INDIANA UNIVERSITY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Performance Optimization on Model Synchronization in Parallel Stochastic Gradient Descent Based SVM

Vibhatha Abeykoon, Geoffrey Fox, Minje Kim
Digital Science Center
USA

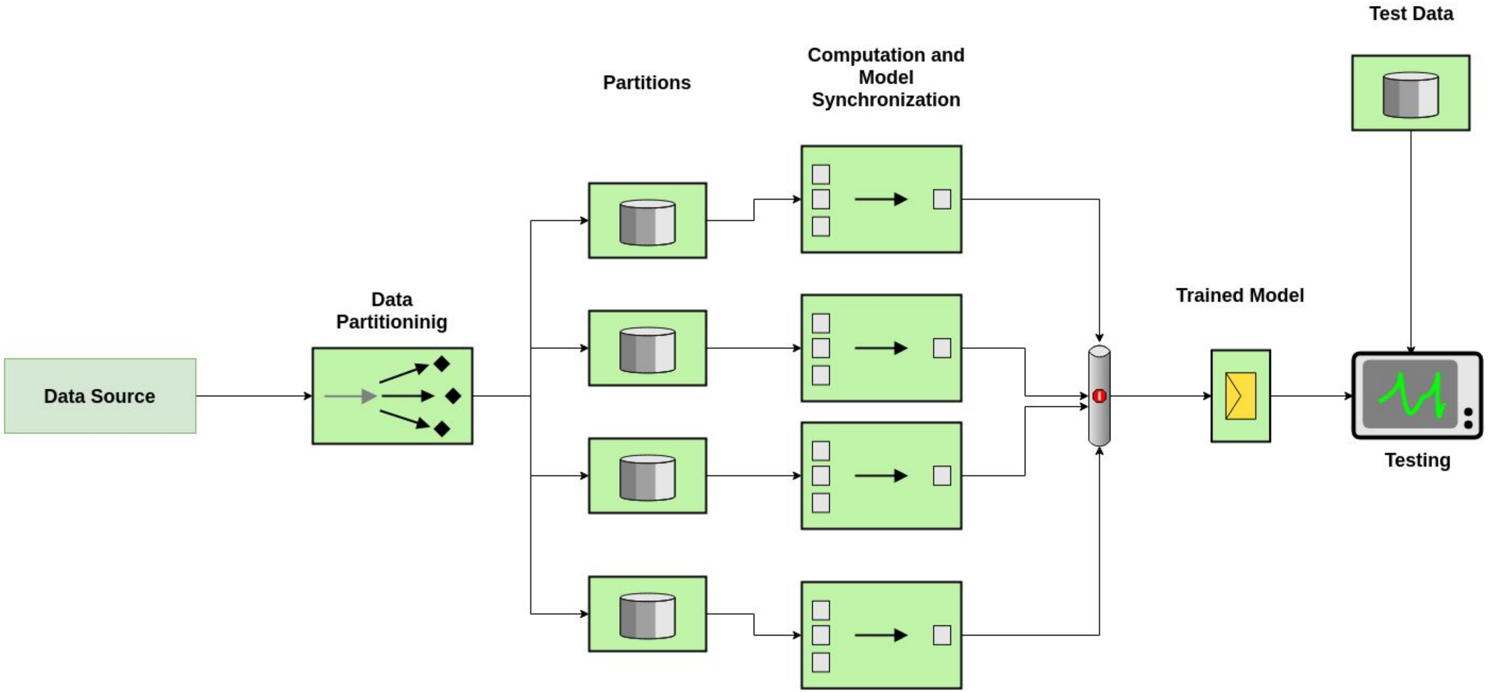
Related Work

- Pegasos SVM
- DC-SVM
- pPackSVM
- Parallel SGD
- Parallel SGD For High Level Architectures

Objective

- Effect of mini-batch based model synchronization on SGD based SVM algorithm convergence.
- Evaluate efficiency of the training model based on execution time and testing accuracy upon batch size.

System Architecture



Anatomy of Datasets

DataSet	Training Data (60% / 80%)	Cross-Validation Data (60% / 80%)	Testing Data (60%, 80%)	Sparsity(%)	Features
Ijcnn1	21,000 / 28,000	7,000 / 3,500	7,000 / 3,500	40.91	22
Webspam	210,000 / 280,000	70,000 / 35,000	70,000 / 35,000	99.9	254
Epsilon	240,000 / 320,000	80,000 / 40,000	80,000 / 40,000	44.9	2000

Objective Function and Equations

$$J^t = \min_{w \in R^d} \frac{1}{2} ||w||^2 + C \sum_{x,y \in S} g(w; (x, y))$$

$$g(w; (x, y)) = \max(0, 1 - y\langle w|x\rangle)$$

$$w = w - \alpha \nabla J^t, \ \ \alpha = \frac{1}{1+t}$$

$$\nabla J^t = \begin{cases} w & \text{if } \max(0, 1 - y\langle w|x\rangle) = 0 \\ w - C x_i y_i & \text{Otherwise} \end{cases}$$

$$y\langle w|x\rangle = y_i w^\top x_i$$

Key Factors

- Cross-Validation accuracy
  - Cross-validation accuracy defines how far the model-in-training is close towards the expected accuracy.
  - Greedy approach would overfit the model to the training data by deviating it from a higher testing accuracy.
  - Cross-validation can be done as soon as the model is being synchronized over the distributed models or ***per epoch***.
  - This is an expensive operation when the number of cross-validation samples are higher and the dimensionality of a datapoint is higher.

Key Factors

- Value of the Objective Function
  - The value of the objective function tells how far is the algorithm from convergence.
  - When this value is a less fluctuating value, we can determine the convergence of the algorithm.
  - This step is also expensive depending on the number of samples and features in a data point.
  - This is also can be calculated once a model synchronization is done over the distributed models or ***per epoch***.

Algorithm Initialization

- Weight vectors are initialized with a Gaussian distribution.
  - Inbuilt C++ libraries are used for this (uniform_real_distribution<0,1>).
- Training data are shuffled with a random algorithm before starting the training.

