An Efficient Task-based All-Reduce for Machine Learning Applications

Zhenyu Li, James Davis, Stephen Jarvis

University of Warwick

WARWICK
THE UNIVERSITY OF WARWICK

WARWICK
THE UNIVERSITY OF WARWICK

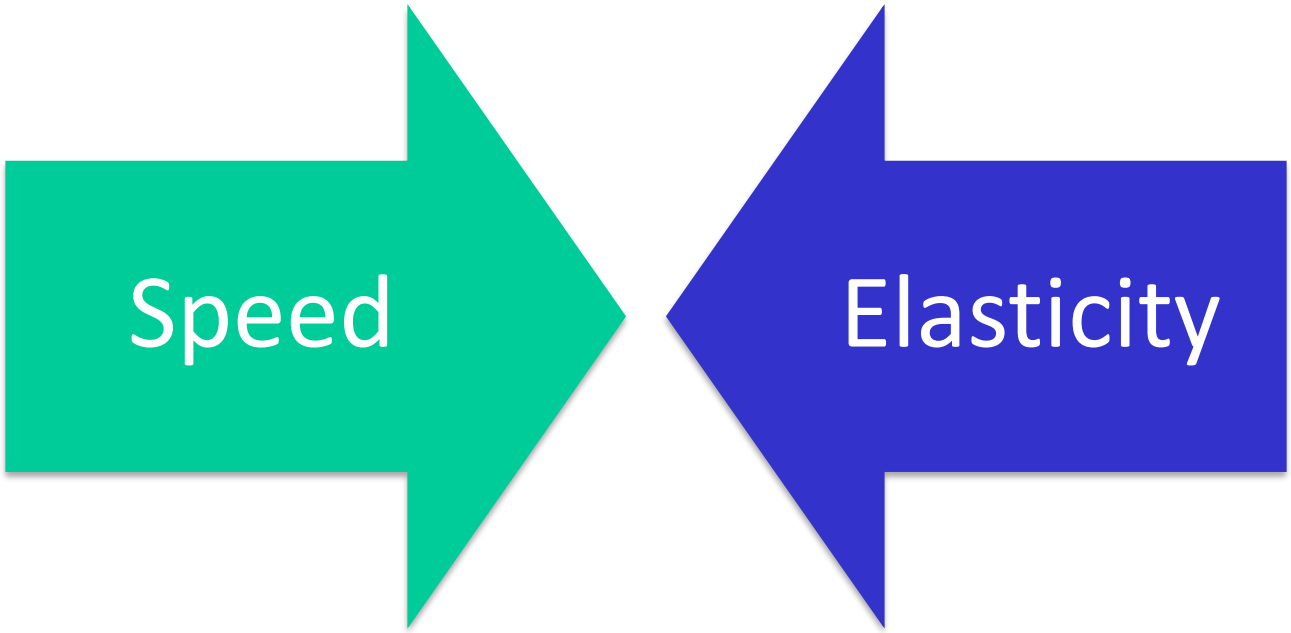
Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

Data Analytics & Scientific Computing



Data Analytics & Scientific Computing

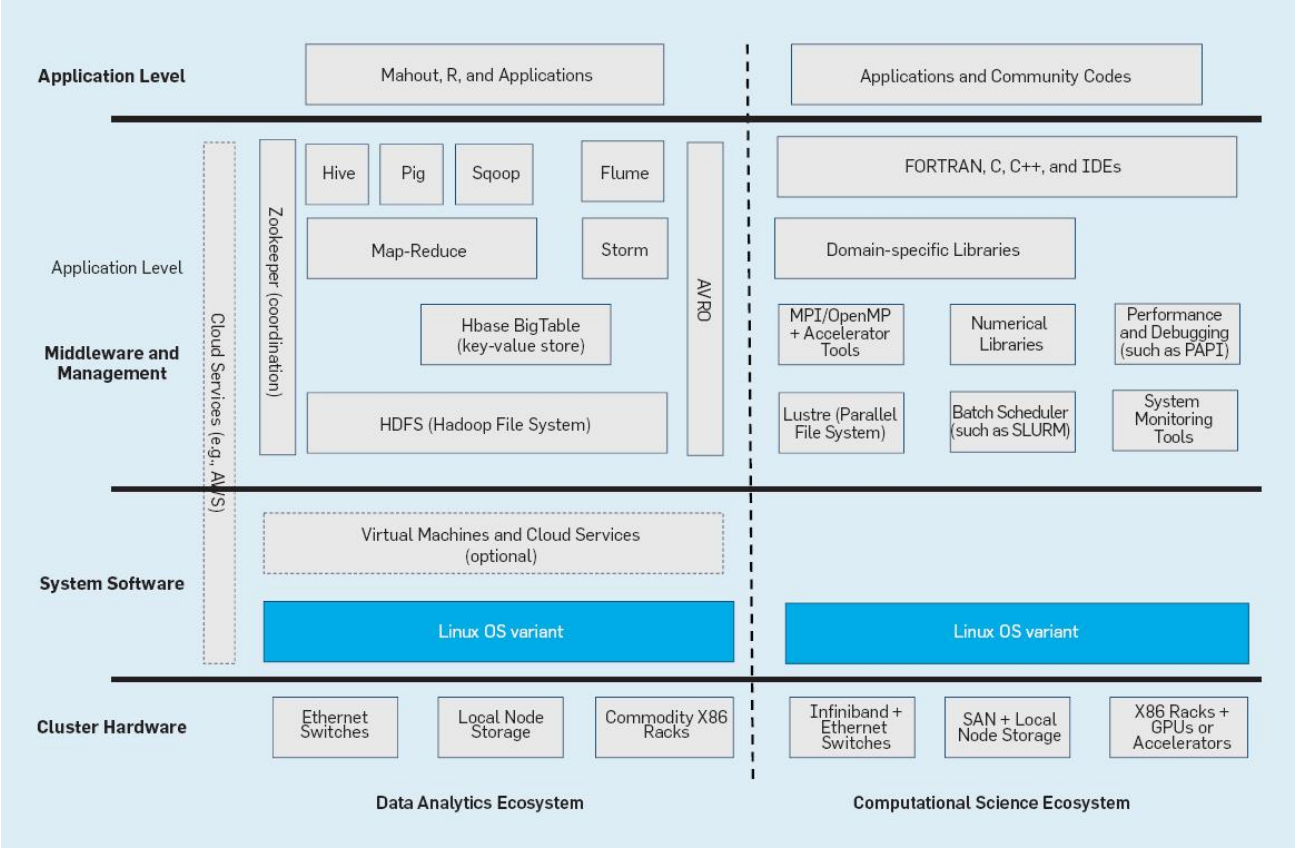
- Memory, I/O Bound
- Scale-Out/Horizontal
- Asynchronous Tasks
- Elastic Allocation

