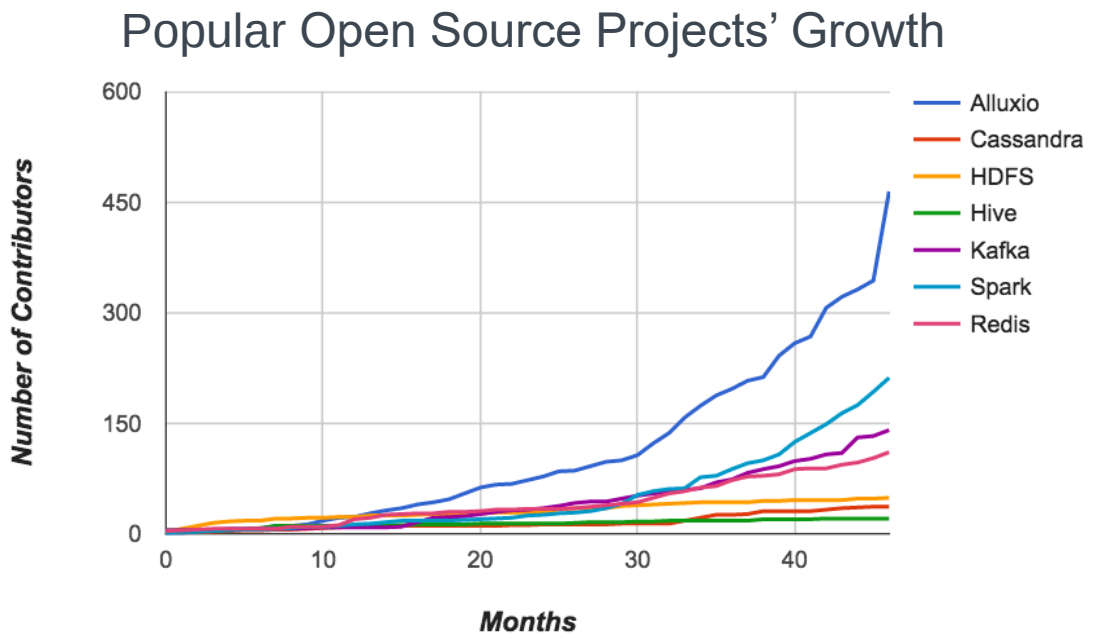
- 2012年12月，Alluxio（Tachyon）发布了第一个版本0.1.0
- 2017年1月，Alluxio的最新发布版本为1.4



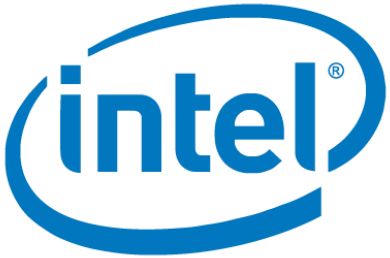


Alluxio的发展

- 自2013年4月开源以来，已有超过100个组织机构的500多贡献者参与到Alluxio的开发中。包括阿里巴巴，Alluxio，百度，卡内基梅隆大学，IBM，Intel，南京大学，Red Hat，UC Berkeley和Yahoo。
- 活跃的开源社区



INDUSTRY ADOPTION



The Register®
Biting the hand that feeds IT

Data Center ▸ Storage

Multi-silo data-sucker Alluxio inks deal with Dell EMC

EMC² ECS

Follows startup's agreement with Huawei

HUAWEI

Huawei and Alluxio Jointly Release Big Data Storage Acceleration Solution, Speeding Up Big Data Application Popularization

DATA + FOLLOW THIS TOPIC

8 data trends on our radar for 2017

From deep learning to decoupling, here are the data trends to watch in the year ahead.

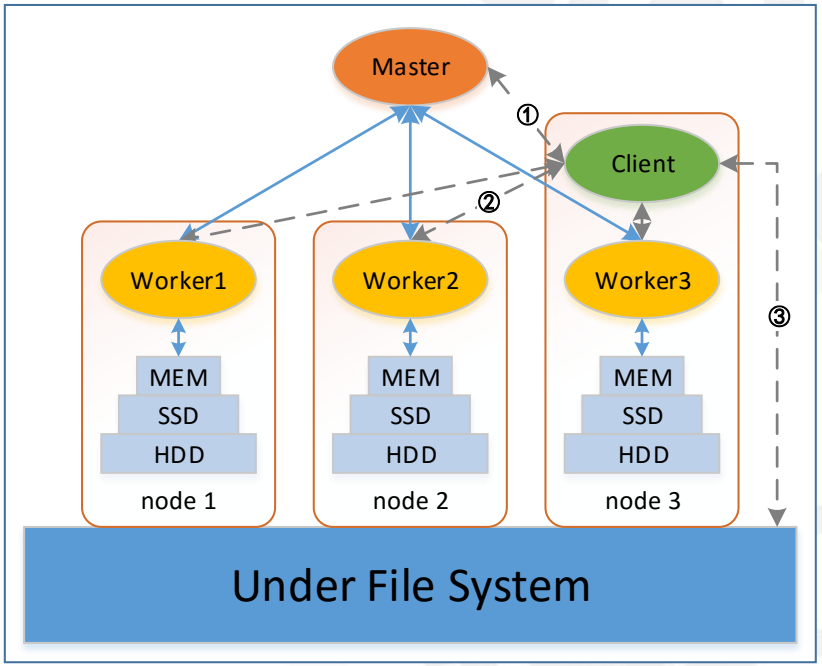
By Ben Lorica, January 3, 2017

6. The decoupling of storage and computation will accelerate.

The UC Berkeley AMPLab project ended last November, but the team behind Apache Spark and Alluxio are far from the only ones to highlight the separation of storage and computation. As noted above, popular object stores in the cloud and even some recent deep learning architectures emphasize this pattern.

Alluxio整体架构

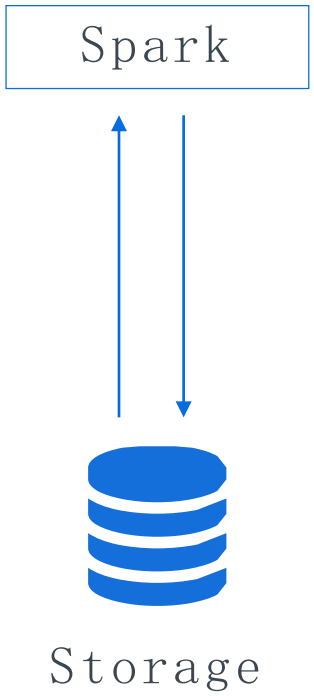
- Master-Worker
  - Master
    - 管理全部元数据
    - 监控各个Worker状态
  - Worker
    - 管理本地MEM、SSD和HDD
- Client
  - 向用户和应用提供访问接口
  - 向Master和Worker发送请求
- Under File System
  - 用于备份



应用场景1：加速远程存储的I/O访问

- 场景: 计算和存储分离
  - 可以满足不同的计算和存储硬件要求
  - 能够独立地扩展计算层和存储层
  - 将数据存储在传统的文件系统/SAN和对象存储中
  - 通过大数据计算框架分析现有数据
- 限制：
  - 访问数据需要远程I/O

