Basic GBT Benchmark

10/17/2018

Bo Peng
IPCC@Indiana University

Outline

- Progress
  - basic performance evaluation
  - hotspot analysis
  - optimzation test
- Discussion
  - categorical features
  - interface to support distributed version

Performance Evaluatioin

- GBT Implementations
  - xgboost 0.80, https://github.com/dmlc/xgboost
  - daal DAAL 2019. Revision: 30360, https://github.com/intel/daal
- Experiment setup
  - benchm-ml, https://github.com/szilard/benchm-ml
  - A minimal benchmark for scalability, speed and accuracy of commonly used open source implementations (R packages, Python scikit-learn, H2O, xgboost, Spark MLlib etc.) of the top machine learning algorithms for binary classification (random forests, gradient boosted trees, deep neural networks etc.).
  - Binary Classification: airline dataset
  - Experiment B: learn_rate = 0.1 max_depth = 6 n_trees = 300.
  - Accuracy is measured by AUC.

Machine

- CPU: Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
- Cores: 24 Cores (thread# set to 24)
- RAM: 128 GB

Dataset

dataset	train#	test#	features#	sparsity	size
synset	750K	250K	50	dense	420 MB
	x5,x10		x2,x4		17 GB ~40 times
airline	1m	100K	690	dense(one-hot encoding)	2.6 GB/63 MB
higgs	10m	1m	28	dense	4.8 GB

synset

- **sklearn.datasets.make_classification**
- Generate a random n-class classification problem.

  This initially creates clusters of points normally distributed (std=1) about vertices of an 2-dimensional hypercube with sides of length 2 and assigns an equal number of clusters to each class. It introduces interdependence between these features and adds various types of further noise to the data.

Airline

- The Airline data set consists of flight arrival and departure details for all commercial flights from 1987 to 2008. The approximately 120MM records (CSV format), occupy 120GB space.
- Training datasets of sizes 1M, 10M are generated using years 2005 and 2006. A test set of size 100K is generated from the same (using year 2007).
- Categorical features

Month	DayofMonth	DayOfWeek	DepTime	UniqueCarrier	Origin	Dest	Distance	dep_delayed_15min
c-4	c-26	c-2	1828	XE	LEX	IAH	828	N
c-12	c-11	c-1	1212	UA	DEN	MCI	533	N
c-10	c-1	c-6	935	OH	HSV	CVG	325	N
c-11	c-26	c-6	930	OH	JFK	PNS	1028	N
c-12	c-6	c-2	1350	MQ	DFW	LBB	282	Y

- One-hot encoding for categorical features

0	6:1.0	30:1.0	44:1.0	50:1828.0	72:1.0	238:1.0	520:1.0	689:828.0
0	3:1.0	14:1.0	43:1.0	50:1212.0	69:1.0	155:1.0	560:1.0	689:533.0
0	1:1.0	12:1.0	48:1.0	50:935.0	66:1.0	209:1.0	455:1.0	689:325.0
0	2:1.0	30:1.0	48:1.0	50:930.0	66:1.0	228:1.0	611:1.0	689:1028.0
1	3:1.0	39:1.0	44:1.0	50:1350.0	64:1.0	156:1.0	543:1.0	689:282.0

Higgs

- https://archive.ics.uci.edu/ml/datasets/HIGGS#
- This is a classification problem to distinguish between a signal process which produces Higgs bosons and a background process which does not.
- The data has been produced using Monte Carlo simulations. The first 21 features (columns 2-22) are kinematic properties measured by the particle detectors in the accelerator. The last seven features are functions of the first 21 features; these are high-level features derived by physicists to help discriminate between the two classes.
- Example:

0.87	-0.64	0.23	0.33	-0.69	0.75	-0.25	-1.09	0.00	1.37	-0.65	0.93	1.11	1.14	-1.58	-1.05	0.00	0.66	-0.01	-0.05	3.10	1.35	0.98	0.98	0.92	0.72	0.99	0.88	1.00
0.91	0.33	0.36	1.50	-0.31	1.10	-0.56	-1.59	2.17	0.81	-0.21	1.27	2.21	0.50	-1.26	0.73	0.00	0.40	-1.14	0.00	0.00	0.30	0.83	0.99	0.98	0.78	0.99	0.80	1.00