- CPU Bound
- Scale-Up/Vertical
- Parallel Processes
- Static Allocation (MPI 1)

Data Analytics & Scientific Computing

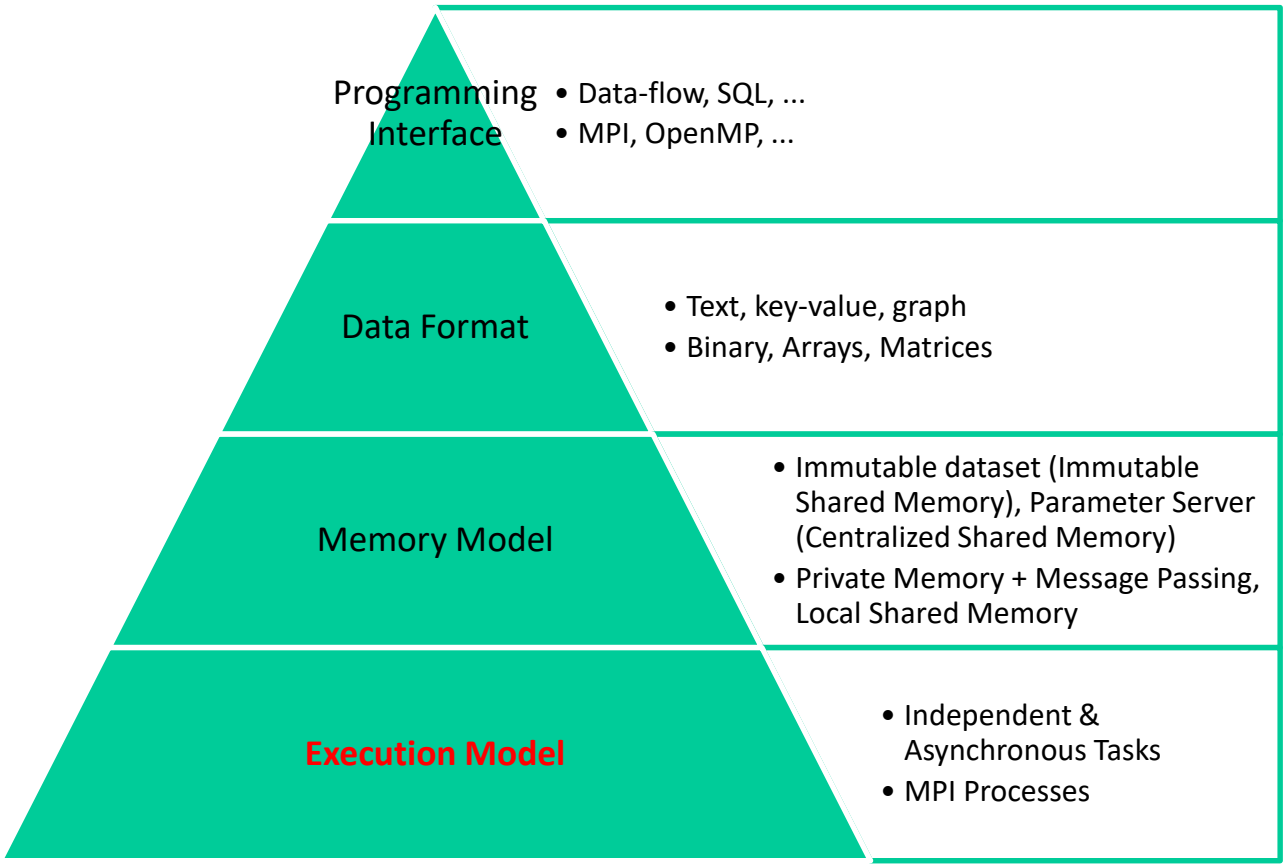
Data Analytic	Scientific Computing
Memory, I/O Bound	CPU Bound
Scale-Out/Horizontal	Scale-Up/Vertical
Asynchronous Tasks	Parallel Processes
Elastic Allocation	Static Allocation (MPI 1)

Data Analytics & Scientific Computing

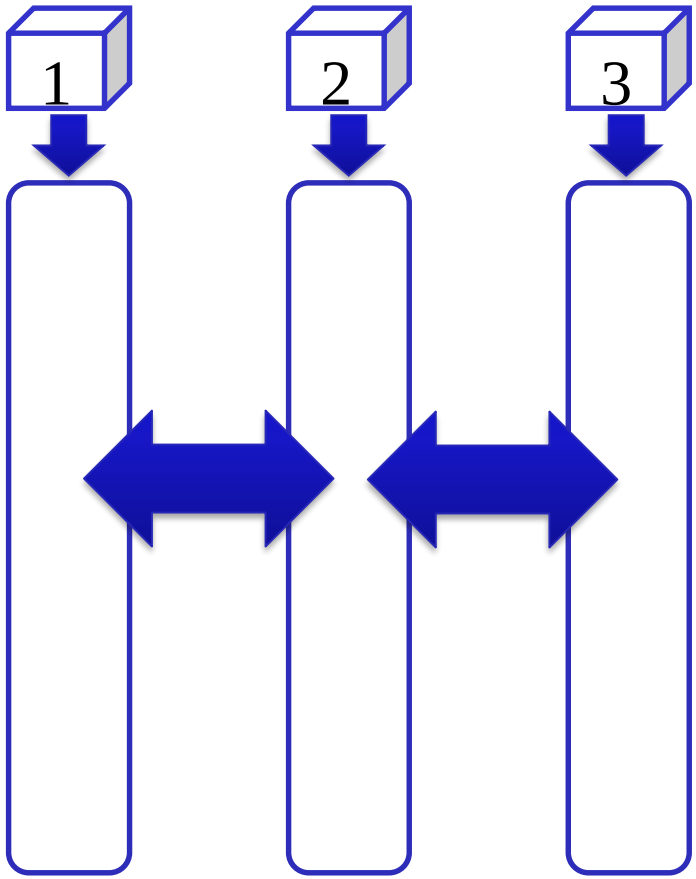


Source: Daniel A. Reed and Jack Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (June 2015), 56-68. DOI: https://doi.org/10.1145/2699414