Algorithm Implementation

- We used OpenMPI 3.0.0 (C++)
- AllReduce collective was used to do model synchronization and later averaging was done over each process.
- Learning rate is an adaptive diminishing function.
  - Function of number of epochs

Distributed Algorithm

- Data is shuffled at distributed data loading
- Each machine receives an equal amount of data points for processing [guarantee the load balancing]
- Each distributed model is initialized with the same weight vector
- Distributed models are synchronized on the initial block size
- After each synchronization barrier, an allreduce is called to sum up the distributed models and the global model is gained by averaging through the number of machines used.
- Per synchronization, calls cross-validate() and calc-objective-value()

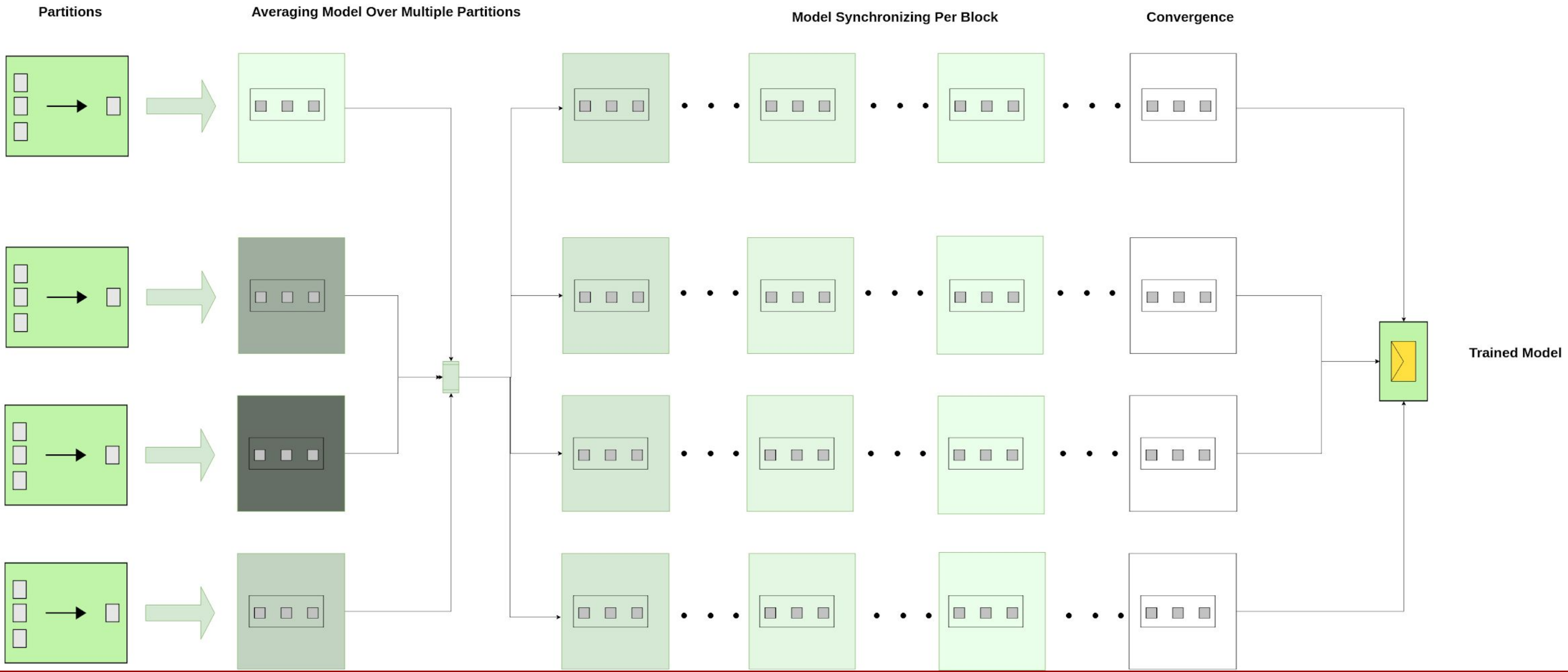
Distributed Algorithm

- Cross-validation calculation time is directly proportional to the number of cross-validation samples and number of features per data point.
- Objective function calculation time is also directly proportional to the number of cross-validation samples and number of features per data point.
- Each synchronization barrier is costly if this is done after processing data per the predefined block size (mini-batch size).

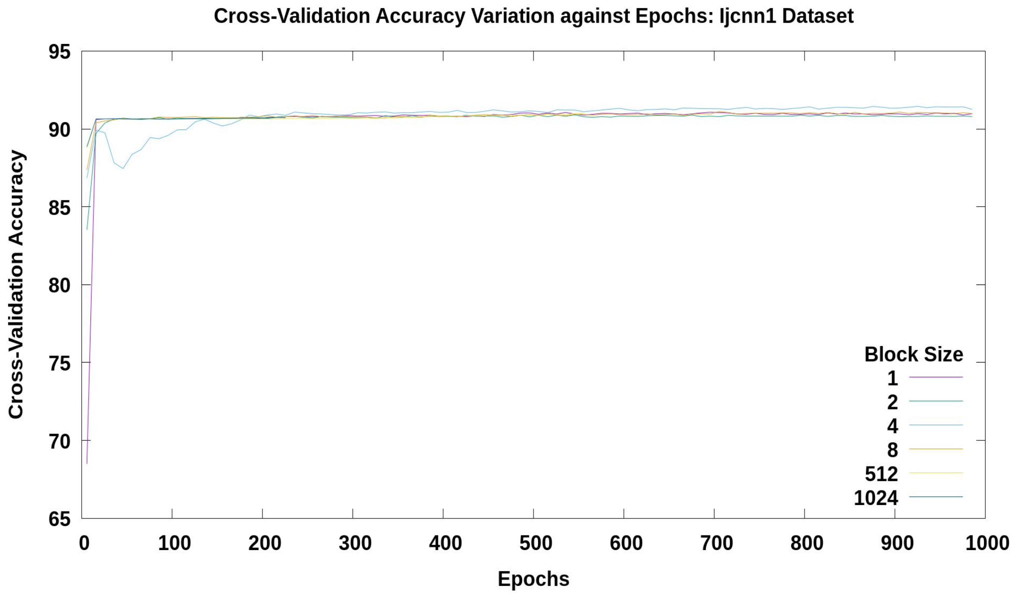
Model Update vs Cross-Validation

- Model update involves d_t (=d_all / (K)) amount of data points
- Cross-Validation involved d_c amount of data points
- d_all = all data points , K = number of machines , b = block size
- d_t data points per machine
- d_c data points per cross-validation
  - This can also be done in parallel and final accuracies can be averaged over distributed models (improves performance).
- Per epoch there is one cross-validation called and d_t / b number of calls of model synchronization
- A single cross validation step and model update step per data point roughly take same time per data point and block based calls priovides a gain.