不使用ALLUXIO访问远程存储

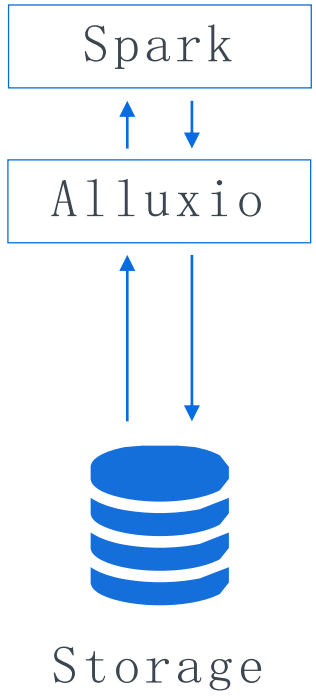


Low latency, memory throughput

High latency, network throughput

使用ALLUXIO访问远程存储

将数据放在Alluxio中以
加速数据访问



实际案例：百度

Baidu的项目经理和分析师需要运行交互式的查询以获得关于他们产品和商业的insights

- 200+ nodes deployment
- 2+ petabytes of storage
- Mix of memory + HDD

Baidu 百度

> *The performance was amazing. With Spark SQL alone, it took 100-150 seconds to finish a query; using Alluxio, where data may hit local or remote Alluxio nodes, it took 10-15 seconds.*
>
> - Baidu

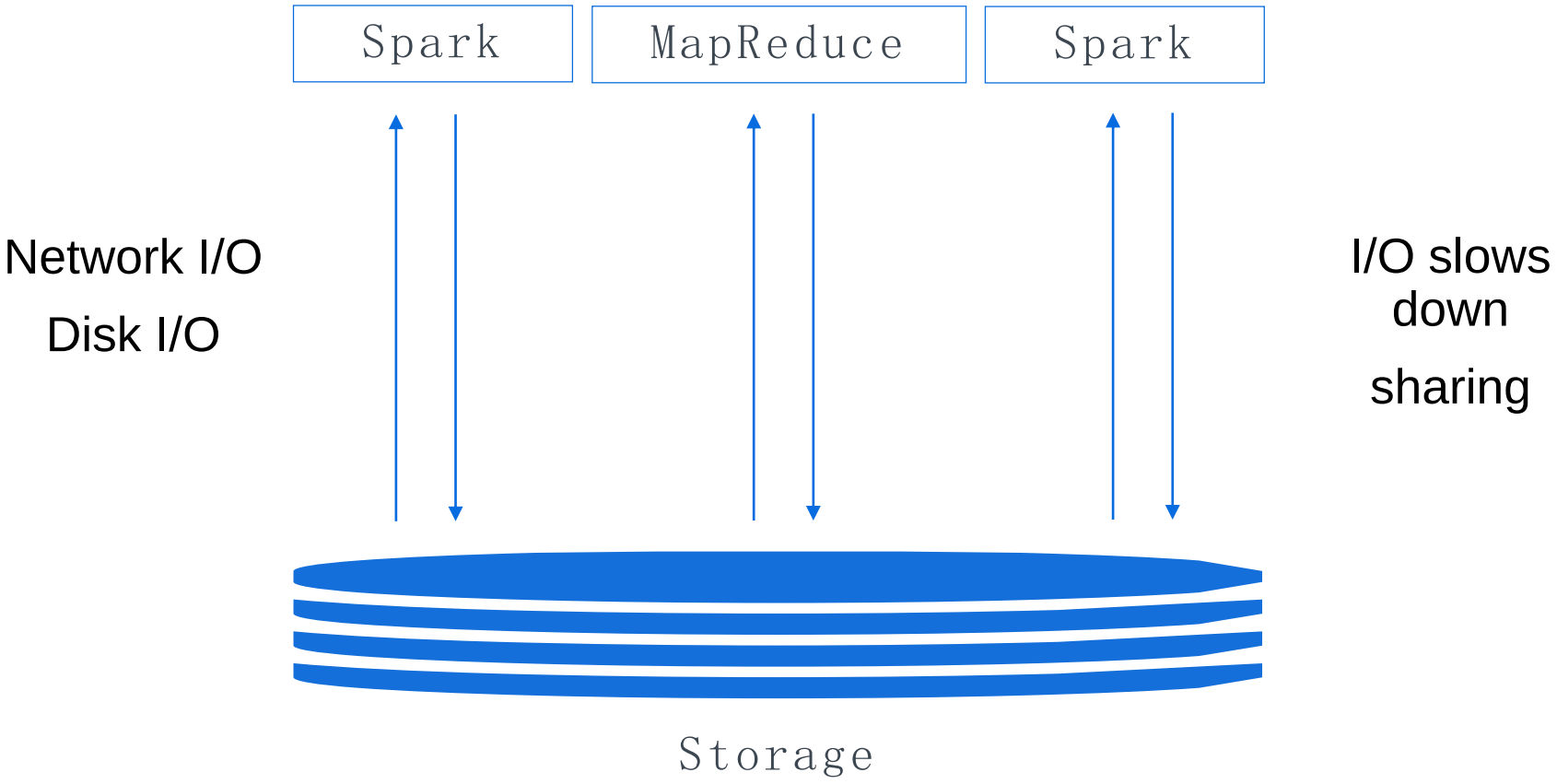
结果:

- 通过使用Alluxio，性能加速了30x
- Alluxio集群运行稳定，并管理着50TB的集群内存空间
- 通过使用Alluxio, 原先需要15分钟才能执行完的批处理查询，转换为了可以30秒内可以处理完的交互式查询

应用场景2：不同应用以内存速度共享数据

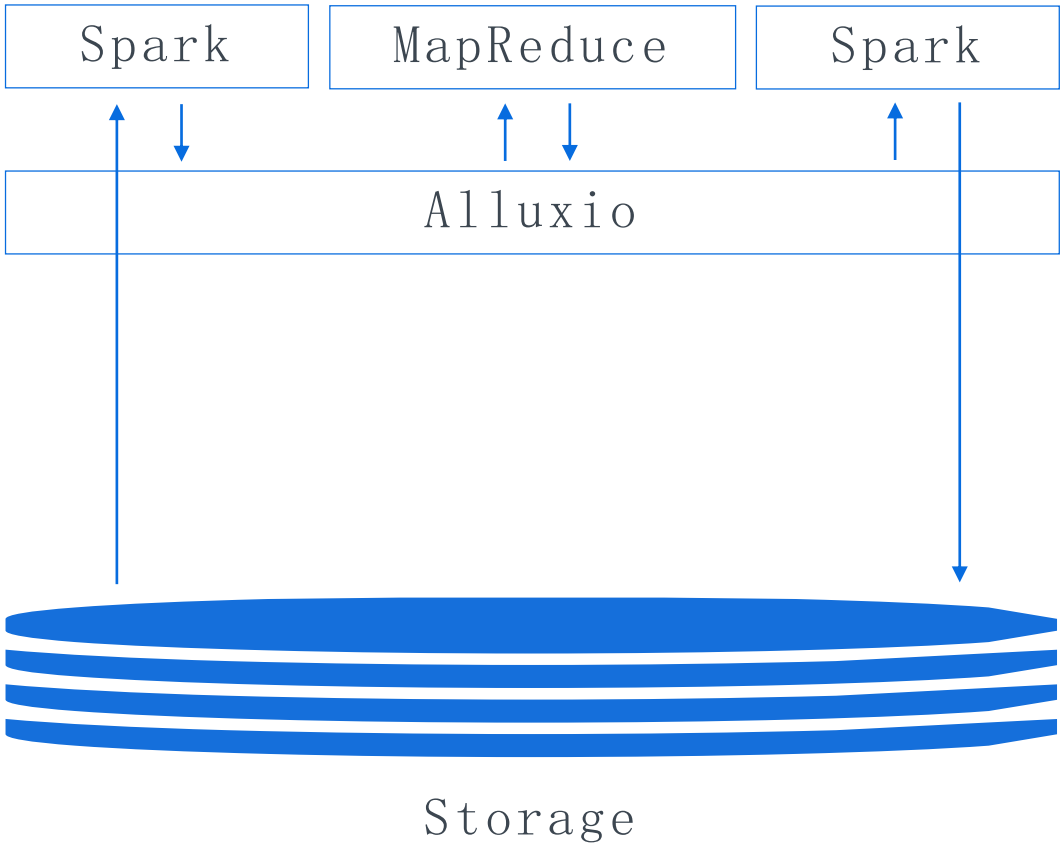
- 场景: 数据共享式架构
  - 流水线: 一个作业的输出是下一个作业的输入
  - 应用、作业、上下文（ contexts ）读取相同的数据
- 限制：
  - 共享数据需要I/O