Data Analytics & Scientific Computing

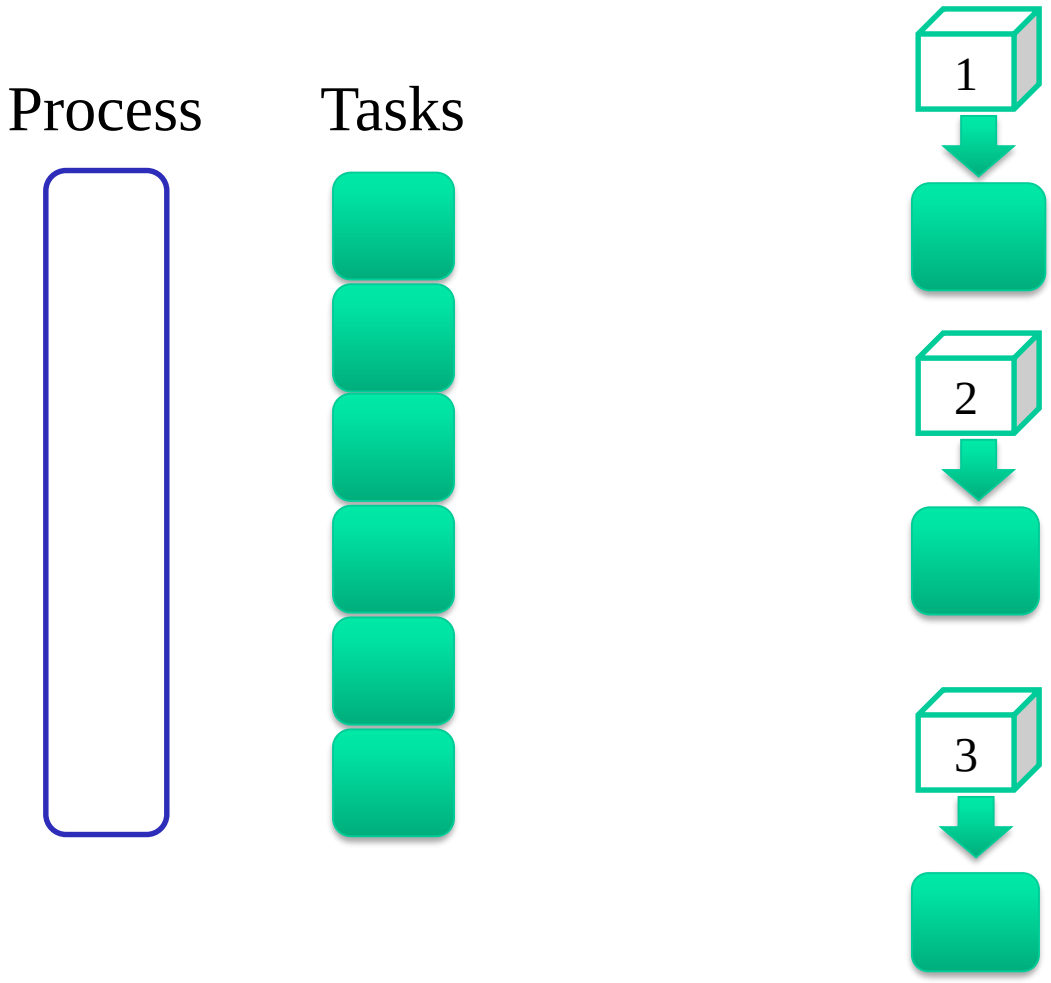


Data Analytics & Scientific Computing



SIMD: Single Instruction, Multiple Data

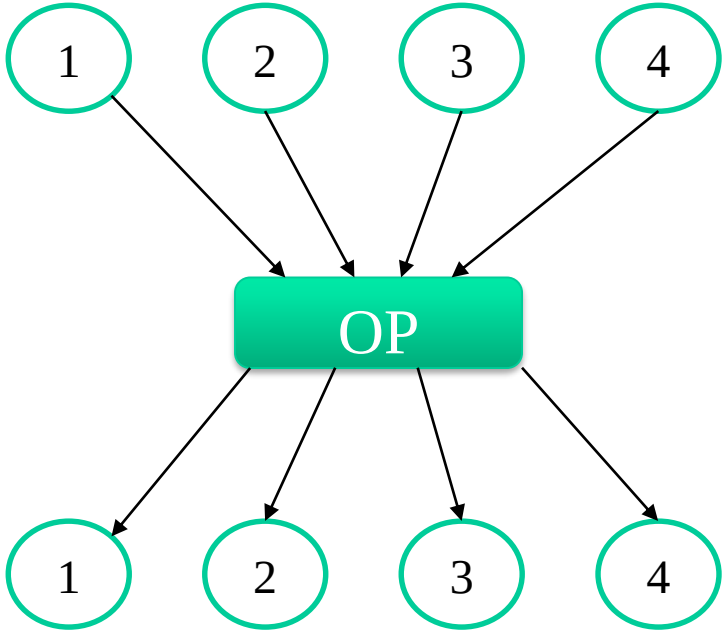
Data Analytics & Scientific Computing



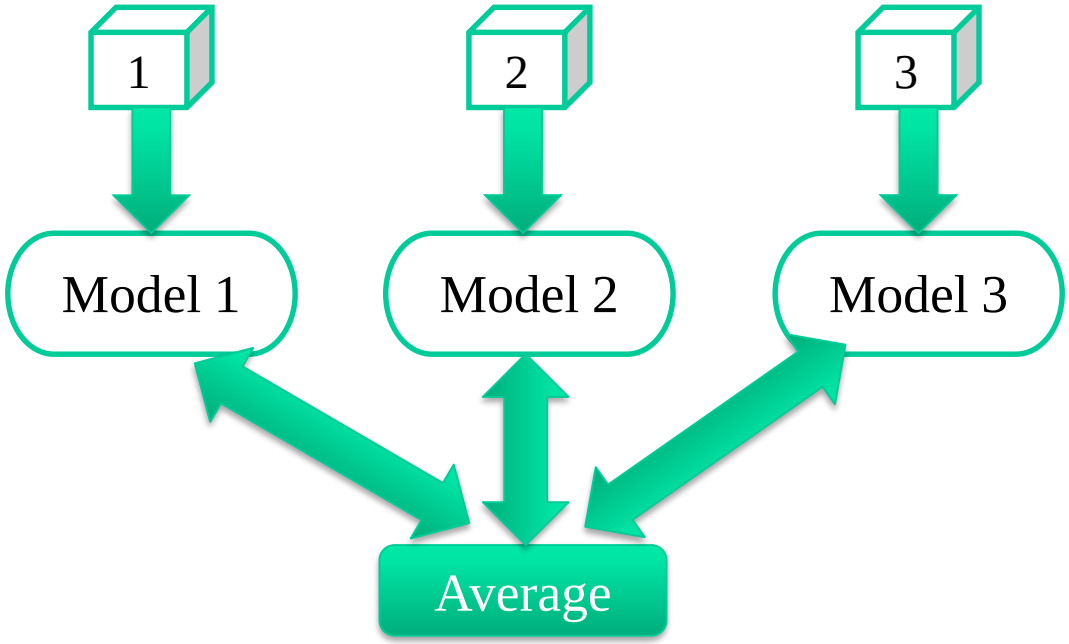
Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

What is All-Reduce?



Distributed Model Training



Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

All-Reduce with MPI

Principles

- Vector halving & doubling