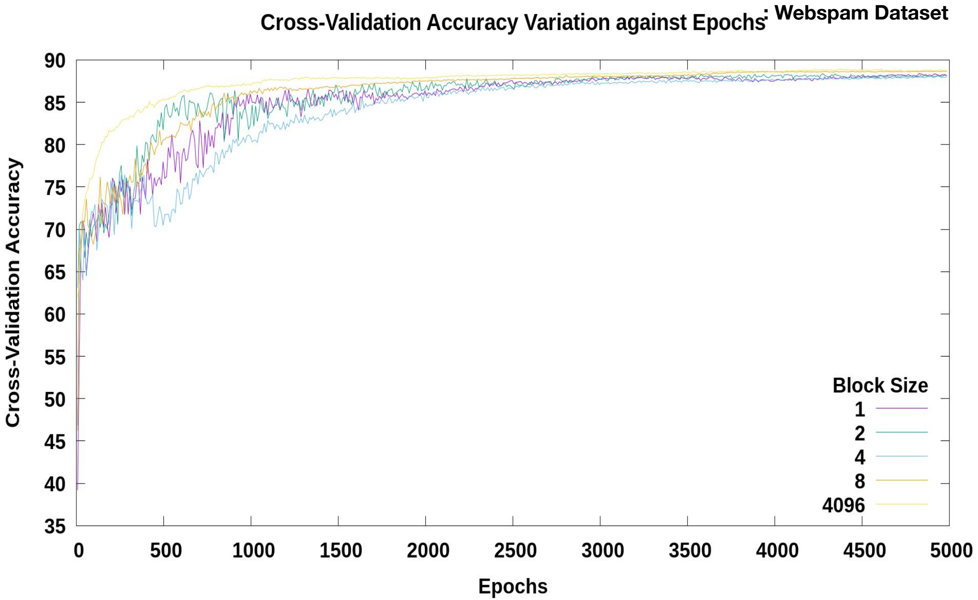
Model Synchronization



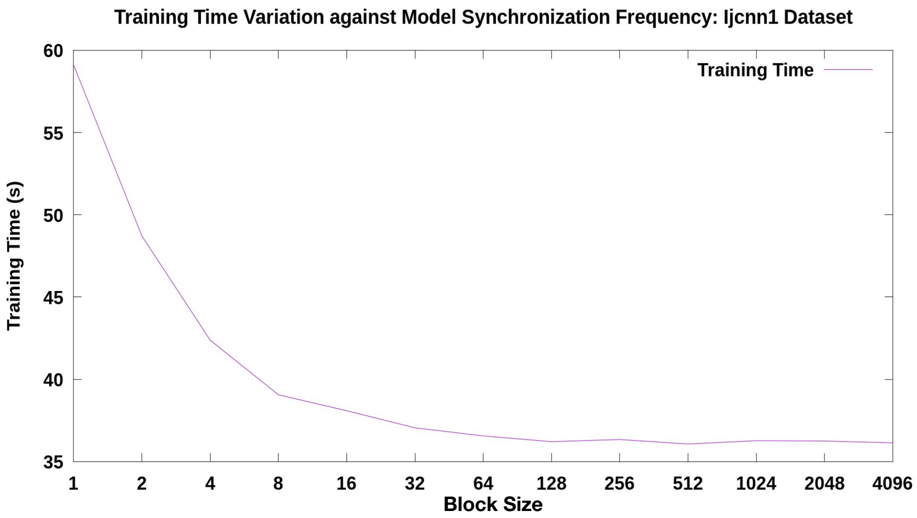
Cross Validation Accuracy Variation [Sequential Mode] - Ijcnn1 Dataset



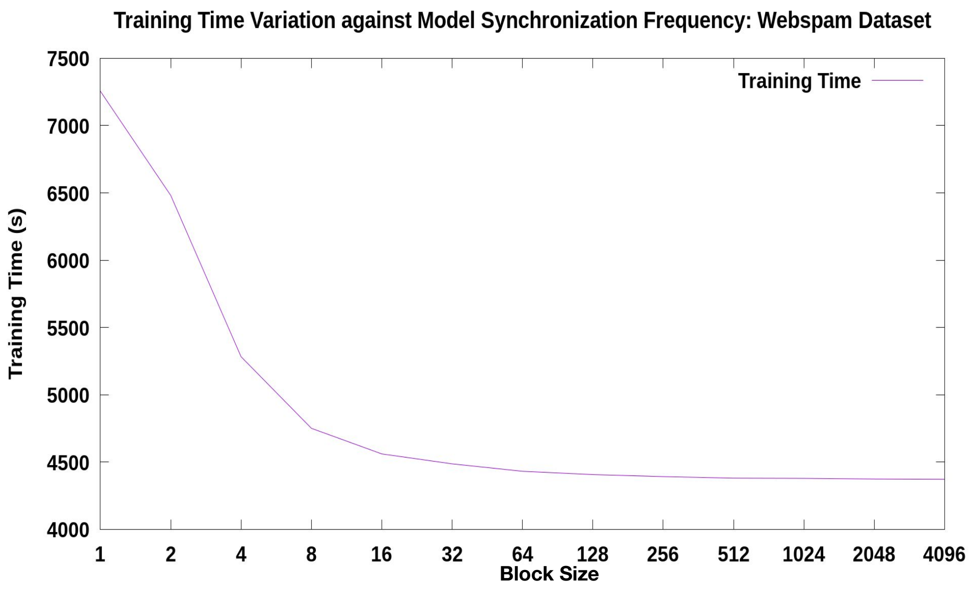
Cross Validation Accuracy Variation [Sequential Mode] - Webspam Dataset



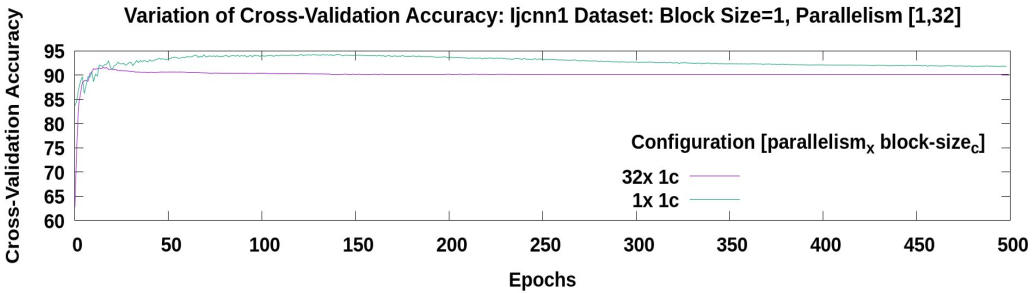
Training Time Variation [Sequential Mode] - Ijcnn1 Dataset