Feature Value Distribution

feature id	synset	higgs	airline
0	746182	27642	2
1	746047	5001	2
2	746073	6284	2
3	746066	1228758	2
4	746047	2138808	2
5	746098	45068	2
6	746099	5999	2
7	746055	6284	2
8	746126	3	2
9	745976	37341	2
10	746103	5999	2

- the distribution of unique values determines the computation complexity of findBestSplit()
- uniq values for each feature

Parameters

Parameter	xgboost	daalgbt
learning rate	eta	shrinkage
histogram algorithm	• tree_method • exact: Exact greedy algorithm. • approx: quantile sketch and gradient histogram. • hist: Fast histogram with bins caching.	• SplitMethod • Exact greedy method • Inexact method for splits finding: bucket continuous features to discrete bins
bin size	• max_bin • by default 256 for hist	• maxBins • by default 256
global or local histogram		• memorySavingMode • true: build histogram on fly • false: build once in initialization

Feature Value Distribution using Histogram

- daalgbt
- IndexedFeatures().numIndices(size_t iCol)

feature id	synset	higgs	airline
0	257	253	2
1	257	248	2
2	257	252	2
3	257	256	2
4	257	256	2
5	257	254	2
6	257	248	2
7	257	251	2
8	257	3	2
9	257	254	2
10	257	248	2

- xgboost tree_method = hist
- GHistIndexMatrix
- updater_fast_hist

feature id	synset	higgs	airline
0	254	254	1
1	254	254	1
2	254	254	1
3	254	254	1
4	254	254	1
5	254	254	1
6	254	254	1
7	254	254	1
8	254	2	1
9	254	254	1
10	254	254	1

Synset Results

trainer	train# 1m			train# 5m			train# 10m		
	f# 50	f# 100	f# 200	f# 50	f# 100	f# 200	f# 50	f# 100	f# 200
xgb-hist	25.38	82.18	144.41	46.42	132.28	243.63	87.68	221.05	407.13
xgb-exact	121.88	783.57	2406.4	206.58	1280.8	3518.46	358.52	2504.71	7462.44
daal-gnu-inexact	36.56	193.9	386.5	64.70	350.8	686.0	124.0	635.2	1343.3
daal-gnu-exact	250.51	1323.9	2576.58	432.50	2571.73	4597.02	789.95	3708.14	7838.2
daal-icc-inexact	27.96	170.79	359.72	51.88	373.56	607.47	109.18	696.58	1447.43
daal-icc-exact	211.47	1054.67	2128.24	366.36	2252.4	4248.7	596.03	2964.5	5952.8


TrainingTime .vs. Trainer
1400
1200
1000
800
600
400
200
0
xgb-exact
xgb-hist
daal-icc-inexact
daal-icc-exact
1mx50
5mx50
10mx50


Training Time .vs. RowNumber
700
600
500
400
300
200
100
0
xgb-exact
xgb-hist
daal-icc-inexact
daal-icc-exact
1mx50
5mx50
10mx50

Higgs Results

dataset	trainer	Parameter	training time(s)	AUC
higgs	xgb	Exact	1387.3	0.742
		Approx	1354.1	0.742
		Hist	229.7	0.742
	daal-gnu	Exact	776.5	0.742
		Inexact	494.6	0.742
		Exact + memSave	3686.3	0.710
		Inexact + memSave	3669.1	0.712
	daal-icc	Exact	653.0	0.742
		Inexact	402.7	0.742
		Exact + memSave	3550.0	0.715
		Inexact + memSave	3564.8	0.712

Airline Results

dataset	trainer	Parameter	training time(s)	AUC
Airline-Sparse	xgb	Exact	43.1	0.557
		Hist	18.7	0.559
Airline-Dense	xgb	Exact	391.9	0.557
		Hist	47.6	0.556
	daal-gnu	Exact	383.8	0.555
		Inexact	362.9	0.555
	daal-icc	Exact	319.79	0.556
		Inexact	309.71	0.555

