Tachyon: memory-speed data sharing

Ali Ghodsi, Haoyuan (HY) Li,
Matei Zaharia, Scott Shenker, Ion Stoica
Memory trumps everything else

- RAM throughput increasing **exponentially**
- Disk throughput increasing **slowly**
Realized by many...

• Frameworks already leverage memory
  – e.g. Spark, Piccolo, GraphX...
Example: Spark

- Fast in-memory data processing within a job
  - Keep only one copy in-memory copy JVM
  - Track lineage of operations used to derive data
  - Upon failure, use lineage to re-compute data
Challenge 1

execution engine & storage engine same JVM process
Challenge 1

execution engine & storage engine same JVM process

- block 1
- block 3

- Spark memory block manager

- block 1
- block 2
- block 3
- block 4

- HDFS disk

- crash
Challenge 1

JVM crash: lose all cache

execution engine & storage engine same JVM process
Challenge 2

**JVM heap overhead:**
GC & duplicate memory per job

execution engine & storage engine
same JVM process (GC & duplication)
Challenge 3

Different jobs share data:
Slow writes to disk

storage engine &
exection engine
same JVM process
(slow writes)
Challenge 3

Different frameworks share data: Slow writes to disk

storage engine & execution engine same JVM process (slow writes)
Tachyon

Reliable data sharing at memory-speed within and across cluster frameworks/jobs
Challenge 1 revisited

execution engine & storage engine same JVM process

Spark Task

Spark memory block manager

Tachyon in-memory
Challenge 1 revisited

execution engine & storage engine same JVM process

- Spark memory
  - block manager

- Tachyon in-memory
  - block 1
  - block 3
  - block 4

- HDFS disk
  - block 1
  - block 2
  - block 3
  - block 4

crash
Challenge 1 revisited

**JVM crash: keep memory-cache**

execution engine & storage engine same JVM process

- Tachyon in-memory
- HDFS disk

- block 1
- block 3
- block 4
Challenge 2 revisited

**Off-heap memory storage**

No GC & one memory copy

Execution engine & storage engine same JVM process (no GC & duplication)
Challenge 3 revisited

Different frameworks share at memory-speed

execution engine & storage engine same JVM process (fast writes)
Tachyon and Spark

Spark’s of **off-JVM-heap RDD-store**
- In-memory RDDs (serialized)
- Fault-tolerant cache

Enables
- avoiding GC overhead
- fine-grained executors
- fast RDD sharing
Tachyon research vision

Vision
• Push **lineage** down to storage layer
• Use memory aggressively

Approach
• One in-memory copy
• Rely on recomputation for fault-tolerance
Architecture
Comparison with in Memory HDFS

**Write Throughput**
- TFS
- MemHDFS

**Read Throughput**
- TFS
- MemHDFS
Further Improve Spark’s Performance

Grep
Master Faster Recovery

[Graph showing time in seconds for different millions of files]
Open Source Status

- **New release**
  - V0.4.0 (July 2014)
  - 20 Developers (7 from Berkeley, 13 from outside)
  - 11 Companies
  - Writes go synchronously to under filesystem
    (No lineage information in Developer Preview release)
  - MapReduce and Spark can run without any code change
    (ser/de becomes the new bottleneck)
Using HDFS vs Tachyon

• Spark
  
  val file = sc.textFile("hdfs://ip:port/path")

• Shark
  
  CREATE TABLE orders_cached AS SELECT * FROM orders;

• Hadoop MapReduce
  
  hadoop jar examples.jar wordcount hdfs://localhost/input hdfs://localhost/output
Using HDFS vs Tachyon

• **Spark**

```scala
val file = sc.textFile("tachyon://ip:port/path")
```

• **Shark**

```sql
CREATE TABLE orders_tachyon AS SELECT * FROM orders;
```

• **Hadoop MapReduce**

```bash
hadoop jar examples.jar wordcount
tachyon://localhost/input
tachyon://localhost/output
```
Thanks to Redhat!

Tachyon is in Fedora 20

Thanks to Redhat!
Future Research Focus

• Integration with HDFS caching

• Memory Fair Sharing

• Random Access Abstraction

• Mutable Data Support
Acknowledgments

Calvin Jia, Nick Lanham, Grace Huang, Mark Hamstra, Bill Zhao, Rong Gu, Hobin Yoon, Vamsi Chitters, Joseph Jin-Chuan Tang, Xi Liu, Qifan Pu, Aslan Bekirov, Reynold Xin, Xiaomin Zhang, Achal Soni, Xiang Zhong, Dilip Joseph, Srinivas Parayya, Tim St. Clair, Shivaram Venkataraman, Andrew Ash
Tachyon Summary

• As more workloads move into memory, big data data sharing across frameworks will become a bottleneck
  – Tachyon provides in-memory, fault-tolerant data sharing across frameworks
Thanks!

- More: https://github.com/amplab/tachyon