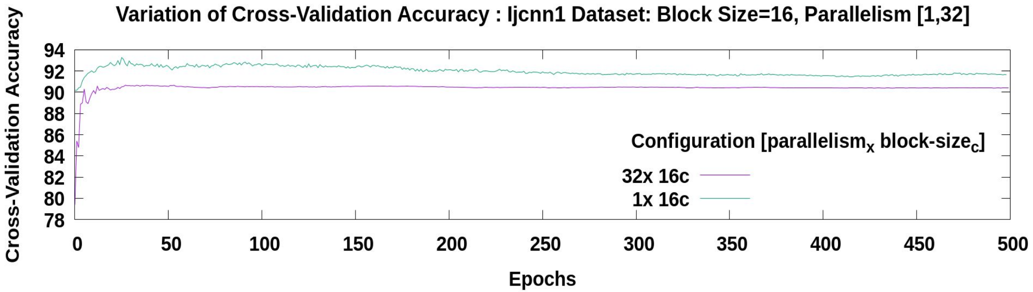


Training Time Variation [Sequential Mode] - Webspam Dataset

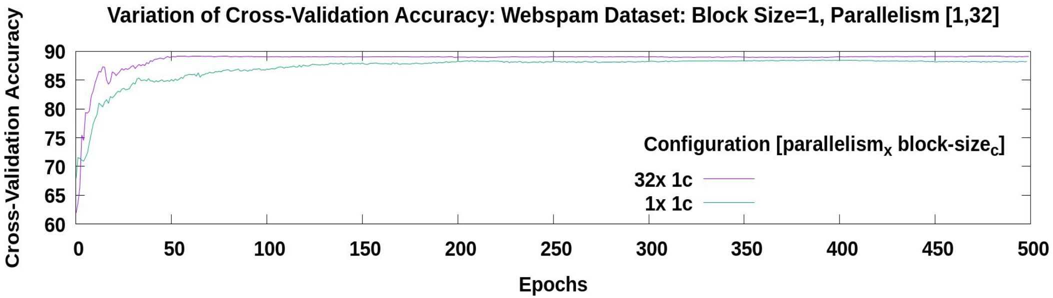


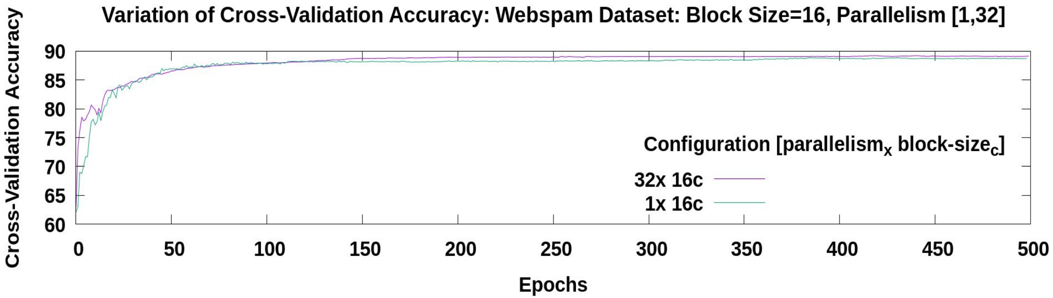
Cross-Validation Accuracy Variation Against Parallelism - Ijcnn1 Dataset



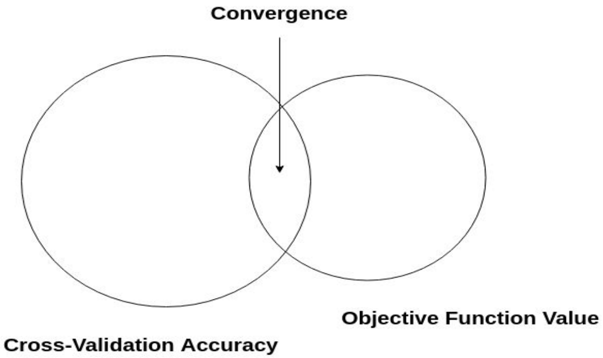


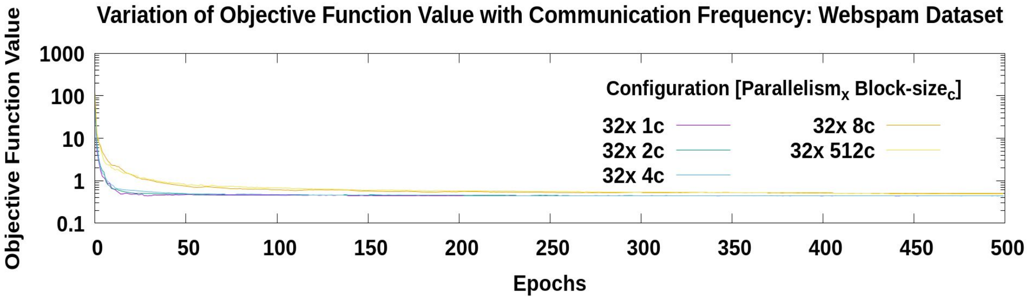
Cross-Validation Accuracy Variation Against Parallelism - Webspam Dataset