Hotspot Analysis

- Intel(R) VTune(TM) Amplifier 2019 (build 570779) Command Line Tool
- higgs 50 iterations
- hotspots
- memory-access

xgboost + Higgs + hist

Elapsed Time: 60.005s

CPU Time:	717.259s
Effective Time:	462.088s
Spin Time:	254.691s
Imbalance or Serial Spinning:	254.621s
Lock Contention:	0.030s
Other:	0.040s
Overhead Time:	0.480s
Total Thread Count:	24
Paused Time:	0s

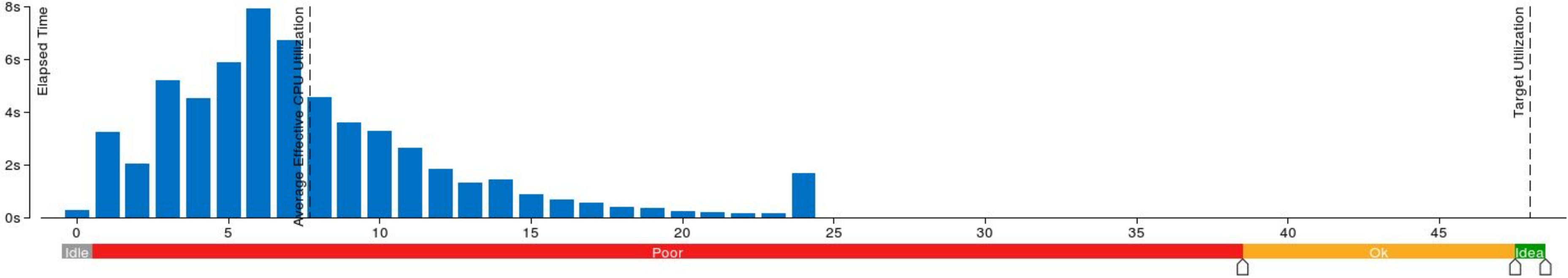
Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically res
performance.

Function	Module	CPU Time
xgboost::common::GHistBuilder::BuildHist._omp_fn.3	xgboost-orig-vtune	234.468s
gomp_simple_barrier_wait	libgomp.so.1	177.689s
xgboost::tree::FastHistMaker::Builder::ApplySplitDenseData._omp_fn.1	xgboost-orig-vtune	119.539s
gomp_team_barrier_wait_end	libgomp.so.1	73.952s
xgboost::tree::FastHistMaker::Builder::InitNewNode	xgboost-orig-vtune	20.544s
[Others]		91.067s

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



daalgbt-icc + Higgs + inexact

Elapsed Time: 88.418s

CPU Time:	1626.862s
Effective Time:	1616.193s
Spin Time:	9.958s
Imbalance or Serial Spinning:	4.588s
Lock Contention:	0.030s
Other:	5.341s
Overhead Time:	0.710s
Total Thread Count:	24
Paused Time:	0s

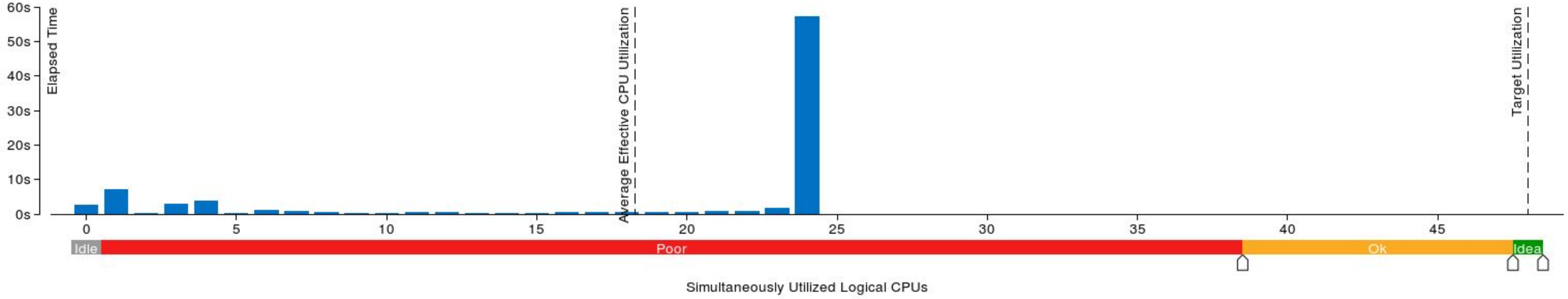
Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall applica performance.

