Parallel Streaming Decision Tree

Anxiang Zhang, Ke Ding

Abstract

We implemented the sequential version, the OpenMP version, the OpenMPI version and the CUDA version for decision tree with histogram and compared the performance of four implementations over several datasets.

Introduction

Decision Tree is widely used in Machine Learning. It is simple and intuitive.

```
Function BuildTree(n,A) // n: samples (rows), A: attributes
  If empty(A) or all n(L) are the same
    status = leaf
    class = most common class in n(L)
  else
    status = internal
    a <- bestAttributeSplitPoint(n,A)
    LeftNode = BuildTree(n(a=1), A \ {a})
    RightNode = BuildTree(n(a=0), A \ {a})
  end
end
```

Approach

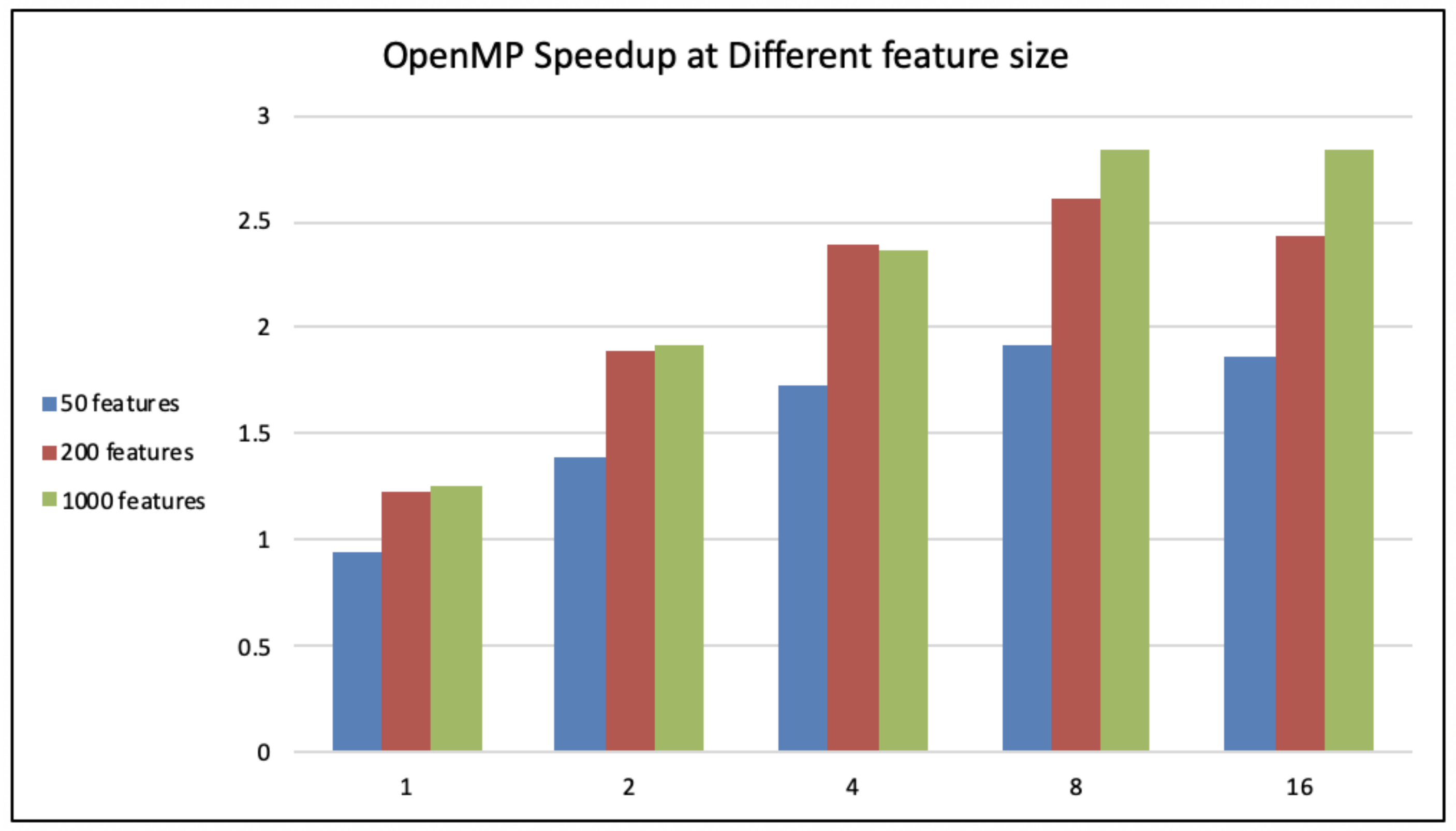
- Node parallel
  - When splitting nodes, parallel over all unlabeled leaves in tree
  - Pro: most intuitive way to speed up
  - Con: unbalanced workload
- Feature parallel
  - When find the best split in tree, parallel over different features
  - Pro: reduced communication cost O(P)
  - Con: manually synchronize best split point information for all features
- Data parallel
  - When update histograms with values, parallel over data in dataset
  - Pro: huge parallelism in large data size
  - Con: manually synchronize the updated histograms in each worker. The histograms are the contention