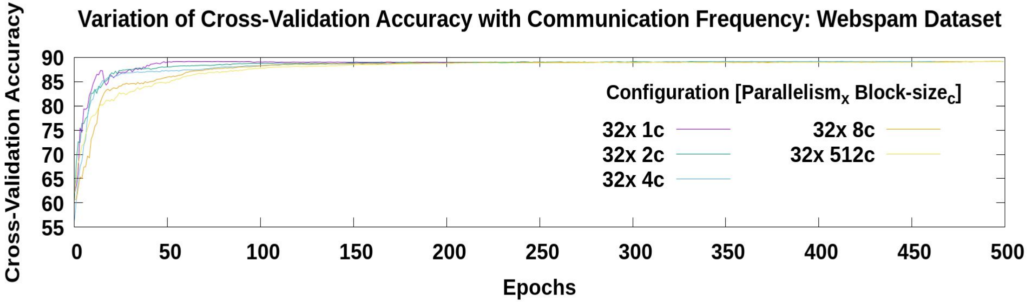




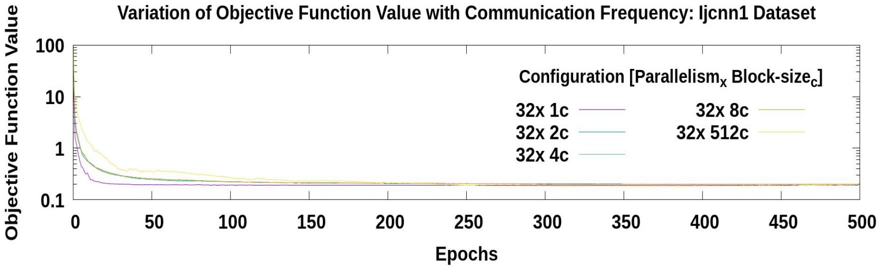
Convergence with Parallelism

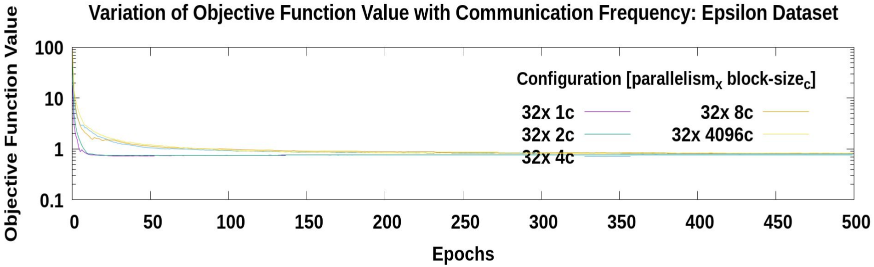


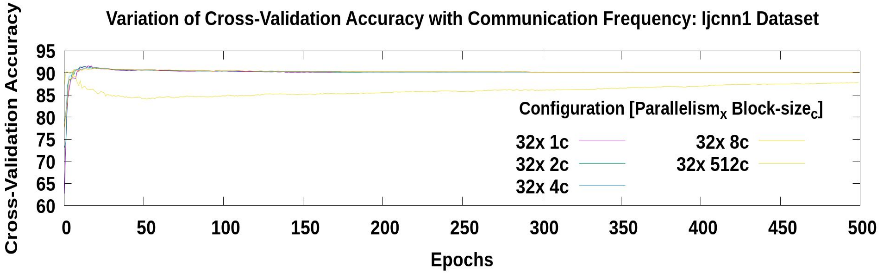


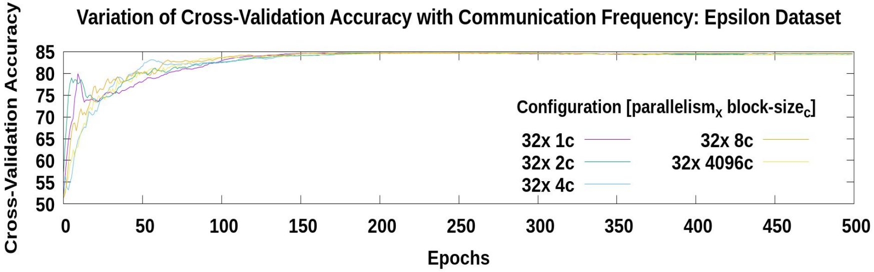


Convergence With Parallelism Cont...