不使用ALLUXIO进行数据共享



使用ALLUXIO进行数据共享

Memory-speed sharing



实际案例：巴克莱银行（BARCLAYS）

巴克莱银行通过查询抽取数据，并运行机器学习算法来训练风控模型

- 6 node deployment
- 1TB of storage
- Memory only

BARCLAYS

> *Thanks to Alluxio, we now have the raw data immediately available at every iteration and we can skip the costs of loading in terms of time waiting, network traffic, and RDBMS activity.*
>
> - Barclays

结果

- 巴克莱银行的处理流程的每轮的迭代时间从几小时降低至几秒钟
- Alluxio使得之前在给定时间内很难运行的工作变得可能
- 通过将数据缓存在Alluxio内存中，加载和存储的I/O开销变成只有几秒钟

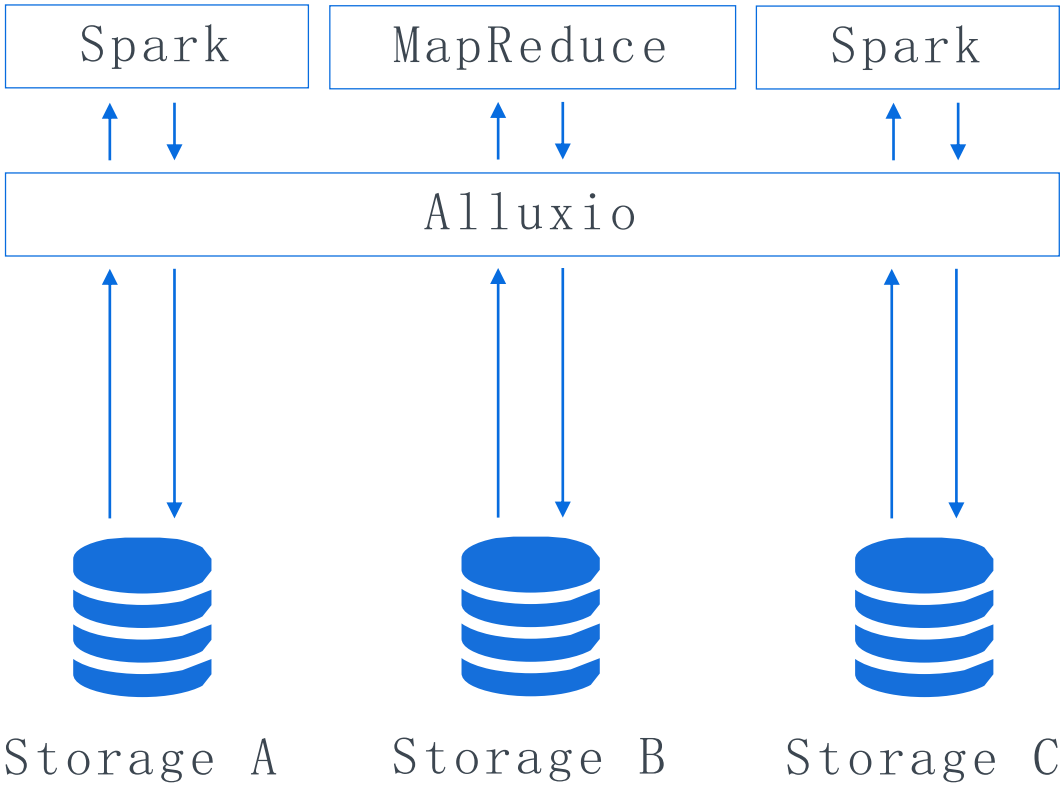
应用场景3：统一不同存储的数据访问

- Scenario: Multiple Storage Systems
  - Most enterprises have multiple storage systems
  - New (better, faster, cheaper) storage systems arise
- Limitation
  - Accessing data from different systems requires different APIs

使用ALLUXIO访问数据

Flexible,

simple



no application changes,

new mount point

实际案例：去哪儿网

去哪儿网使用实时机器学习算法服务于他们的线上广告

- 200+ nodes deployment
- 6 billion logs (4.5 TB) daily
- Mix of Memory + HDD

去哪儿? Qunar.Com 聪 明 你 的 旅 行

> *We've been running Alluxio in production for over 9 months, Alluxio's unified namespace enable different applications and frameworks to easily interact with data from different storage systems.*
>
> - Qunar

结果

- 通过使用Alluxio，在Spark流处理、Spark批处理以及Flink作业直接的数据共享变得高效
- 系统提升了大约15 – 300倍
- 采用的Alluxio层次化存储功能，对集群的存储资源（内存、HDD）进行了分层管理

Alluxio 1.4版本重要特性介绍

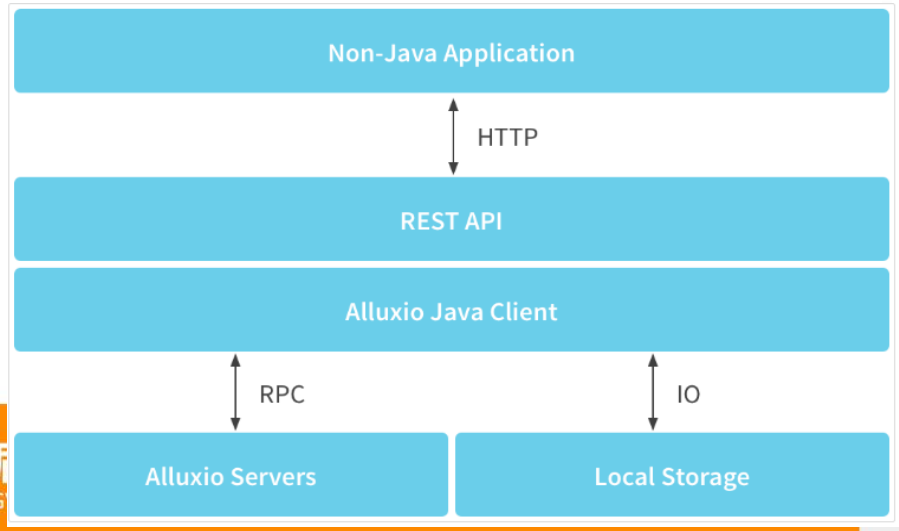
- 优化Alluxio底层对象存储API
  - 优化的对象存储连接器
    - Alluxio 1.4.0在对象存储上的小文件和小规模读取的性能方面有了重大改进。改进UFS API提升了对象存储中获取元数据的性能。

	创建	删除
S3A	5 倍提升，与 v1.3 比较	3 倍提升，与 v1.3 比较
Ceph	10 倍提升，与 v1.3 比较	15 倍提升，与 v1.3 比较

  - 简化的对象存储集成
    - 如今集成新的对象存储只需要原先实现的方法中的一半即可。
    - 在源代码行数方面，现在仅需要不到400行代码即可完成新的对象存储的集成，不到原来的一半。

