Various GBM Performance Topics

Damien Soukhavong (Laurae)
SAP BI / Data Science Consultant, Planeum

Budapest BI Forum
November 2019

Laurae (Damien)

Follow

Laurae’s Data Science & Design curated posts

Editor of Imploding Gradients and Data Science & Design

5 Following 887 Followers ·


Edit profile
Damien (Laurae)
@Laurae_Cht
Laurae's. Creating & Designing wizard for Data Science in general
Paris, France
95 Following 368 Followers

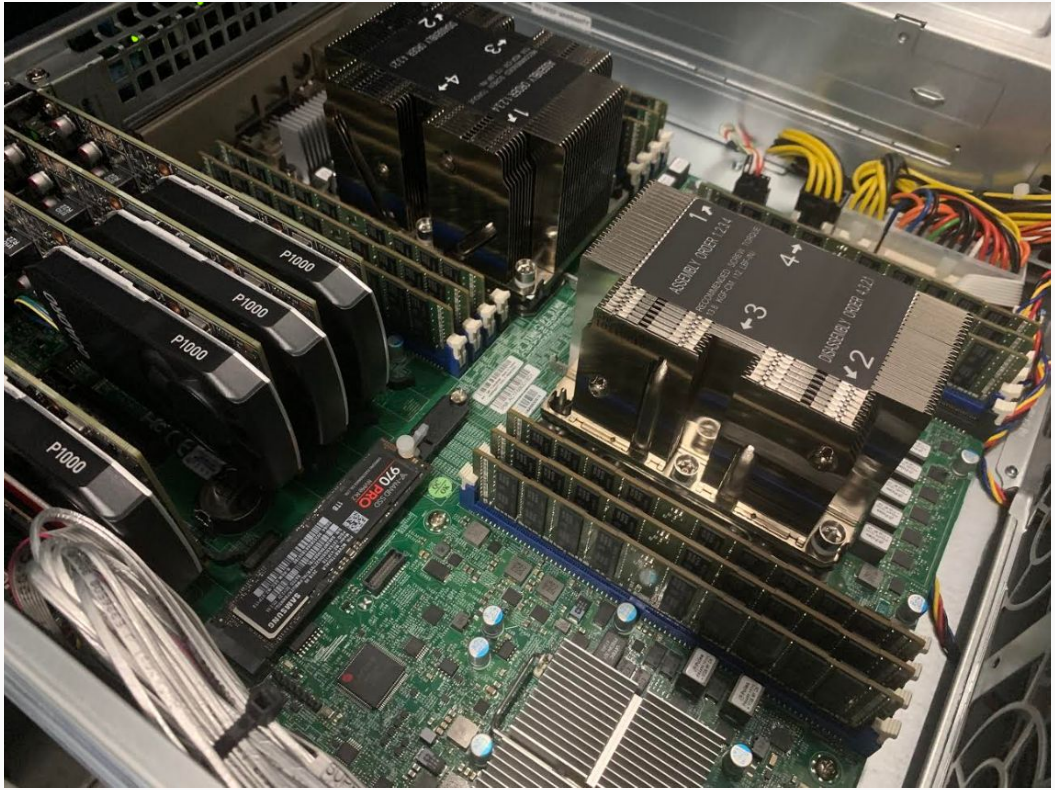
Disclaimer:

I am not representing my employer (Planeum) in this talk

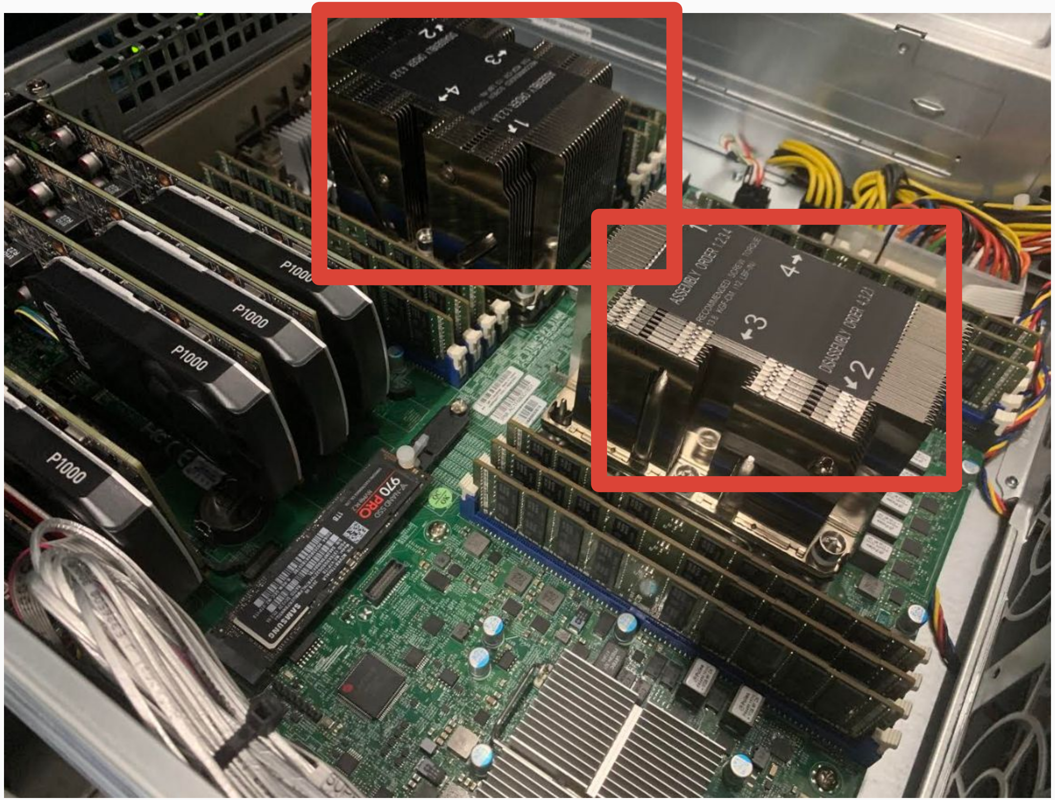
I cannot confirm nor deny if Planeum is using any of the methods, tools, results etc. mentioned in this talk

Hardware Used for Benchmarking

Hardware Used for Benchmarking



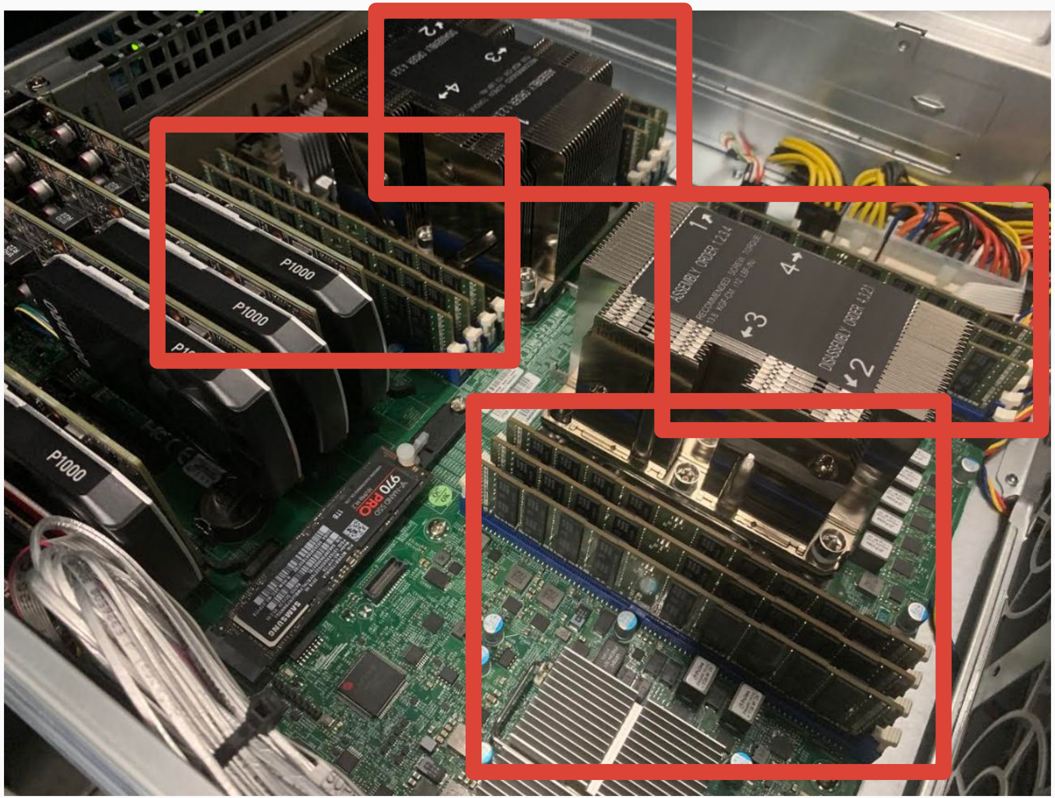
Hardware Used for Benchmarking



CPU: 2x Dual Xeon 6154

- 36 cores / 72 threads, 400W
- All turbo 3.7 GHz (if ~~AVX256/512~~)