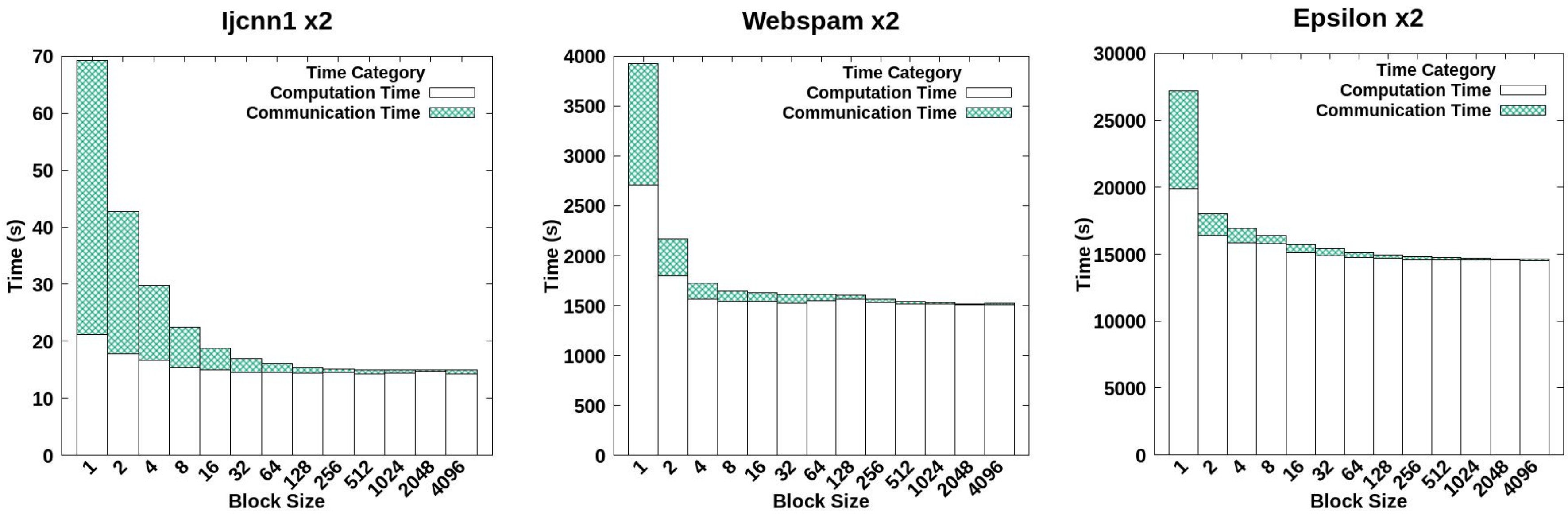




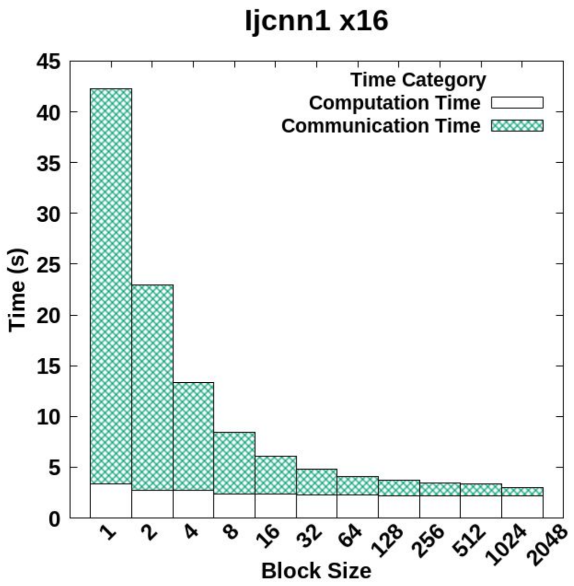
Understanding Performance

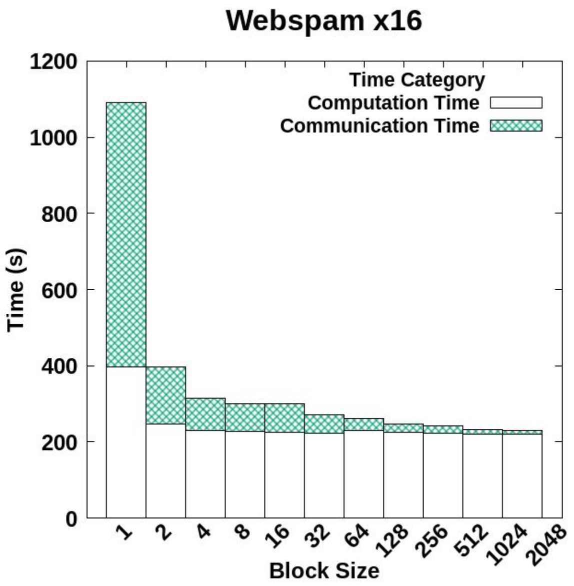
- Understanding the performance of the algorithm in terms of parallelism level and block size, in terms of times.
  - Time to update one point (0.5625 us/epoch - epsilon x32 b=1)
  - Time to check for convergence (0.375 us/epoch - epsilon x32 b=1) (objective function evaluation)
  - Time for MPI collective (3.5625 us/epoch - epsilon x32 b=1) (model synchronization, i.e allreduce)

