CLBlast: A Tuned BLAS Library

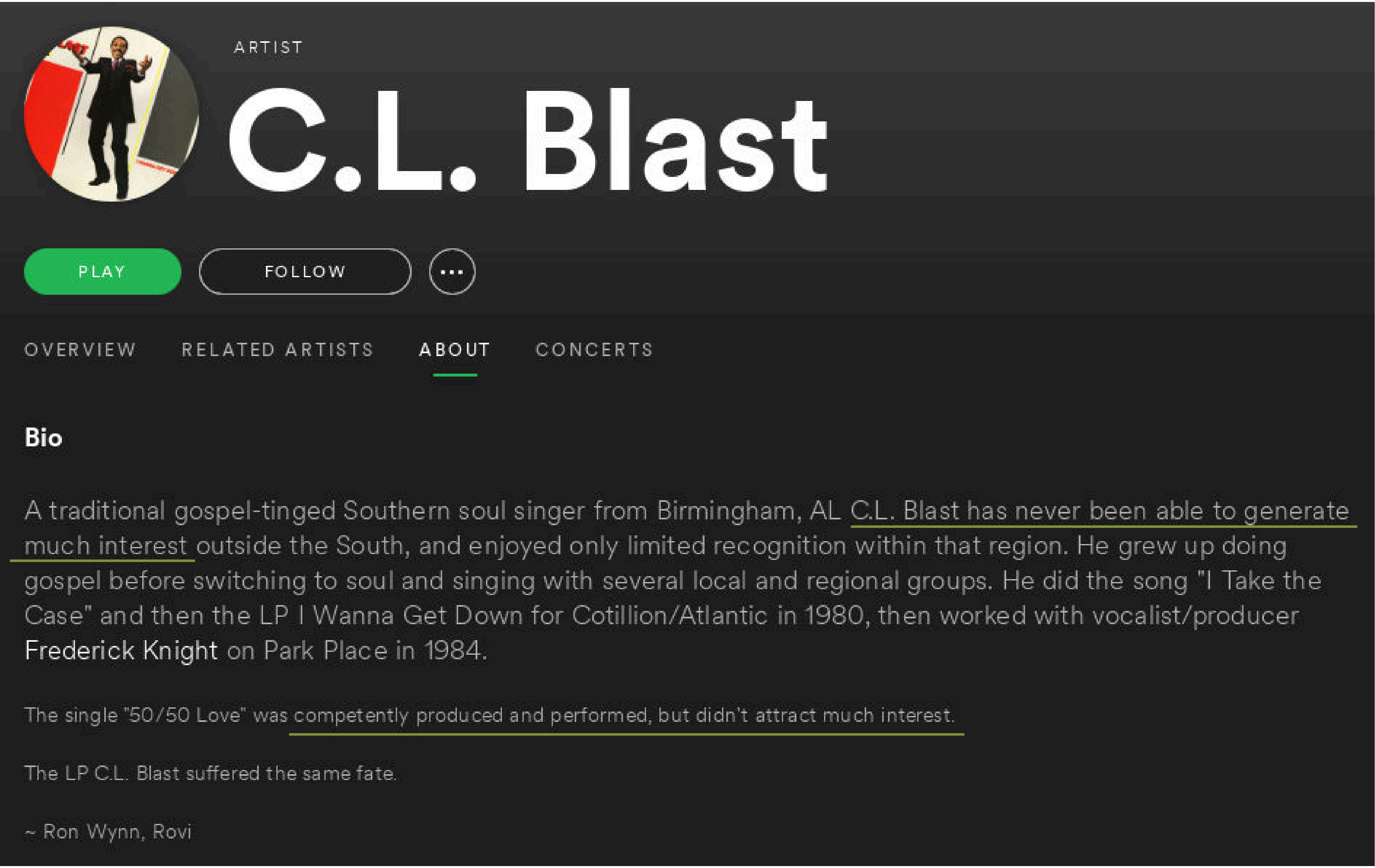
Cedric Nugteren
May 16, 2018

http://github.com/cnugteren/clblast
http://cnugteren.github.io/clblast

CLBlast?

C.L. BLAST
I WANNA G
C.L. BLAST
C.L. BLAST LAY ANOTHER LOG ON THE FIRE
THE COMPLETE JUANA SESSIONS : 1976 TO 1984

CLBlast?



C.L. Blast

Bio

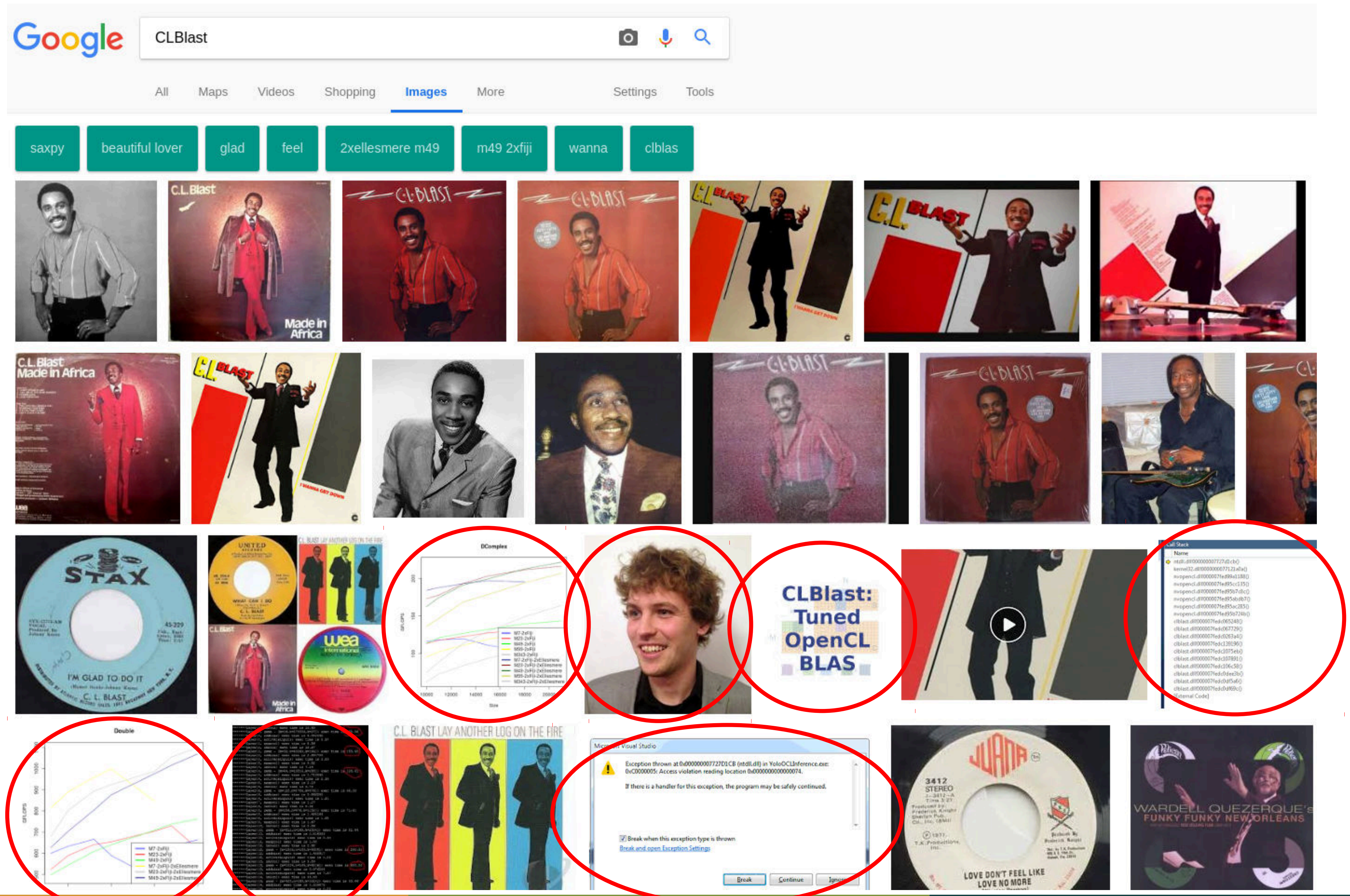
A traditional gospel-tinged Southern soul singer from Birmingham, AL C.L. Blast has never been able to generate much interest outside the South, and enjoyed only limited recognition within that region. He grew up doing gospel before switching to soul and singing with several local and regional groups. He did the song "I Take the Case" and then the LP I Wanna Get Down for Cotillion/Atlantic in 1980, then worked with vocalist/producer Frederick Knight on Park Place in 1984.

The single "50/50 Love" was competently produced and performed, but didn't attract much interest.

The LP C.L. Blast suffered the same fate.

~ Ron Wynn, Rovi

CLBlast?



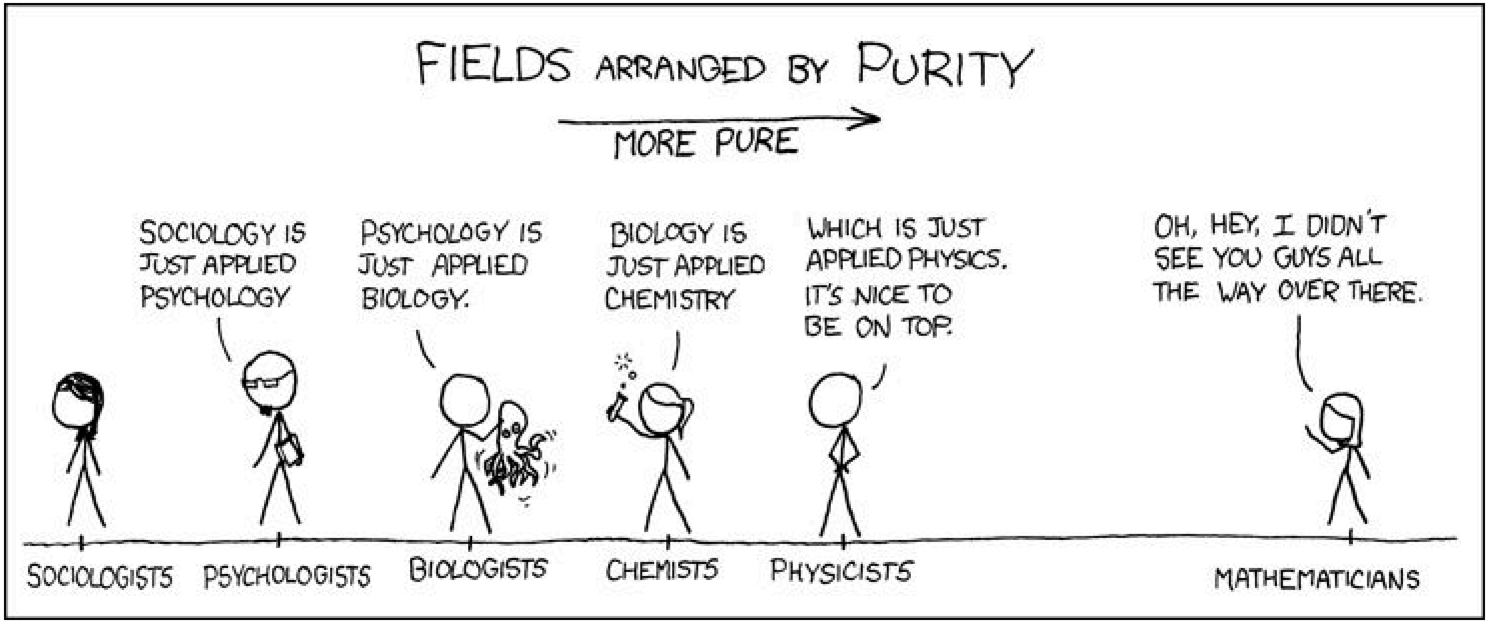
Exception thrown at 0x000000007727D1CB (ntdll.dll) in YoloOCLInference.exe: 0xC0000005: Access violation reading location 0x0000000000000074.