Hardware Used for Benchmarking



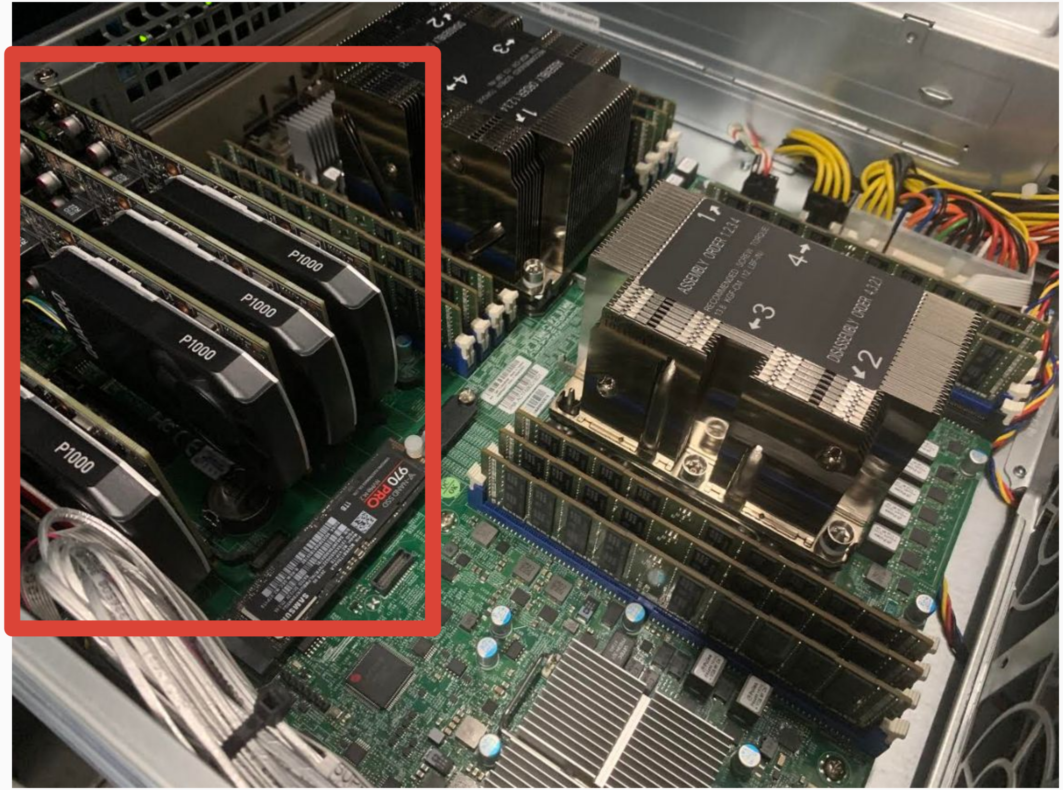
CPU: 2x Dual Xeon 6154

- 36 cores / 72 threads, 400W
- All turbo 3.7 GHz (if ~~AVX256/512~~)

RAM: 768 GB RAM

- 12x 32 GB RAM
- DDR4 2666 MHz

Hardware Used for Benchmarking



CPU: 2x Dual Xeon 6154

- 36 cores / 72 threads, 400W
- All turbo 3.7 GHz (if ~~AVX256/512~~)

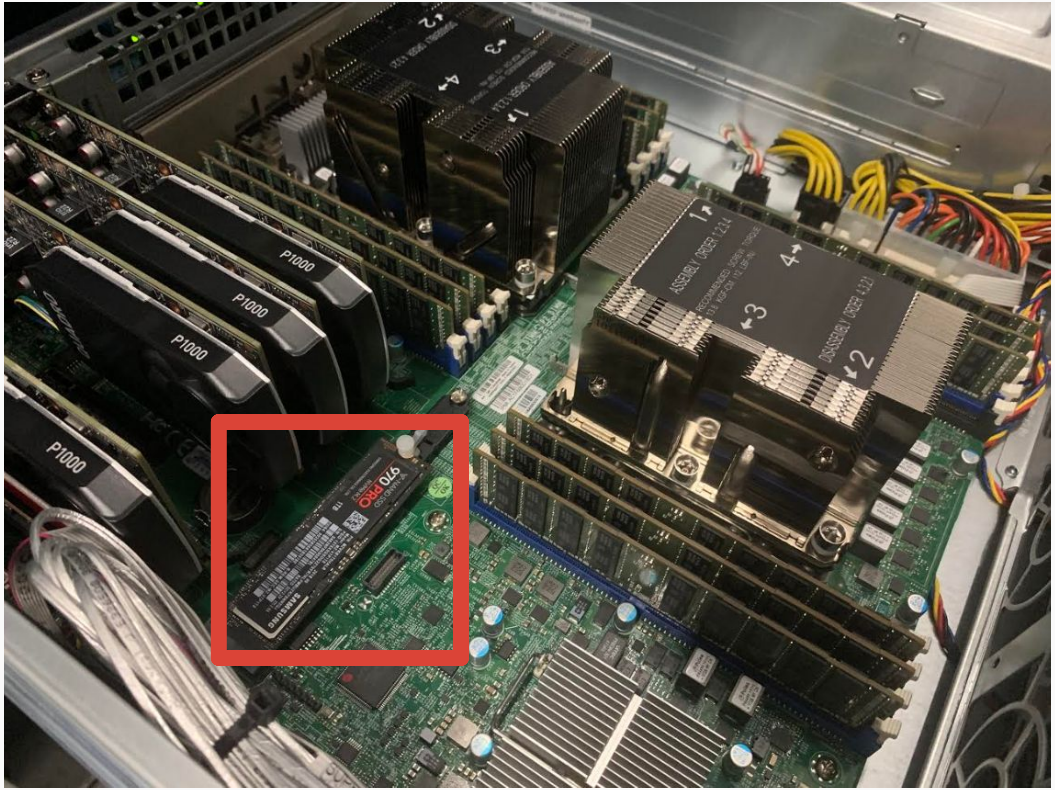
RAM: 768 GB RAM

- 12x 32 GB RAM
- DDR4 2666 MHz

GPU: 4x NVIDIA PNY Quadro P1000

- Equivalent of NVIDIA 1050
- Low Profile HHHL: Half Height Half Length
- 4 GB RAM

Hardware Used for Benchmarking



CPU: 2x Dual Xeon 6154

- 36 cores / 72 threads, 400W
- All turbo 3.7 GHz (if ~~AVX256/512~~)

RAM: 768 GB RAM

- 12x 32 GB RAM
- DDR4 2666 MHz

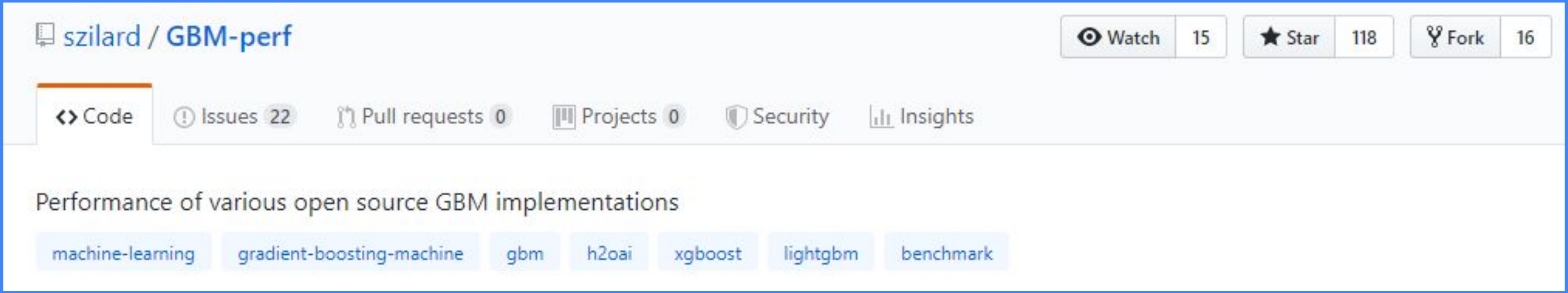
GPU: 4x NVIDIA PNY Quadro P1000

- Equivalent of NVIDIA 1050
- Low Profile HHHL: Half Height Half Length
- 4 GB RAM

Drive: Samsung 970 Pro (NVMe)