Function	Module	CPU Time
daal::algorithms::gbt::training::internal::TreeBuilderIndexed<float, (daal::CpuType)3>::findSplitOneFeature	daalgbt-icc-vtune	1539.425s
daal::algorithms::gbt::training::internal::TreeBuilderIndexed<float, (daal::CpuType)3>::finalizeBestSplit	daalgbt-icc-vtune	35.329s
daal::algorithms::gbt::training::internal::TreeBuilder<float, (daal::CpuType)3>::buildSplit	daalgbt-icc-vtune	19.159s
daal::algorithms::gbt::training::internal::TreeBuilder<float, (daal::CpuType)3>::makeLeaf	daalgbt-icc-vtune	9.899s
[TBB Scheduler Internals]	libtbb.so.2	6.170s
[Others]		16.880s

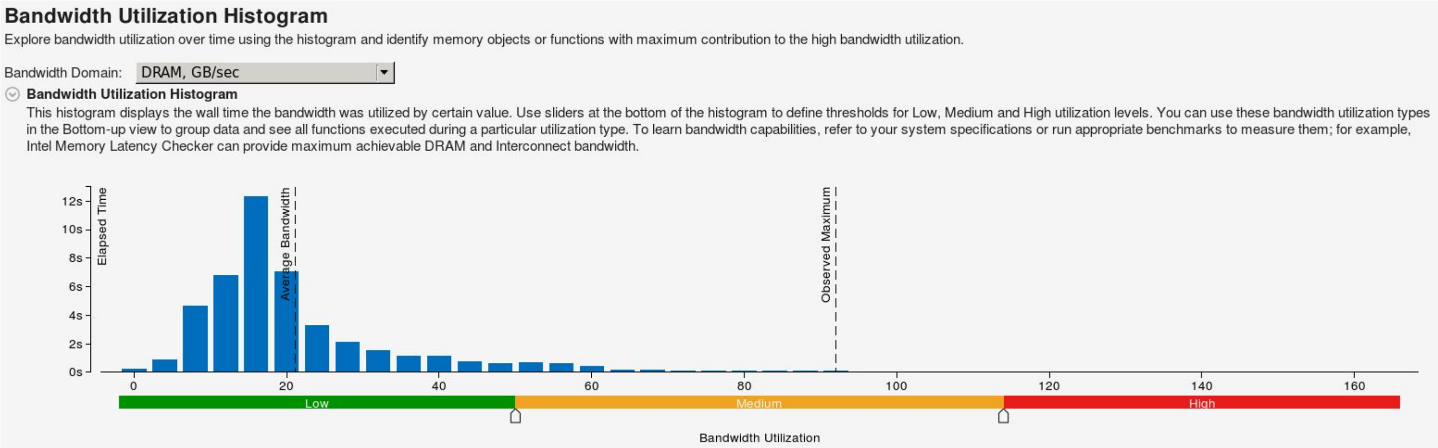
Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

