A Reliable Memory-Centric Distributed Storage System

Haoyuan Li, Tachyon Nexus
haoyuan@tachyonnexus.com
September 22, 2015 @ SDC 2015
• Team consists of Tachyon creators, top contributors, people from UC Berkeley, Google, CMU, VMware, Stanford, Facebook, etc.

• $7.5 million Series A from Andreessen Horowitz

• Committed to Tachyon Open Source
WE’RE HIRING!
Outline

• Overview
  – Motivation
  – Tachyon Architecture
  – Using Tachyon

• Open Source
  – Status
  – Production Use Cases

• Roadmap
Outline

• **Overview**
  – Motivation
  – Tachyon Architecture
  – Using Tachyon

• **Open Source**
  – Status
  – Production Use Cases

• **Roadmap**
Tachyon: Born in UC Berkeley AMPLab

Cluster manager

Parallel computation framework

Reliable, distributed memory-centric storage system
Why Tachyon?
Memory is Fast

- RAM throughput increasing **exponentially**
- Disk throughput increasing **slowly**

**Memory-locality** key to interactive response times
Memory is **Cheaper**

![Memory Cost Over Time Graph](source: jcmit.com)

- **DRAM Cost Line**: Approximately 50% decrease every 18 months.
- **Value Streams**:
  - **Wall Street**
  - **Google**
  - **The rest of us**

**Source**: jcmit.com
Realized by many...

**DBMS2**

April 7, 2012

Many kinds of memory-centric data management

I'm frequently asked to generalize in some way about in-memory or memory-centric data management. I can start:

- The desire for human real-time interactive response naturally leads to

---

HANA In-Memory Platform

SAP Business Objects
SAP Applications
3rd Party SAP Applications

**cloudera**

IMPALA
Is the Problem Solved?
Missing a Solution for the Storage Layer
An Example: Spark

• Fast, in-memory data processing framework
  – Keep **one in-memory** copy inside JVM
  – Track **lineage** of operations used to derive data
  – Upon failure, use lineage to recompute data
Issue 1

Data Sharing is the bottleneck in analytics pipeline: Slow writes to disk

storage engine & execution engine same process (slow writes)
Issue 1

Data Sharing is the bottleneck in analytics pipeline: Slow writes to disk

storage engine & execution engine same process (slow writes)
Issue 2

Cache loss when process crashes

- execution engine & storage engine same process
- Spark Task
- Spark memory block manager
- HDFS / Amazon S3
Issue 2

**Cache loss when process crashes**

execution engine & storage engine same process

- Block 1
- Block 3
- Block 2
- Block 4

Spark memory block manager

HDFS / Amazon S3
Issue 2

Cache loss when process crashes

execution engine & storage engine same process

crash

block 1
block 2
block 3
block 4

HDFS / Amazon S3
Issue 3

In-memory Data Duplication & Java Garbage Collection

execution engine & storage engine same process (duplication & GC)
Tachyon

*Reliable* data sharing at *memory-speed* within and across cluster frameworks/jobs
Technical Overview

Ideas
• A memory-centric storage architecture
• Push lineage down to storage layer
• Manage tiered storage

Facts
• One data copy in memory
• Re-computation for fault-tolerance
Eco-System

Spark
Hadoop MapReduce
Apache HBase
Flink
H2O
Cloudera Impala

TACHYON

S3
GlusterFS
HDFS
openstack
NFS
ceph
Tachyon Memory-Centric Architecture
Tachyon Memory-Centric Architecture
Lineage in Tachyon
Issue 1 revisited

Memory-speed data sharing among jobs in different frameworks

execution engine & storage engine same process (fast writes)
Issue 2 revisited

*Keep in-memory data safe, even when a job crashes.*

execution engine & storage engine same process

```
Spark Task

Spark memory block manager

Tachyon
in-memory

block 1
block 3
block 4
```
Issue 2 revisited

Keep *in-memory* data safe, even when a job crashes.