If there is a handler for this exception, the program may be safely continued.

☑ Break when this exception type is thrown

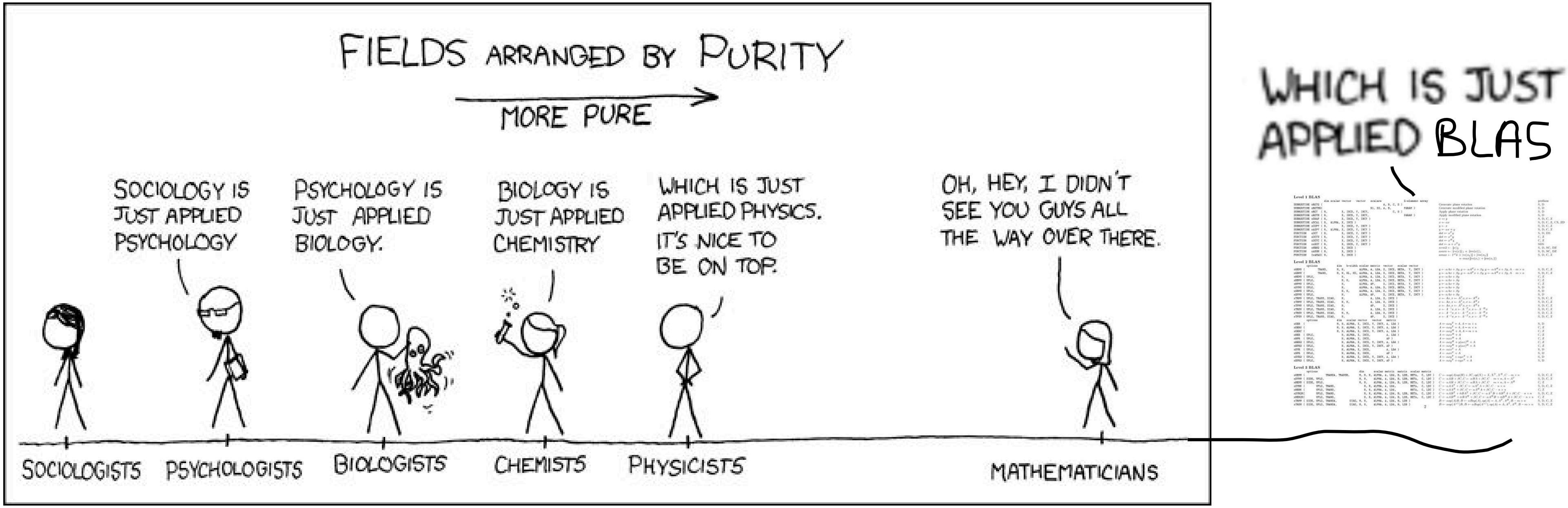
Break and open Exception Settings

What do we use a BLAS library for?


FIELDS ARRANGED BY PURITY
MORE PURE
SOCIOLOGY IS JUST APPLIED PSYCHOLOGY
PSYCHOLOGY IS JUST APPLIED BIOLOGY.
BIOLOGY IS JUST APPLIED CHEMISTRY
WHICH IS JUST APPLIED PHYSICS. IT'S NICE TO BE ON TOP.
OH, HEY, I DIDN'T SEE YOU GUYS ALL THE WAY OVER THERE.
SOCIOLOGISTS
PSYCHOLOGISTS
BIOLOGISTS
CHEMISTS
PHYSICISTS
MATHEMATICIANS

What do we use a BLAS library for?

BLAS: “Basic Linear Algebra Subpgrams”


FIELDS ARRANGED BY PURITY
MORE PURE
SOCIOLOGY IS JUST APPLIED PSYCHOLOGY
PSYCHOLOGY IS JUST APPLIED BIOLOGY.
BIOLOGY IS JUST APPLIED CHEMISTRY
WHICH IS JUST APPLIED PHYSICS. IT'S NICE TO BE ON TOP.
OH, HEY, I DIDN'T SEE YOU GUYS ALL THE WAY OVER THERE.
WHICH IS JUST APPLIED BLAS
SOCIOLOGISTS
PSYCHOLOGISTS
BIOLOGISTS
CHEMISTS
PHYSICISTS
MATHEMATICIANS

But why a new BLAS Library?

- NVIDIA’s cuBLAS is great, or is it?

But why a new BLAS Library?

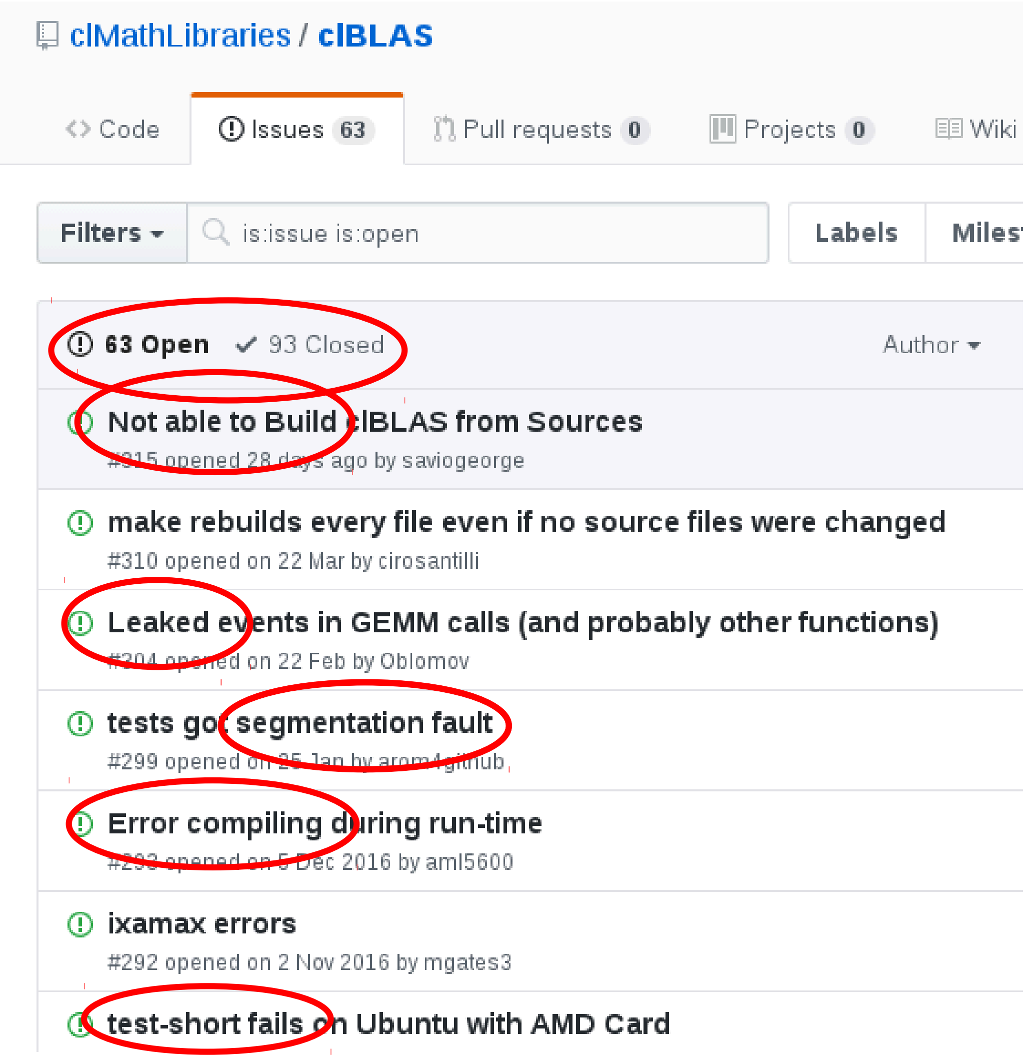
- NVIDIA's cuBLAS is great, or is it?

But why a new BLAS Library?

- NVIDIA's cuBLAS is great, or is it?
  - Not portable, not customisable, not open-source, ...

But why a new BLAS Library?

- NVIDIA's cuBLAS is great, or is it?
  - Not portable, not customisable, not open-source, ...
- Is AMD's clBLAS great?
  - Not performance portable, not well engineered, ...
  - Discontinued, superseeded by incomplete ROCblas



63 Open ✓ 93 Closed

Not able to Build clBLAS from Sources
#315 opened 28 days ago by saviogeorge

make rebuilds every file even if no source files were changed
#310 opened on 22 Mar by cirosantilli

Leaked events in GEMM calls (and probably other functions)
#304 opened on 22 Feb by Oblomov

tests got segmentation fault
#299 opened on 25 Jan by arom4github

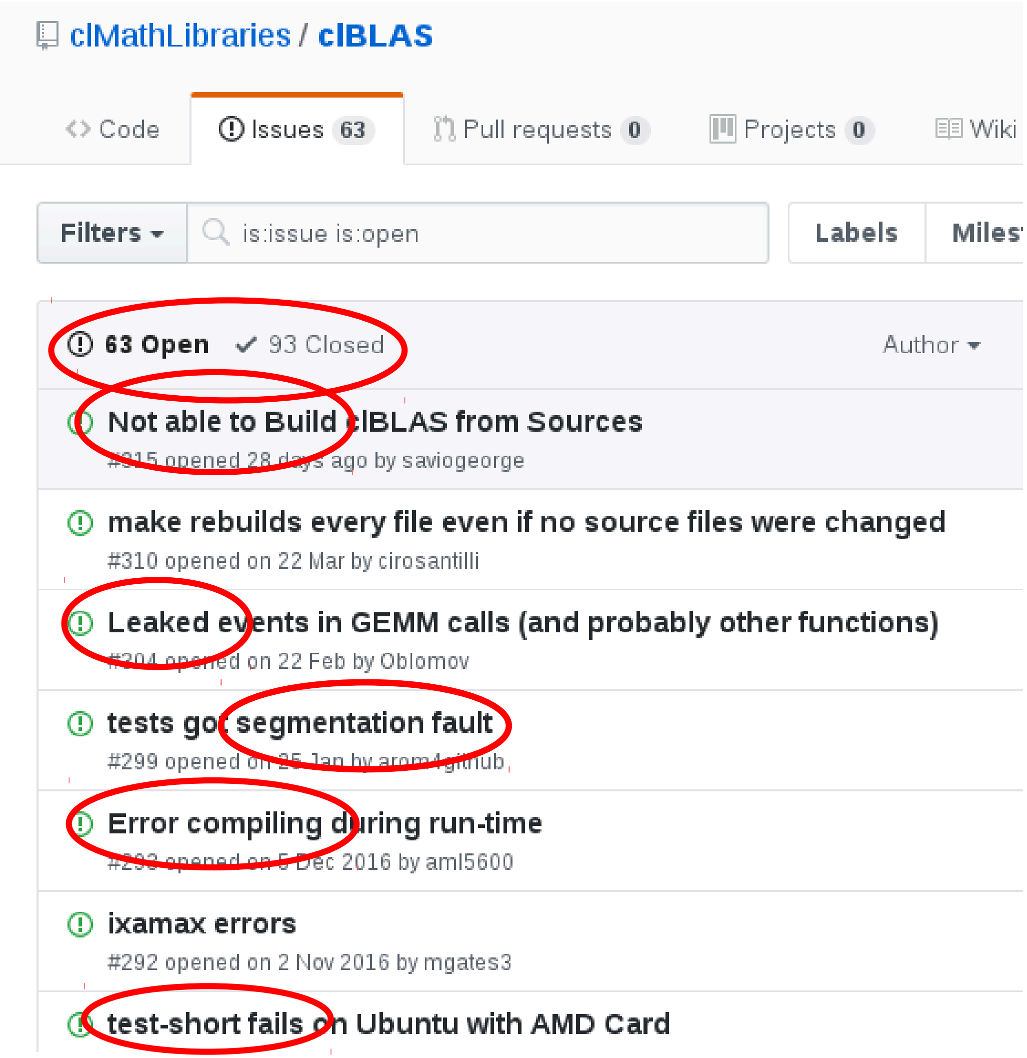
Error compiling during run-time
#293 opened on 5 Dec 2016 by aml5600

ixamax errors
#292 opened on 2 Nov 2016 by mgates3

test-short fails on Ubuntu with AMD Card

But why a new BLAS Library?

