MALT: Distributed Data-Parallelism for Existing ML Applications

Hao Li, Asim Kadav, Erik Kruus, Cristian Ungureanu
asim@nec-labs.com

ML transforms data into insights

Large amounts of data is being generated by user-software interactions, social networks, and hardware devices.

Timely insights depend on providing accurate and updated machine learning (ML) models using this data.

Large learning models, trained on large datasets often improve model accuracy [1].

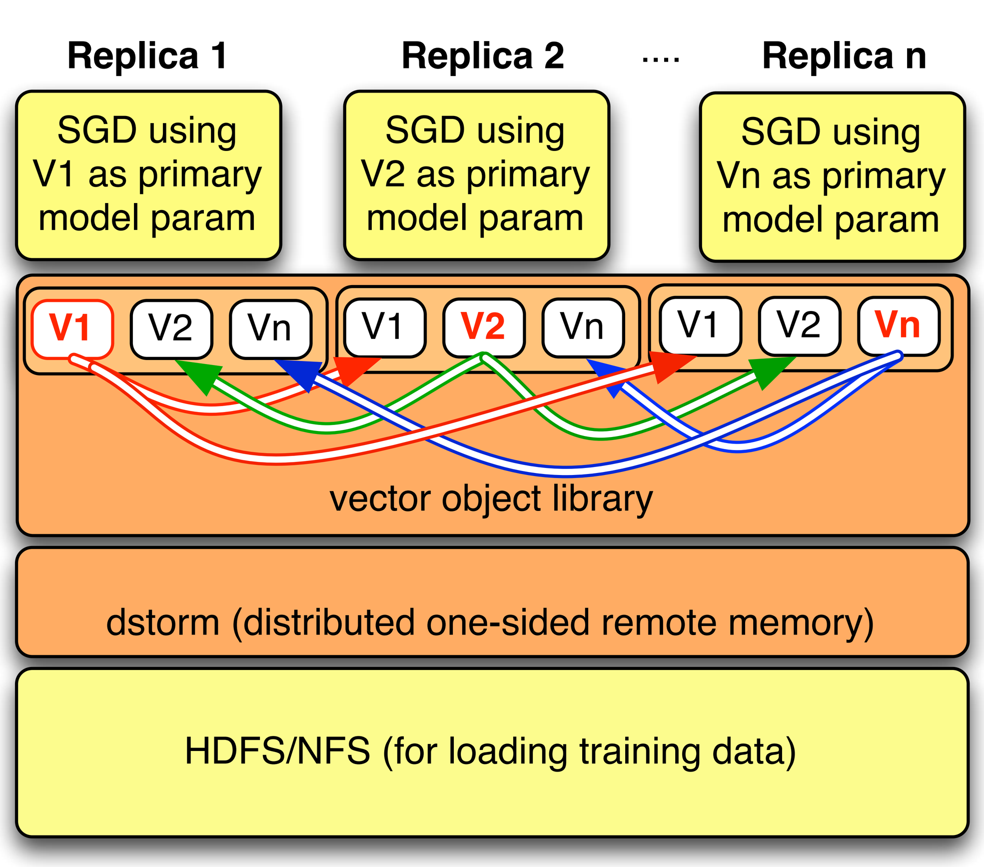
Properties of ML applications

Machine learning tasks have all of the following properties:

- Fine-Grained and Incremental: ML tasks perform repeated model updates over new input data.
- Asynchronous: ML tasks may communicate asynchronously. E.g. communicating model information, back-propagation etc.
- Approximate: ML applications are stochastic and often an approximation of the trained model is sufficient.
- Need Rich Developer Environment: Developing ML applications requires a rich set of libraries, tools and graphing abilities which is often missing in many highly scalable systems.

Our Solution: MALT

Goal: Provide an efficient library for providing data-parallelism to existing ML applications.