- Execution engine & storage engine same process
- Crash
- Tachyon in-memory
- Block 1
- Block 3
- Block 4
- HDFS / Amazon S3

Keep in-memory data safe, even when a job crashes.
Issue 3 revisited

No in-memory data duplication, much less GC

execution engine & storage engine same process (no duplication & GC)

Spark Task
Spark mem

Spark Task
Spark mem

Tachyon in-memory

block 1
block 3
block 4

HDFS / Amazon S3

block 1
block 2
block 3
block 4
Comparison with In-Memory HDFS

Write Throughput

- Tachyon Write
- MemHDFS Write
- Theoretical Replication (2 copies) Based Write

Throughput (GB/Sec)

Number of Machines
Outline

• Overview
  – Motivation
  – Tachyon Architecture
  – Using Tachyon
• Open Source
  – Status
  – Production Use Cases
• Roadmap
Open Source Status

• Started at UC Berkeley AMPLab in Summer 2012

• Apache License 2.0, Version 0.7.1 (August 2015)

• Deployed at > 50 companies (July 2014)

• 30+ Companies Contributing
Contributors Growth

- v0.1: Dec '12, 1
- v0.2: Apr '13, 3
- v0.3: Oct '13, 15
- v0.4: Feb '14, 30
- v0.5: Jul '14, 46
- v0.6: Mar '15, 70
- v0.7: Jul '15, 111
Codebase Growth

- v0.2: 465 commits (Apr '13)
- v0.3: 696 commits (Oct '13)
- v0.4: 1080 commits (Feb '14)
- v0.5: 1610 commits (Jul '14)
- v0.6: 2884 commits (Mar '15)
- v0.7: 5021 commits (Jul '15)
Thanks to Our Contributors!
Reported Tachyon Usage

The Future Architecture of a Data Lake: In-memory Data Exchange Platform Using Tachyon and Apache Spark

Tachyon is the in-heap distributed file system. Tachyon is the fast file system. Tachyon makes it simple to deploy a distributed file system. Tachyon scales to multi-terabyte file sizes.

Pivotal is the high-availability database. Pivotal is the high-availability database. Pivotal is the high-availability database.

GigaOM

IBM Research

8.17.2015

Tachyon for ultra-fast Big Data processing

Editor’s note: This article is by cloud analytics infrastructure expert Gil Vernik, IBM Research-Haifa.

Today’s massive growth in data sets means that storage is increasingly becoming a critical bottleneck for system workloads. My storage team in Haifa, Israel wants to analyze and understand these massive volumes of data, and we need to store them somewhere reliable. Although disk space is an option, it’s too slow to carry out fast Big Data processing. In-memory computing, which keeps the data in a server’s RAM for fast access and processing, offers a good solution for processing Big Data workloads – but it’s limited and expensive.

Enter Tachyon, a memory-centric distributed storage system that offers processing at memory-speed and reliable storage. Its software works with servers in clusters so there’s plenty of room for storage, and a unique proprietary feature eliminates the need for replication to ensure fault tolerance. Now, we’ve connected Tachyon to Swift so it can work effortlessly with Swift and SoftLayer. The result? Tachyon is even more flexible and efficient.
Under Filesystem Choices
(Big Data, Cloud, HPC, Enterprise)
Use Case: Baidu

- Framework: SparkSQL
- Under Storage: Baidu’s File System
- Storage Media: MEM + HDD
- 100+ nodes deployment
- 1PB+ managed space
- 30x Performance Improvement
Use Case: a SAAS Company

• Framework: Impala

• Under Storage: S3

• Storage Media: MEM + SSD

• 15x Performance Improvement
Use Case: an Oil Company

• Framework: Spark

• Under Storage: GlusterFS

• Storage Media: MEM only

• Analyzing data in traditional storage
Use Case: a SAAS Company

- Framework: Spark
- Under Storage: S3
- Storage Media: SSD only
- Elastic Tachyon deployment
Outline

• Overview
  – Motivation
  – Tachyon Architecture
  – Using Tachyon

• Open Source
  – Status
  – Production Use Cases

• Roadmap
New Features

- Lineage in Storage (alpha)
- Tiered Storage (alpha)
New Features

• Lineage in Storage (alpha)
• Tiered Storage (alpha)
  • Data Serving
  • Support for New Hardware
  • ...
• Your New Feature!
Tachyon’s Goal?
Distributed Memory-Centric Storage: Better Assist Other Components

Welcome Collaboration!

JIRA New Contributor Tasks
• Website: http://tachyon-project.org
• Github: https://github.com/amplab/tachyon
• Meetup: http://www.meetup.com/Tachyon
• Email: haoyuan@tachyonnexus.com