- NVIDIA's cuBLAS is great, or is it?
  - Not portable, not customisable, not open-source, ...
- Is AMD's clBLAS great?
  - Not performance portable, not well engineered, ...
  - Discontinued, superseeded by incomplete ROCblas
- Don't vendors ship their hardware with their own libraries?
  - Not a portable solution, and actually not always true...



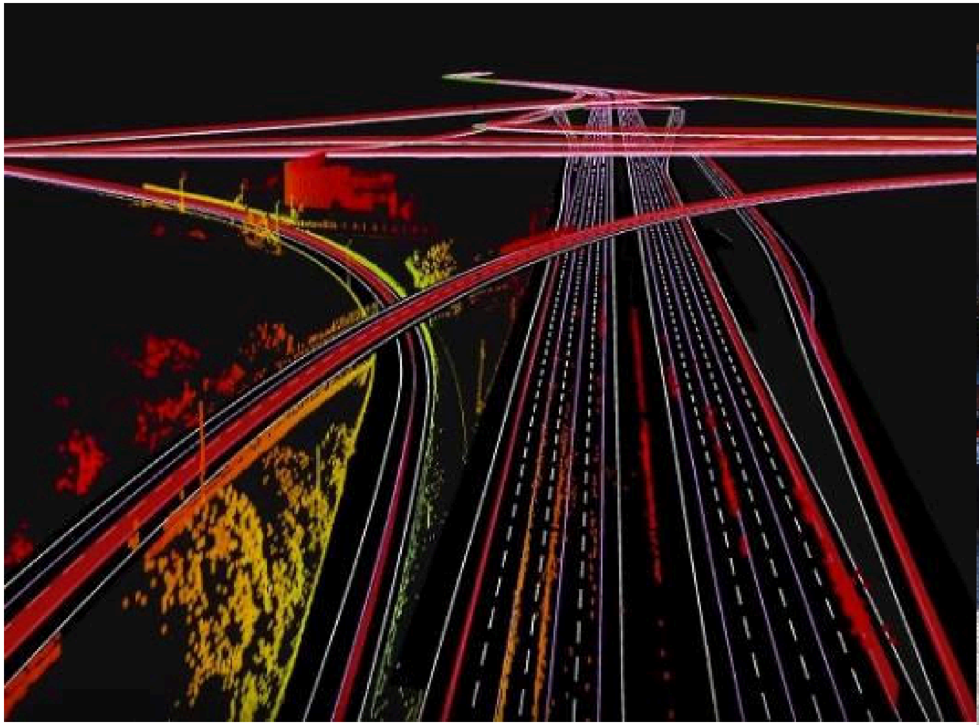
clMathLibraries / clBLAS

63 Open ✓ 93 Closed

- Not able to Build clBLAS from Sources
  #315 opened 28 days ago by saviogeorge
- make rebuilds every file even if no source files were changed
  #310 opened on 22 Mar by cirosantilli
- Leaked events in GEMM calls (and probably other functions)
  #304 opened on 22 Feb by Oblomov
- tests got segmentation fault
  #299 opened on 25 Jan by arom4github
- Error compiling during run-time
  #293 opened on 5 Dec 2016 by aml5600
- ixamax errors
  #292 opened on 2 Nov 2016 by mgates3
- test-short fails on Ubuntu with AMD Card

Why is BLAS Important for TomTom ?

- HDMap making → Deep-learning
- Deep-learning → Fast BLAS libraries



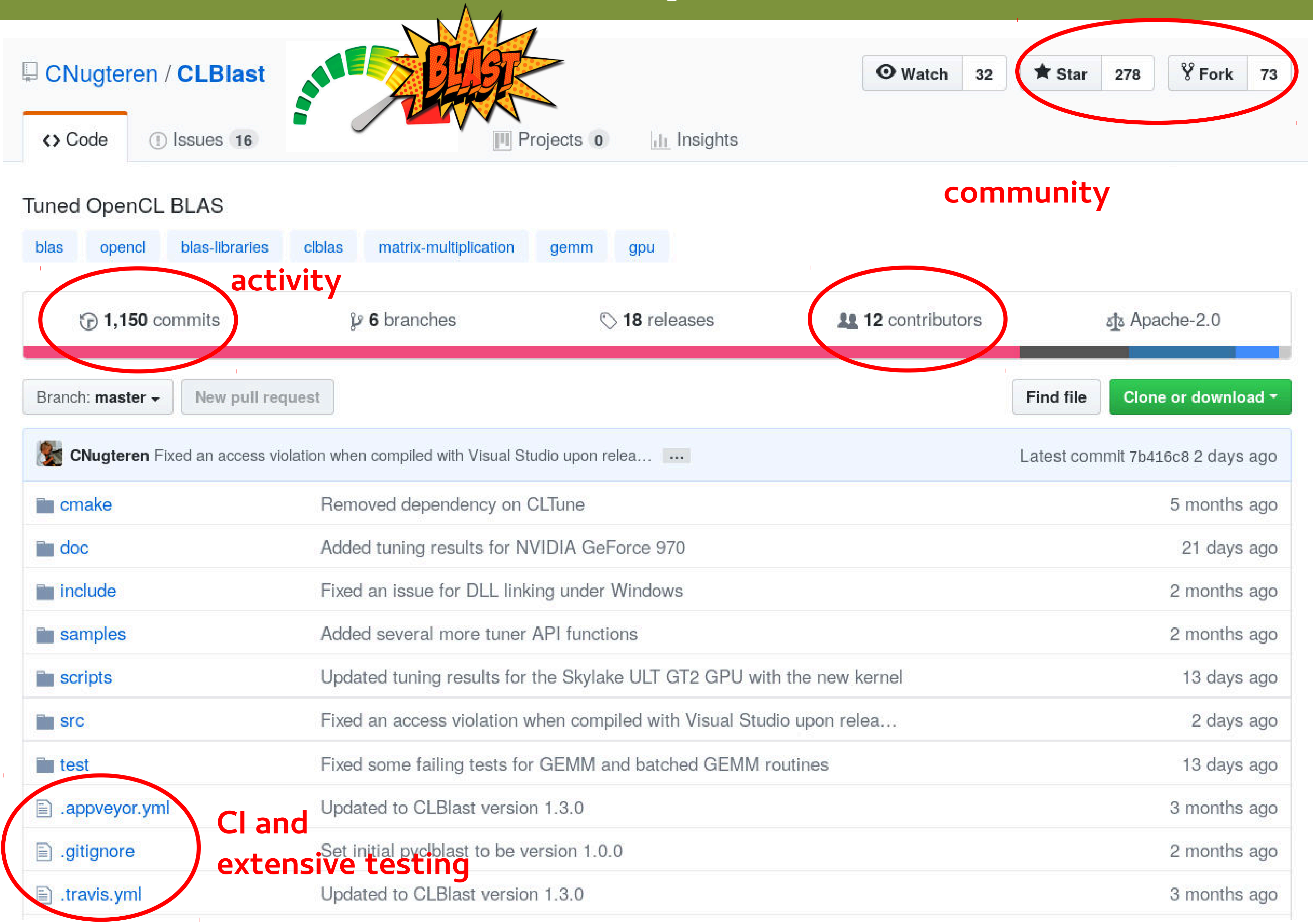
Introducing CLBlast

- CLBlast: Modern C++11 OpenCL BLAS library
- Implements all BLAS routines for all precisions (S, D, C, Z)
- Accelerates all kinds of applications:
  - Fluid dynamics, quantum chemistry, linear algebra, finance, etc.
  - Some extra focus on deep learning

Introducing CLBlast

- CLBlast: Modern C++11 OpenCL BLAS library
- Implements all BLAS routines for all precisions (S, D, C, Z)
- Accelerates all kinds of applications:
  - Fluid dynamics, quantum chemistry, linear algebra, finance, etc.
  - Some extra focus on deep learning
- Already integrated into various projects:
  - JOCLBlast (Java bindings)
  - PyCLBlast (Python bindings)
  - ArrayFire (GPU accelerated library and applications)
  - OpenCL fork of Caffe (github.com/dividiti/ck-caffe)
  - OpenCL fork of TF (github.com/hughperkins/tensorflow-cl)

Introducing CLBlast



CNugteren / **CLBlast**

Watch 32 | Star 278 | Fork 73

Code | Issues 16 | Projects 0 | Insights

community

Tuned OpenCL BLAS

blas | opencl | blas-libraries | clblas | matrix-multiplication | gemm | gpu

activity

1,150 commits | **6** branches | **18** releases | **12** contributors | Apache-2.0

Branch: **master** | New pull request | Find file | Clone or download

CNugteren Fixed an access violation when compiled with Visual Studio upon relea… … Latest commit 7b416c8 2 days ago

cmake	Removed dependency on CLTune	5 months ago
doc	Added tuning results for NVIDIA GeForce 970	21 days ago
include	Fixed an issue for DLL linking under Windows	2 months ago
samples	Added several more tuner API functions	2 months ago
scripts	Updated tuning results for the Skylake ULT GT2 GPU with the new kernel	13 days ago
src	Fixed an access violation when compiled with Visual Studio upon relea…	2 days ago
test	Fixed some failing tests for GEMM and batched GEMM routines	13 days ago
.appveyor.yml	Updated to CLBlast version 1.3.0	3 months ago
.gitignore	Set initial pyclblast to be version 1.0.0	2 months ago
.travis.yml	Updated to CLBlast version 1.3.0	3 months ago