- 1 GB RAM
- 36 GB SLC cache

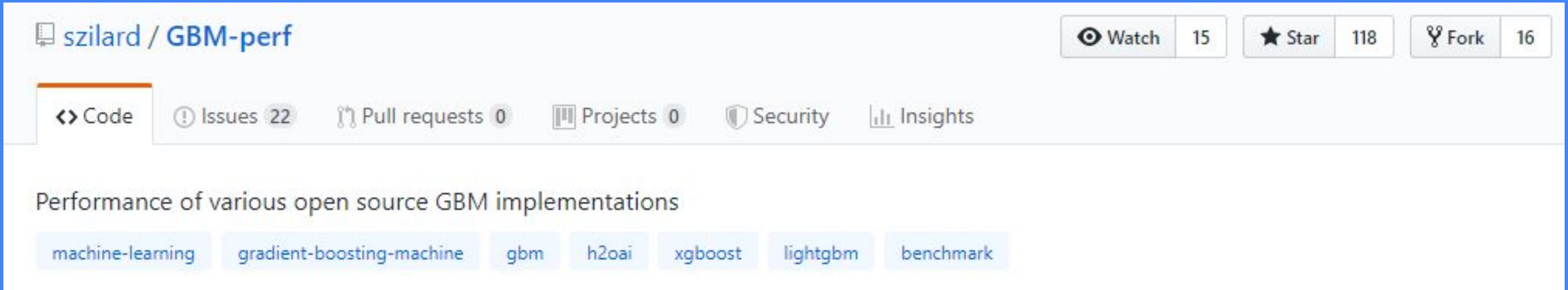
Lowly threaded GBM is quick



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

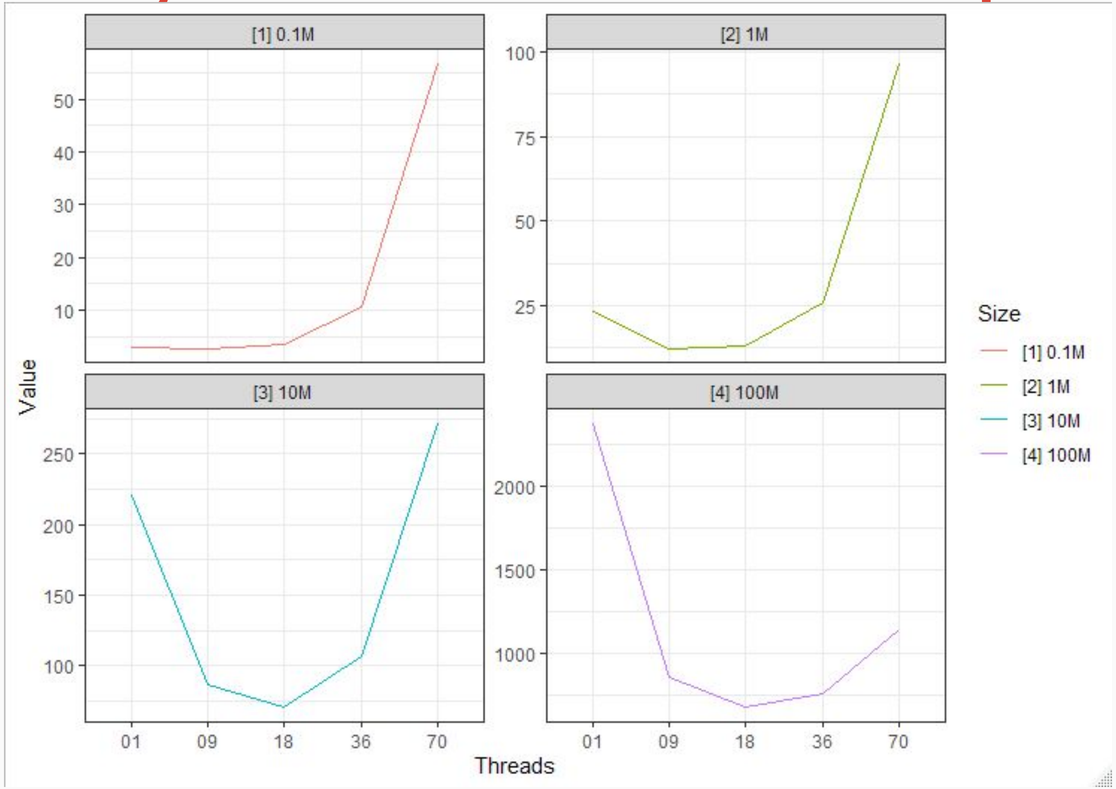
Lowly threaded GBM is quick

xgboost



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Lowly threaded GBM is quick



xgboost hist, CPU

Lowly threaded GBM is quick

```
ggplot2::ggplot(
    data.frame(
        Size = rep(c("[1] 0.1M", "[2] 1M", "[3] 10M", "[4] 100M"), each = 5),
        Threads = rep(c("01", "09", "18", "36", "70"), 4),
        Value = c(3.062, 2.735, 3.407, 10.515, 56.710,
                  23.293, 12.254, 12.929, 25.980, 96.465,
                  220.200, 86.092, 70.121, 106.479, 271.683,
                  2373.128, 858.772, 675.223, 756.661, 1142.271)),
  ggplot2::aes(x = Threads,
               y = Value,
               group = Size,
               color = Size)) +
  ggplot2::facet_wrap(~ Size,
                      scales = "free_y") +
  ggplot2::geom_line() +
  ggplot2::theme_bw()
```

xgboost hist, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	3.062	**2.735**	3.407	10.515	**56.710**
1M	23.293	**12.524**	12.929	25.980	**96.465**
10M	220.200	86.092	**70.121**	106.479	**271.683**
100M	**2373.128**	858.772	**675.223**	756.661	1142.271

Comparison unit: seconds

xgboost hist, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	1x	**1.12x**	0.89x	0.29x	**0.05x**
1M	1x	**1.86x**	1.80x	0.90x	**0.24x**
10M	1x	2.56x	**3.14x**	2.07x	**0.81x**
100M	**1x**	2.76x	**3.51x**	3.14x	2.08x

Comparison unit: 1 model thread speed

xgboost hist, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	1x	**1.12x**	0.89x	0.29x	**0.05x**
1M	1x	**1.86x**	1.80x	0.90x	**0.24x**
10M	1x	2.56x	**3.14x**	2.07x	**0.81x**
100M	**1x**	2.76x	**3.51x**	3.14x	2.08x

Comparison unit: 1 model thread speed

- xgboost has difficulties to scale (even on a log scale)
- Poor threading leads to slow training times
- Large data (100M) does not benefit from using lot of threads

xgboost hist, CPU

Lowly threaded GBM is quick

hcho3 commented on Jul 6, 2017 — Collaborator

@Laurae2 I've submitted patch #2493 to improve multithreaded scaling of fast histogram. It's still slower than LightGBM for a few runs; I'll submit a follow-up patch to further improve speed.

Sponsored(?) by intel

- **Fix unstable results in hist method** ✓
  #4767 by SmirnovEgorRu was closed on Aug 13
- **Optimizations for CPU** ✓
  #4529 by SmirnovEgorRu was merged on Jun 27 • Approved
- **CPU optimizations** ✗
  #4433 by SmirnovEgorRu was closed on Jun 2 • Changes requested
- **Optimizations of pre-processing for 'hist' tree method** ✓
  #4310 by SmirnovEgorRu was merged on Apr 17 • Approved
- **Performance optimizations for CPUs [part 2]** ✗
  #4278 by SmirnovEgorRu was closed on May 25
- **Performance optimizations for Intel CPUs** ✓
  #3957 by SmirnovEgorRu was merged on Jan 9

Mixed bag of results:

- Sometimes faster
- Sometimes significantly slower

xgboost hist, CPU

Lowly threaded GBM is quick

Sponsored(?) by intel

Performance optimizations for Intel CPUs #3957

Merged hcho3 merged 15 commits into dmlc:master from unknown repository on Jan 9

Conversation 26 · Commits 15 · Checks 0 · Files changed 7

SmirnovEgorRu commented on Dec 2, 2018 — Contributor

In this issue some performance optimizations for multi-core CPUs were offered.
I prepared these as a pull-request. Also, I added SW prefetching, which helps to improve performance additionally.
It was tested for HIGGS data set (1m x 28) in comparison with Intel® DAAL

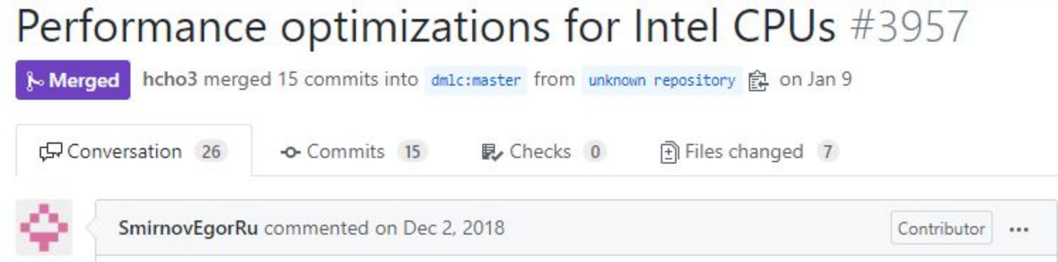
Intel Data Analytics Acceleration Library

