INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP / 2021 / CHICAGO / USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

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*

清華大學 Tsinghua University

UC San Diego

Background

Big data, Spark, distribured machine learning in Spark

Big Data

The "3V" of big data

Volume Velocity Variety

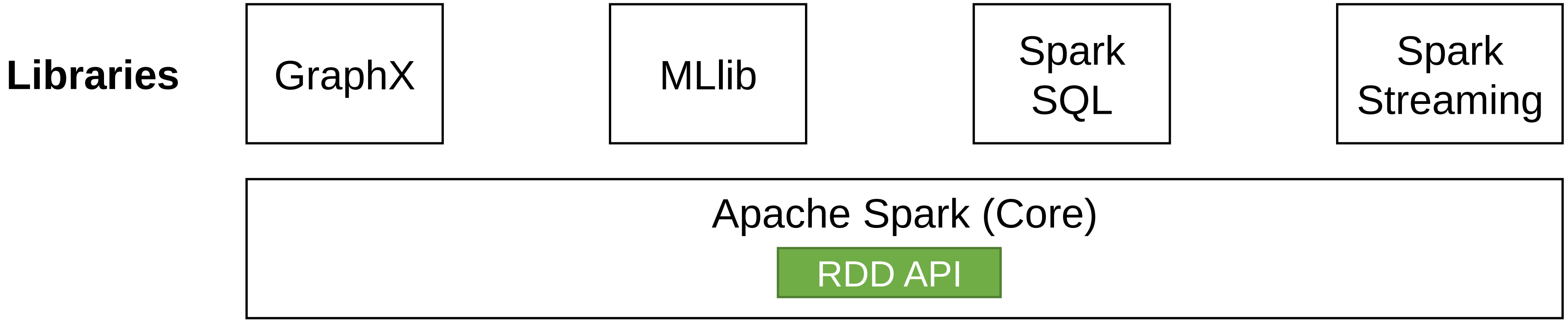
- Big Data: **large** **growing** data sets that include **heterogeneous** formats: structured, unstructured and semi-structured data[1].

Data Source	Storage	Resource Management	Data Analytics	Data Visualize
	hadoop HDFS	hadoop YARN	hadoop MapReduce	matplotlib
	amazon S3	kubernetes	APACHE Spark	D3 Data-Driven Documents
		Apache MESOS	Flink	
			HIVE	

[1] Oussous, Ahmed, et al. "Big Data technologies: A survey." Journal of King Saud University-Computer and Information Sciences 30.4 (2018): 431-448.

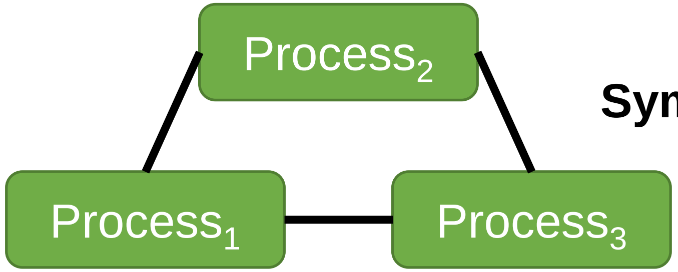
Apache Spark

- 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.



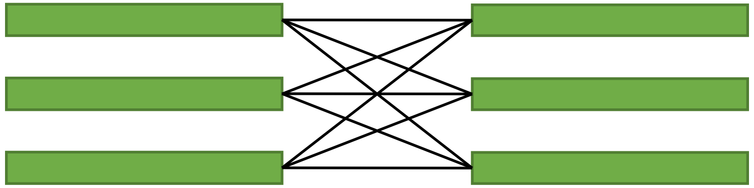
Distributed Machine Learning Training in Spark

MPI

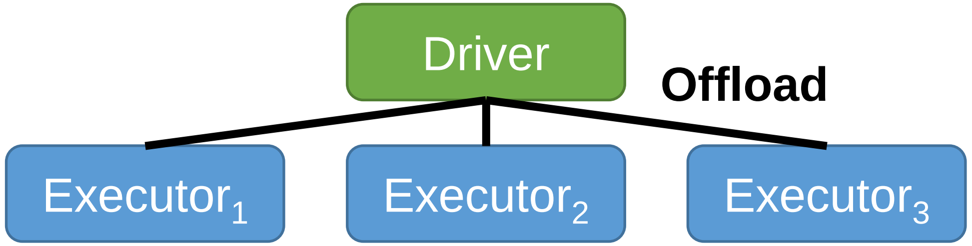


Symmetric

`MPI_Allreduce`

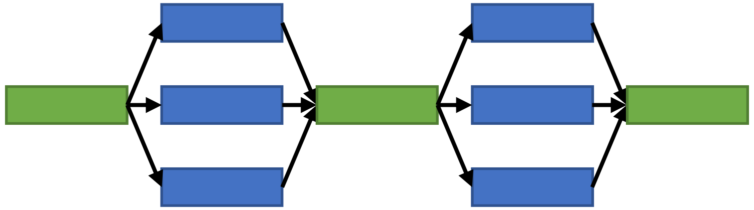


Spark



Offload

`Aggregate` `Aggregate`



`Broadcast` `Broadcast`

Similar to fork-join

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

Motivation

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

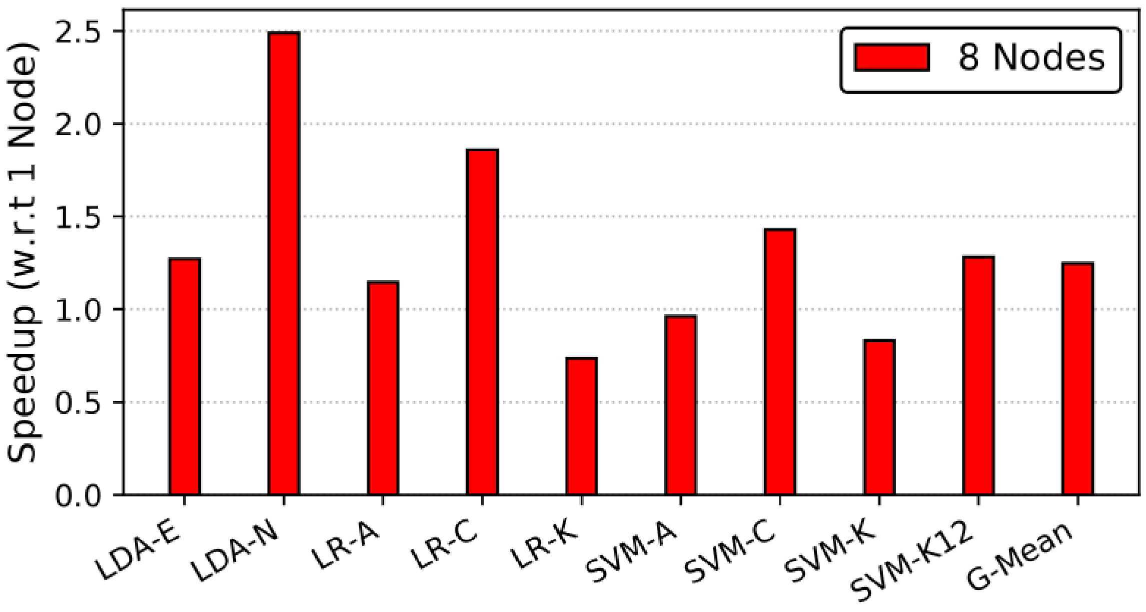
Experiment Configuration

- 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)