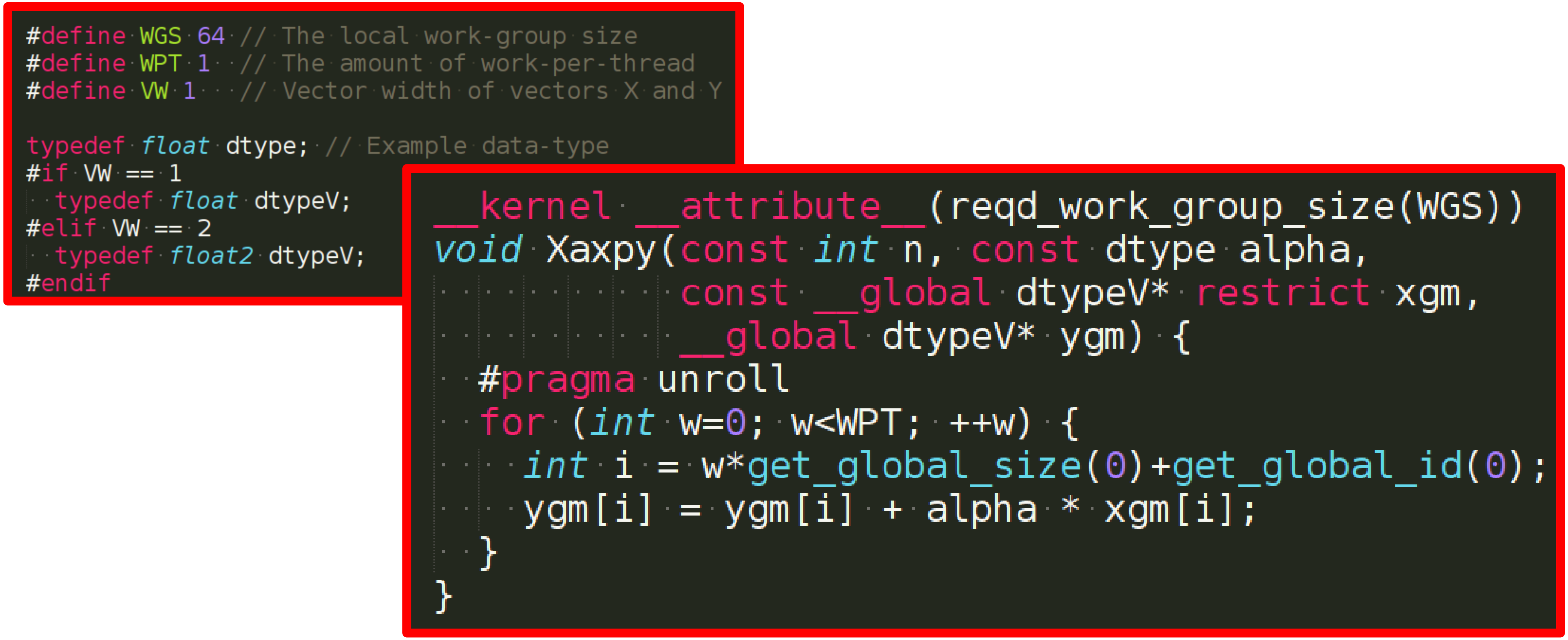
CI and extensive testing

But… is it fast?

- All kernels are generic and tunable thanks to integration of the CLTune auto-tuner (presented at MCSoC ‘15 and GTC ‘16)

But... is it fast?

- All kernels are generic and tunable thanks to integration of the CLTune auto-tuner (presented at MCSoC ‘15 and GTC ‘16)



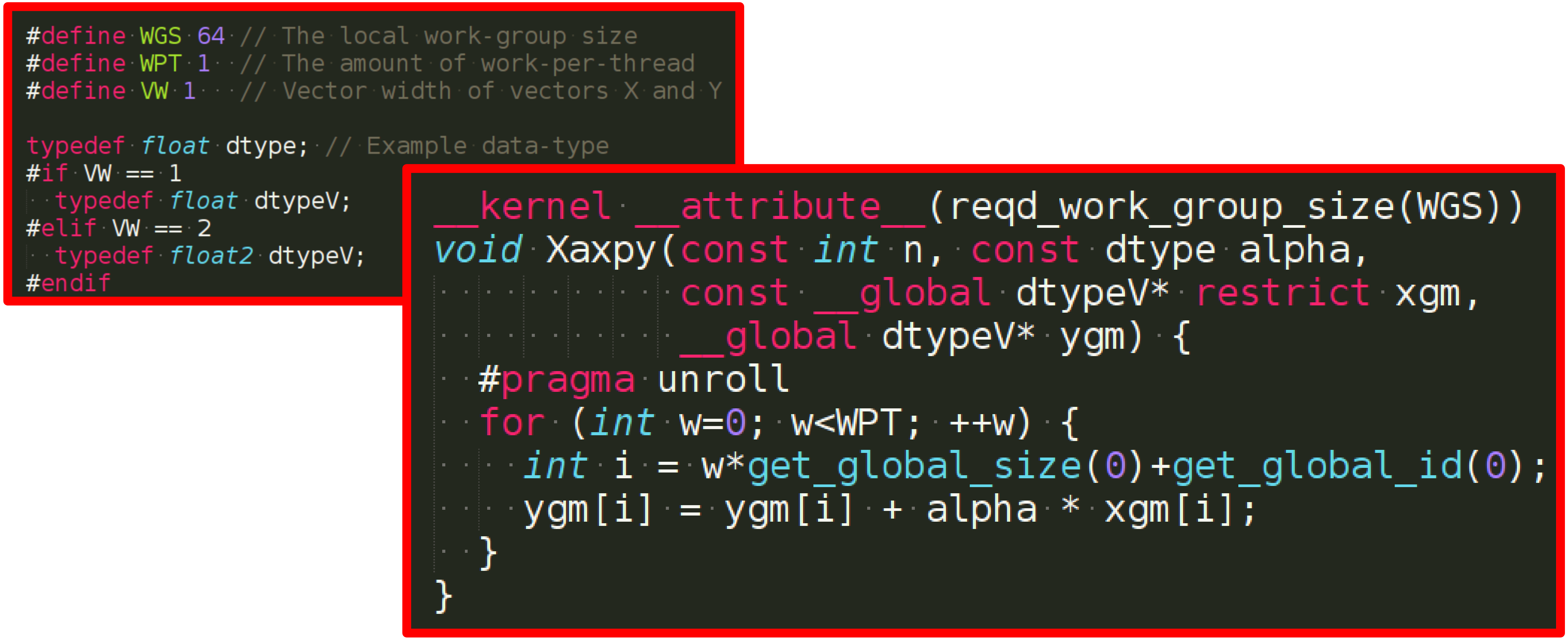
```
#define WGS 64 // The local work-group size
#define WPT 1  // The amount of work-per-thread
#define VW 1   // Vector width of vectors X and Y

typedef float dtype; // Example data-type
#if VW == 1
  typedef float dtypeV;
#elif VW == 2
  typedef float2 dtypeV;
#endif
```

```
__kernel __attribute__(reqd_work_group_size(WGS))
void Xaxpy(const int n, const dtype alpha,
           const __global dtypeV* restrict xgm,
           __global dtypeV* ygm) {
  #pragma unroll
  for (int w=0; w<WPT; ++w) {
    int i = w*get_global_size(0)+get_global_id(0);
    ygm[i] = ygm[i] + alpha * xgm[i];
  }
}
```

But... is it fast?

- All kernels are generic and tunable thanks to integration of the CLTune auto-tuner (presented at MCSoC ‘15 and GTC ‘16)



```
#define WGS 64 // The local work-group size
#define WPT 1  // The amount of work-per-thread
#define VW 1   // Vector width of vectors X and Y

typedef float dtype; // Example data-type
#if VW == 1
  typedef float dtypeV;
#elif VW == 2
  typedef float2 dtypeV;
#endif
```

```
__kernel __attribute__(reqd_work_group_size(WGS))
void Xaxpy(const int n, const dtype alpha,
           const __global dtypeV* restrict xgm,
           __global dtypeV* ygm) {
  #pragma unroll
  for (int w=0; w<WPT; ++w) {
    int i = w*get_global_size(0)+get_global_id(0);
    ygm[i] = ygm[i] + alpha * xgm[i];
  }
}
```

- Tuned out-of-the-box for 50+ common devices
  - For new devices: run the auto-tuner when installing CLBlast

CLBlast Benchmark Results

AXPY
regular
(in GB/s)

AXPY
odd
(in GB/s)

GEMV
regular
(in GB/s)

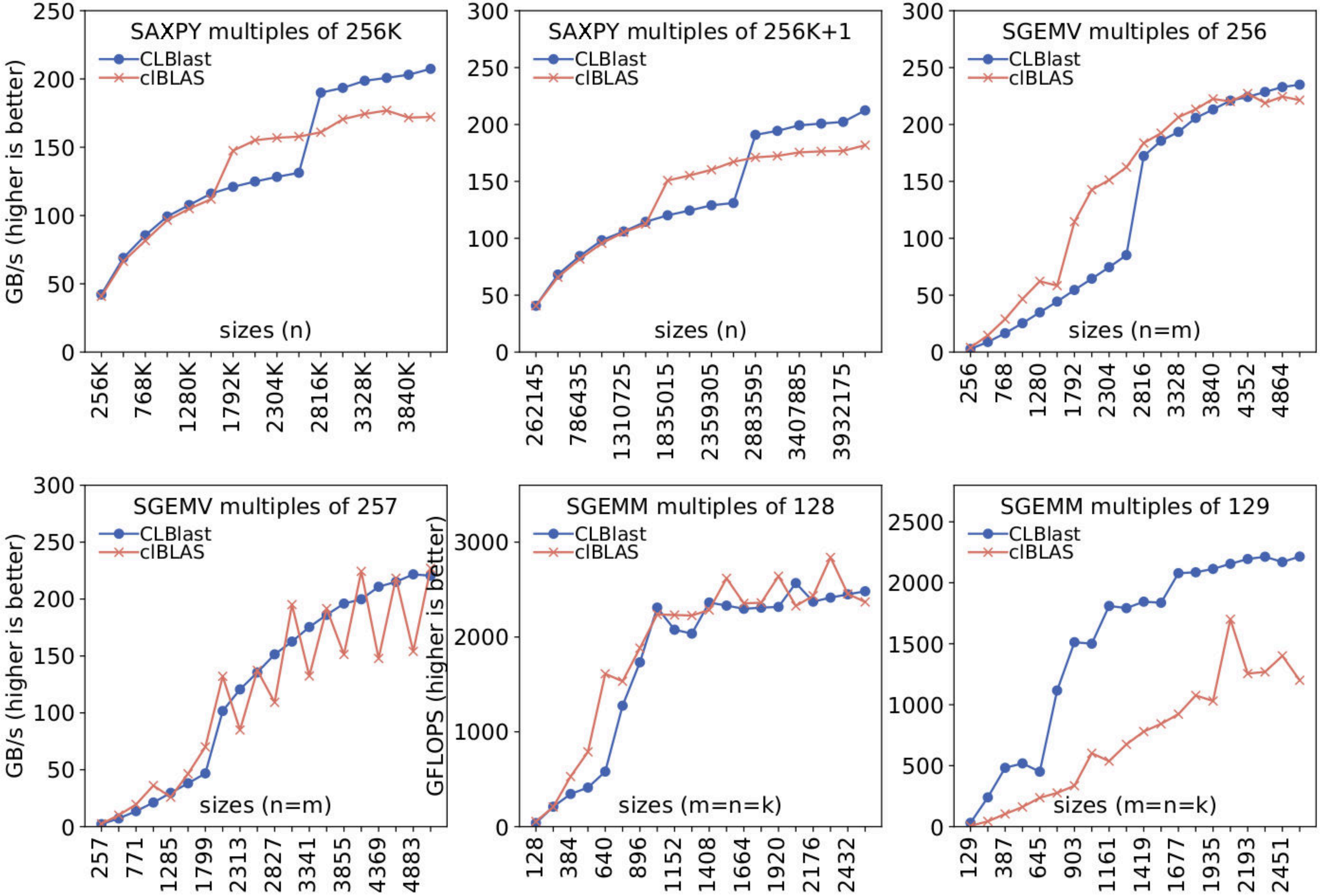
GEMV
odd
(in GB/s)

GEMM
regular
(in GFLOPS)

GEMM
odd
(in GFLOPS)