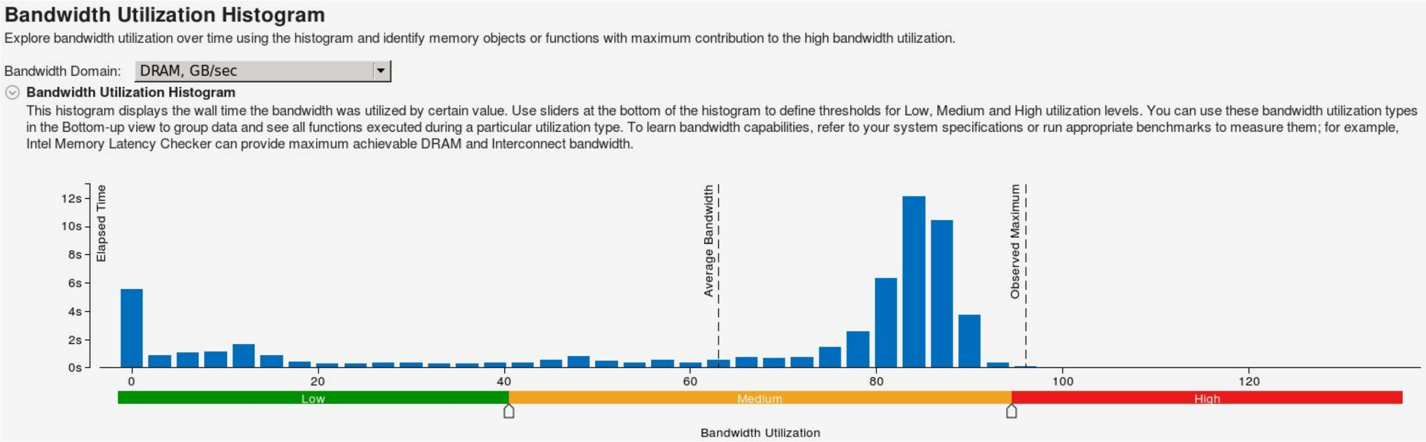


Higgs + Hist

xgboost



daal



Higgs + Hist

- xgboost+ hist

Elapsed Time: 45.520s		
CPU Time:	615.350s	
Memory Bound:	49.4%	**of Pipeline Slots**
L1 Bound:	10.6%	of Clockticks
DRAM Bound:		
DRAM Bandwidth Bound:	0.1%	of Elapsed Time
NUMA: % of Remote Accesses:	28.7%	
QPI Bandwidth Bound:	1.8%	of Elapsed Time
Loads:	218,035,540,870	
Stores:	76,620,798,555	
LLC Miss Count:	**2,784,417,055**	
Average Latency (cycles):	35	
Total Thread Count:	24	
Paused Time:	0s	

- daal + inexact

Elapsed Time: 58.054s		
CPU Time:	1522.305s	
Memory Bound:	85.1%	**of Pipeline Slots**
L1 Bound:	0.3%	of Clockticks
DRAM Bound:		
DRAM Bandwidth Bound:	53.1%	of Elapsed Time
NUMA: % of Remote Accesses:	49.4%	
QPI Bandwidth Bound:	72.8%	of Elapsed Time
Loads:	575,007,749,715	
Stores:	170,644,119,170	
LLC Miss Count:	**10,134,858,055**	
Average Latency (cycles):	72	
Total Thread Count:	36	
Paused Time:	0s	

Higgs + Exact

xgboost

Elapsed Time: 120.091s

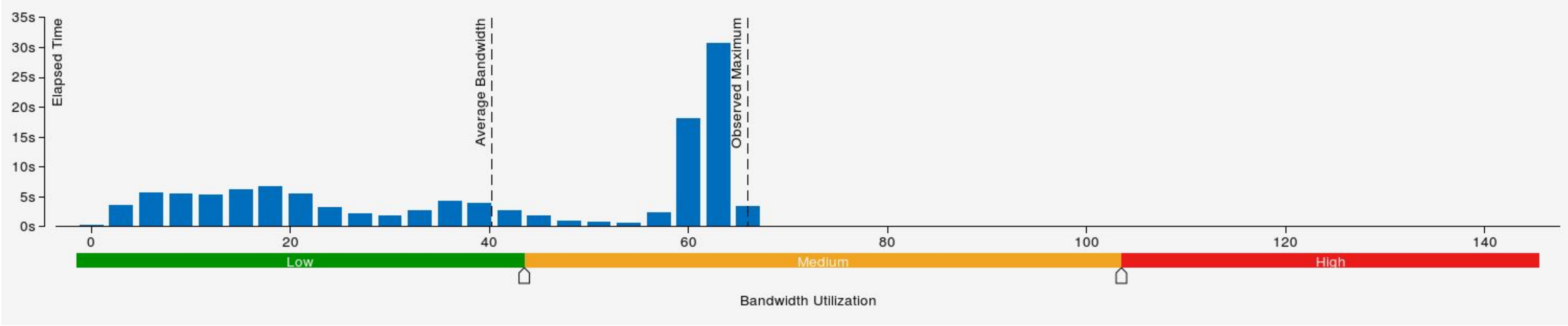
CPU Time:	2151.915s	
Memory Bound:	70.8%	**of Pipeline Slots**
L1 Bound:	1.5%	of Clockticks
DRAM Bound:		
DRAM Bandwidth Bound:	0.0%	of Elapsed Time
NUMA: % of Remote Accesses:	47.8%	
QPI Bandwidth Bound:	57.5%	of Elapsed Time
Loads:	1,187,592,626,710	
Stores:	471,369,640,665	
LLC Miss Count:	**56,153,869,030**	
Average Latency (cycles):	54	
Total Thread Count:	24	
Paused Time:	0s	