- Distance halving & doubling
- Ring algorithms

Factors

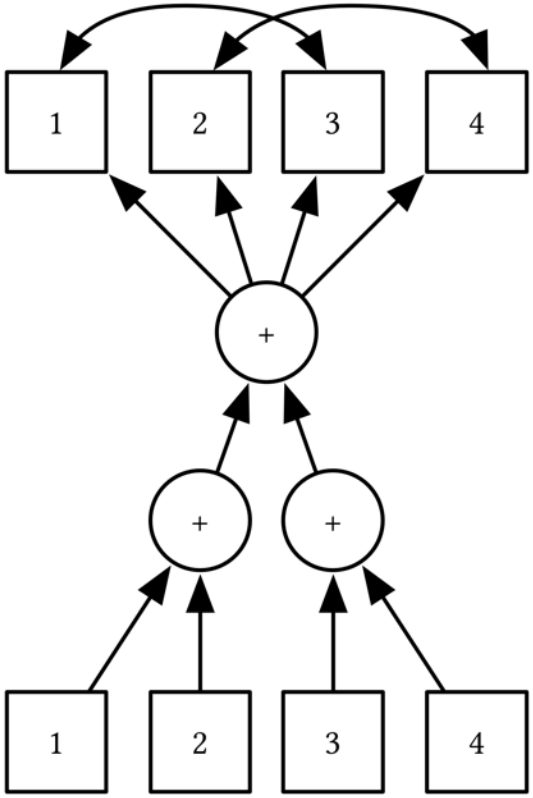
- Vector length
- Number of nodes
- Network bandwidth and latency
- Network topology / Degree of freedom

Algorithms

- Binary Tree
- Recursive doubling / Butterfly
- Recursive halving & doubling
- ...

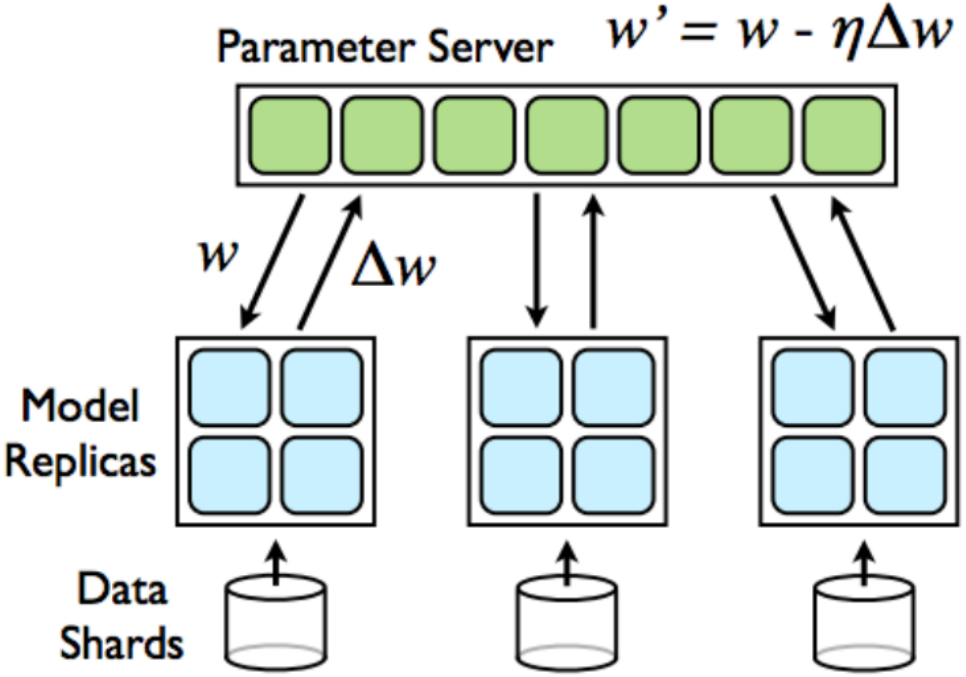
Reference: Rabenseifner R. Optimization of collective reduction operations. InInternational Conference on Computational Science 2004 Jun 6 (pp. 1-9). Springer, Berlin, Heidelberg.