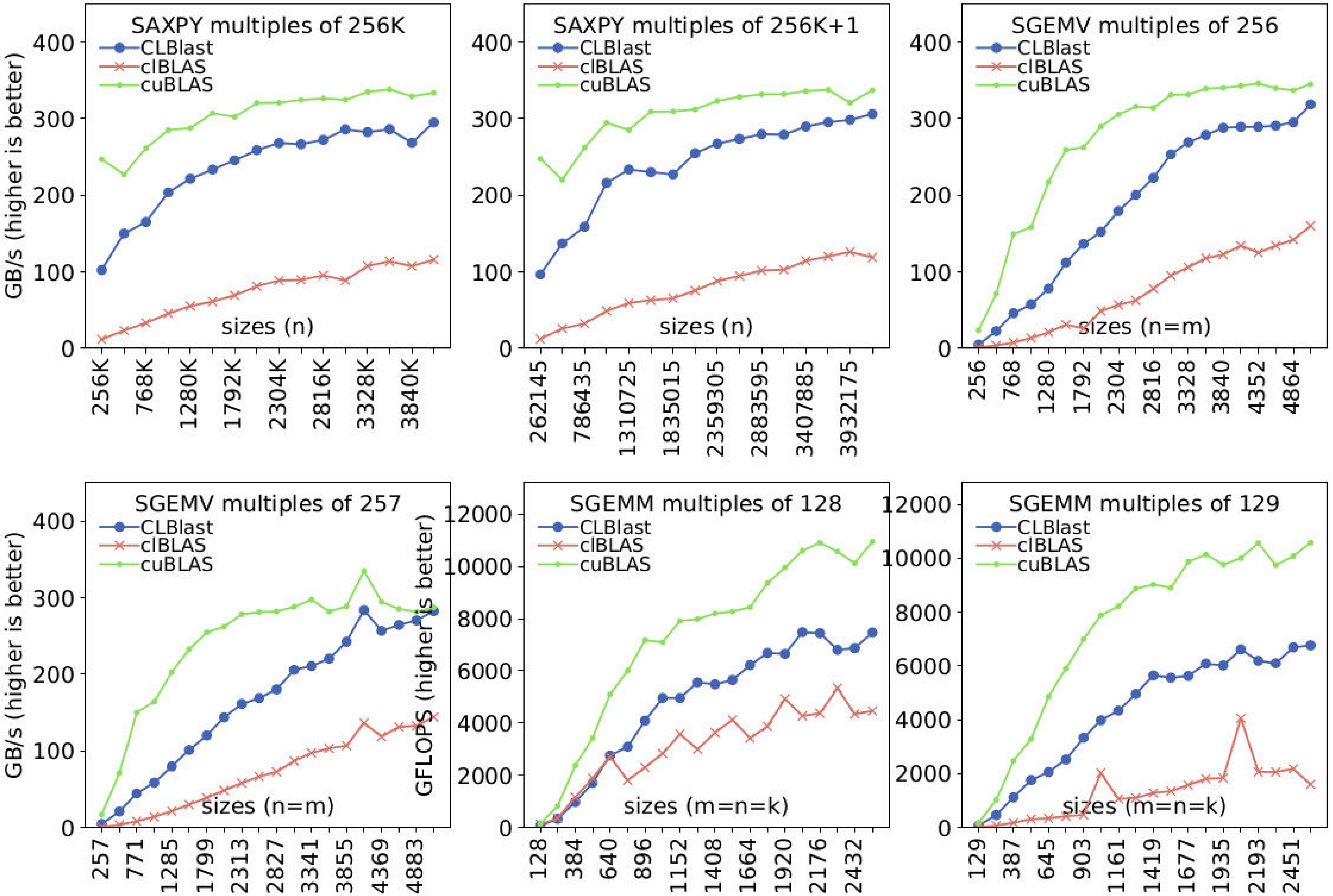
- Higher is better
- More results at http://cnugteren.github.io/clblast

CLBlast on AMD Radeon HD7970



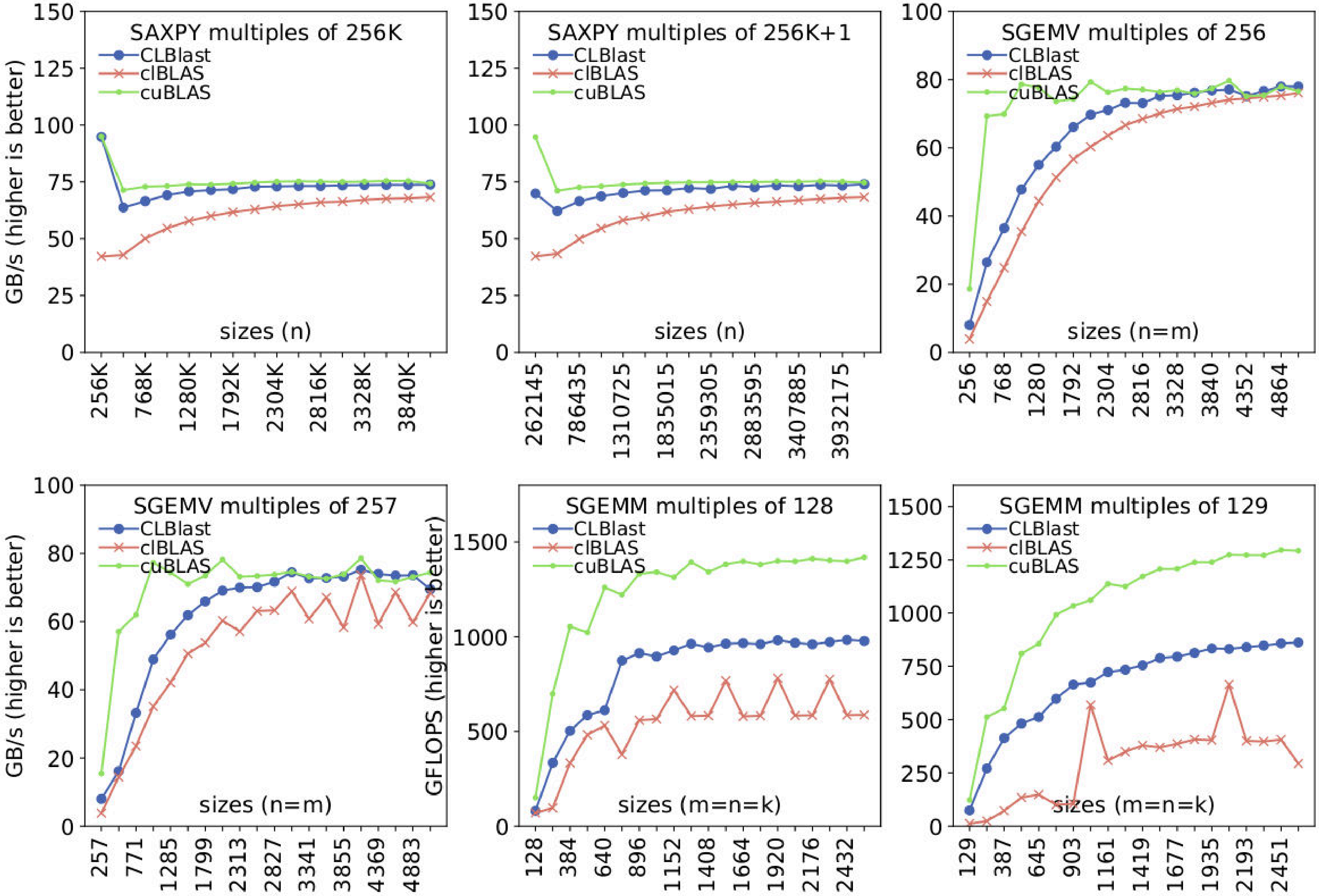
- On-par or better than clBLAS (especially for odd-sized GEMM)

CLBlast on NVIDIA Titan X Pascal



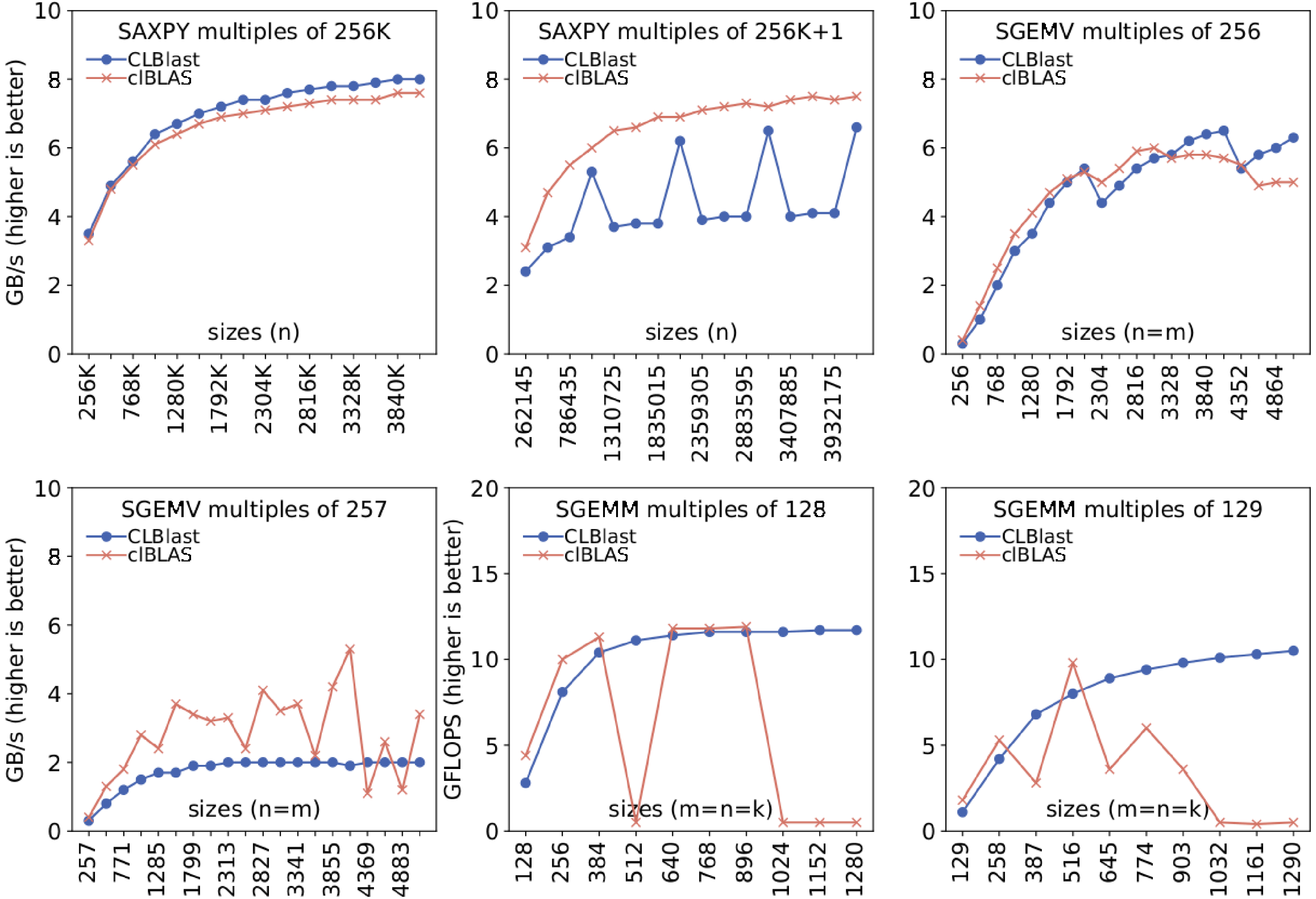
- Much better than clBLAS, reasonably close to cuBLAS