daal-icc

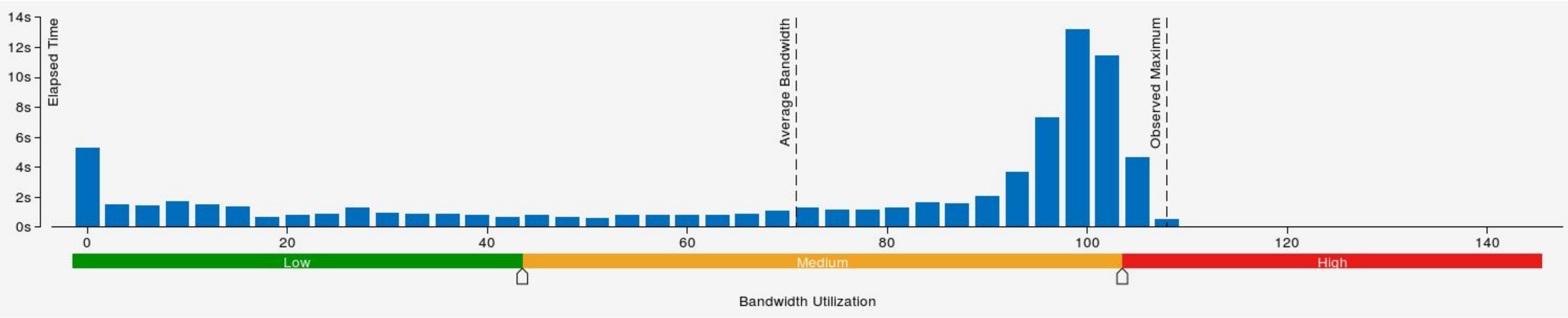
Elapsed Time: 80.058s

CPU Time:	1954.195s	
Memory Bound:	80.1%	**of Pipeline Slots**
L1 Bound:	0.5%	of Clockticks
DRAM Bound:		
DRAM Bandwidth Bound:	39.5%	of Elapsed Time
NUMA: % of Remote Accesses:	28.3%	
QPI Bandwidth Bound:	58.7%	of Elapsed Time
Loads:	549,260,977,335	
Stores:	194,983,849,340	
LLC Miss Count:	**22,292,837,490**	
Average Latency (cycles):	94	
Total Thread Count:	36	
Paused Time:	0s	

Higgs + Exact



35s
30s
25s
20s
15s
10s
5s
0s
Elapsed Time
Average Bandwidth
Observed Maximum
0
20
40
60
80
100
120
140
Low
Medium
High
Bandwidth Utilization



14s
12s
10s
8s
6s
4s
2s
0s
Elapsed Time
Average Bandwidth
Observed Maximum
0
20
40
60
80
100
120
140
Low
Medium
High
Bandwidth Utilization

benmark results

- daalgbt provides very good cpu and memory utilization, leading to better performance in exact mode for normal dense dataset
- xgboost support sparse dataset by default, and it has an improved histogram algorithm, leading to better performance on sparse dataset and in fast_hist mode
-

Discussion & Questions

- Categorical features
  - feature bundle is a useful feature for sparse dataset, (xgboost supports in 0.80)
  - Q: how tree split is done on an unordered categorical feature in daalgbt?
- Interface to support distributed version
  - need allreduce in the processes of
    - build histogram (feature index)
    - findBestSplit(local GHSum)
  - currently tree expands supports feature level and node level parallelism
  - Q: Is it possible to reduce the number of communication by aggregating to single allreduce for all the nodes in the same tree level?

End

“All benchmarks are wrong, but some are useful”