Implementation

- OpenMP
  - Data parallel + Feature parallel
  - Reorder the loop to avoid race condition
- OpenMPI
  - Data parallel + Feature parallel
  - Message-passing version
  - When facing contention, introduce local variables, then send to master and merge
- CUDA
  - Four kernels
  - Serialize the dataset, rewrite some operations
  - Data parallel + Feature parallel
  - Fixed thread number and block number for kernels

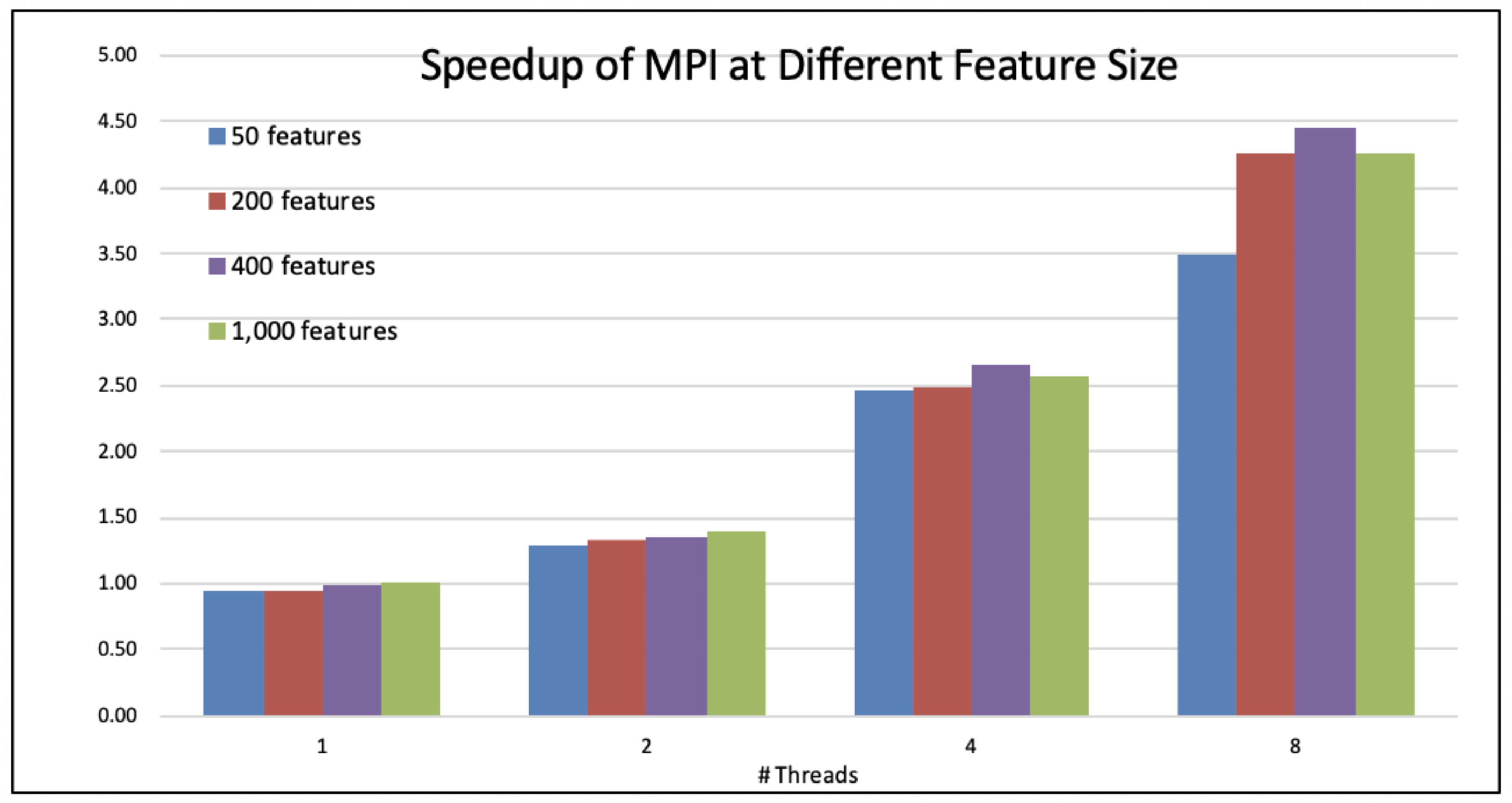
Kernel name	Thread num	Block num
histogram_update	# features	#leaves
calculate_feature_value	128	(#features+127)/128
calculate_gain_deltas	max bin size	#features
navigate_sample	128	(#data+127)/128