CLBlast on NVIDIA GeForce GTX 750Ti



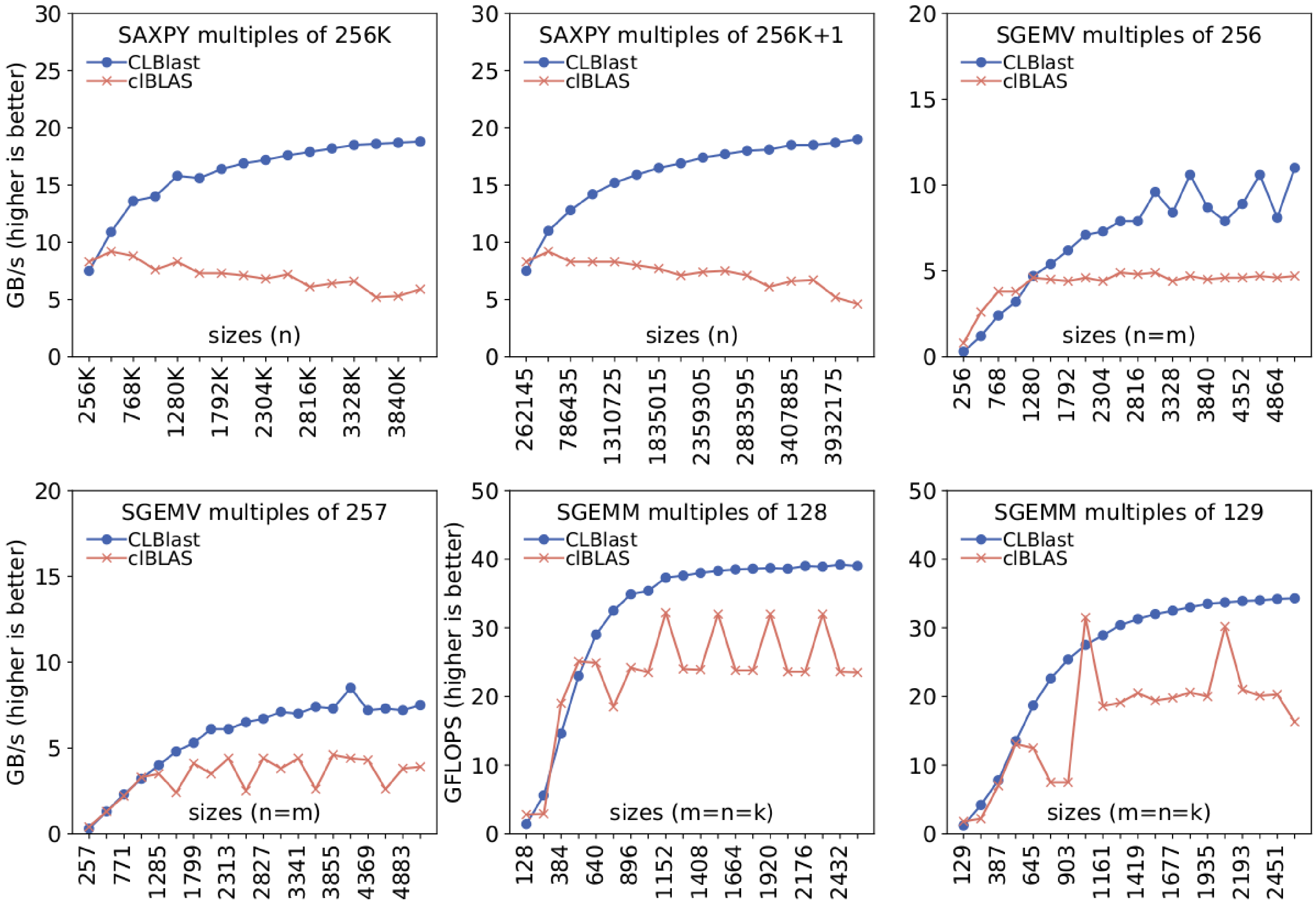
- Much better than clBLAS, reasonably close to cuBLAS

CLBlast on ARM Mali T628



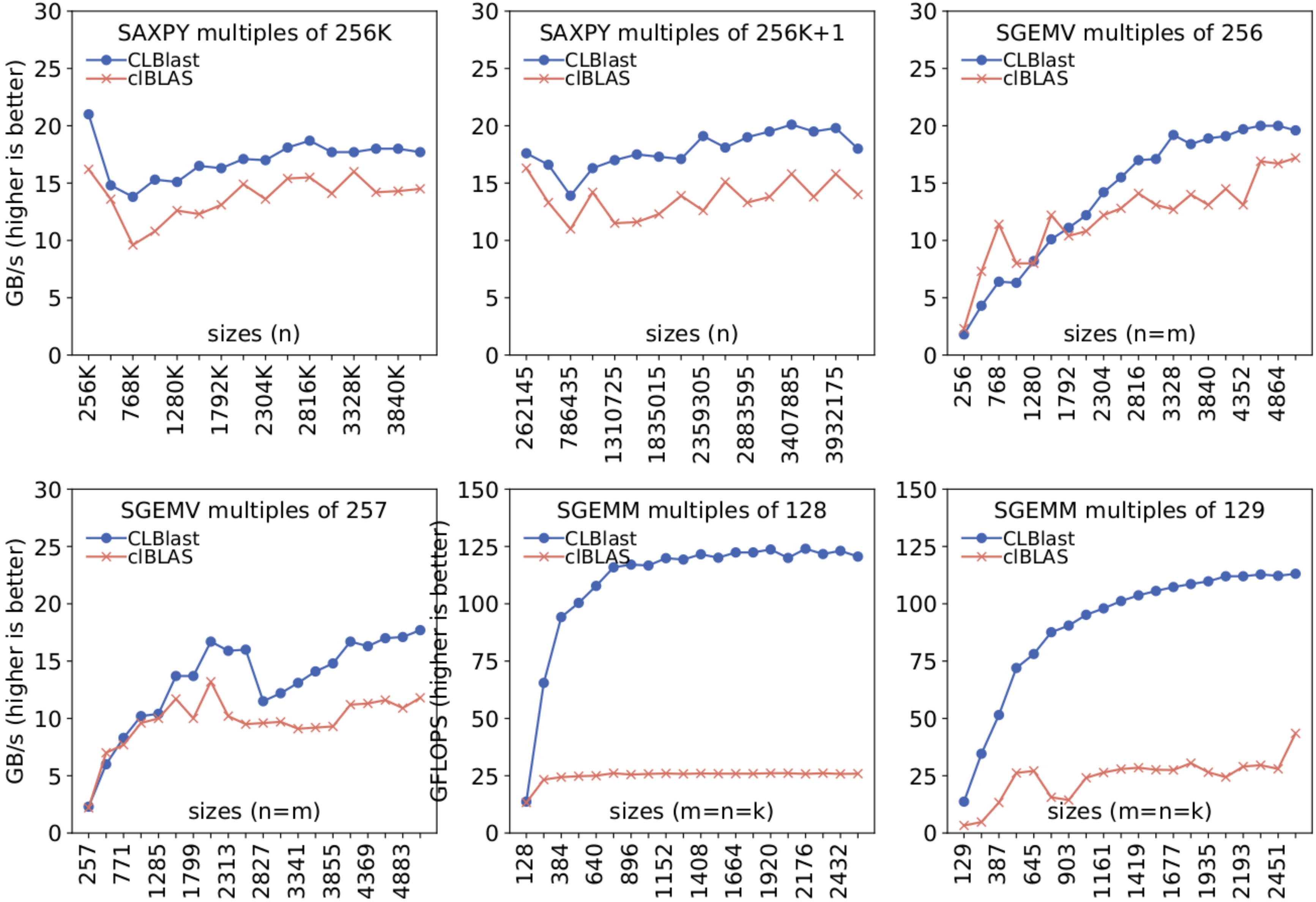
- GEMM much better for CLBlast

CLBlast on Intel Core i5-6200U CPU



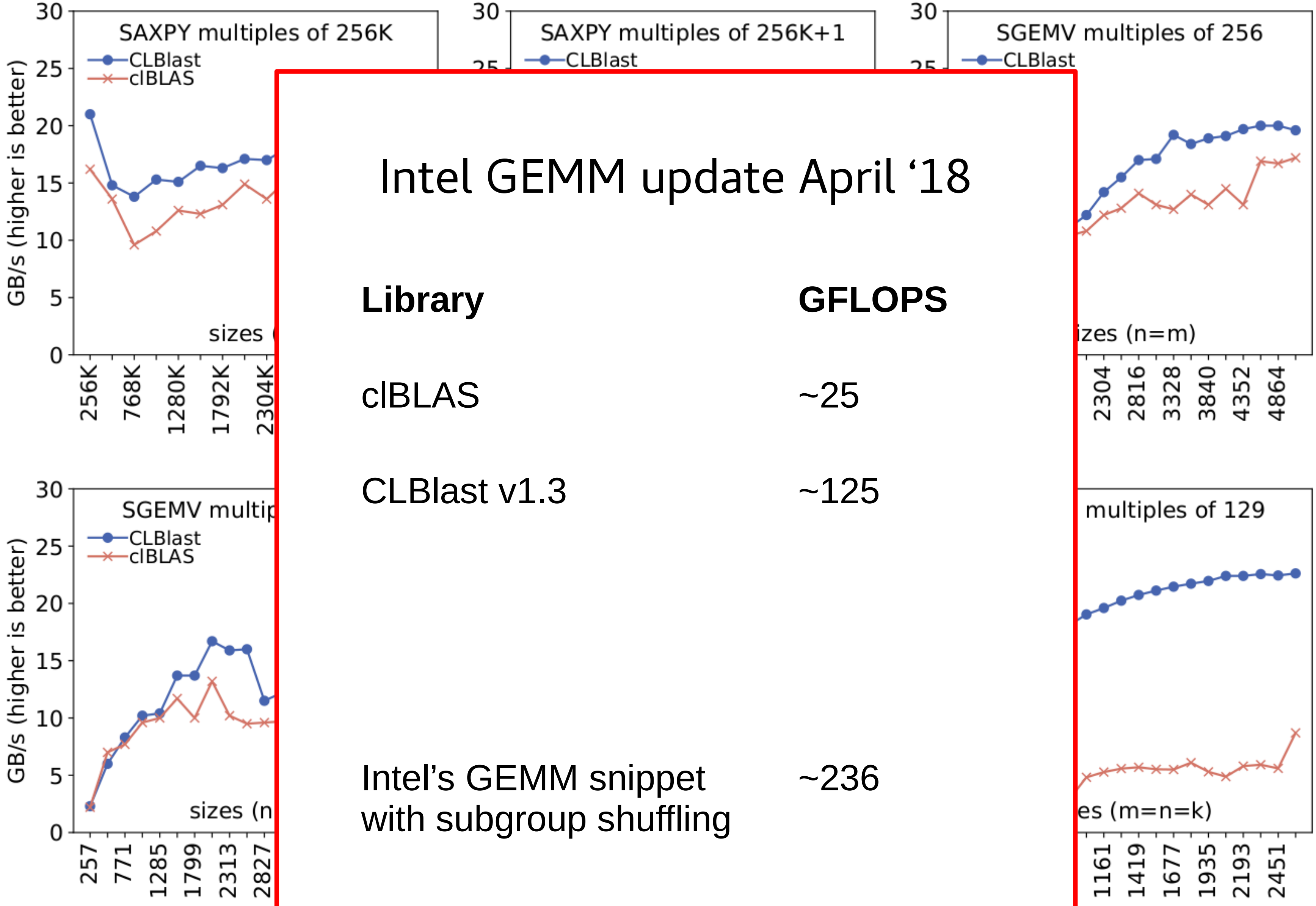
- On-par or better than clBLAS

CLBlast on Intel Skylake ULT GT2 GPU



- On-par or better than clBLAS (especially for GEMM)