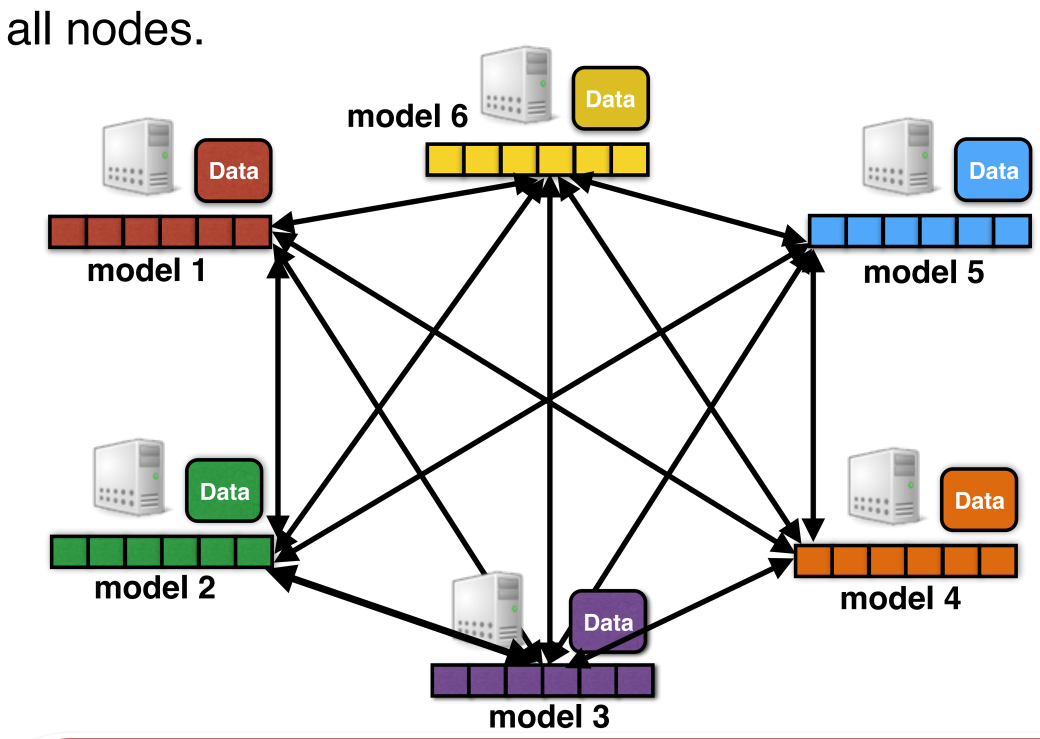


MALT performs peer-to-peer machine learning. It provides abstractions for fine-grained in-memory updates using one-sided RDMA, limiting data movement costs when training models. MALT allows machine learning developers to specify the dataflow and apply communication and representation optimizations.

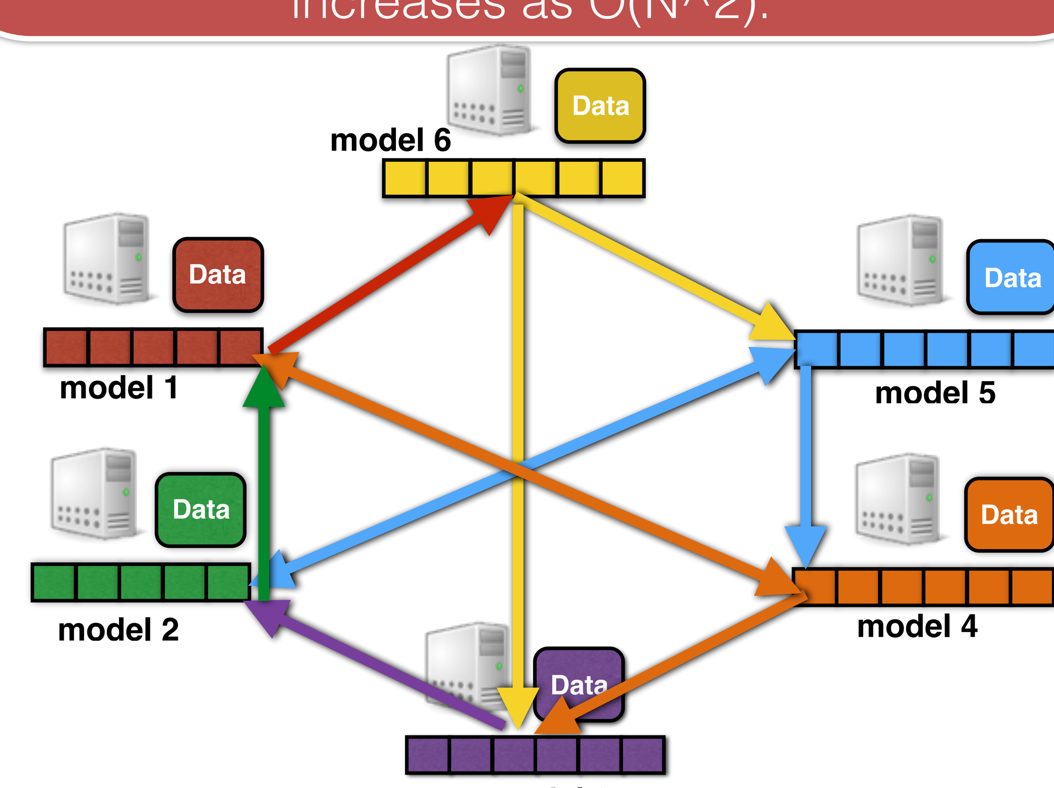
Design requirements	MALT's solution
Efficient model communication	MALT provides a **scatter-gather** API. **scatter** allows sending of model updates to the peer replicas. Local **gather** function applies any user-defined function on the received values.
Asynchronous	Models train and **scatter** updates to per-sender receive queues. This mechanism when used with one-sided RDMA writes, ensure no interruption to the receiver CPU.
Approximate	MALT allows different consistency models to trade-off consistency and training time.
Re-use developer environment	Works with existing applications. Currently integrated with SVM-SGD, HogWild-MF and NEC RAPID.