Alluxio 1.4版本重要特性介绍

- 文件系统原生REST接口
  - 新引入的REST接口提供了同Alluxio native Java API的对等性，其目的是促进非Java环境与Alluxio的交互。
    - REST接口通过Alluxio代理进程实现，该进程使用一个内部Alluxio Java客户端代理REST接口和Alluxio服务器之间的通信
    - 为了获得最佳性能，推荐将Alluxio代理与Alluxio服务进程放在一起。这可以让非java应用以内存级的速度访问Alluxio中的数据，同时最小化Alluxio代理和Alluxio服务之间额外网络的开销。



Alluxio 1.4版本重要特性介绍

- 数据包流（Packet Streaming）
  - Alluxio 1.4.0引入了一种新的网络传输协议，旨在充分利用Alluxio组件之间的可用网络带宽
    - 在标准网络中高达2倍的IO性能改进，以及在高延迟吞吐量产品环境下取得更好的结果
  - 减少了网络传输期间使用的缓存，并且依赖于使用连续流传输协议取代数据传输中的请求-响应协议
    - 通过使用这种方法，能确保网络管道持续饱和，因为我们不需要发送周期性的额外数据请求

Alluxio 1.4版本重要特性介绍

- 支持Apache Hive（Contributed By PASALab）
  - 将Apache Hive集成运行在Alluxio上,具体看使用文档(http://www.alluxio.org/docs/master/en/Running-Hive-with-Alluxio.html)

- 提升了与基于YARN的应用的集成工作
  - 支持在用户/权限打开的模式下将YARN的应用运行在Alluxio上

Alluxio 1.4版本重要特性介绍

- 提升了Alluxio Master中的锁管理机制
  - 使得基于MapReduce的计算框架能够有更高并发的元数据操作性能
  - 对缓慢底层文件系统进行元数据操作（例如在云存储中进行重命名）的性能能够提升1个数量级左右
- 指定层级写操作
  - 用户能够将数据显式地写到Alluxio存储层级中的指定层，而不是原先默认的最顶层

内容

- Alluxio应用场景分析与1.4版本新特性介绍
- **基于Alluxio的Spark DataFrame/RDD性能调优**
- 基于Alluxio提升HDFS集群的性能和SLA稳定性
- 总结