CLBlast on Intel Skylake ULT GT2 GPU

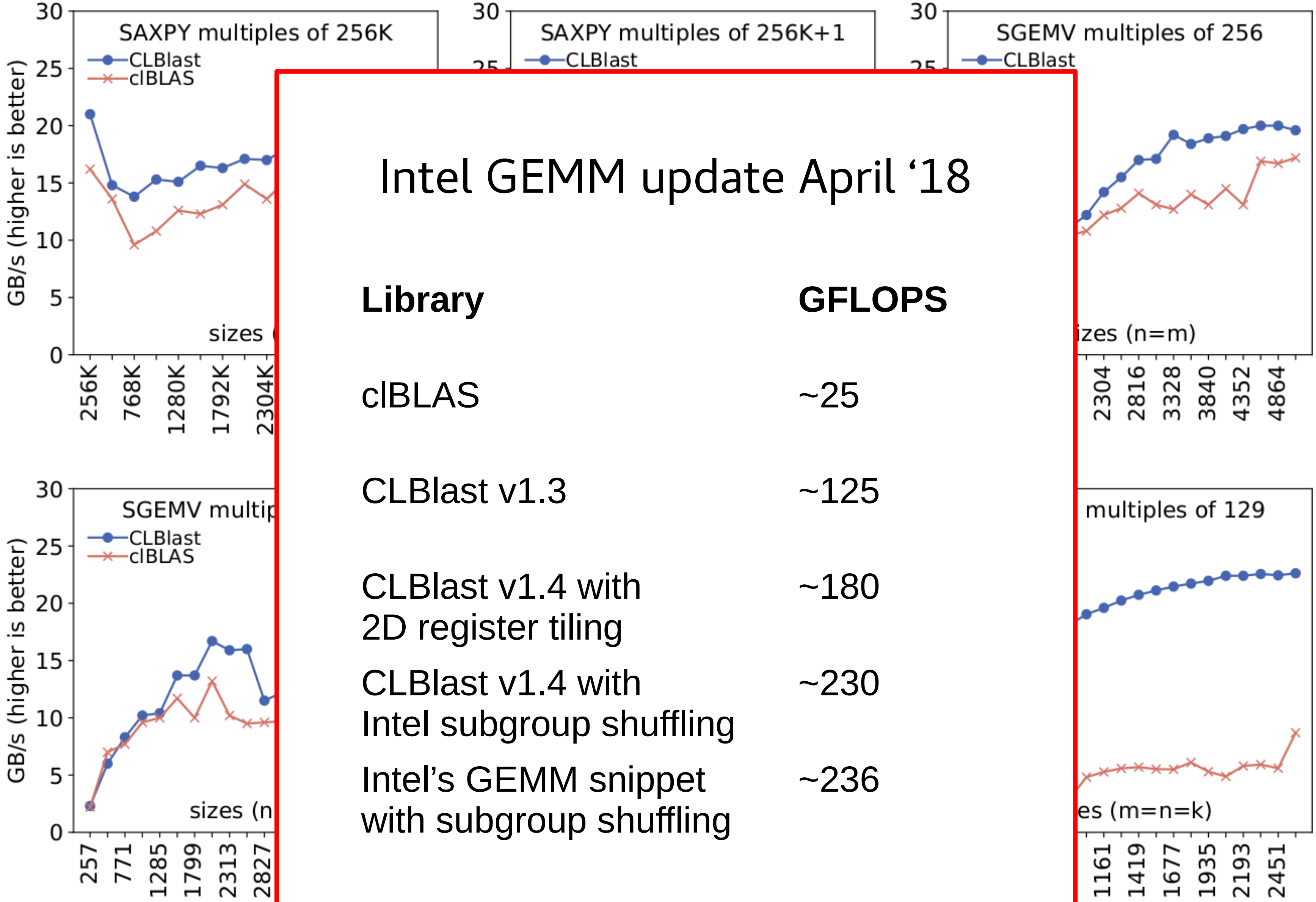


Intel GEMM update April ‘18

Library	GFLOPS
clBLAS	~25
CLBlast v1.3	~125
Intel’s GEMM snippet with subgroup shuffling	~236

- On-par or bet

CLBlast on Intel Skylake ULT GT2 GPU



Intel GEMM update April ‘18

Library	**GFLOPS**
clBLAS	~25
CLBlast v1.3	~125
CLBlast v1.4 with 2D register tiling	~180
CLBlast v1.4 with Intel subgroup shuffling	~230
Intel’s GEMM snippet with subgroup shuffling	~236

- On-par or bet

CLBlast for Deep Learning

- What can we do for the deep-learning community?
  - Problem-specific tuning
  - Half-precision floating-point (FP16)
  - Batched routines

Tuning Only for a Single Size?

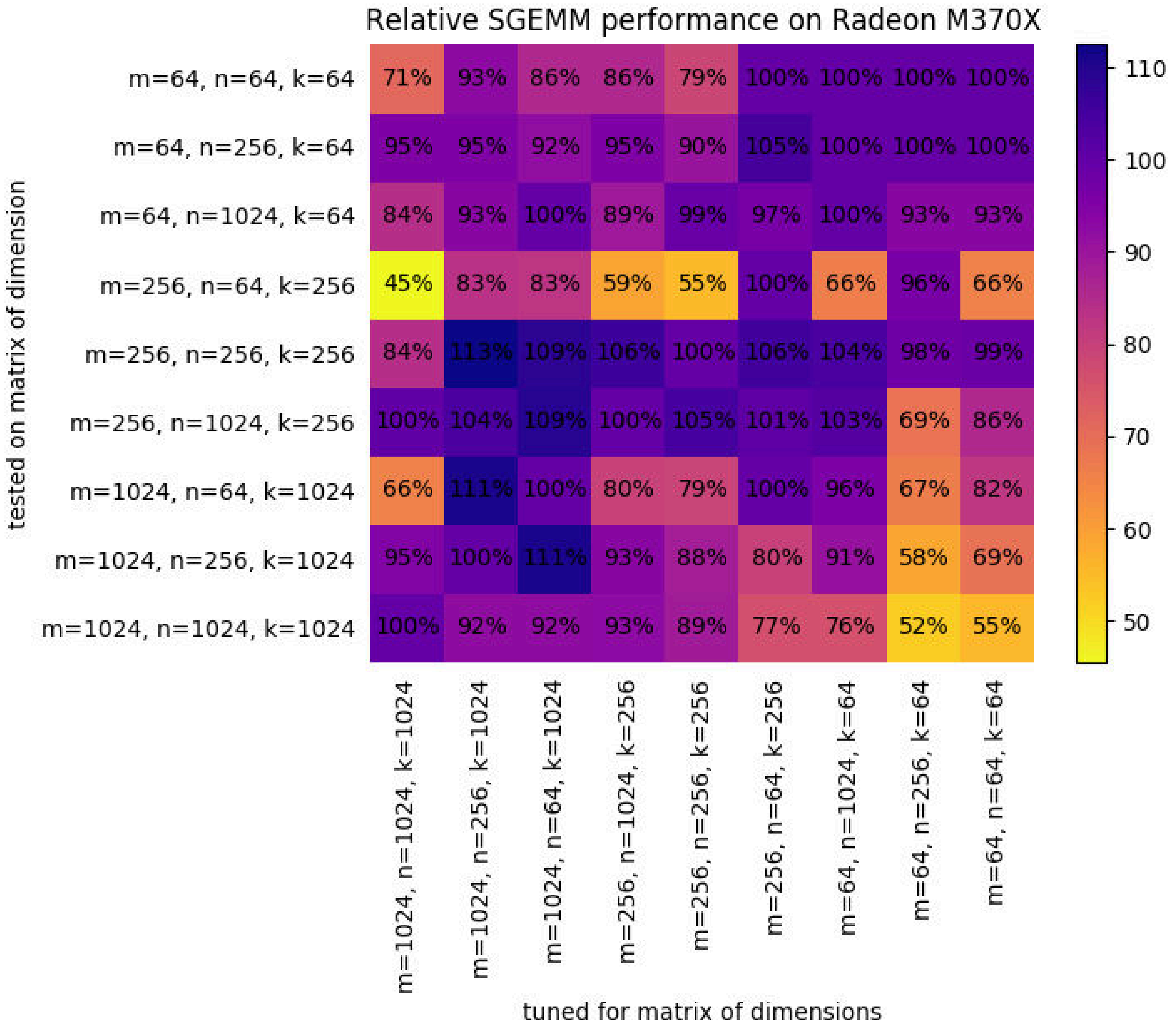
- Default GEMM tuning:
  - 1024x1024 matrices
- Deep-learning:
  - Various but fixed matrix sizes (dependent on network layout)
  - Typically smaller and/or rectangular matrices

Tuning Only for a Single Size?

- Default GEMM tuning:
  - 1024x1024 matrices
- Deep-learning:
  - Various but fixed matrix sizes (dependent on network layout)
  - Typically smaller and/or rectangular matrices
- Potential for optimal performance in CLBlast:
  - Tuning for a custom size possible
  - C++ API to change parameters at run-time