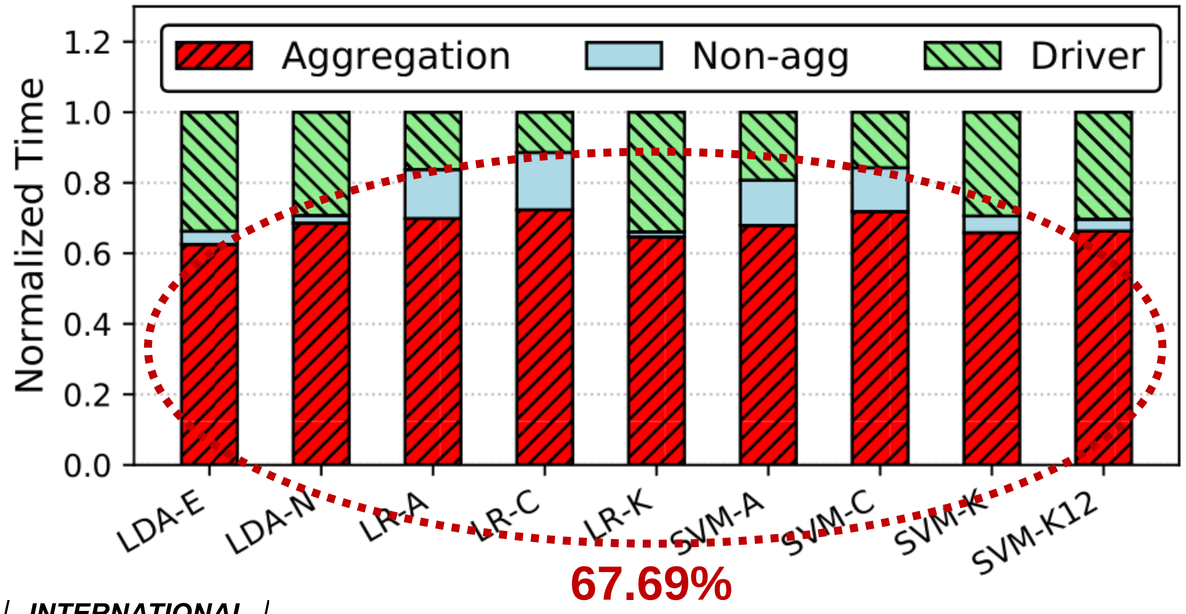
Scalability Issue in MLlib

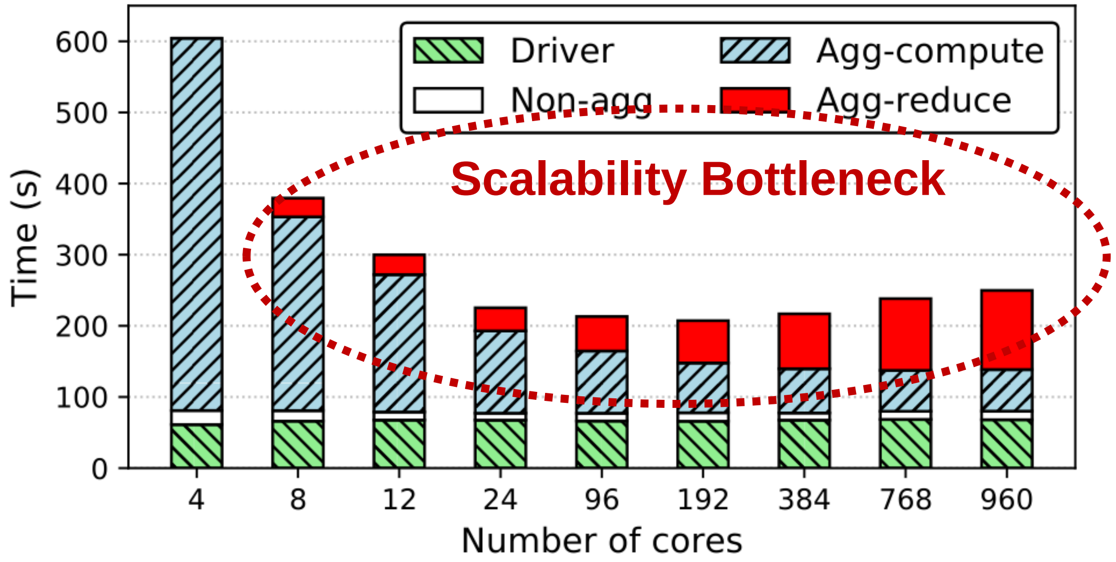
- Poor scalability: 1.25 × speedup on 8 machines w.r.t 1 machine



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



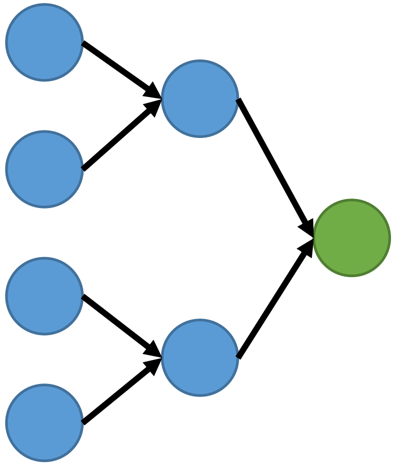


Strong scalability of LDA-N

The Cause of Reduction Scalability

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

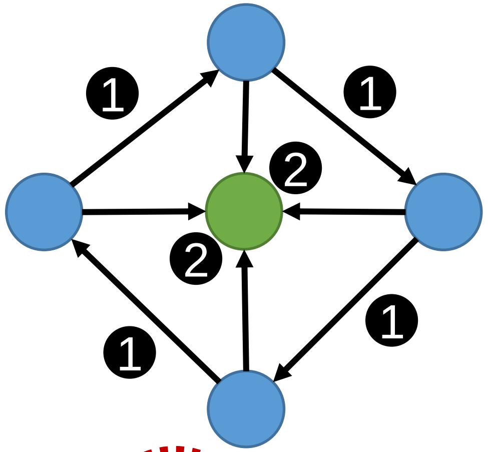
Tree-based



$$t = \log P \times \frac{M}{B}$$

not scalable

Ring-based + Gather



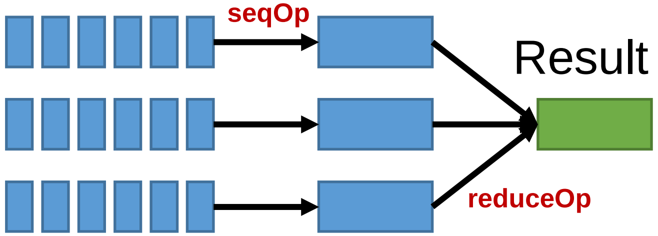
$$t = \frac{P-1}{P} \times \frac{M}{B} + \frac{M}{B}$$