实验设置

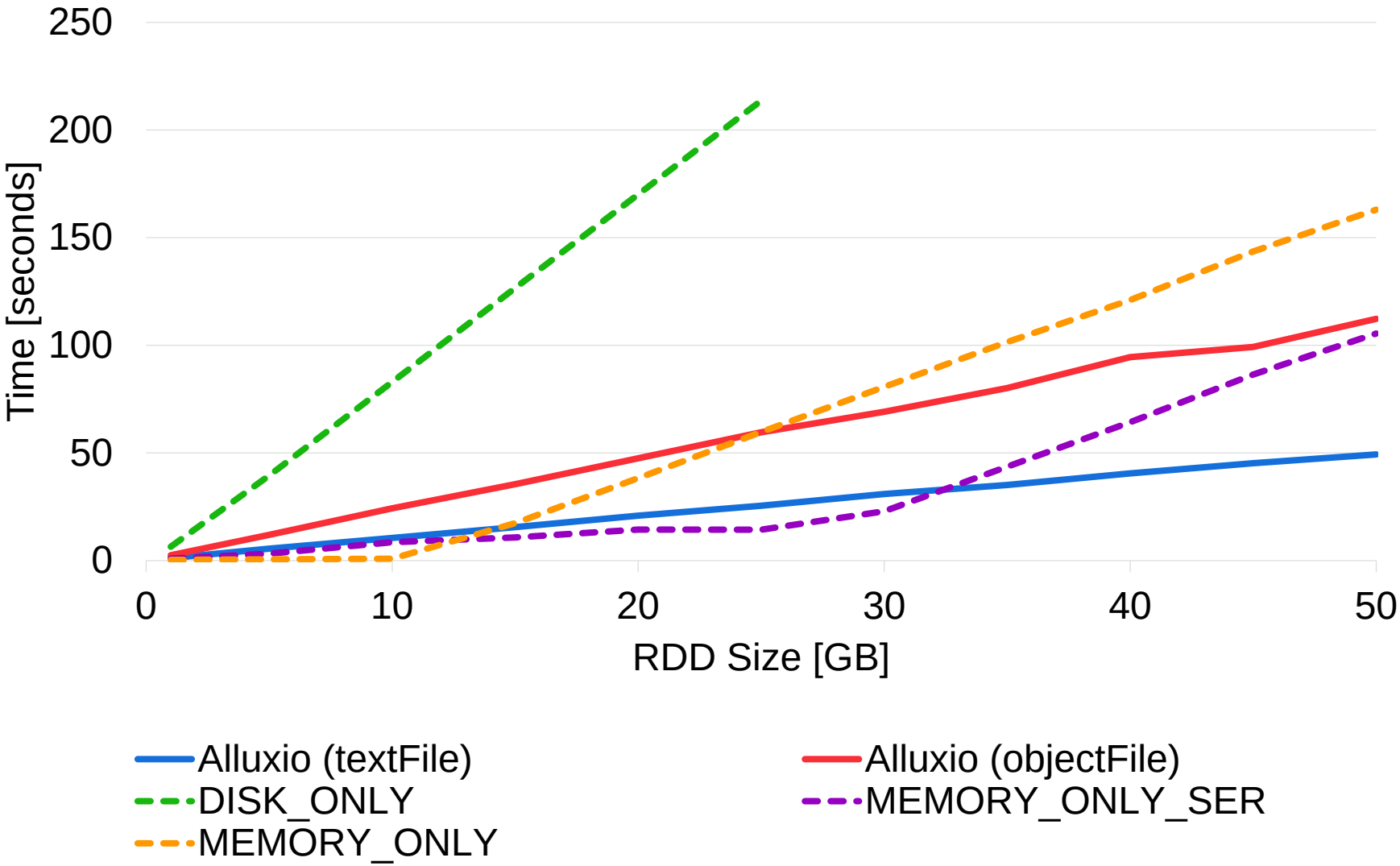
Spark 2.0.0 + Alluxio 1.2.0

Single worker: Amazon r3.2xlarge（61 GB MEM, 8-core CPU）

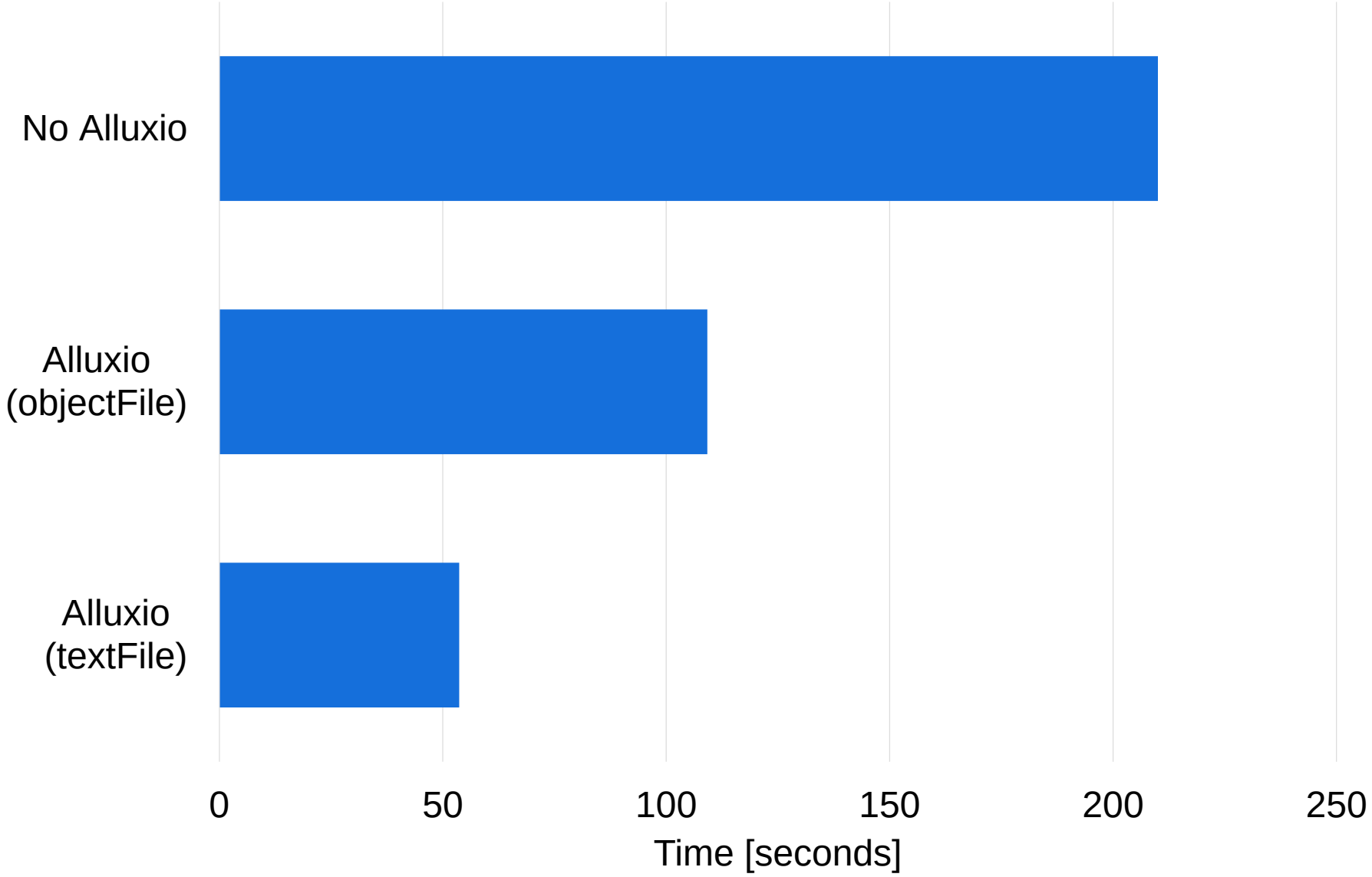
Comparisons:

- Alluxio
- Spark Storage Level: MEMORY_ONLY
- Spark Storage Level: MEMORY_ONLY_SER
- Spark Storage Level: DISK_ONLY