	Intel® DAAL	Current XGBoost	Proposed changes
Time, ms	1344	8600	4238

xgboost hist, CPU

Lowly threaded GBM is quick

Sponsored(?) by intel



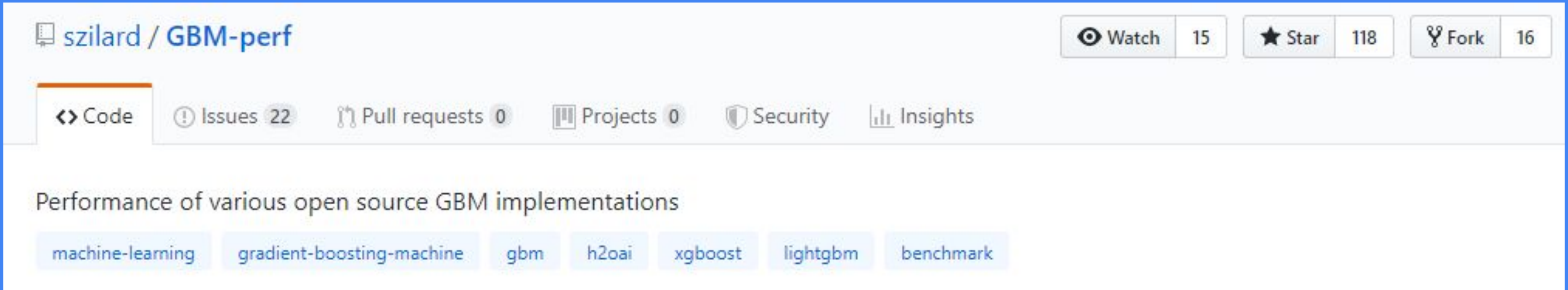
Commit	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads	32 Threads
`dade7c3` (Jan 7, 2019)	32.160	20.538	21.005	17.161	16.432	**16.849**
`5f151c5` (Jan 8, 2019)	31.504	25.475	18.222	16.094	15.943	**579.058**

New multithreaded scaling behavior depends on the dataset

xgboost hist, CPU

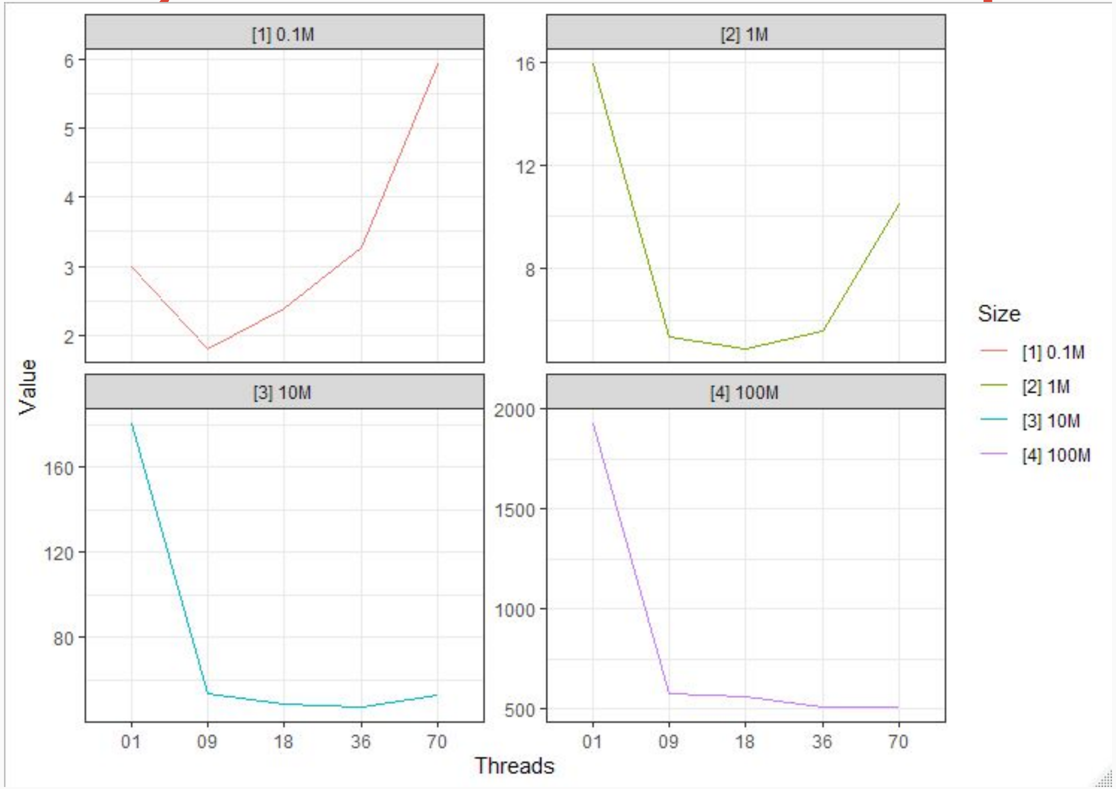
Lowly threaded GBM is quick

LightGBM



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Lowly threaded GBM is quick



LightGBM, CPU

Lowly threaded GBM is quick

```
ggplot2::ggplot(
    data.frame(
        Size = rep(c("[1] 0.1M", "[2] 1M", "[3] 10M", "[4] 100M"), each = 5),
        Threads = rep(c("01", "09", "18", "36", "70"), 4),
        Value = c(2.983, 1.801, 2.389, 3.266, 5.943,
                  15.919, 5.363, 4.891, 5.568, 10.487,
                  180.748, 53.689, 48.620, 47.033, 53.234,
                  1930.816, 578.734, 560.296, 507.580, 507.627)),
  ggplot2::aes(x = Threads,
               y = Value,
               group = Size,
               color = Size)) +
  ggplot2::facet_wrap(~ Size,
                      scales = "free_y") +
  ggplot2::geom_line() +
  ggplot2::theme_bw()
```

LightGBM, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	2.983	**1.801**	2.389	3.266	**5.943**
1M	**15.919**	5.363	**4.891**	5.568	10.487
10M	**180.748**	53.689	48.260	**47.033**	53.234
100M	**1930.816**	578.834	560.296	**507.580**	**507.627**

Comparison unit: seconds

LightGBM, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	1x	**1.66x**	1.25x	0.91x	**0.50x**
1M	**1x**	2.97x	**3.25x**	2.86x	1.52x
10M	**1x**	3.37x	3.75x	**3.84x**	3.40x
100M	**1x**	3.34x	3.45x	**3.80x**	**3.80x**

Comparison unit: 1 model thread speed

LightGBM, CPU

Lowly threaded GBM is quick

Size	1 Thread	9 Threads	18 Threads	36 Threads	70 Threads
0.1M	1x	**1.66x**	1.25x	0.91x	**0.50x**
1M	**1x**	2.97x	**3.25x**	2.86x	1.52x
10M	**1x**	3.37x	3.75x	**3.84x**	3.40x
100M	**1x**	3.34x	3.45x	**3.80x**	**3.80x**

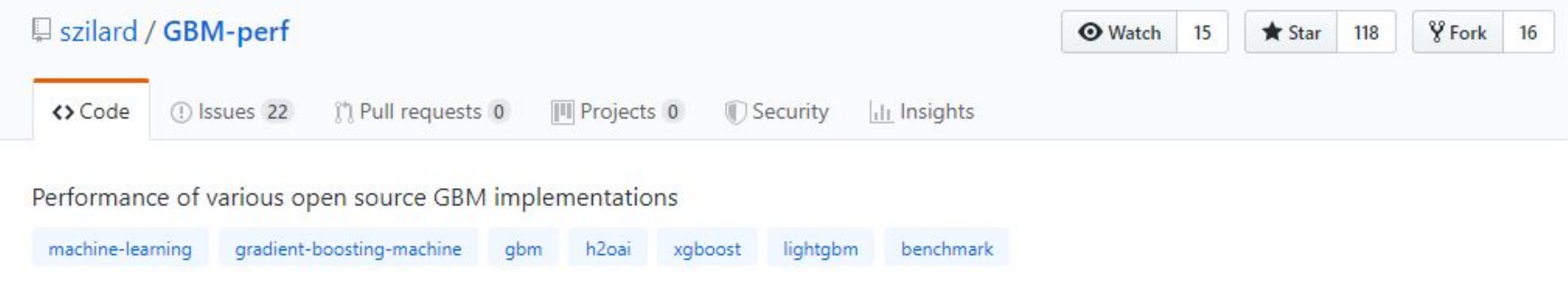
Comparison unit: 1 model thread speed

- LightGBM scales okay-ish with the number of threads
- Good threading makes scaling easy to predict
- Large data (100M) benefits a bit from using lot of threads