Network-efficient learning

In a peer-to-peer learning, instead of sending model info. to all replicas, MALT sends model updates to log(N) nodes, such that (i) the graph of all nodes is connected (ii) the model updates are disseminated uniformly across all nodes.



Traditional: all-reduce exchange of model information. As number of nodes (N) increase, the total number of updates transmitted in the network increases as O(N^2).



MALT model propagation: Each machine sends updates to log(N) nodes (to N/2 + i and N/4 + i for node i). As N increases, the outbound nodes follows Halton sequence (N/2, N/4, 3N/4, N/8, 3N/8..).and the total number of updates transmitted increases as O(N logN).

Serial SGD

```
procedure SERIALSGD
Gradient g;
Parameter W;

for epoch = 1 : maxEpochs do

  for i = 1 : maxData do
    g = cal gradient(data[i]);
    W = W + g;

return W
```

Transforming serial SGD (Stochastic Gradient Descent) to data-parallel SGD.

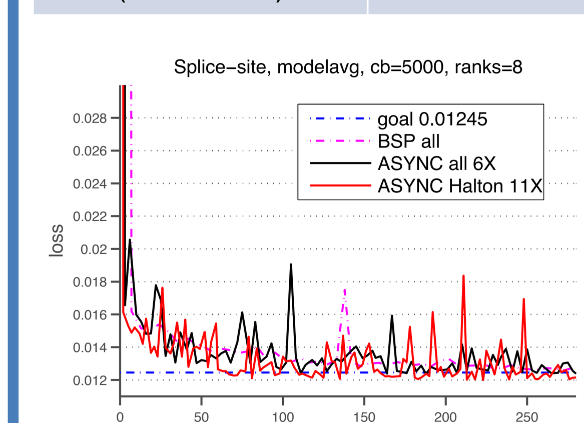
```
procedure PARALLELSGD
  maltGradient g(SPARSE, ALL);
  Parameter W;

  for epoch = 1 : maxEpochs do

    for i = 1 : maxData/totalMachines do
      g = cal gradient(data[i]);
      g.scatter(ALL);
      g.gather(AVG);
      W = W + g;

return W
```