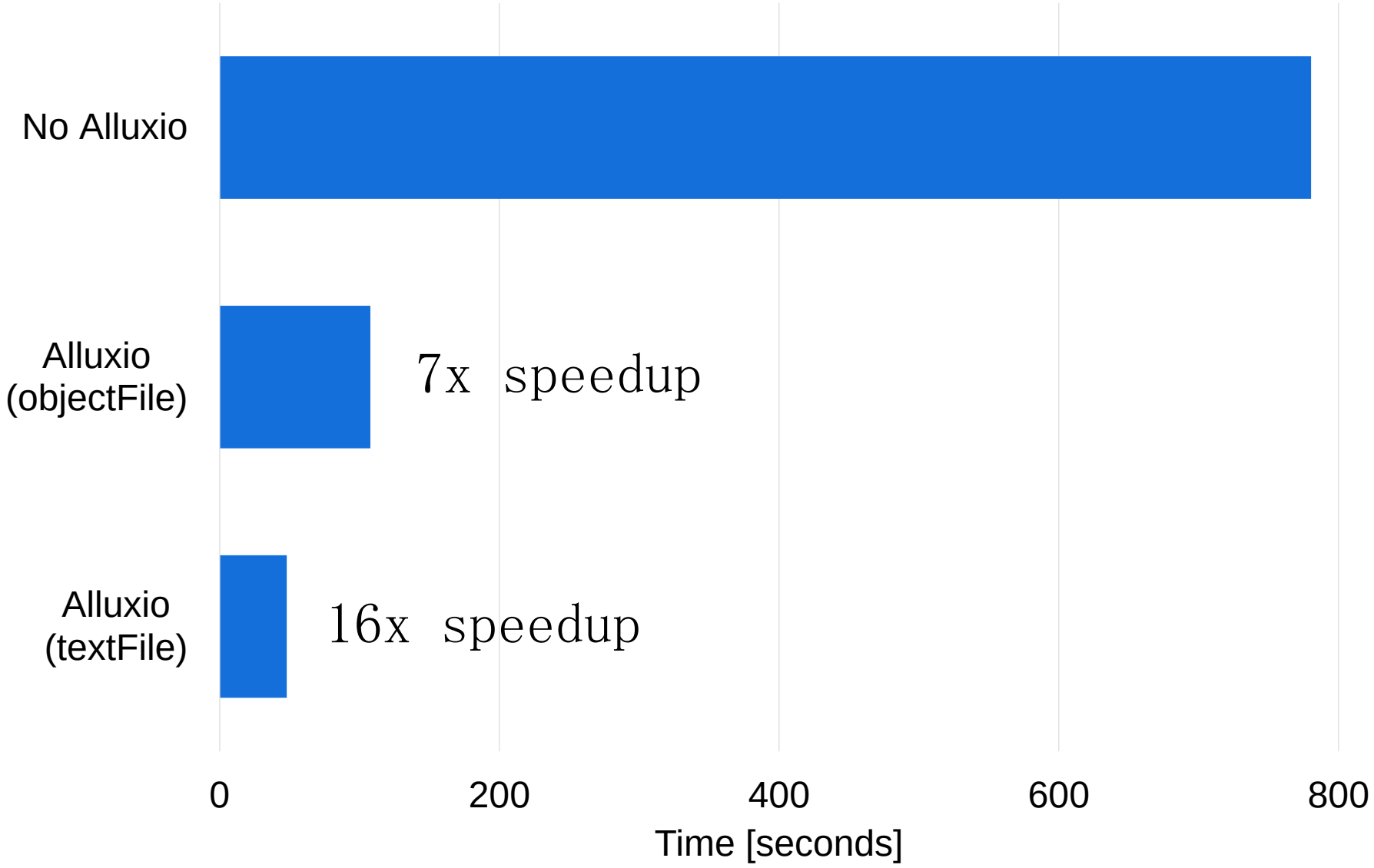
READING CACHED RDD



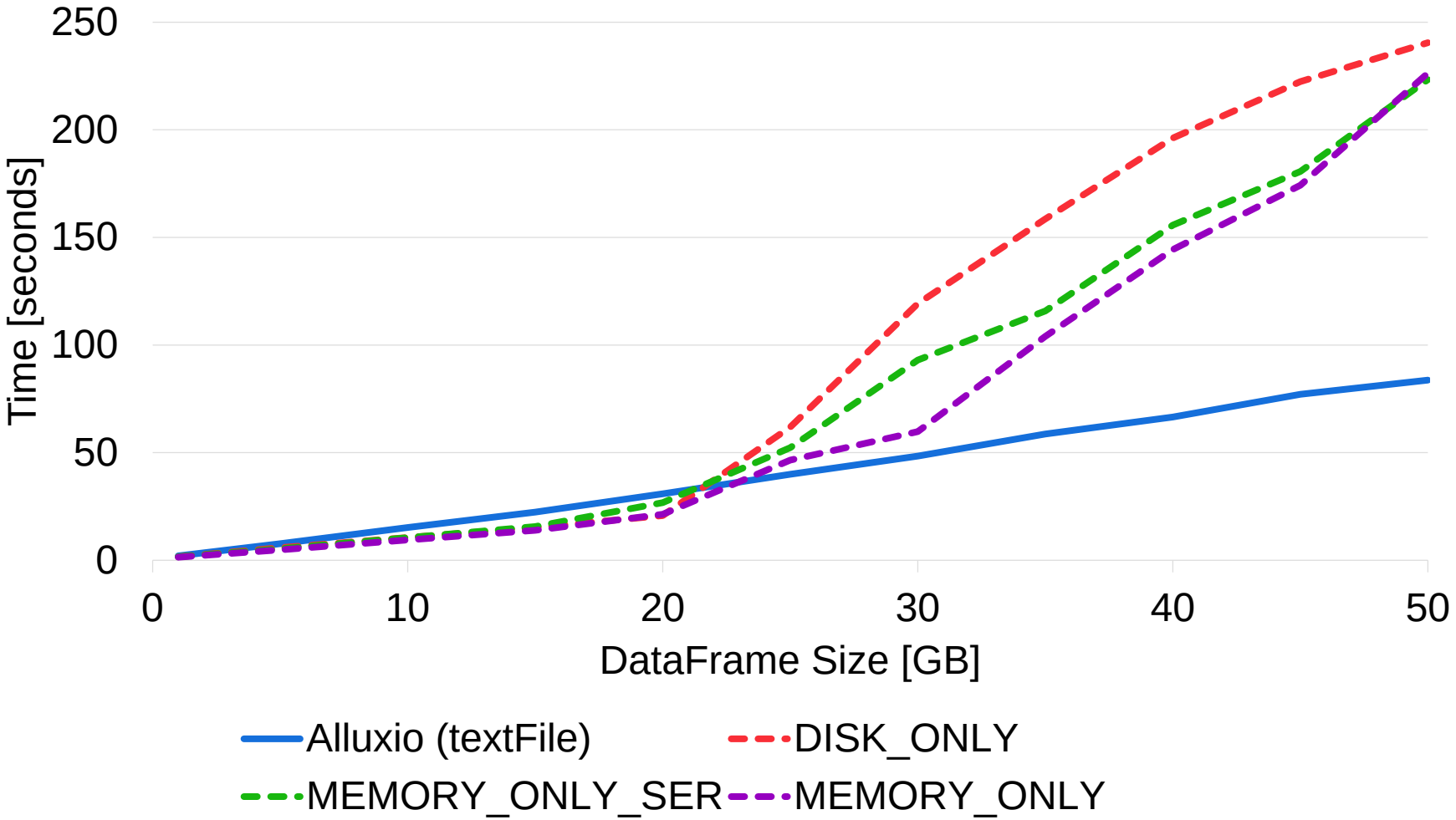
新场景: READ 50 GB RDD (SSD)



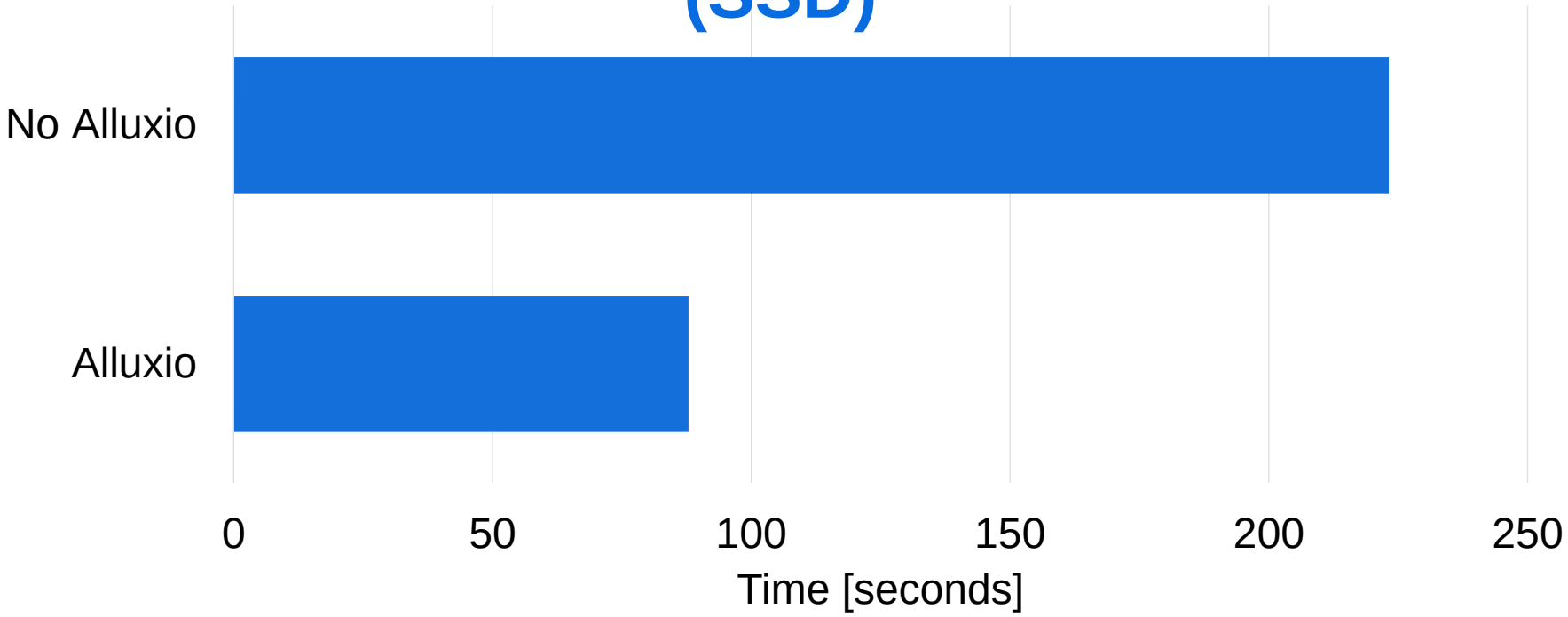
新场景：READ 50 GB RDD (S3)