All-Reduce with DataFlow



Reduce - Broadcast

All-Reduce with Parameter Servers



Updating weights with Parameter Servers

Image Reference: Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, Senior A, Tucker P, Yang K, Le QV, Ng AY. Large scale distributed deep networks. InAdvances in neural information processing systems 2012 (pp. 1223-1231).

Butterfly All-Reduce with Spark?

"the butterfly pattern introduces complex dependency that slows down the computation"

Why?

- Creates new RDD at each stage
- Shuffle & Dependencies
- Scheduling Overhead (Task Start-up and Result Collection)
- Synchronization Overhead

Reference: https://github.com/apache/spark/pull/506

Problem?

Continue with reduce – broadcast.

Use more parameter servers.

Use MPI, but assume tasks are parallel.

Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

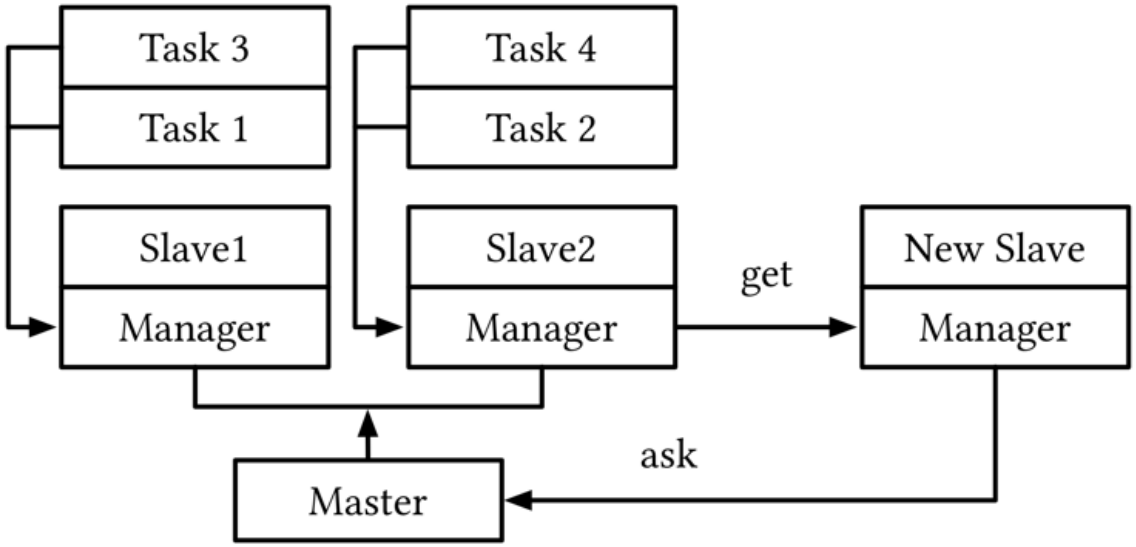
All-Reduce for Tasks

What are we trying to address?