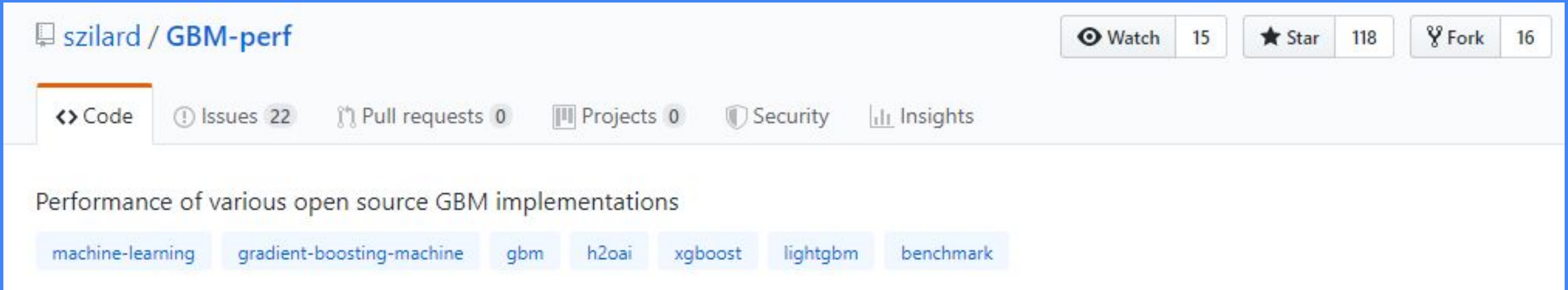
LightGBM, CPU

Lowly threaded GBM is quick

See more: https://github.com/szilard/GBM-perf



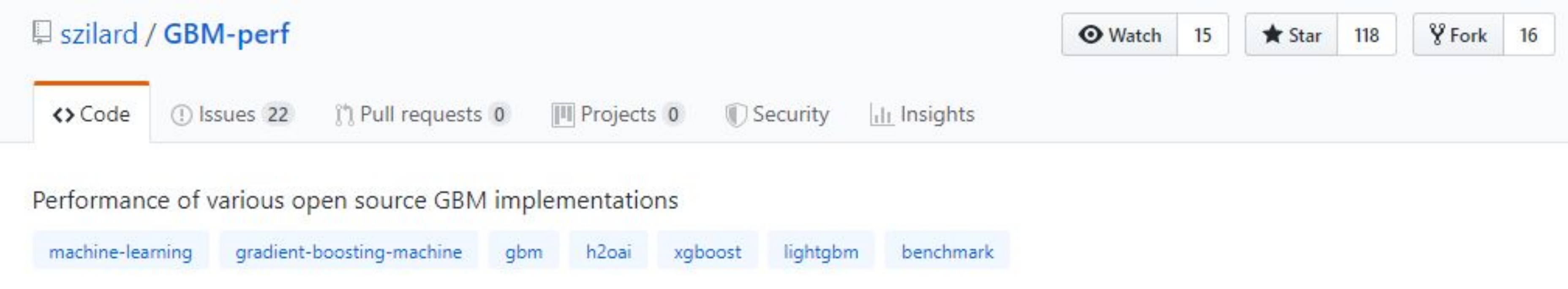
Weak GPU for GBM is quick



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Weak GPU for GBM is quick

xgboost



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Weak GPU for GBM is quick

	0.1M	1M	10M	100M
CPU (Best)	2.735	12.524	70.121	675.223
Quadro P1000	17.529	38.528	103.154	CRASH
GPU Perf. (%)	15.6%	32.5%	67.9%	N/A

Comparison unit: seconds

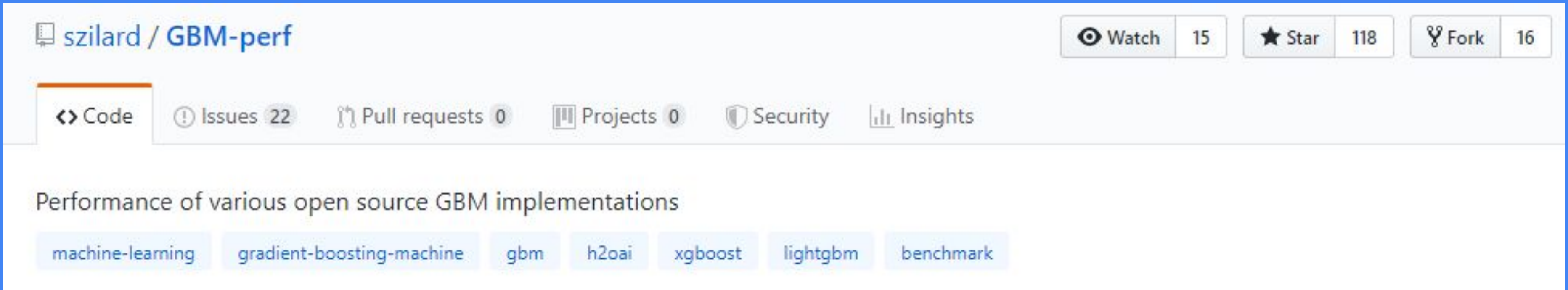
N.B: Dual Xeon 6154 (~$7,000) vs Quadro P1000 (~$400)

Very high GPU usage (can be better using better GPU)

xgboost hist, GPU

Weak GPU for GBM is quick

LightGBM



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Weak GPU for GBM is quick

	0.1M	1M	10M	100M
CPU (Best)	1.801	4.891	47.033	507.580
Quadro P1000	18.345	22.179	62.929	396.233
GPU Perf. (%)	9.8%	22.1%	74.7%	128.1%

Comparison unit: seconds

N.B: Dual Xeon 6154 (~$7,000) vs Quadro P1000 (~$400)

Very low GPU usage (you can use many in parallel!)

LightGBM, GPU

Weak GPU for GBM is quick

Very low GPU usage for LightGBM?

Example using R:

- To train 10x LightGBM models in parallel
- On Airline 10M dataset

Only ~50% GPU usage, if GPU had more RAM, it could train 20 in parallel without any potential compute bottleneck

```
Fri May 31 12:23:45 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.56       Driver Version: 418.56       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro P1000        Off  | 00000000:3B:00.0 Off |                  N/A |
| 34%   35C    P0    N/A /  N/A |   4017MiB /  4040MiB |     48%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Quadro P1000        Off  | 00000000:86:00.0 Off |                  N/A |
| 34%   17C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Quadro P1000        Off  | 00000000:AF:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Quadro P1000        Off  | 00000000:D8:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     13473      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13487      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13501      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13515      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13532      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13549      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13563      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13577      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13594      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13612      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13626      C   /usr/local/lib64/R/bin/exec/R                333MiB |
|    0     13640      C   /usr/local/lib64/R/bin/exec/R                333MiB |
+-----------------------------------------------------------------------------+
```

LightGBM, GPU

Weak GPU for GBM is quick

xgboost

```
Fri May 31 12:37:07 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.56       Driver Version: 418.56       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro P1000        Off  | 00000000:3B:00.0 Off |                  N/A |
| 34%   27C    P0    N/A /  N/A |   3344MiB /  4040MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Quadro P1000        Off  | 00000000:86:00.0 Off |                  N/A |
| 34%   17C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Quadro P1000        Off  | 00000000:AF:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Quadro P1000        Off  | 00000000:D8:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

 Processes:                                                       GPU Memory
  GPU       PID   Type   Process name                             Usage
=============================================================================
    0     23079      C   /usr/local/lib64/R/bin/exec/R               1109MiB
    0     23096      C   /usr/local/lib64/R/bin/exec/R               1109MiB
    0     23113      C   /usr/local/lib64/R/bin/exec/R               1109MiB
```

LightGBM

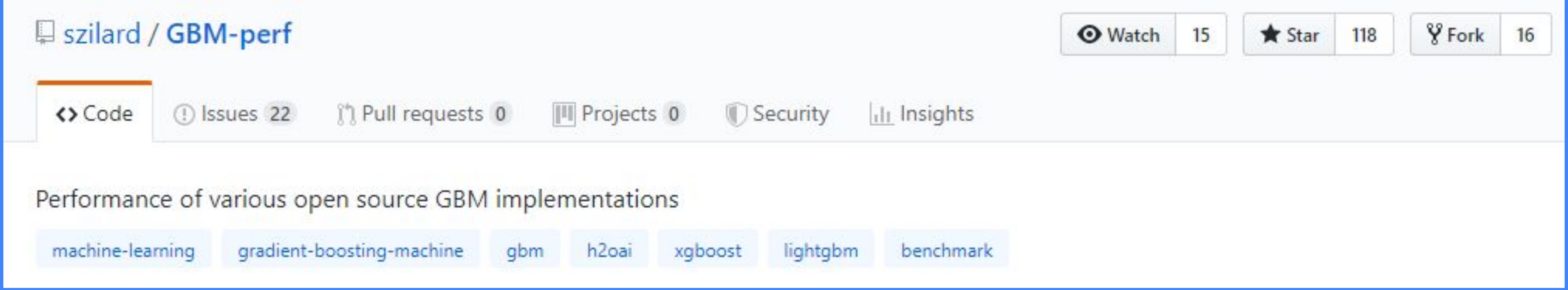
```
Fri May 31 12:23:45 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.56       Driver Version: 418.56       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro P1000        Off  | 00000000:3B:00.0 Off |                  N/A |
| 34%   35C    P0    N/A /  N/A |   4017MiB /  4040MiB |     48%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Quadro P1000        Off  | 00000000:86:00.0 Off |                  N/A |
| 34%   17C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Quadro P1000        Off  | 00000000:AF:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Quadro P1000        Off  | 00000000:D8:00.0 Off |                  N/A |
| 34%   18C    P8    N/A /  N/A |     17MiB /  4040MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

 Processes:                                                       GPU Memory
  GPU       PID   Type   Process name                             Usage
=============================================================================
    0     13473      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13487      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13501      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13515      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13532      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13549      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13563      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13577      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13594      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13612      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13626      C   /usr/local/lib64/R/bin/exec/R                333MiB
    0     13640      C   /usr/local/lib64/R/bin/exec/R                333MiB
```