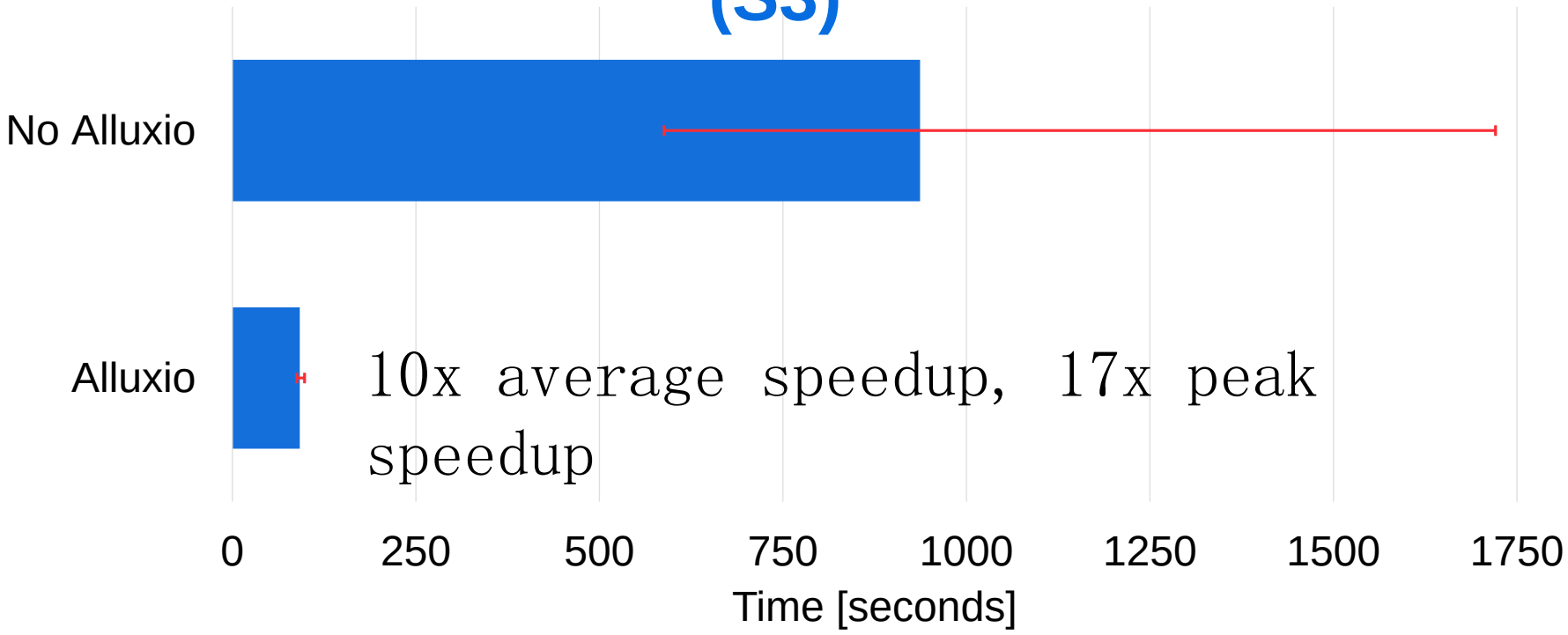
READING CACHED DATAFRAME (PARQUET)



新场景：READ 50 GB DATAFRAME (SSD)



新场景：READ 50 GB DATAFRAME (S3)



内容

- Alluxio基本原理回顾与1.4的最新特性介绍
- 基于Alluxio的Spark DataFrame/RDD性能调优
- **基于Alluxio提升HDFS集群的性能和SLA稳定性**
- 总结

Alluxio提升HDFS性能和SLA稳定性

- Alluxio可以给与HDFS共同部署的计算集群的两大好处
  - 提升高达10倍的性能提升
  - 性能的高可预测性使得SLA（service-level agreement服务级别协议）很容易满足
    - 例：作业运行时间的变化范围从100秒以上缩短至2秒

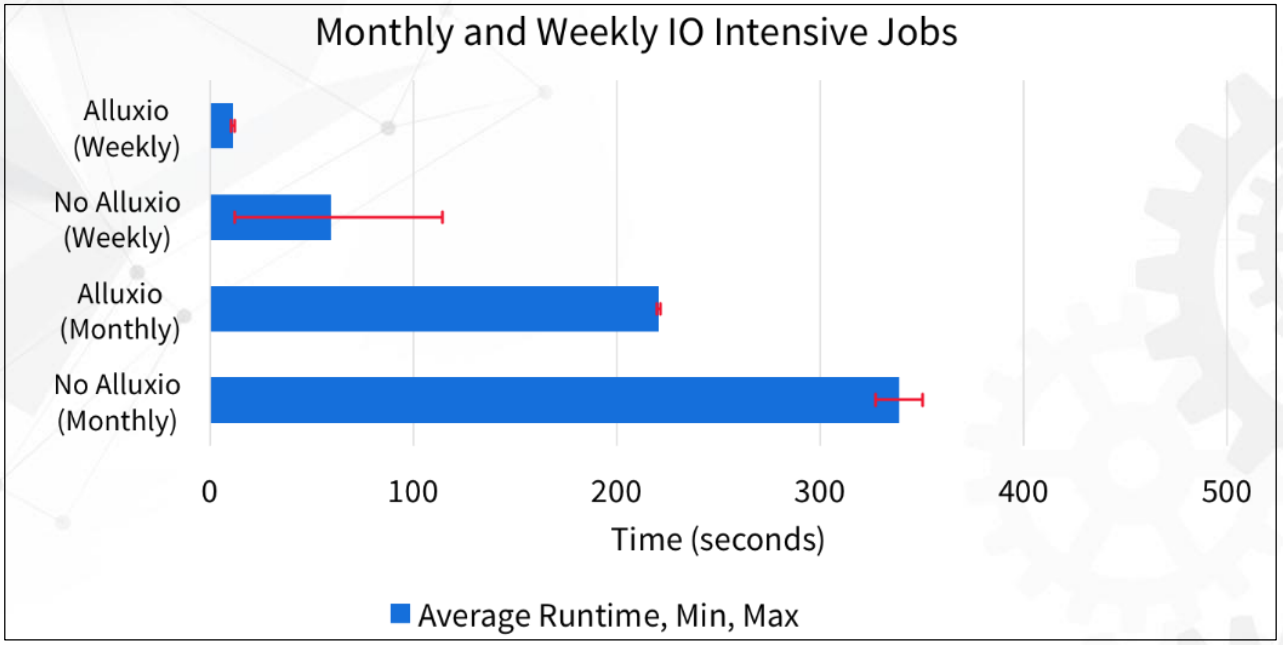
实验设置

为了模拟不同数据规模的多租户，实验设置如下：

- 在任何时间点，集群上会运行两个作业：月作业和周作业；
- 每一个作业使用一半的CPU和内存资源；
- 新的作业会在前一个同类型作业完成时立即开始；
- 实验在两个不同的软件栈上进行，其中一个使用Alluxio (即Spark + Alluxio + HDFS)，另外一个不使用Alluxio (即Spark + HDFS);

作业名	描述
Weekly IO	运行在前一个月数据上的I/O密集型作业
Monthly CPU	运行在前一个月数据上的CPU密集型作业
Monthly IO	运行在前一周数据上的IO密集型作业
Weekly CPU	运行在前一周数据上的CPU密集型作业