Data-Parallel SGD with MALT

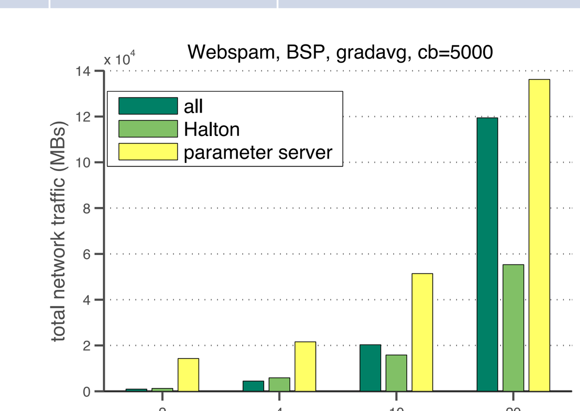
Results

We integrate MALT with three applications: SVM[3], matrix factorization[4] and neural network[5]. MALT requires reasonable developer efforts and provides speedup over existing methods.

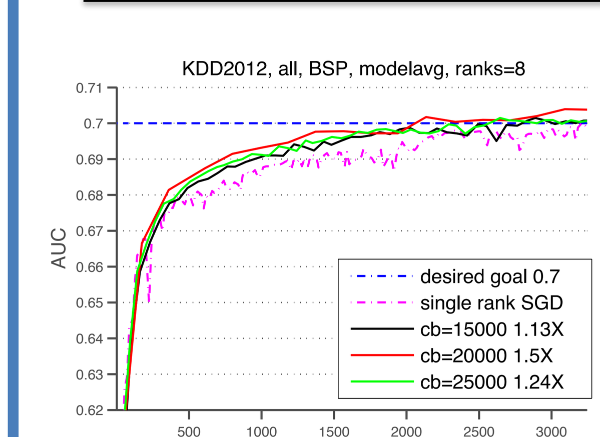
Application (Dataset)	Model	# Parameters	Dataset size (uncompressed)
Document Classification (RCV1)	SVM	47K	480 MB
Image classification (PASCAL - alpha)	SVM	500	1 GB
DNA detection (DNA)	SVM	800	10 GB
Genome detection (splice-site)	SVM	11M	250 GB
Webspam detection (webspam)	SVM	16.6M	10 GB
Collaborative filtering (netflix)	Matrix Factorization	14.9M	1.6 GB
Ad prediction (KDD 2012)	Neural networks	12.8M	3.1 GB



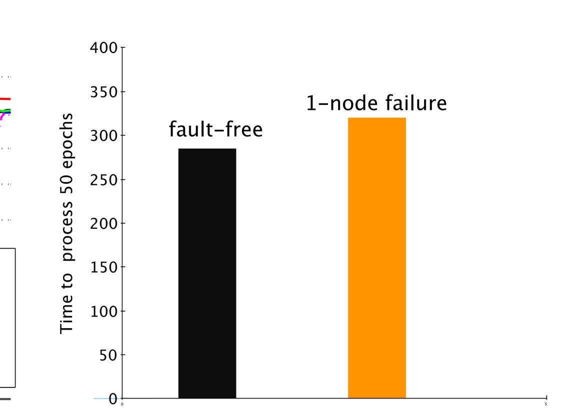
SVM Convergence (loss vs time in seconds) for SVM using splice dataset for MALT-all and MALT-Halton. We find that MALT-Halton converges faster than MALT-all.



Network costs for MALT-all, MALT-Halton and the parameter server for the whole network for the webspam workload. We find that MALT-Halton reduces network communication costs and provides fast convergence.



Neural networks AUC (area under curve) vs time (in seconds) for a three layer neural network for text learning (click prediction) using KDD 2012 data.



Fault tolerance: Time taken to converge for the DNA dataset with fault-free and a single rank failure case. MALT is able to recover from the failure and train the model correctly.

We demonstrate that MALT outperforms single machine performance for small workloads and can efficiently train models over large datasets that span multiple machines (See our paper in EuroSys 2015 for more results).

References and Related Work

[1] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. Intelligent Systems, IEEE, 24(2):8–12, 2009.
[2] J. Dean et. al., Large scale distributed deep networks, NIPS 2012.
[3] L. Bottou. Large scale machine learning with SGD. COMPSTAT 2010.
[4] B. Recht et. al., HogWild: A lock free approach to parallelizing stochastic descent, NIPS 2011.
[5] B. Bai e.t. al. SSI: Supervised Semantic Indexing. ACM CIKM 2009.