N.B: xgboost uses 100% GPU for 1 model… no benefits from parallel training (applies to my NVIDIA Quadro P1000 which is not a powerful GPU)

LightGBM, GPU

Multi GPU xgboost



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Multi GPU xgboost

	0.1M	1M	10M	100M
CPU (Best)	2.735	12.524	70.121	675.223
Quadro P1000	17.529	38.528	103.154	CRASH
4 Quadro P1000	18.838	36.877	64.994	232.947
4 GPU Perf. (%)	14.5%	34.0%	107.9%	289.9%

Comparison unit: seconds

N.B: Dual Xeon 6154 (~$7,000) vs Quadro P1000 (~$400) vs 4 Quadro P1000 (~$1,600)

Multi GPU xgboost requires NCCL = only for Linux

xgboost hist, GPU

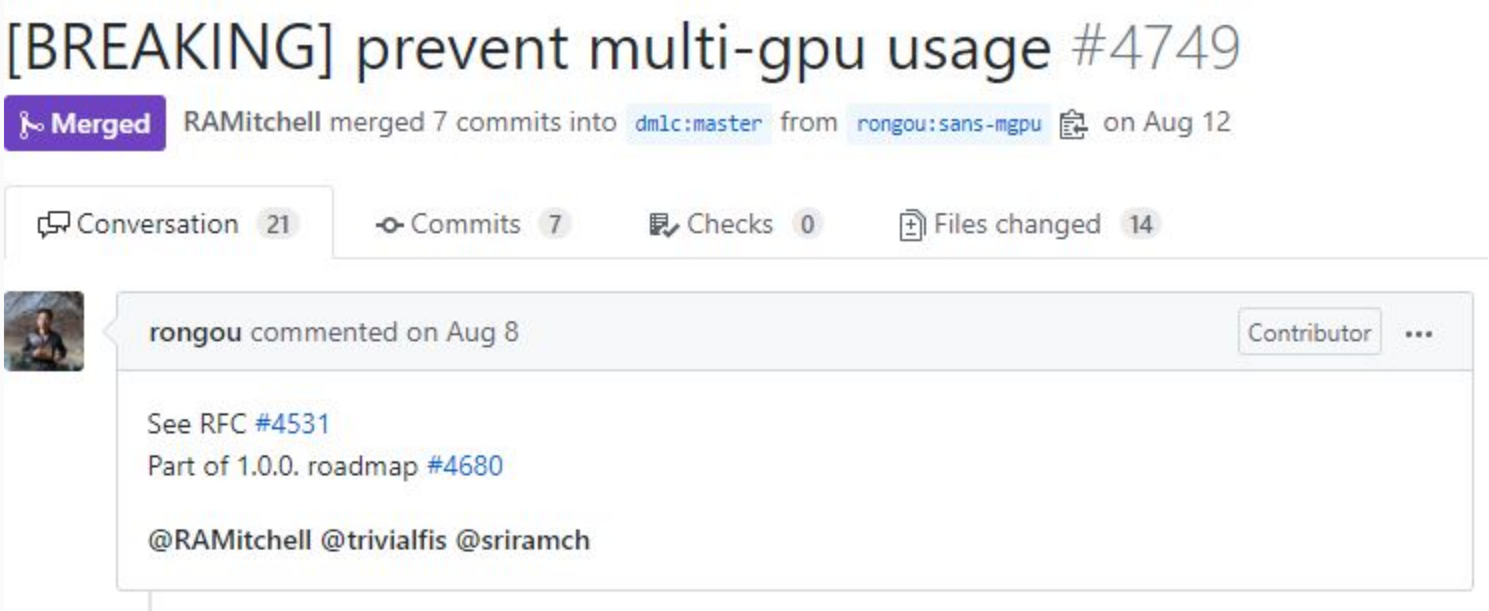
Multi GPU

~~MI i PU~~

~~Multi-GPU~~

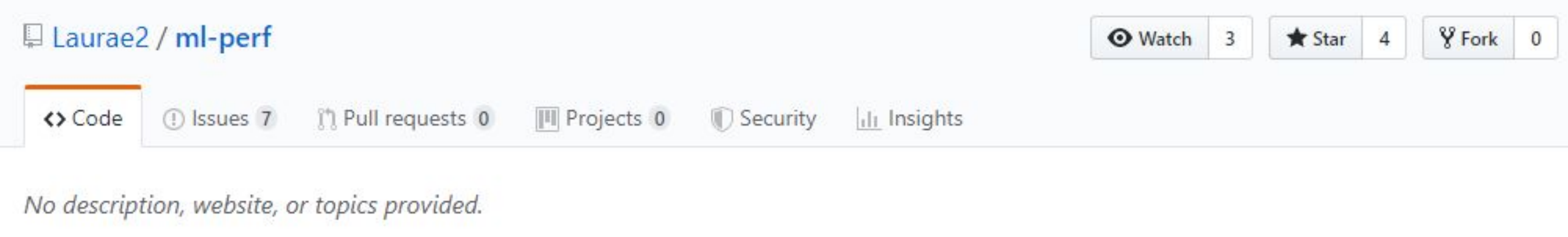
If you want to use xgboost with multiple GPU, use a commit before August 12, 2019.

Multi GPU xgboost



xgboost hist, GPU

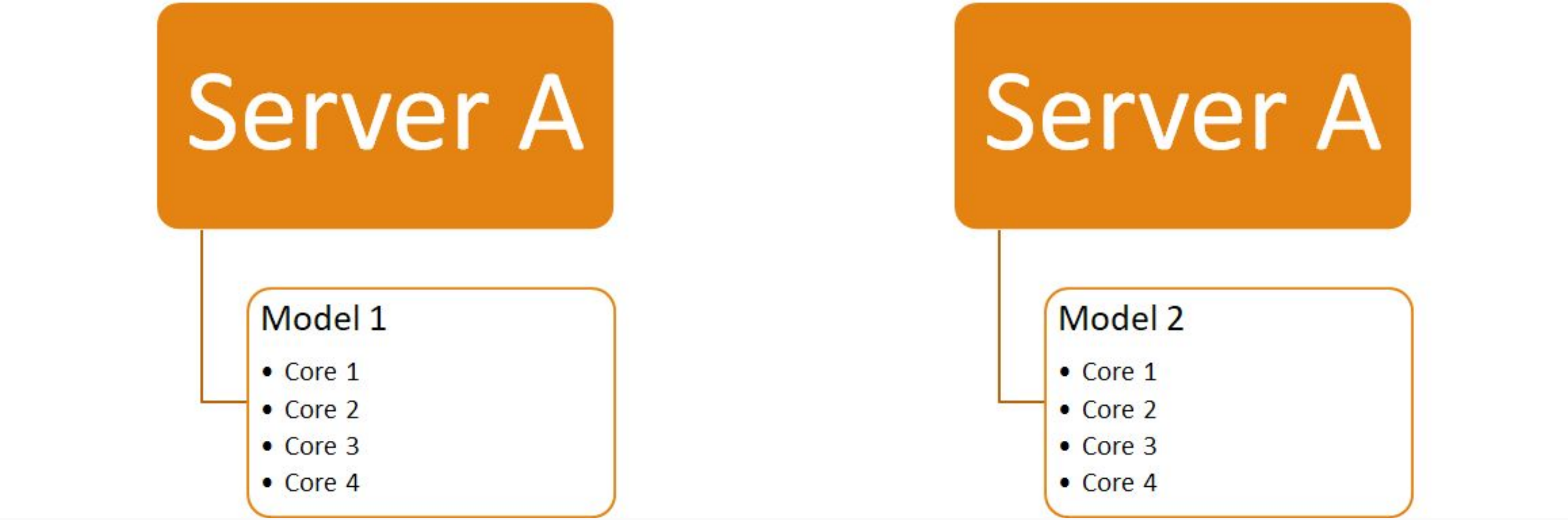
Concurrent Scaling



Data: https://github.com/szilard/benchm-ml/blob/master/0-init/2-gendata.txt

Concurrent Scaling

Sequential Parallel training
“Train each model using all cores”