scalable

```
abstract class RDD[T] {
 def aggregate[U](zeroValue: U)(
  seqOp: (U, T) => U,
  reduceOp: (U, U) => U): U

 def treeAggregate[U](zeroValue: U)(
  seqOp: (U, T) => U,
  reduceOp: (U, U) => U,
  depth: Int = 2): U
}
```

Values (T)



Aggregator (U)

No way to split aggregators

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

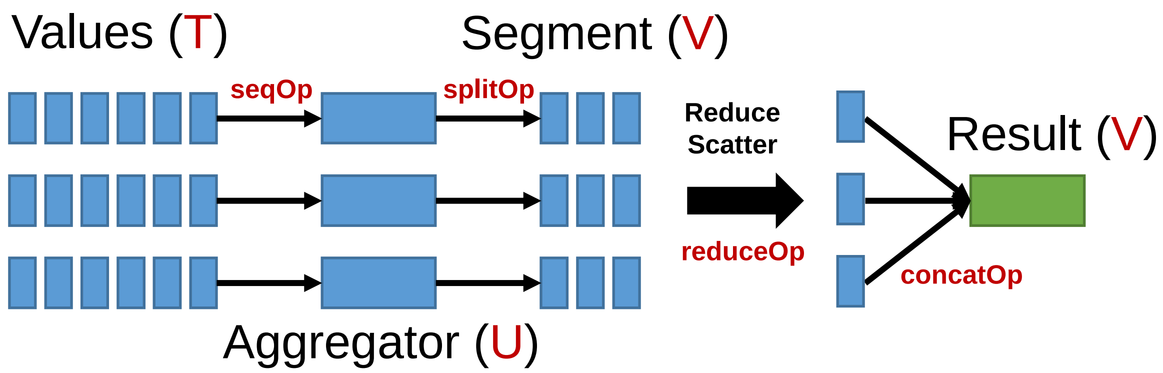
Challenge 1: Aggregation interface should include aggregator-splitting semantics.

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

```
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
}
```

Segment Index

Num of Segments



Low-latency Inter-Executor Communication

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

Spark

Executor 0 | Executor 1 | Executor 2

Block Manager (Shared Object Storage)

Spark

Executor 0 | Executor 1 | Executor 2

Block Manager (Shared Object Storage) | Scalable Communicator (Explicit Message Passing)

Neighborhood communication on a ring-based topology only

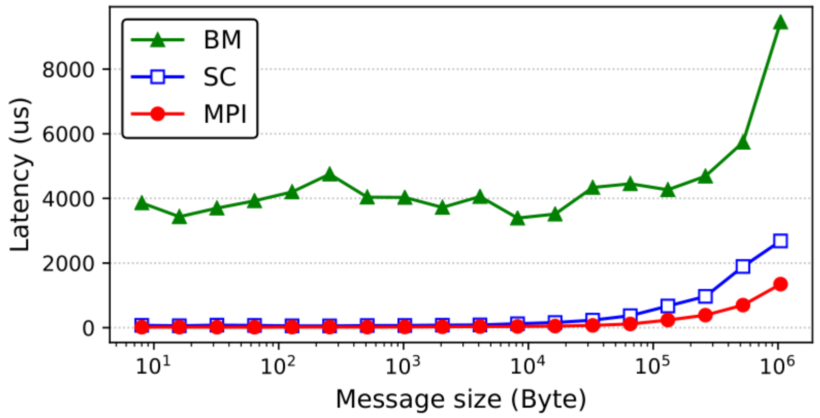
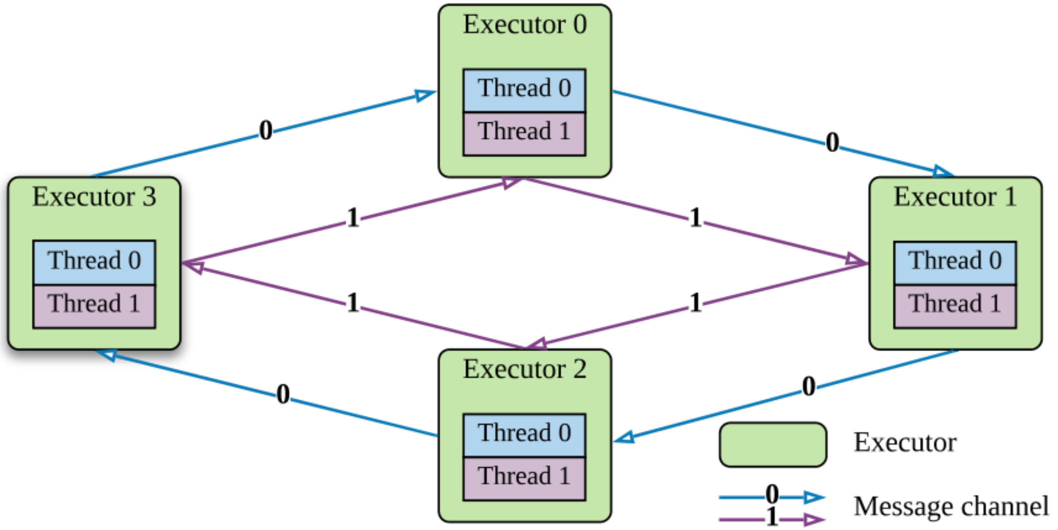
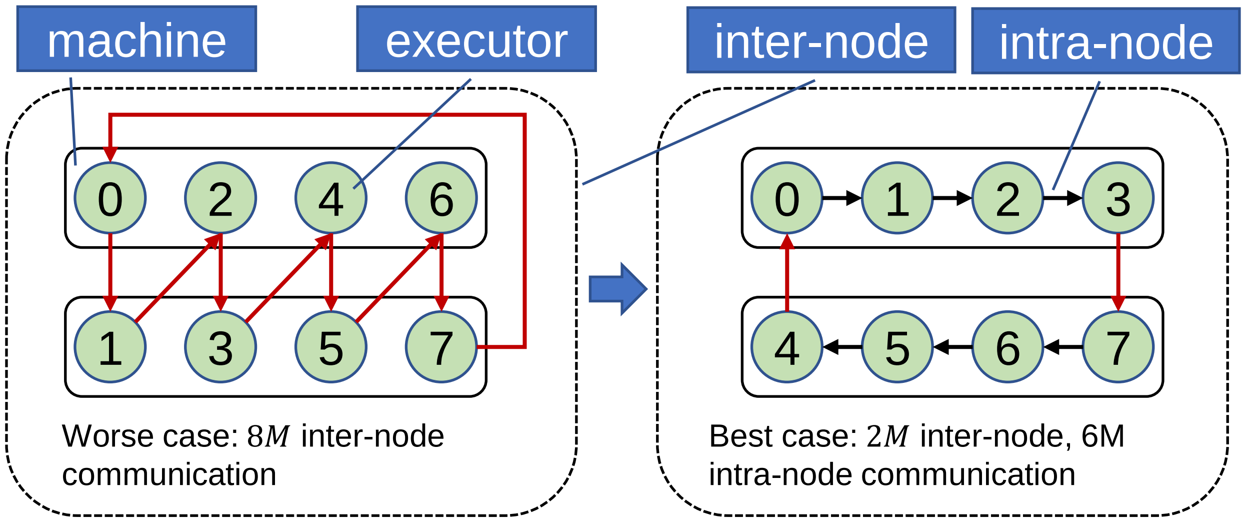