场景 1



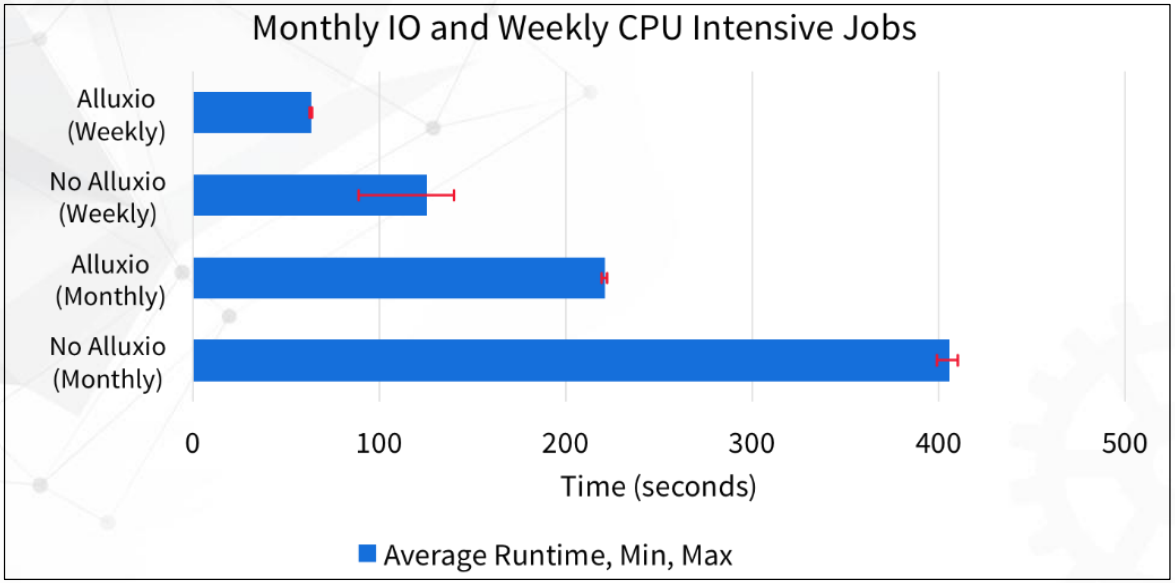
周作业和月作业都是IO密集型

结论：

1. Alluxio对这两种作业的性能提升都很明显。

2. 在不使用Alluxio的情况下，作业执行性能的波动范围较大（见图中 用红线标出的最小和最大范围），甚至可能比使用Alluxio的情况**慢10倍**以上。

场景 2



月作业I/O密集型，周作业CPU密集型

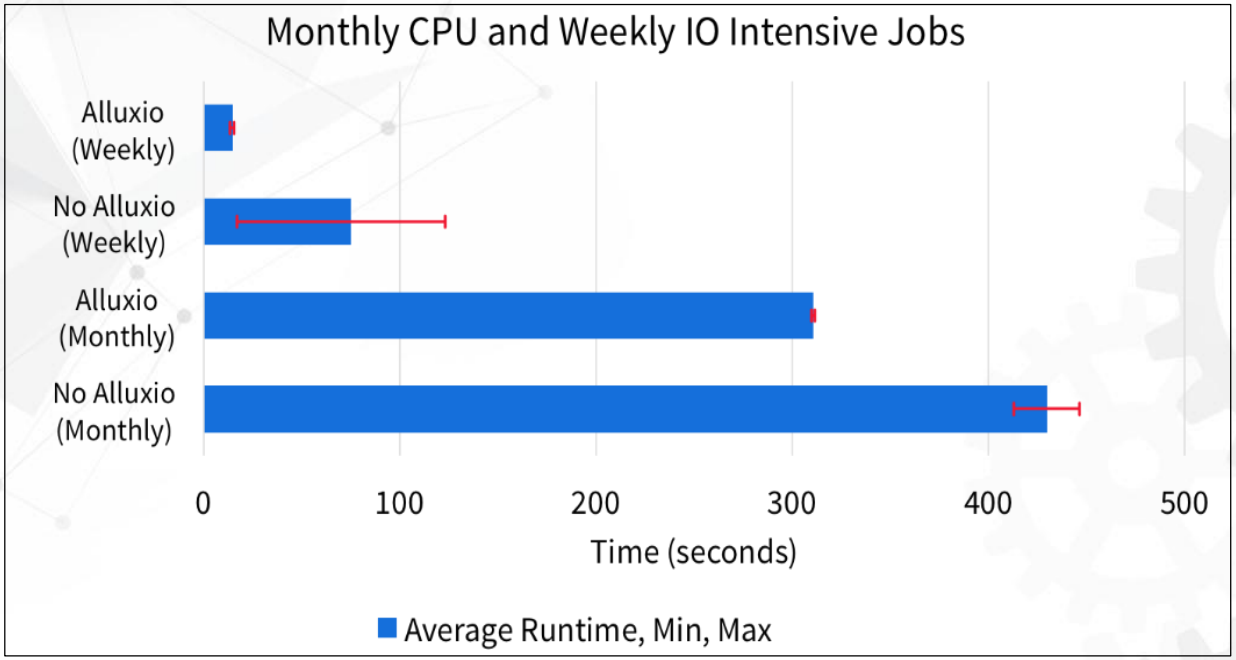
结论：

1. Alluxio还是同时提高了两种任务的性能

2. 周作业性能提升没有之前的IO密集型作业明显（此时性能主要受机器的CPU吞吐量影响）

3.月作业在使用Alluxio的情况下表现出极大的优势，因为场景1中月作业的性能影响因素在此场景下仍然适用

场景 3

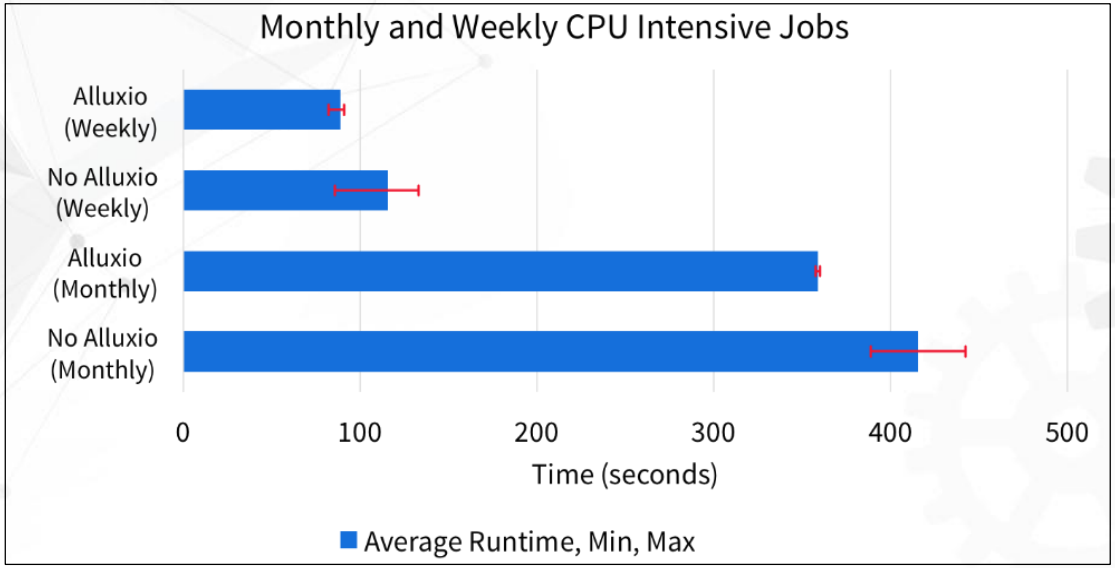


月作业是CPU密集型，周作业是I/O密集型

结论：

1. Alluxio极大地加速周作业，因为数据完全在内存中
2. CPU密集型的月作业的执行性能也有所提升，因为使用Alluxio避免了周作业与月作业竞争磁盘资源。

场景 4



月作业和周作业都是CPU密集型

结论：

1. Alluxio带来的性能提升不明显，因为两种任务中I/O吞吐量都不是瓶颈。
2. 然而，通过持续地管理在内存中的数据，Alluxio使得作业的性能更为稳定。

内容

- Alluxio应用场景分析与1.4版本新特性介绍
- 基于Alluxio的Spark DataFrame/RDD性能调优
- 基于Alluxio提升HDFS集群的性能和SLA稳定性
- **总结**

技术特点小结

- *将计算与数据共同安置（co-located），提供内存级访问速度*
- *通过统一的命名空间虚拟化底层不同存储系统*
- *横向可扩展的系统架构*
- *文件系统API（软件层的解决方案）*

系统优势小结

Unification

新的应用可以访问任意存储里的任意数据

Performance

多个数量级以上的性能提升

Flexibility

计算和存储的选择变得独立，可以根据需要而进行购买

提升Hadoop/Spark应用小结

- Alluxio可以在多个方面帮助Hadoop/Spark应用
  - Alluxio可以直接在内存中保存大规模的数据来加速Spark应用
  - Alluxio能够共享内存中的数据
  - Alluxio可以提供稳定和可预测的性能

欢迎使用Alluxio中文文档！

- http://alluxio.org/documentation/master/cn/index.html

个人微博：顾荣_NJU
电子邮箱：gurong@nju.edu.cn

THANKS

SequeMedia 盛拓传媒 IT168.com ITPUB ChinaUnix