1. All-Reduce in a Task-based setting
2. Alternative all-reduce algorithms
3. Exploit finer-grained parallelism

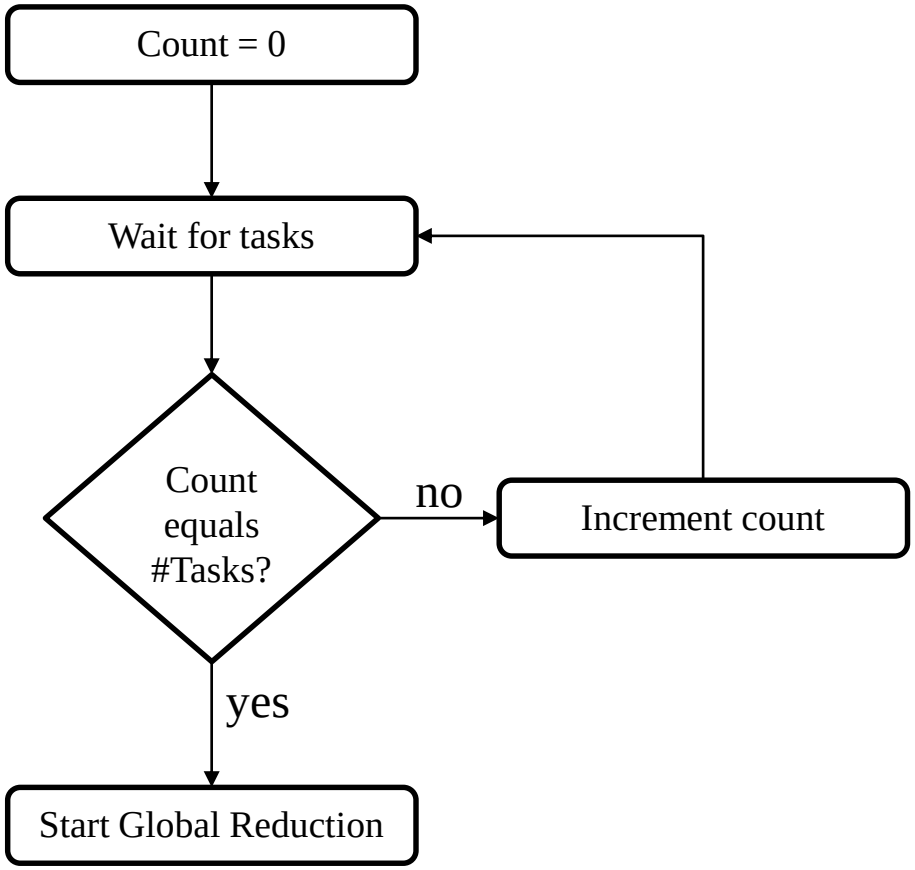
All-Reduce in a Task-based setting

Overall Architecture

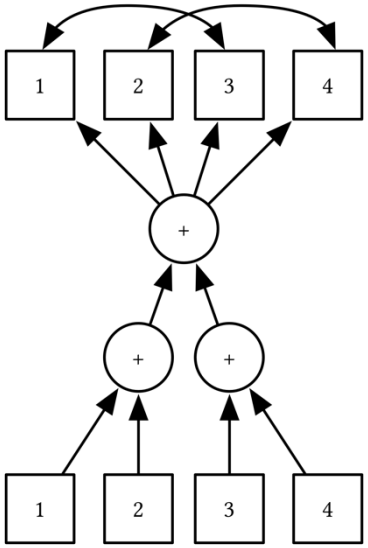


All-Reduce in a Task-based setting

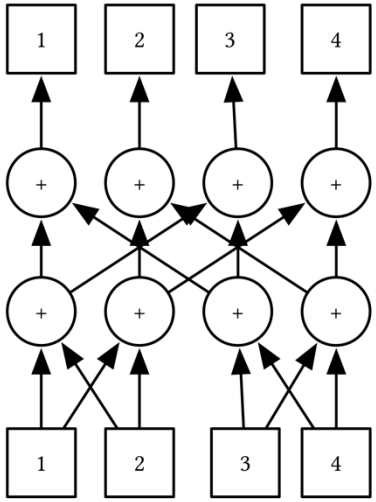
Driver Side Implementation



Alternative all-reduce algorithms

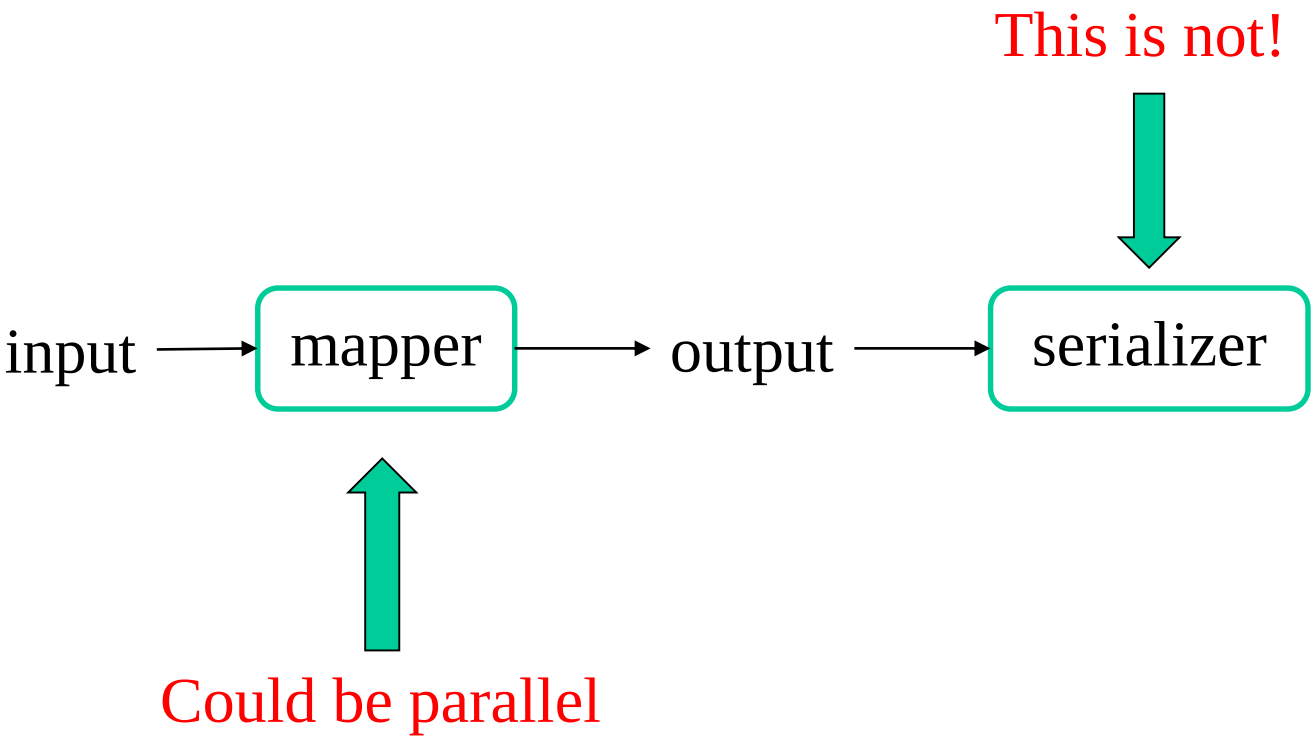


Reduce - Broadcast

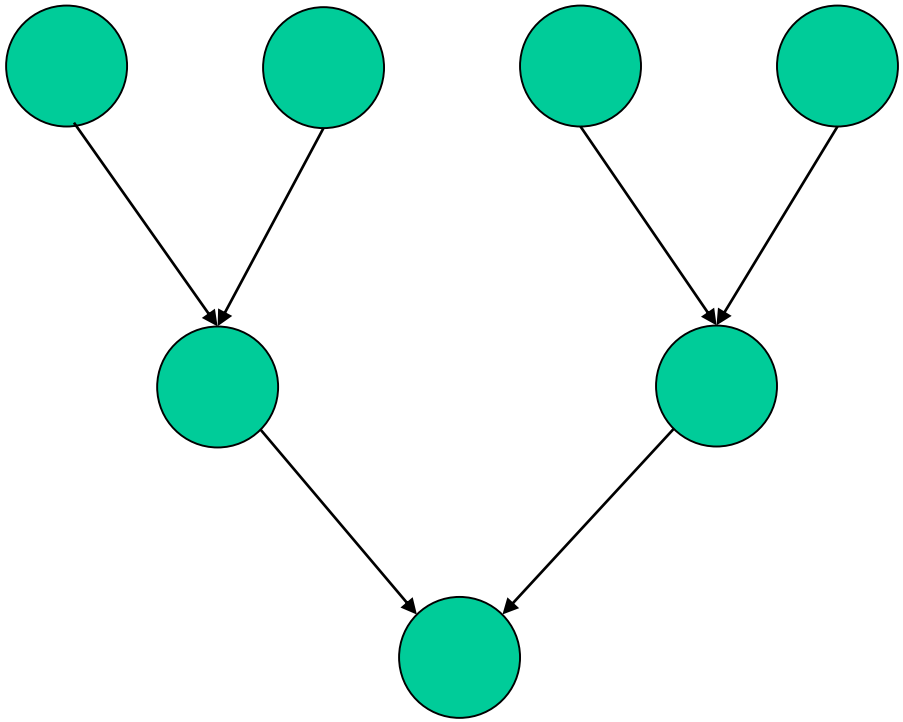


Butterfly

Finer-grained parallelism



Finer-grained parallelism

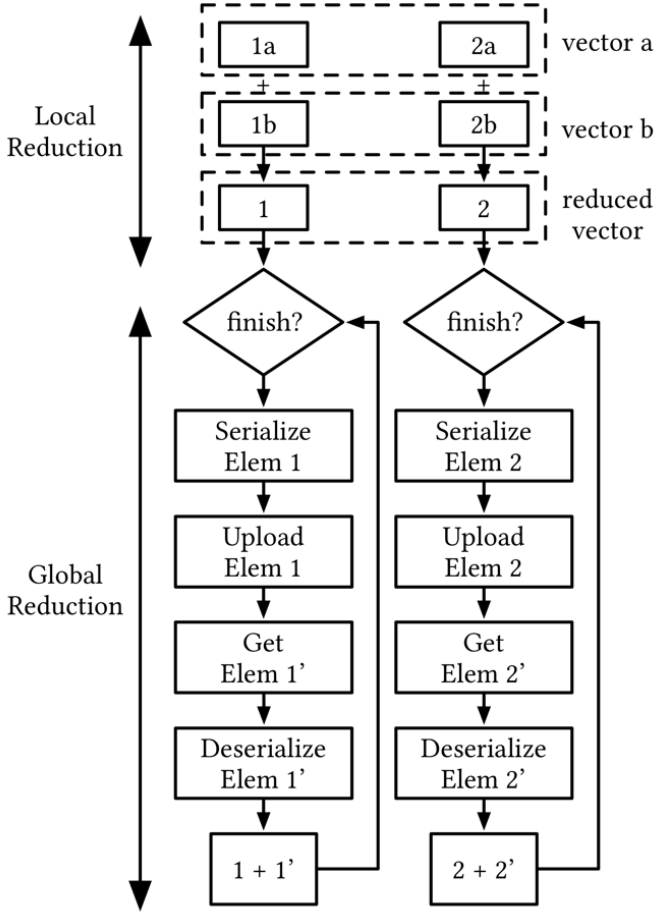


Finer-grained parallelism

Vector Interface

- *Init(key, numTasks, func):* Creates a shared variable for the given key with the number of tasks and a reduction function, the context of all-reduce is maintained by the returned handle;
- *Commit(vector):* Commits a vector for reduction, the func- tion does not block;
- *Get:* Get the globally reduced vector, block until comple- tion;

Finer-grained parallelism



Auto-Parallelization

Outline

1. Background
    1. Data Analytic & Scientific Computing (Tasks vs. Message-Passing)
    2. All-Reduce & Machine Learning
    3. All-Reduce Implementations (Apache Spark, Tensorflow, MPI)
2. Problem?
3. All-Reduce for Tasks
    1. Algorithm
    2. Interface & Architecture
    3. Implementation
4. Results
    1. Empirical Results
    2. Neural Network Results
5. Conclusion & Future Work

Experiment Setup

Table 1: Hardware & Software Specification of the Test Cluster

Component	Detail
Nodes	1 Driver Node, 32 Executor Nodes
Cores per Node	20
CPU	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz
Memory	64GB
Harddisk	Locally Attached (HDD & SSD)
Interconnect	Mellanox Technologies MT26428
Software	Centos/Linux-2.6, Hadoop 2.7, Spark-2.1.1

