Sparker: Efficient Reduction for More Scalable Machine Learning with Spark

Bowen Yu, Huanqi Cao, Tianyi Shan†, Haojie Wang, Xiongchao Tang, Wenguang Chen

Tsinghua University

† University of California San Diego
Background

Big data, Spark, distributed machine learning in Spark
Big Data

- Big Data: **large growing** data sets that include **heterogeneous** formats: structured, unstructured and semi-structured data[1].

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Storage</th>
<th>Resource Management</th>
<th>Data Analytics</th>
<th>Data Visualize</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="data-source-car.png" alt="Car" /></td>
<td><img src="storage-hadoop.png" alt="Hadoop" /></td>
<td><img src="resource-management-kubernetes.png" alt="Kubernetes" /></td>
<td><img src="data-analytics-apache-spark.png" alt="Apache Spark" /></td>
<td><img src="data-visualize-matplotlib.png" alt="Matplotlib" /></td>
</tr>
<tr>
<td><img src="data-source-phone.png" alt="Phone" /></td>
<td><img src="storage-amazon-s3.png" alt="Amazon S3" /></td>
<td><img src="resource-management-apache-mesos.png" alt="Apache Mesos" /></td>
<td><img src="data-analytics-apache-hive.png" alt="Apache Hive" /></td>
<td><img src="data-visualize-data-driven-documents.png" alt="Data-Driven Documents" /></td>
</tr>
<tr>
<td><img src="data-source-cloud.png" alt="Cloud" /></td>
<td><img src="storage-hadoop-yarn.png" alt="Hadoop YARN" /></td>
<td><img src="resource-management-flink.png" alt="Flink" /></td>
<td><img src="data-analytics-matplotlib.png" alt="Matplotlib" /></td>
<td></td>
</tr>
</tbody>
</table>
Apache Spark

- Apache Spark is an important big data framework that unifies big data analytics.
- Libraries are built upon Spark’s core module using its RDD API.
Distributed Machine Learning Training in Spark

- Resiliency against frequent failures in commodity clusters
- Auto-scaling for better cluster utilization
- Single-thread abstraction to ease programming

 MPI

Spark

Driver

Executor_1
Executor_2
Executor_3

Symmetric

MPI_Allreduce

Aggregate

Broadcast

Aggregate

Broadcast

Similar to fork-join
Motivation

Despite training machine learning model in Spark has advantages, it has scalability issue.
Experiment Configuration

• Platform BIC
  • Intel Xeon E5-2680 v4
  • 448-core in-house cluster

• Platform AWS
  • Intel Xeon Platinum 8175M
  • 960-core public cloud cluster
  • AWS EC2 (m5d.24xlarge)

• Apache Spark: Spark 2.3.0
• MPI library: MPICH 3.2

• Datasets from libsvm
  • avazu
  • criteo
  • kdd10
  • kdd12

• Datasets from uci
  • enron
  • nytimes

• MLlib Applications
  • Latent Dirichlet Allocation (LDA)
  • Support Vector Machine (SVM)
  • Logistic Regression (LR)
Scalability Issue in MLlib

- Poor scalability: $1.25 \times$ speedup on 8 machines w.r.t 1 machine
Reduction is the Scalability Bottleneck

- **Driver**: computation not offloaded to executors
- **Non-aggregation**: stages unrelated to aggregation
- **Aggregation**: stages related to aggregation operation
  - *Compute*: data-parallel computation
  - *Reduce*: reduction

- **Scalability Bottleneck**

- **Strong scalability of LDA-N**

![Graph showing normalized time and scalability bottleneck](image-url)

![Bar chart showing time in seconds](image-url)
The Cause of Reduction Scalability

**M**: message size  **B**: bandwidth  **P**: number of executors

Tree-based  
\[ t = \log P \times \frac{M}{B} \]

not scalable

Ring-based + Gather  
\[ t = \frac{P - 1}{P} \times \frac{M}{B} + \frac{M}{B} \]

scalable

Abstract class RDD[T]

```
1 class RDD[T] {
2   def aggregate[U](zeroValue: U) {
3     seqOp: (U, T) => U,
4     reduceOp: (U, U) => U : U
5   }
6
7   def treeAggregate[U](zeroValue: U) {
8     seqOp: (U, T) => U,
9     reduceOp: (U, U) => U,
10    depth: Int = 2 : U
11   }
```

Values (T)

Result (U)

Aggregator (U)

No way to split aggregators

M: message size  B: bandwidth  P: number of executors
Sparker
Challenges

- Challenge 1: Aggregation interface should include aggregator-splitting semantics.
- Challenge 2: Low-latency communication among executors is required.
- Challenge 3: Communication amount should be reduced.
Splittable Aggregation Interface

Challenge 1: Aggregation interface should include aggregator-splitting semantics.