Concurrent Scaling

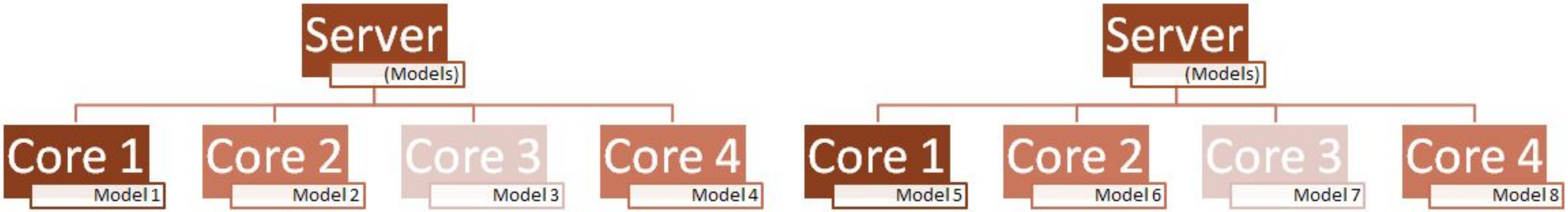
Parallel Threads	Model Threads	Seconds / model
1	1	11.383
1	9	6.565
1	18	6.481
1	35	24.601
1	70	165.947

How to not use parallelism correctly (when data is small)

xgboost hist, CPU

Concurrent Scaling

Parallel Sequential training
"Train each model using 1 core"



Concurrent Scaling

Parallel Threads	Model Threads	Seconds / model (Model)	Seconds / model (Model -> Parallel)	Performance Boost
1	1	11.383	11.389	1x
1	9	6.565	1.456	4.51x
1	18	6.481	0.782	8.29x
1	35	24.601	0.489	50.31x
1	70	165.947	0.428	387.73x

xgboost hist, CPU

Concurrent Scaling

Parallel Threads	Model Threads	Seconds / model (Model)	Seconds / model (Model -> Parallel)	Performance Boost
1	1	11.383	11.389	1x
1	9	6.565	1.456	4.51x
1	18	6.481	0.782	8.29x
1	35	24.601	0.489	50.31x
1	70	165.947	0.428	387.73x

Fastest vs Slowest

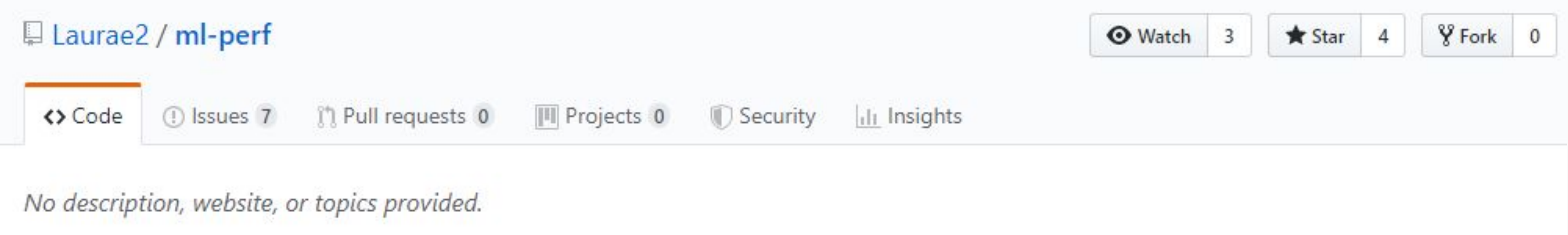
Drawbacks:

- Really fully uses all hardware resources
- Might hit RAM bandwidth limitations
- Might hit L1 / L2 / L3 cache limitations
- Optional: pin process to maximize performance

xgboost hist, CPU

Concurrent Scaling

See more: https://github.com/Laurae2/ml-perf



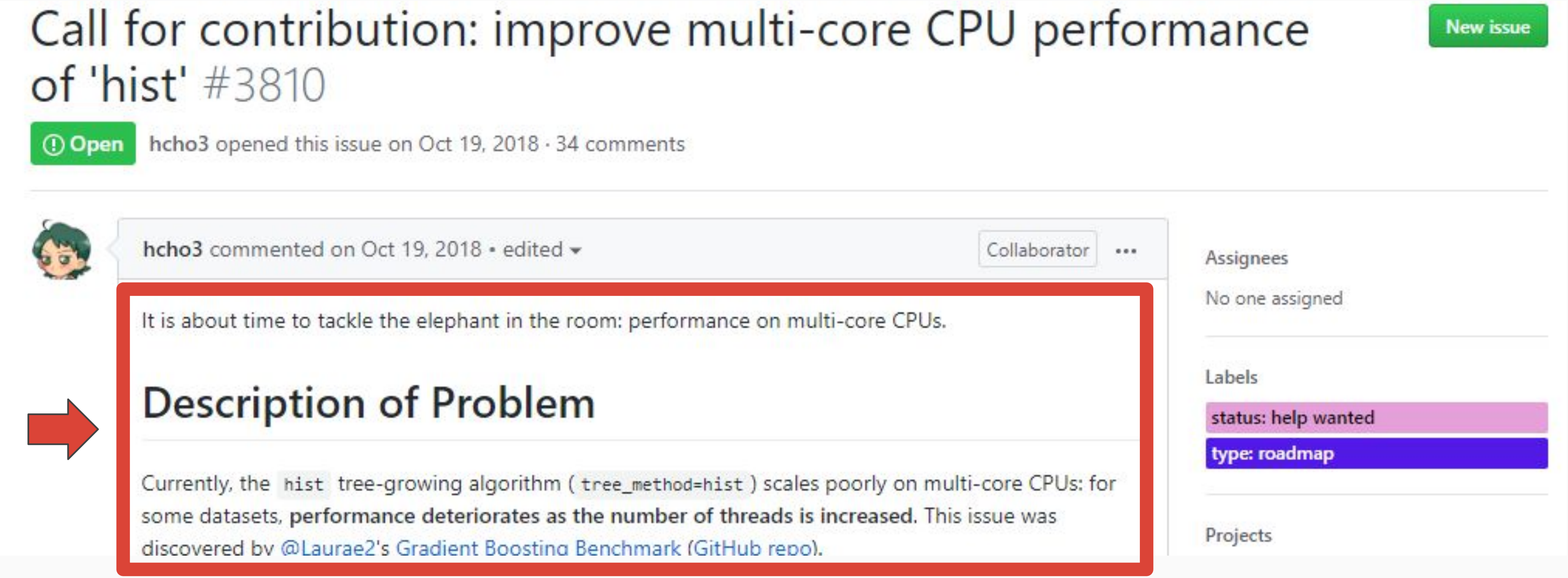
GBM performance optimization

GBM performance optimization

For single model training:

- Use very high frequency CPUs
- Favor desktop CPUs over server CPUs
- Favor CPU overclocking
- Favor recent CPU generations
- Does not need many CPU cores
- Hyperthreading may not help

GBM performance optimization



Call for contribution: improve multi-core CPU performance of 'hist' #3810

New issue

Open hcho3 opened this issue on Oct 19, 2018 · 34 comments

hcho3 commented on Oct 19, 2018 • edited

Collaborator

It is about time to tackle the elephant in the room: performance on multi-core CPUs.

Description of Problem

Currently, the `hist` tree-growing algorithm (`tree_method=hist`) scales poorly on multi-core CPUs: for some datasets, **performance deteriorates as the number of threads is increased.** This issue was discovered by @Laurae2's Gradient Boosting Benchmark (GitHub repo).

Assignees

No one assigned

Labels

status: help wanted

type: roadmap

Projects

GBM performance optimization

For sequential model training:

- Use very high frequency CPUs
- Favor desktop CPUs over server CPUs
- Favor CPU overclocking
- Favor recent CPU generations
- Does not need many CPU cores
- Hyperthreading may not help
- **Favors large tasks**

For parallel model training:

- Use (very) high frequency CPUs
- Favor server CPUs over desktop CPUs
- Favor recent CPU generations
- Need many CPU cores
- Hyperthreading should help (+20~40% performance)
- **Favors short tasks**

GBM performance optimization

intel CORE™ i9 9th Gen

is much better than

intel XEON® inside™

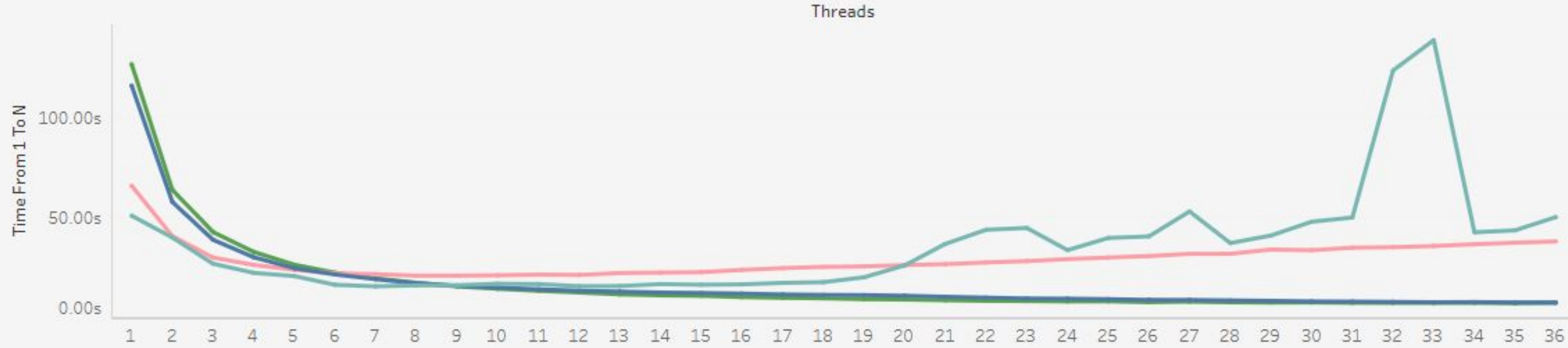
for training GBMs
(in general)

*More details: [https://sites.google.com/view/laurae*pp*/benchmarks](https://sites.google.com/view/lauraepp/benchmarks)*

Model Centric - 1 to N iterations | Model Centric - 0 to N iterations | Memory Centric - 1 to N iterati... | Memory Centric - 0 to N iterati...