Empirical Performance

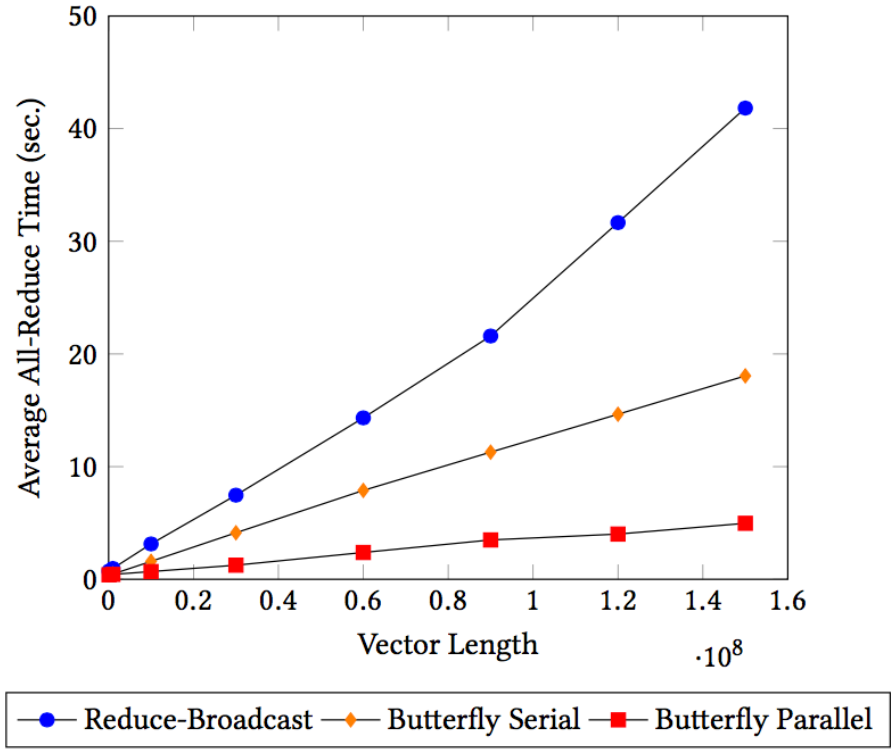


Figure 4: Average All-Reduce Performance on 32 Executors for a Single Iteration

Empirical Performance

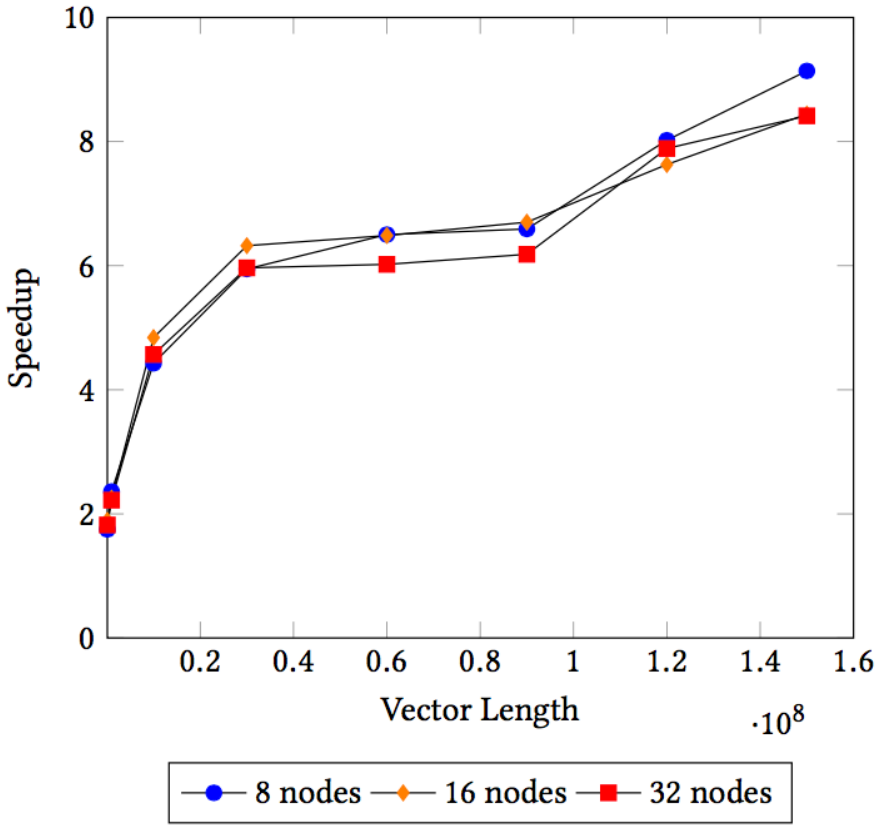


Figure 5: Speed-up of Parallel Butterfly w.r.t Tree-Reduce+Broadcast on 8, 16, 32 nodes

Empirical Performance

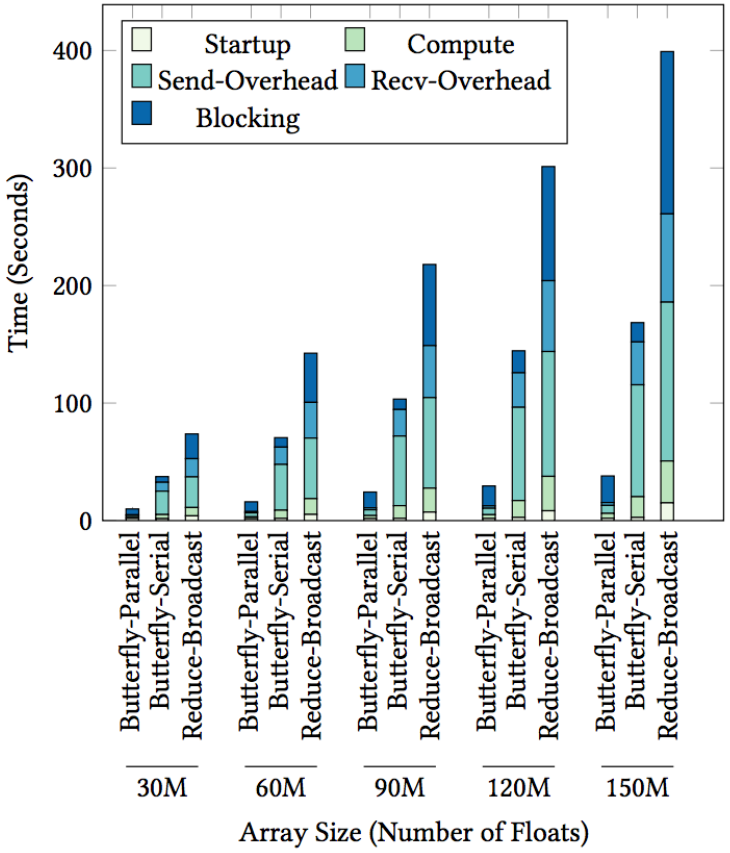


Figure 6: Breakdown of overheads in all-reduce of a large array size for 10 iterations on a 32-node cluster

Application – Neural Net Training

Table 3: All-reduce time in real-world neural network applications across 32 nodes. Original: Reduce-broadcast. New: Butterfly all-reduce.

Dataset	Neural Net	Weight size – log length	Original (sec.)	New (sec.)
Cifar [12]	cuda-convnet [5]	5.2	0.356	0.154
Mnist [16]	LeNet [15]	5.6	0.447	0.184
ImageNet [9]	AlexNet [14]	7.8	17.9	2.4

Conclusion & Future Work

- An all-reduce for task-based frameworks
- Effectiveness, 2x speed-up on small datasets (Cifar, Mnist), 7x speed-up on large datasets (ImageNet)
- Further optimization through RDMA, vector compression
- Further research in the context of dynamic resource allocation

Thank you!

Acknowledgement

This research is supported by Atos IT Services UK Ltd and by the EPSRC Centre for Doctoral Training in Urban Science and Progress (grant no. EP/L016400/1