The aggregator-splitting semantic is included in the splittable aggregation interface.

```java
abstract class RDD[T] {
  def splitAggregate[U, V](zeroValue: U) {
    seqOp: (U, T) => U,
    splitOp: (U, Int, Int) => V,
    reduceOp: (V, V) => V,
    concatOp: Seq[V] => V,
    parallelism: Int = 4): V
  }
```

Values (T) \rightarrow Segment (V) \rightarrow Result (V)
Low-latency Inter-Executor Communication

Challenge 2: Low-latency communication among executors is required.

Figure: The latency of Block Manager is very high.

Neighborhood communication on a ring-based topology only.
Improvements on Scalable Communicator

Improvement 1: Parallel Directed Ring (PDR) to provide abundant CPU power to overcome Java serialization / deserialization overhead.

Improvement 2: Topology-awareness eliminates unnecessary inter-node communication by properly placing executors on the nodes.
Ring-based Reduction Algorithm

- Based on the splittable aggregation interface and the scalable communicator, we implement a ring-based reduction algorithm.
In-Memory Merge

Challenge 3: Communication amount should be reduced.

Independent tasks

With in-memory merge
Evaluation
Experiment Configuration

- **Platform BIC**
  - Intel Xeon E5-2680 v4
  - 448-core in-house cluster

- **Platform AWS**
  - Intel Xeon Platinum 8175M
  - 960-core public cloud cluster
  - AWS EC2 (m5d.24xlarge)

- **Apache Spark**: Spark 2.3.0
- **MPI library**: MPICH 3.2

- **Datasets from libsvm**
  - avazu
  - criteo
  - kdd10
  - kdd12

- **Datasets from uci**
  - enron
  - nytimes

- **MLlib Applications**
  - Latent Dirichlet Allocation (LDA)
  - Support Vector Machine (SVM)
  - Logistic Regression (LR)
Evaluation

- Fig: communication latency vs message size
- Scalable communicator has near-MPI performance and has significantly lower latency than Spark Block Manager
Evaluation

- Fig: communication throughput vs message size
- Unlike MPI, only with Parallel Directed Ring (PDR) can the scalable communicator fully utilize the network bandwidth. This is due to high CPU overhead from Java serialization and deserialization.
Evaluation

- Fig: reduce-scatter time vs number of parallel PDR rings
- Parallel Directed Ring improves the reduce-scatter performance, and topology-awareness further improves the reduce-scatter performance.
Evaluation

- Fig: reduce-scatter time vs the number of executors
- The reduce-scatter performance of scalable communicator is as scalable as MPI (even goes beyond MPI)
Evaluation

- Fig: comparing tree aggregation, tree aggregation with in-memory merge, and split aggregation with in-memory merge.
- For large messages (256MB), in-memory merge improves the aggregation performance, and split aggregation further improves the performance.
- For small messages (1KB), their performance are similar.
Evaluation

- Fig: speedup of end-to-end MLlib applications.
- Sparker (IMM + Split Aggregation) improves the end-to-end MLlib distributed machine learning training performance.

![Speedup chart](image-url)
Evaluation

- Fig: strong scalability of LDA-N on AWS
- Sparker (IMM + Split Aggregation) improves the end-to-end MLlib distributed machine learning training strong scalability due to improved reduction performance.
• A aggregation interface for distributed datasets that supports scalable reduction.
• A low-latency and high-bandwidth communication layer integrated in Spark.
• Improve the end-to-end scalability of Spark’s distributed machine learning.

Thank you!

Bowen Yu, Huanqi Cao, Tianyi Shan, Haojie Wang, Xiongchao Tang, Wenguang Chen
Tsinghua University, University of California San Diego

yubw15@mails.tsinghua.edu.cn, caohq18@mails.tsinghua.edu.cn, tshan@eng.ucsd.edu,
wanghaojie@tsinghua.edu.cn, txc13@tsinghua.org.cn, cwg@tsinghua.edu.cn