Evaluation

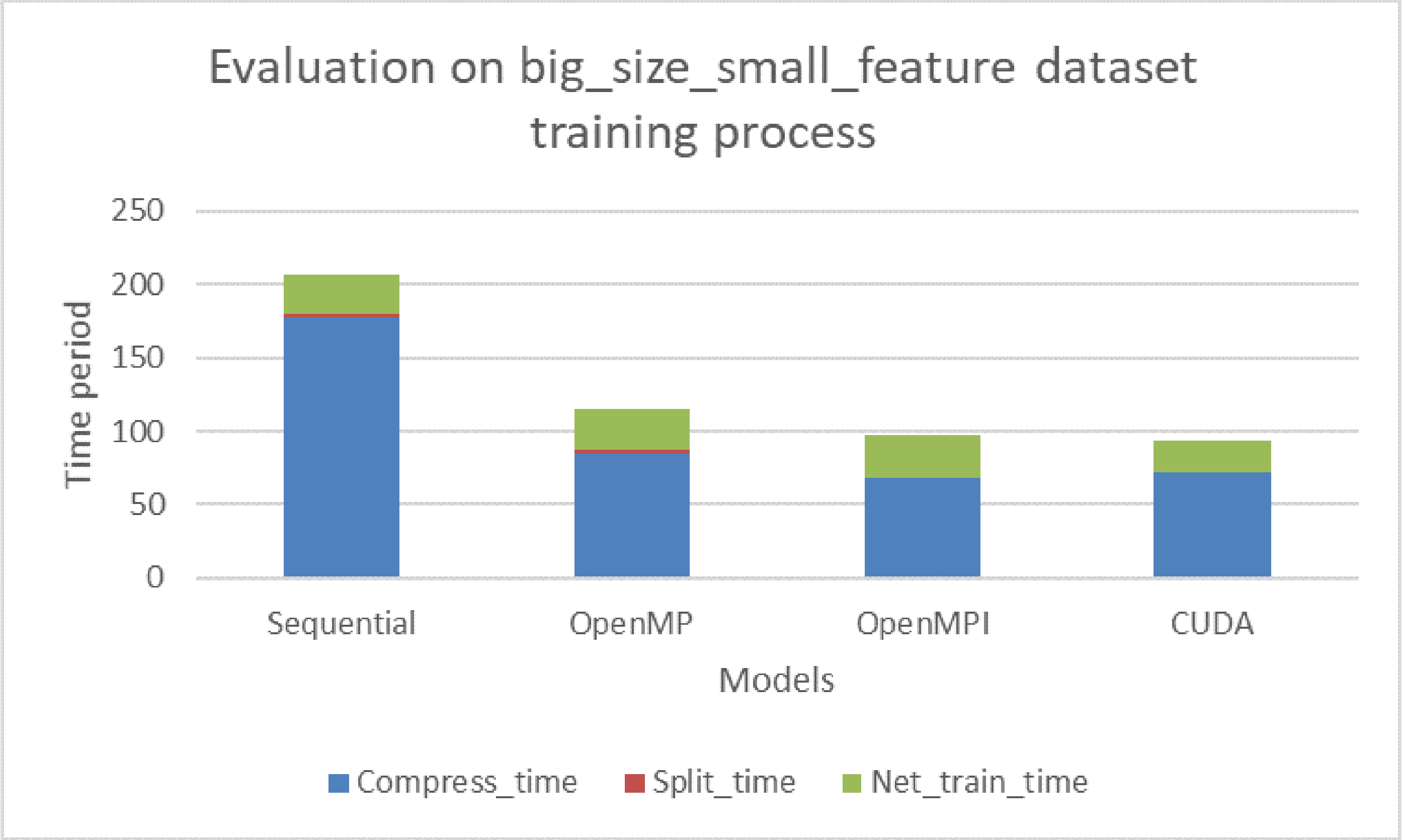


OpenMP speedup over different feature sizes

Evaluation



OpenMP speedup over different feature sizes



Sequential, OpenMP, OpenMPI and CUDA version's compress time, split time and net train time comparison

Takeaways

1. OpenMP scales over feature size, while OpenMPI scales poorly over feature size.
2. OpenMPI could achieve a higher speedup than OpenMPI.
3. On large data size, CUDA could achieve higher speedup than OpenMP and OpenMPI.

References:
[1]: Srivastava, Anurag, et al. "Parallel formulations of decision-tree classification algorithms." High Performance Data Mining. Springer, Boston, MA, 1999. 237-261.
[2]: Ben-Haim, Yael, and Elad Tom-Tov. "A streaming parallel decision tree algorithm." Journal of Machine Learning Research 11.Feb (2010): 849-872.
[3]: Zhang, Huan, Si Si, and Cho-Jui Hsieh. "GPU-acceleration for Large-scale Tree Boosting." arXiv preprint arXiv:1706.08359 (2017).
[4]: Jin, Ruoming, and Gagan Agrawal. "Communication and memory efficient parallel decision tree construction." Proceedings of the 2003 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2003.
[5]: Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011