Figure: The latency of Block Manager is very high.

Improvements on Scalable Communicator



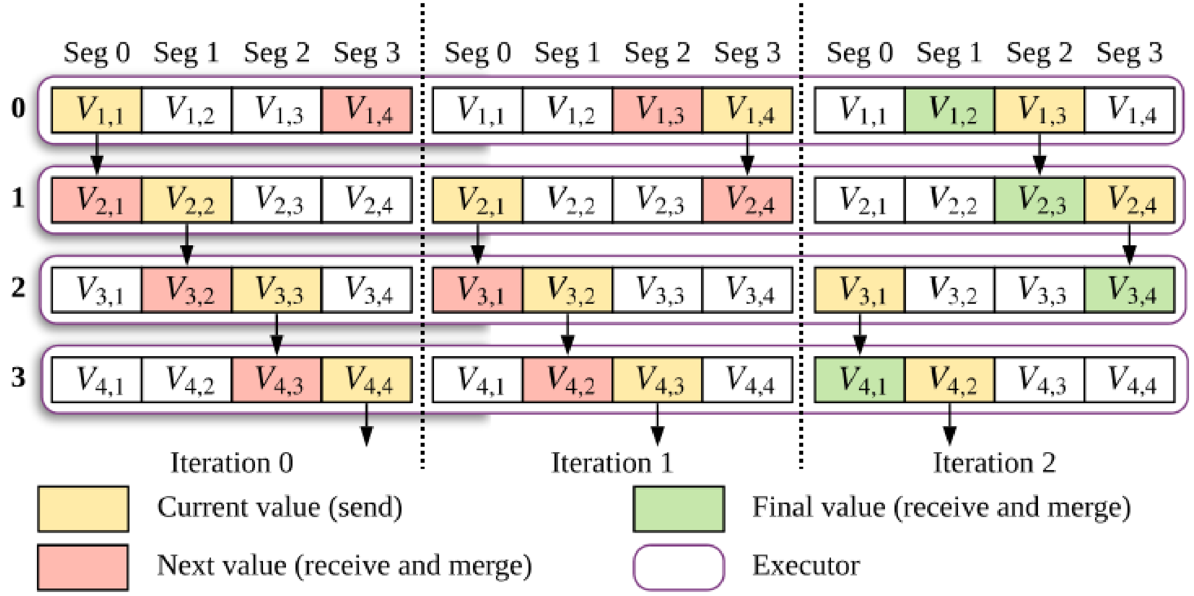
Improvement 1: Parallel Directed Ring (PDR) to provide abundant CPU power to overcome Java serialization / deserialization overhead.



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.



INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

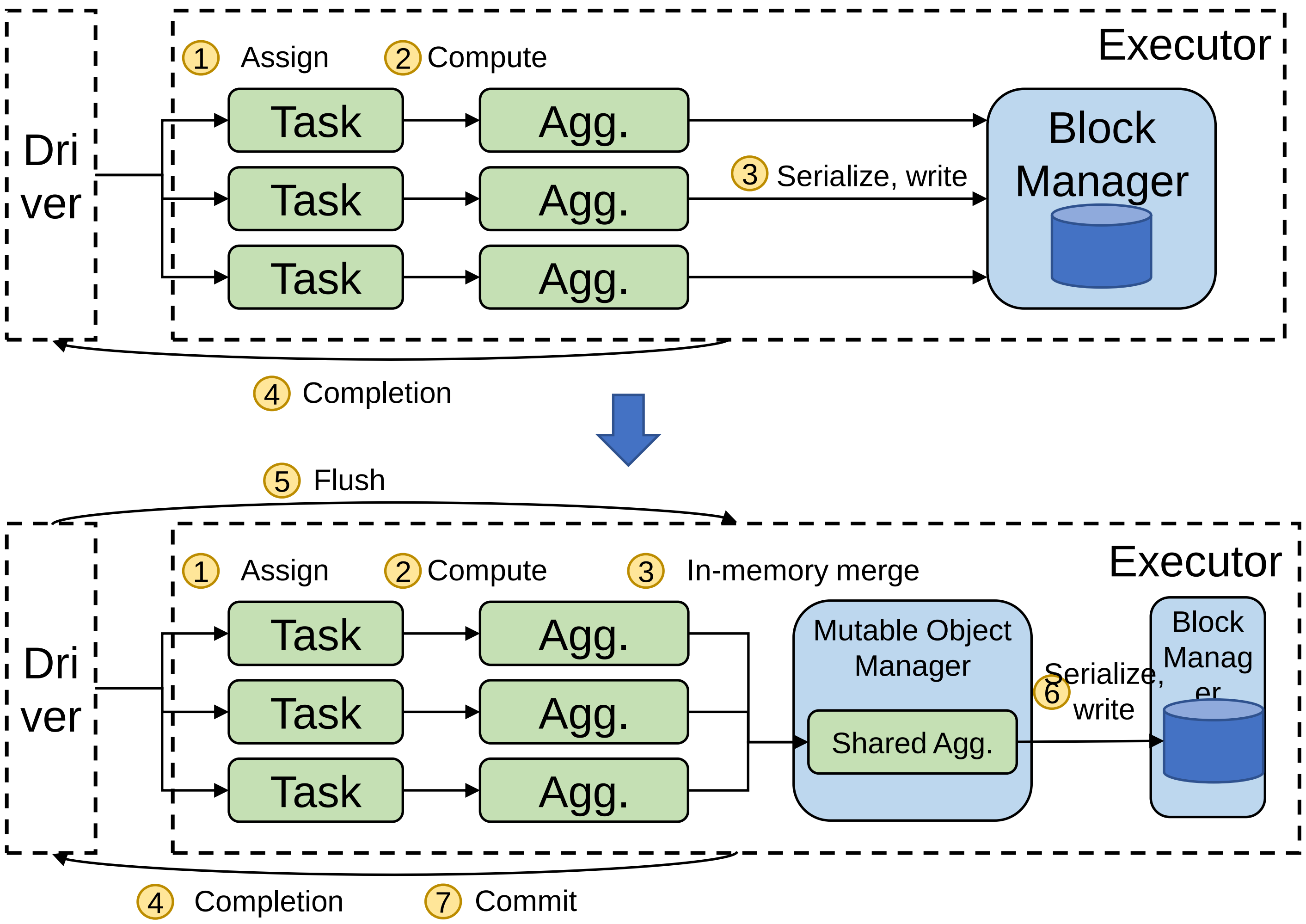
acm In-Cooperation sighpc

In-Memory Merge

Challenge 3: Communication amount should be reduced.

Independent tasks

With in-memory merge



Evaluation

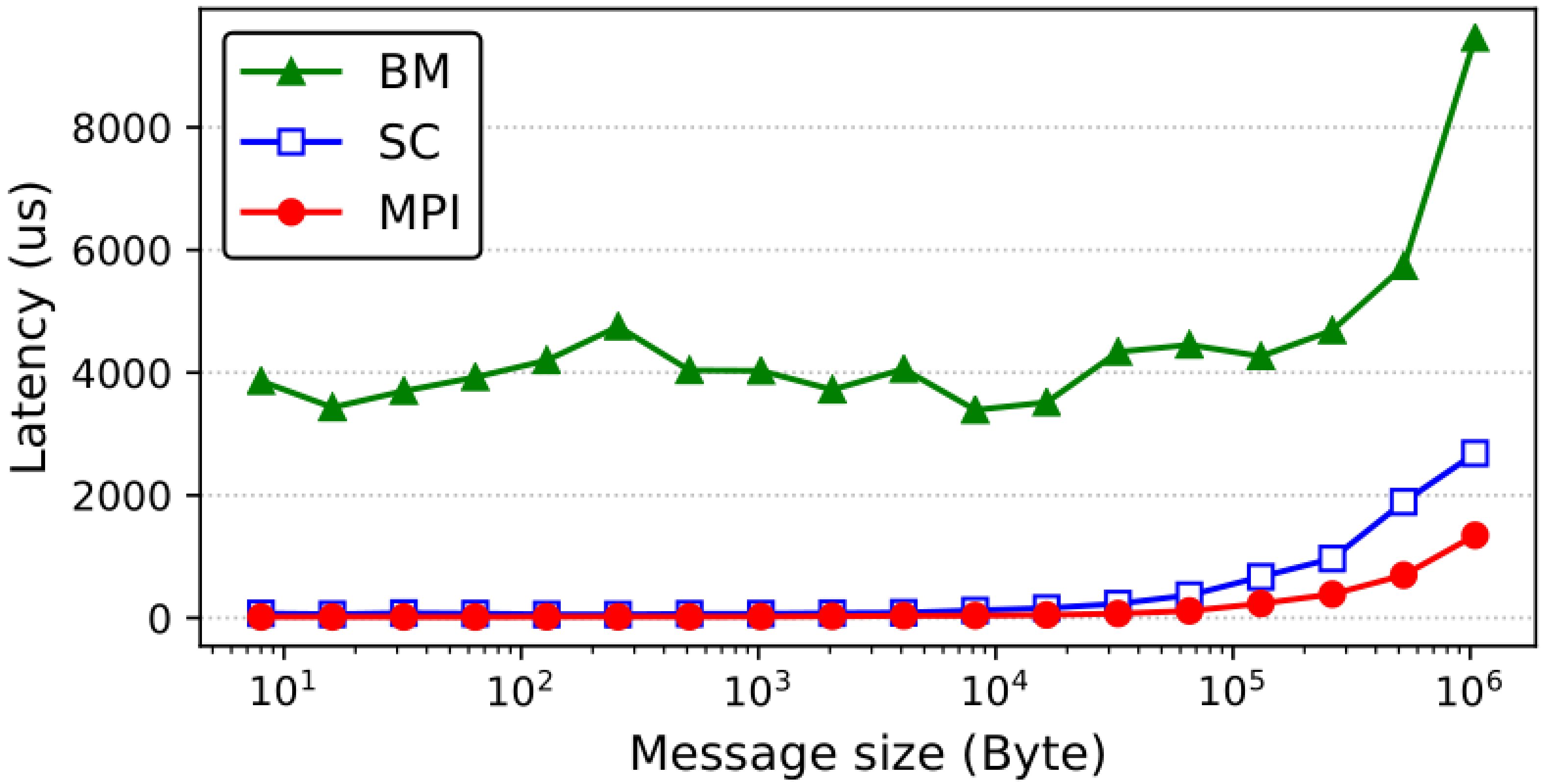
Experiment Configuration

- 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)