Model Centric - 1 to N iterations

Model Centric - Speed Chart (1 to N)



Model Centric - Speed Data (1 to N)

	Memory / Model / Method			
	NUMA		UMA	
	xgboost	LightGBM	xgboost	LightGBM
Threads	hist	gbdt	hist	gbdt
Grand Total	35.82s	17.29s	29.28s	17.20s
1	50.82s	115.34s	65.67s	125.93s
2	40.02s	57.65s	40.70s	63.64s
3	27.00s	38.88s	30.07s	42.72s
4	22.48s	30.22s	26.43s	32.88s
5	20.87s	24.75s	23.98s	26.50s
6	16.56s	21.68s	22.21s	22.56s

Model Centric - Efficiency Data (1 to N)

	Memory / Model / Method			
	NUMA		UMA	
	xgboost	LightGBM	xgboost	LightGBM
Threads	hist	gbdt	hist	gbdt
Grand Total	195.93%	975.06%	238.03%	1166.29%
1	100.00%	100.00%	100.00%	100.00%
2	126.99%	200.09%	161.34%	197.89%
3	188.24%	296.67%	218.41%	294.77%
4	226.03%	381.67%	248.45%	382.99%
5	243.45%	465.96%	273.80%	475.17%
6	306.94%	532.00%	295.64%	558.12%
7	321.71%	596.66%	300.79%	638.18%
8	312.47%	663.79%	311.79%	718.23%

Threads
1 36

Model
- [x] LightGBM
- [x] xgboost

Cancel Apply

Method
- [ ] approx
- [ ] exact
- [x] gbdt
- [x] hist

Cancel Apply

Memory
- [x] NUMA
- [x] UMA

Cancel Apply

Name
- NUMA - xgboost - hist
- NUMA - LightGBM - g...
- UMA - xgboost - hist
- UMA - LightGBM - gbdt

Time From 1 To N
0.00s 1,800.00s

Efficiency From 1 To N
100.00% 1800.00%

+ a b | e a u

GBM performance optimization

Parallelism and scaling on CPU:

“slow” by today’s standards *the fastest GBMs*

	xgboost (exact)	xgboost (hist)	LightGBM
Inherent Parallelism (C++)	Row	Row	Column

GBM performance optimization

Parallelism and scaling on CPU:

"slow" by today's standards *the fastest GBMs*

	xgboost (exact)	xgboost (hist)	LightGBM
Inherent Parallelism (C++)	Row	Row	Column
Scaling with more rows*	Excellent	Poor	Good

* Use more rows on dataset

GBM performance optimization

Parallelism and scaling on CPU:

	"slow" by today's standards		*the fastest GBMs*
	xgboost (exact)	xgboost (hist)	LightGBM
Inherent Parallelism (C++)	Row	Row	Column
Scaling with more rows*	Excellent	Poor	Good
Scaling with more columns**	Good	Poor	Good

* Use more rows on dataset
** Use more columns on dataset

GBM performance optimization

Parallelism and scaling on CPU:

"slow" by today's standards *the fastest GBMs*

	xgboost (exact)	xgboost (hist)	LightGBM
Inherent Parallelism (C++)	Row	Row	Column
Scaling with more rows*	Excellent	Poor	Good
Scaling with more columns**	Good	Poor	Good
Scaling of models***	**Excellent**	**Excellent**	**Excellent**

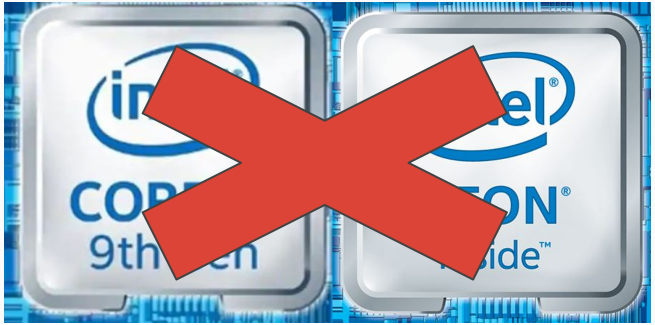
* Use more rows on dataset
** Use more columns on dataset
*** Train models in parallel instead of sequentially

GBM performance optimization

Need speed?

GBM performance optimization

Need speed?

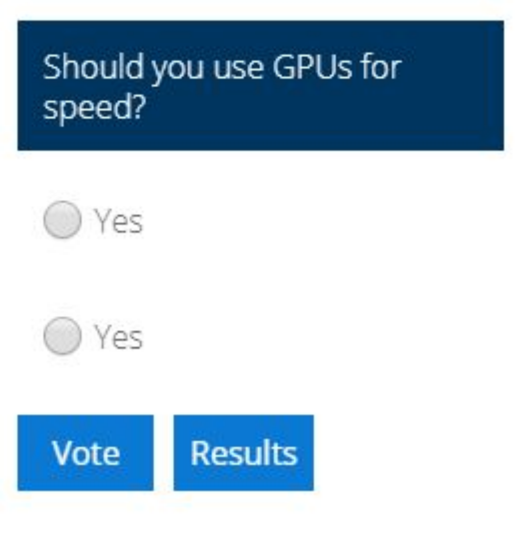


Use GPU*

**with enough RAM to fit dataset in memory*
(and must not care about non-reproducible results)

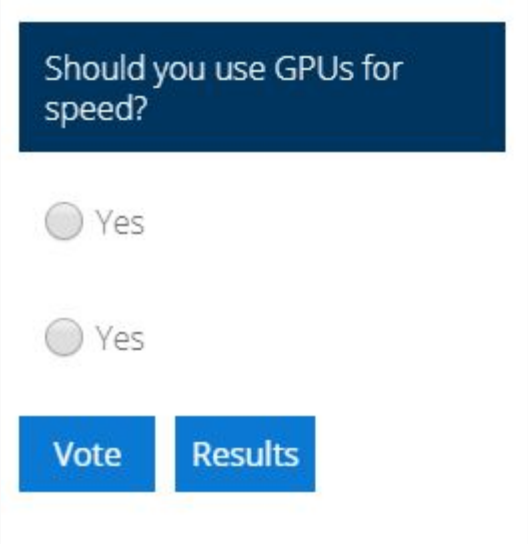
GBM performance optimization

Use GPU*



GBM performance optimization

Use GPU*



GBM performance optimization

Learn hyperparameters:
https://sites.google.com/view/lauraepp/parameters
(especially the most impactful)

Type +
Category +
Parameter +
xgboost +

LightGBM +
Major Impact
Minor Impact
Search

1 - 86 / 86

Parameter
Number of Threads
Learning rate
Number of Iterations
Number of Trees
Early Stopping
Verbosity
Debugging Verbosity
Missing Values
Maximum Depth
Maximum Leaves
Row Sampling
Row Sampling Seed
Row Sampling Frequency
Column Sampling by Tree
Column Sampling by Level
Column Sampling Seed
L1 Regularization
L2 Regularization
L2 Bias Regularization
L2 Categorical Regularization
Categorical Smoothing
Loss Regularization
Hessian Regularization
Minimum Data per Leaf
Histogram Binning
Histogram Binning (Categorical)
Histogram Size
Histogram Sample Construction
Histogram Bin Construction
Matrix Data Type
Sparse Threshold

Parameter name: Number of Threads

Type of parameter: Model

Category of parameter: General

xgboost name: `nthread`
LightGBM name: `num_threads`

Range: `[1, ∞[`
Major Impact: Training Time
Minor Impact: RAM Usage

Description: Number of threads using for training models.

Behavior
Larger is usually better.
Typical: `number of threads of your computer.`
Tips: larger data benefit from more threads, but smaller data has reverse benefits.

xgboost Specific
Multithreaded exact xgboost scales well (nearly linear) as long as the dataset is large enough.
Multithreaded approximate xgboost scales very poorly (parabola) to well (nearly linear) as long as the dataset is large enough.
Multithreaded fast histogram xgboost scales poorly (parabola) to well (nearly linear) as long as the dataset is large enough.
Defaults to `NULL`, the number of threads in your machine.

LightGBM Specific
Multithreaded LightGBM scales well (nearly linear) as long as the dataset is large enough.

dsoukhavong@gmail.com

twitter.com/@Laurae_Cht

github.com/Laurae2

medium.com/@Laurae2