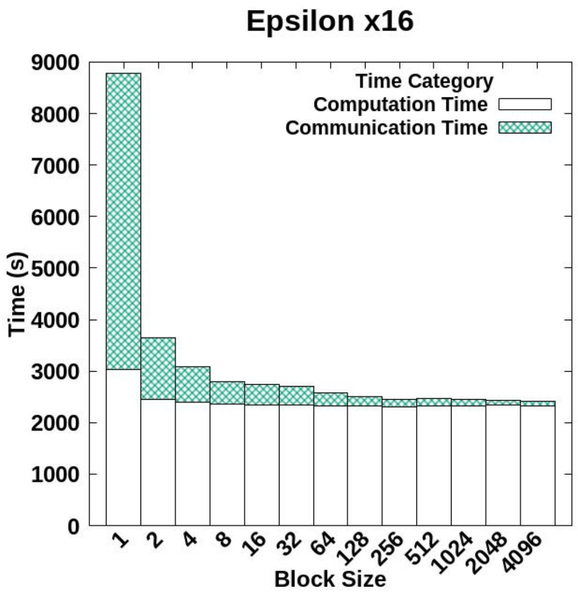
Training Time Breakdown



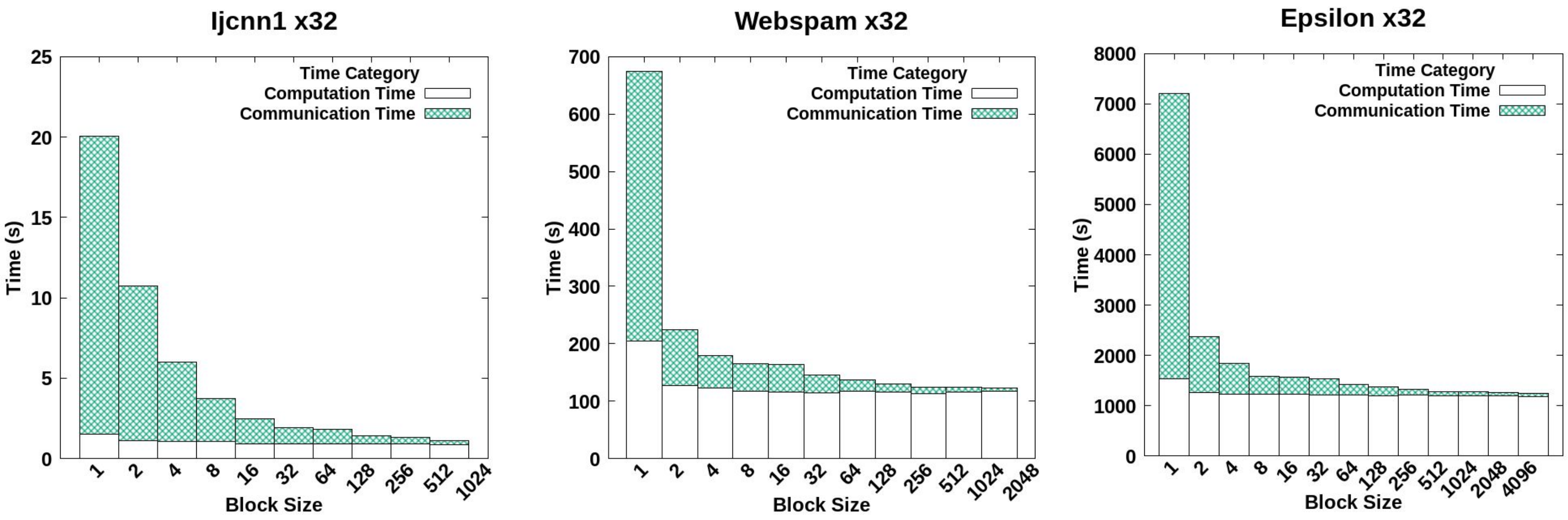
Training Time Breakdown



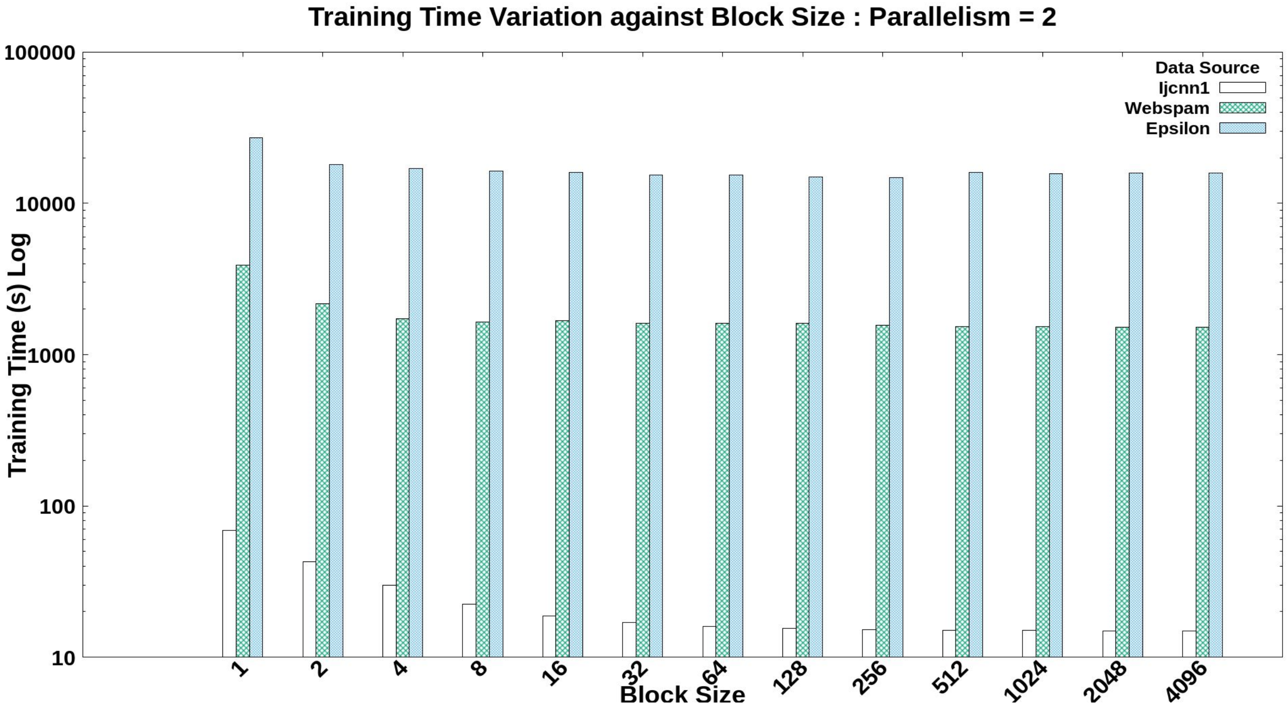




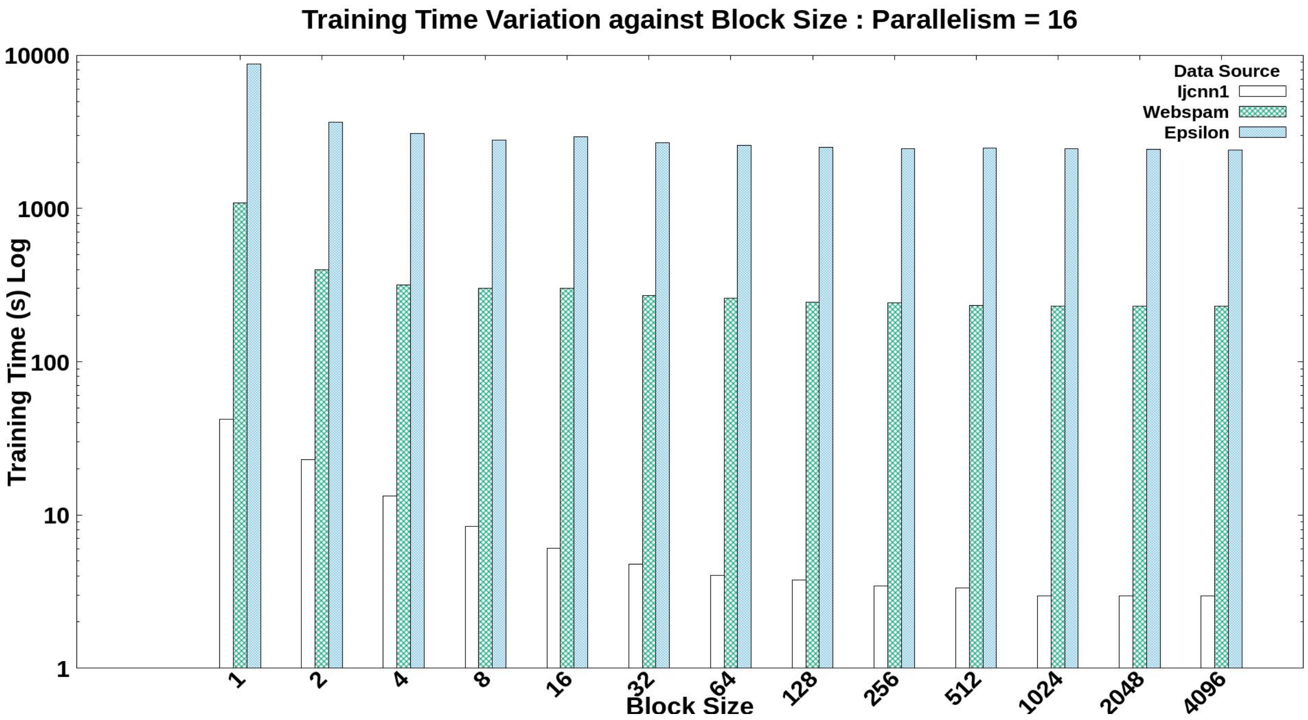
Training Time Breakdown



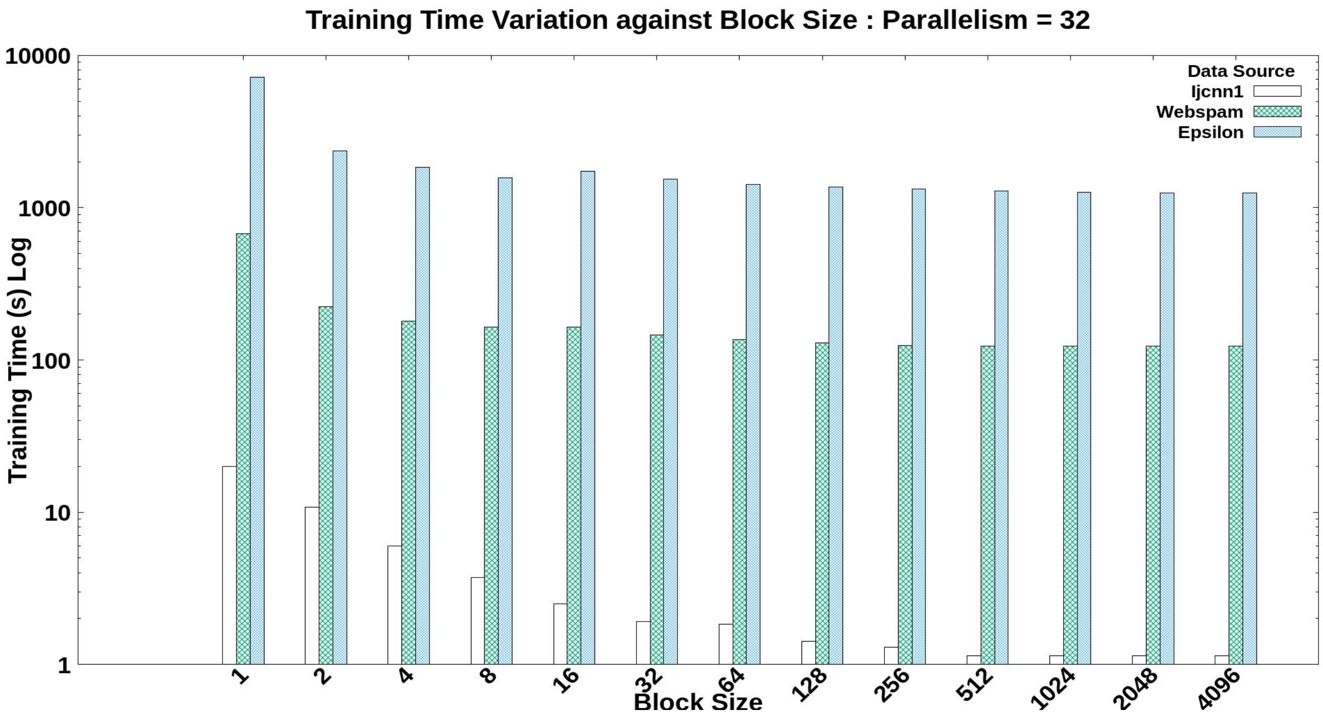
Training Time vs Parallelism



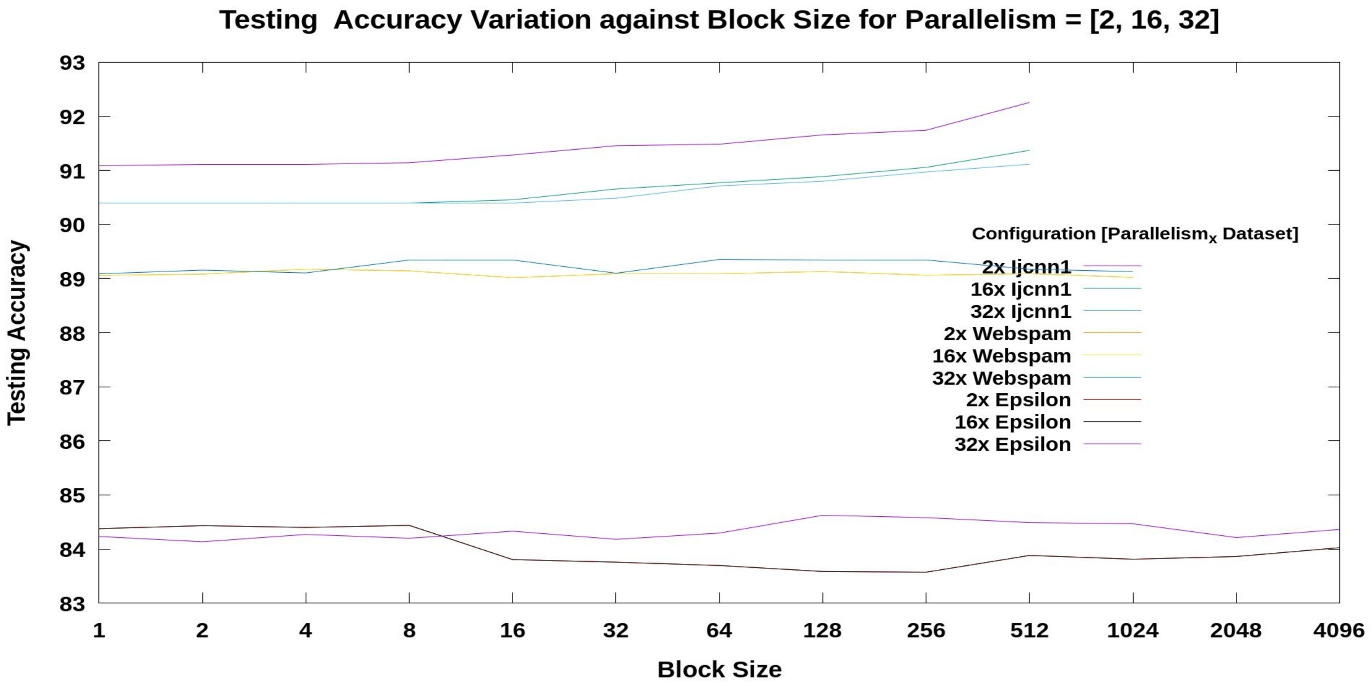
Training Time Vs Parallelism Cont...



Training Time Vs Parallelism Cont...



Testing Accuracy Variation



Summary of Experimental Results

DataSet	Sequential Timing (seconds)	Parallel Timing (seconds)	Sequential Accuracy	Parallel Accuracy	Speed Up (x1 vs x32)
Ijcnn1	22.19	1.37	90.63	91.51	16.2
Webspam	2946.49	120.02	87.69	89.12	24.55
Epsilon	20037.5	968.782	80.06	84.36	21.12

Experiment Environment

- For this we used Juliet Cluster which is a part of the Future Systems cloud environment of Digital Science Center in Indiana University Bloomington
- Configuration of a Node in the Cluster
  - Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
  - Cores Per Socket = 18
  - Sockets = 2
  - Threads Per Core = 2

Extension of Research

- Providing support in both HPC and Dataflow-like computation models.