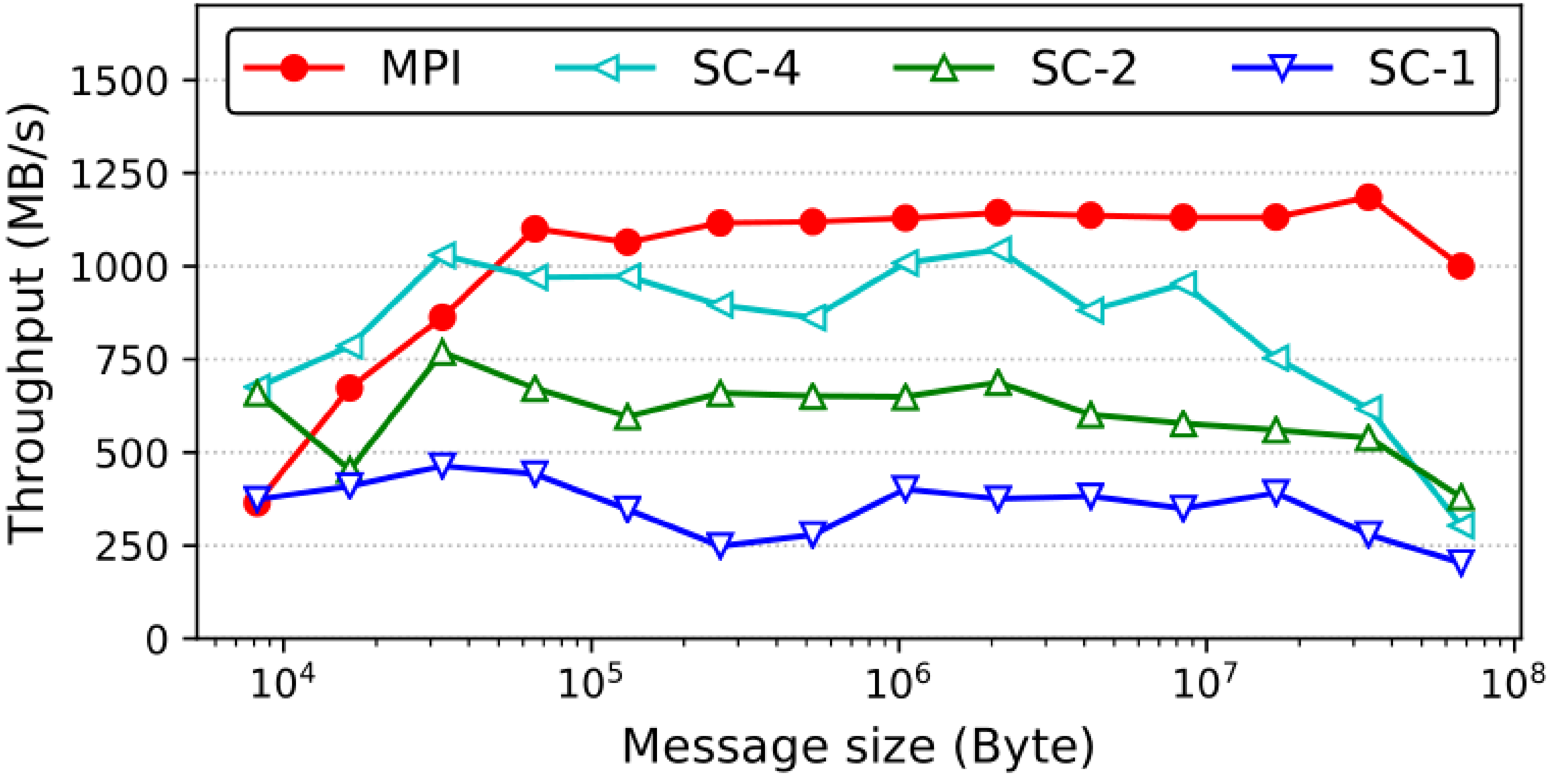
Evaluation

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



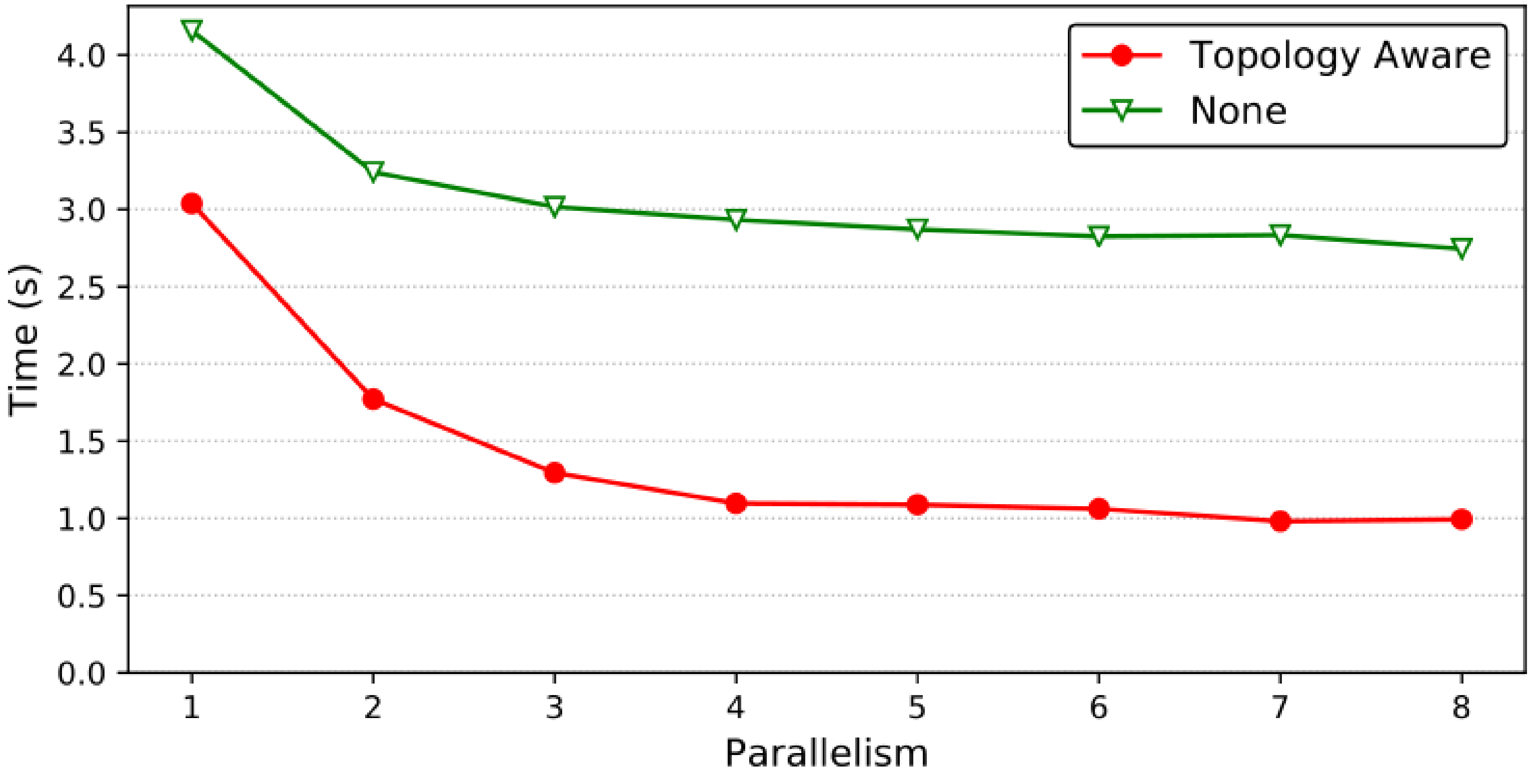
Evaluation

- 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.



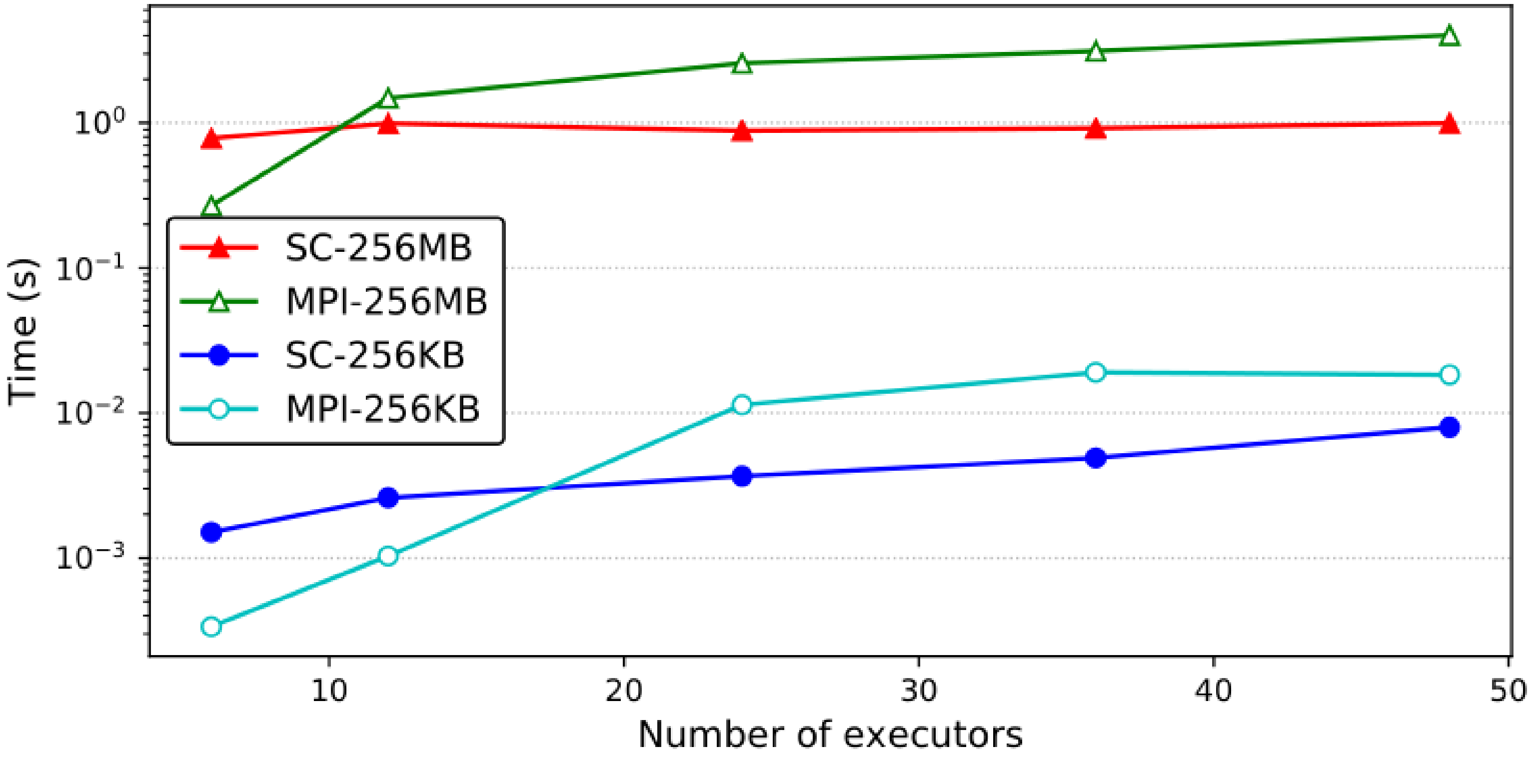
Evaluation

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



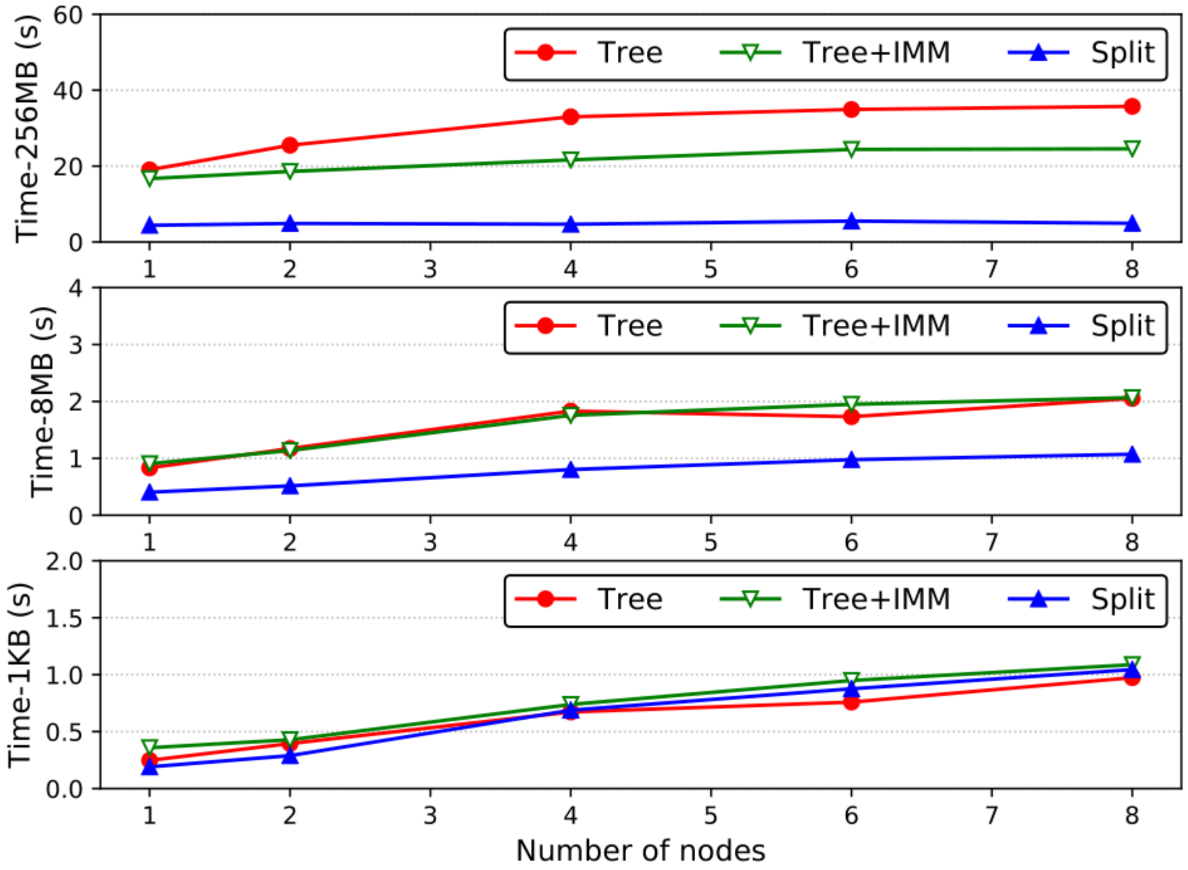
Evaluation

- 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)