- Twister2 SVM (Batch and Streaming [experimental]) https://twister2.gitbook.io/twister2/examples/ml/svm
- Available With Twister2 0.2.0 release. [Twister2 is a framework developed by Indiana University Bloomington as a Big Data Hosting Environment: A composable framework for high-performance data analytics]
- Twister2 TSet: High Performance Iterative Dataflow a paper published (May 10th, 2019) uses this SVM model as an application.

Future Work

- Online training with SGD-based SVM with Twister2
- Supporting multiple kernels and multi-model training infrastructure with Twister2

Thank You

- Code
  - OpenMPI C++: https://github.com/vibhatha/PSGDSVMC [Used in Paper]
  - OpenMPI Java: https://github.com/vibhatha/PSGDSVM
  - OpenMPI Python: https://github.com/vibhatha/PSGDSVMPY
  - Twister2: https://twister2.gitbook.io/twister2/examples/ml/svm
- Paper
  - Pre-print: https://arxiv.org/abs/1905.01219
- Contact
  - https://www.vibhatha.org
  - vibhatha@gmail.com, vlabeyko@iu.edu
  - Digital Science Center [Indiana University Bloomington]
    - Geoffrey Fox's Lab https://www.researchgate.net/lab/Geoffrey-Charles-Fox-Lab-2
    - My Profile https://www.researchgate.net/profile/Vibhatha_Abeykoon2