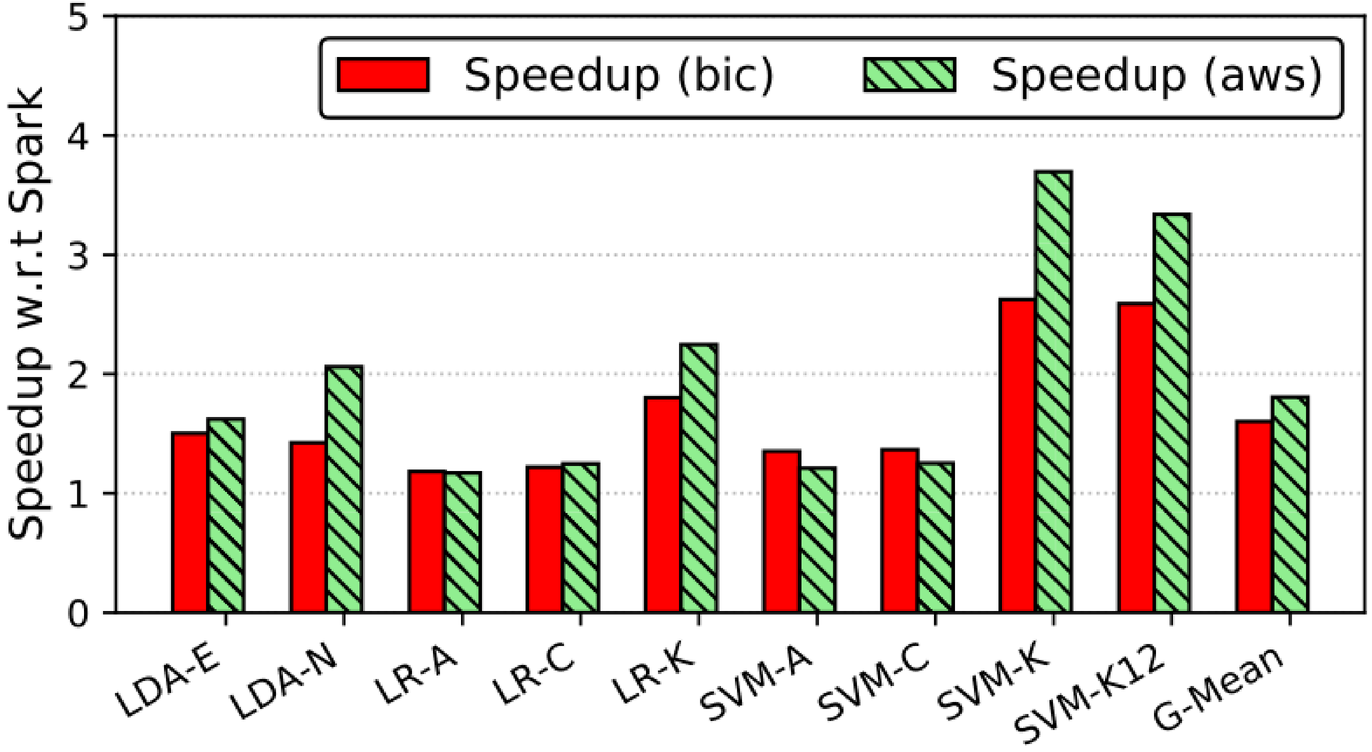
Evaluation

- 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.



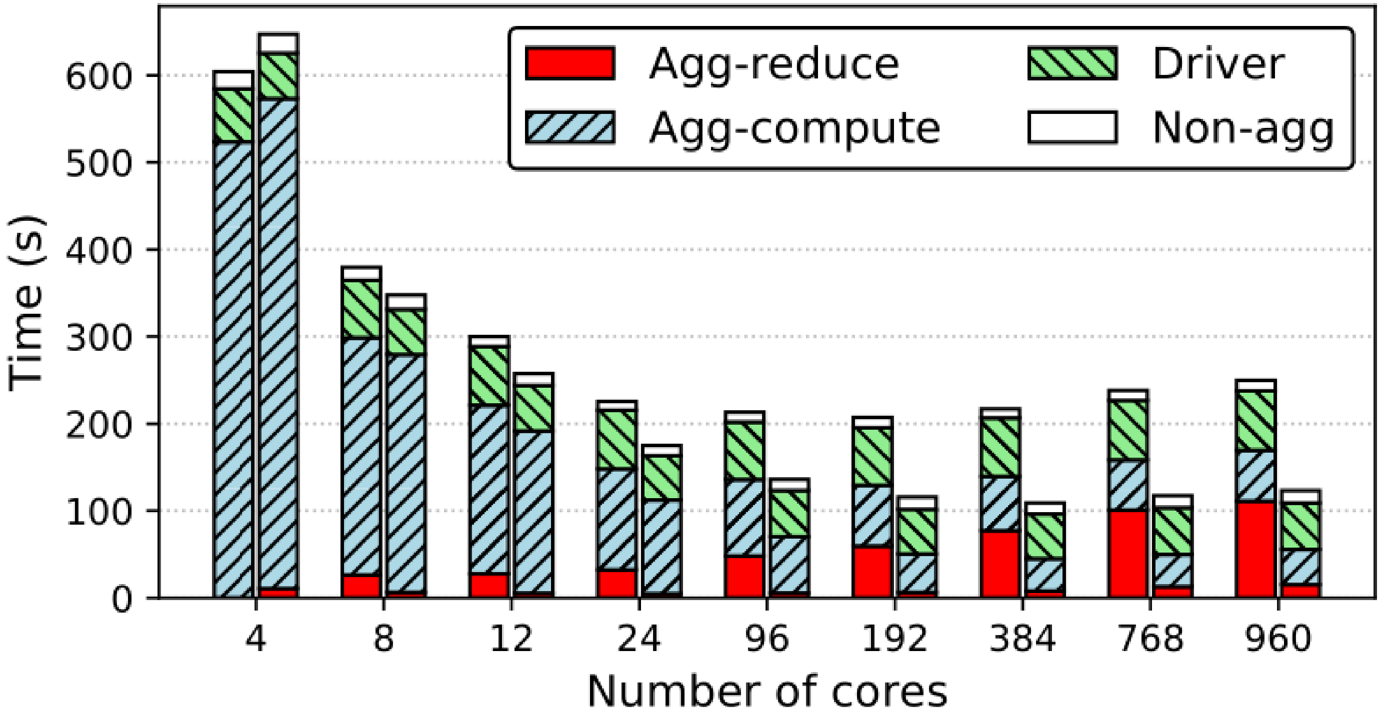
Evaluation

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



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

清華大學 Tsinghua University

UC San Diego