Problem-Specific Tuning

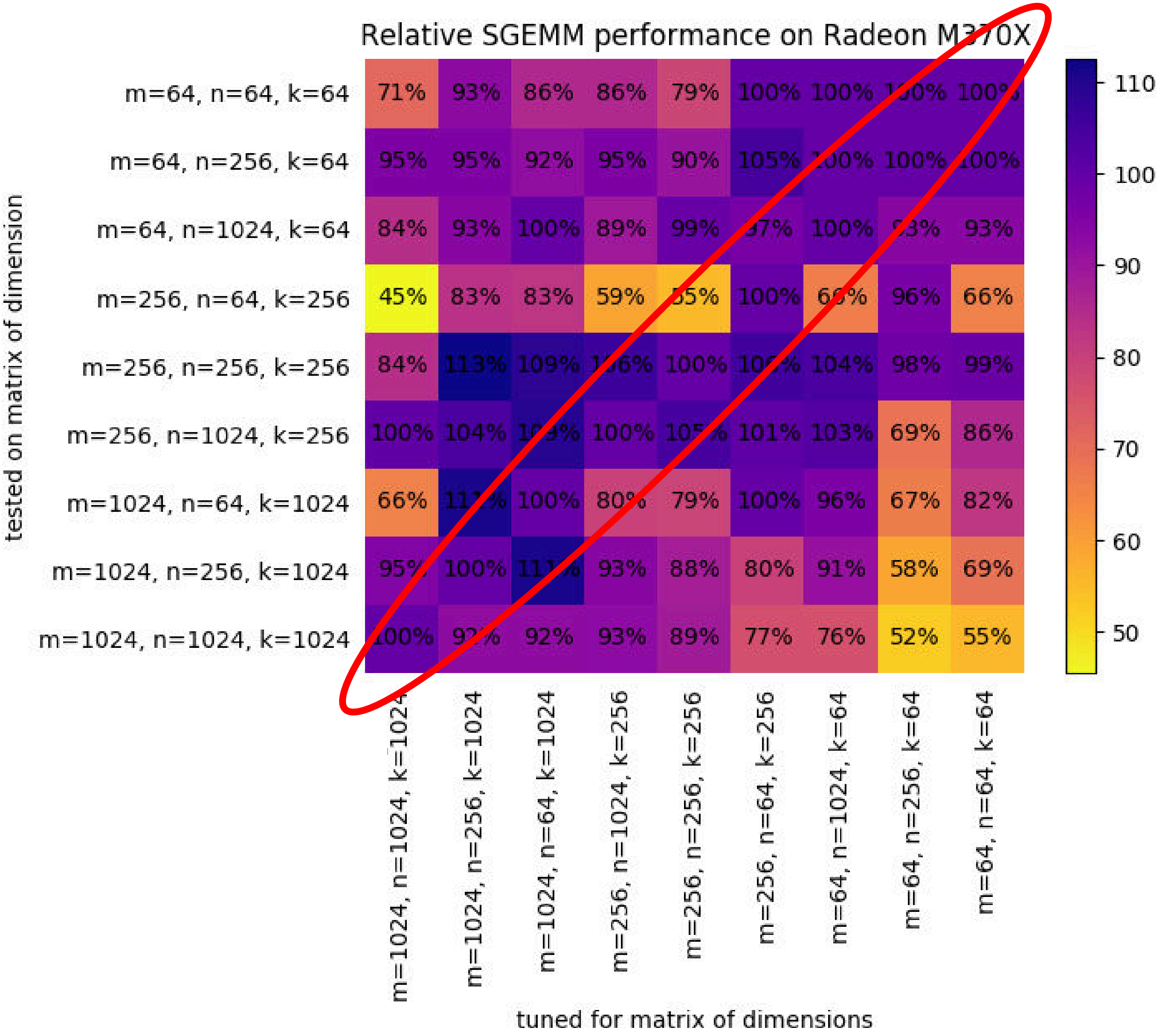
- SGEMM tuning for Radeon M370X GPU



tested on \ tuned for	m=1024, n=1024, k=1024	m=1024, n=256, k=1024	m=1024, n=64, k=1024	m=256, n=1024, k=256	m=256, n=256, k=256	m=256, n=64, k=256	m=64, n=1024, k=64	m=64, n=256, k=64	m=64, n=64, k=64
m=64, n=64, k=64	71%	93%	86%	86%	79%	100%	100%	100%	100%
m=64, n=256, k=64	95%	95%	92%	95%	90%	105%	100%	100%	100%
m=64, n=1024, k=64	84%	93%	100%	89%	99%	97%	100%	93%	93%
m=256, n=64, k=256	45%	83%	83%	59%	55%	100%	66%	96%	66%
m=256, n=256, k=256	84%	113%	109%	106%	100%	106%	104%	98%	99%
m=256, n=1024, k=256	100%	104%	109%	100%	105%	101%	103%	69%	86%
m=1024, n=64, k=1024	66%	111%	100%	80%	79%	100%	96%	67%	82%
m=1024, n=256, k=1024	95%	100%	111%	93%	88%	80%	91%	58%	69%
m=1024, n=1024, k=1024	100%	92%	92%	93%	89%	77%	76%	52%	55%

Problem-Specific Tuning

- SGEMM tuning for Radeon M370X GPU
- Best on the diagonal
- >100% due to random tuning

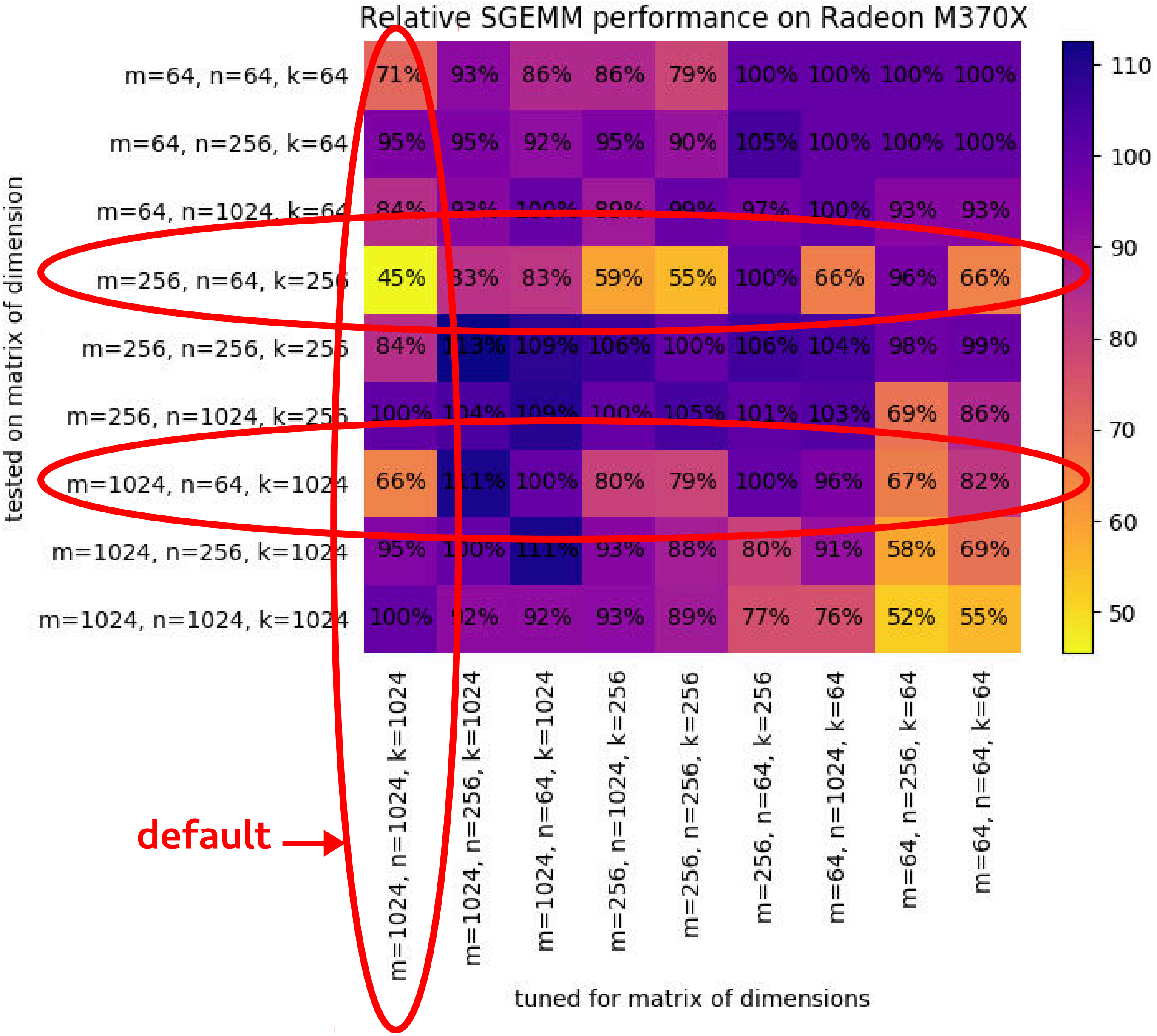


Relative SGEMM performance on Radeon M370X

tested on matrix of dimension \ tuned for matrix of dimensions	m=1024, n=1024, k=1024	m=1024, n=256, k=1024	m=1024, n=64, k=1024	m=256, n=1024, k=256	m=256, n=256, k=256	m=256, n=64, k=256	m=64, n=1024, k=64	m=64, n=256, k=64	m=64, n=64, k=64
m=64, n=64, k=64	71%	93%	86%	86%	79%	100%	100%	100%	100%
m=64, n=256, k=64	95%	95%	92%	95%	90%	105%	100%	100%	100%
m=64, n=1024, k=64	84%	93%	100%	89%	99%	97%	100%	93%	93%
m=256, n=64, k=256	45%	83%	83%	59%	55%	100%	66%	96%	66%
m=256, n=256, k=256	84%	113%	109%	106%	100%	106%	104%	98%	99%
m=256, n=1024, k=256	100%	104%	109%	100%	105%	101%	103%	69%	86%
m=1024, n=64, k=1024	66%	111%	100%	80%	79%	100%	96%	67%	82%
m=1024, n=256, k=1024	95%	100%	111%	93%	88%	80%	91%	58%	69%
m=1024, n=1024, k=1024	100%	92%	92%	93%	89%	77%	76%	52%	55%

Problem-Specific Tuning

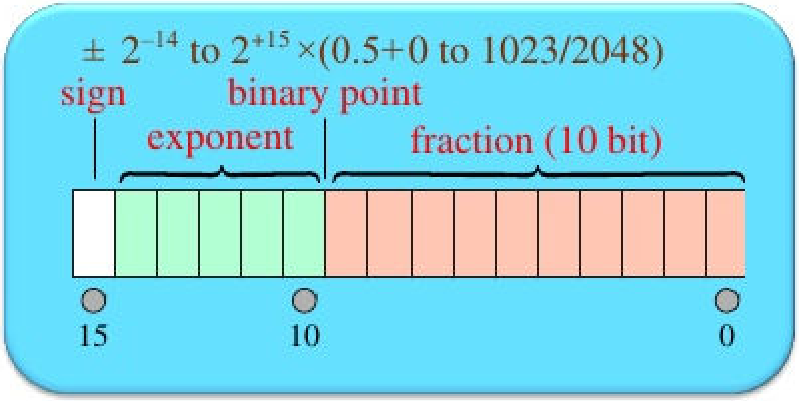
- SGEMM tuning for Radeon M370X GPU
- Best on the diagonal
- >100% due to random tuning
- Gain of ~2x for some cases



tested on \ tuned for	m=1024, n=1024, k=1024	m=1024, n=256, k=1024	m=1024, n=64, k=1024	m=256, n=1024, k=256	m=256, n=256, k=256	m=256, n=64, k=256	m=64, n=1024, k=64	m=64, n=256, k=64	m=64, n=64, k=64
m=64, n=64, k=64	71%	93%	86%	86%	79%	100%	100%	100%	100%
m=64, n=256, k=64	95%	95%	92%	95%	90%	105%	100%	100%	100%
m=64, n=1024, k=64	84%	93%	100%	89%	99%	97%	100%	93%	93%
m=256, n=64, k=256	45%	83%	83%	59%	55%	100%	66%	96%	66%
m=256, n=256, k=256	84%	113%	109%	106%	100%	106%	104%	98%	99%
m=256, n=1024, k=256	100%	104%	109%	100%	105%	101%	103%	69%	86%
m=1024, n=64, k=1024	66%	111%	100%	80%	79%	100%	96%	67%	82%
m=1024, n=256, k=1024	95%	100%	111%	93%	88%	80%	91%	58%	69%
m=1024, n=1024, k=1024	100%	92%	92%	93%	89%	77%	76%	52%	55%

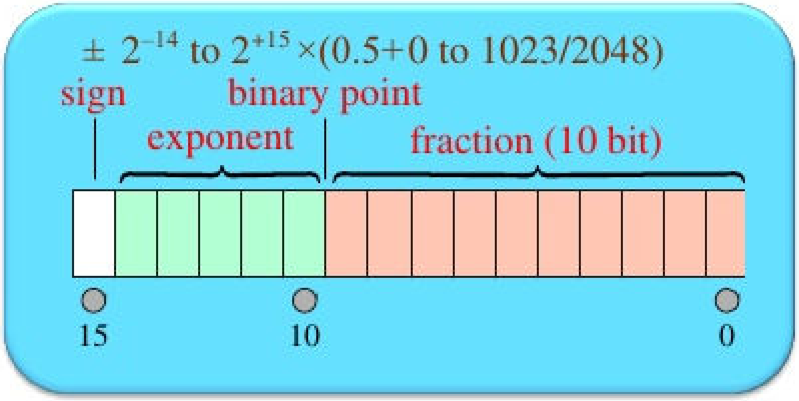
Half-precision floating-point (FP16)

- Double-precision (FP64) not needed for deep-learning
- Even FP32 is too much → introducing half-precision FP16
- Implemented in low-power devices (ARM Mali, Intel GPUs) and deep-learning specific GPUs (Tesla P100, V100)



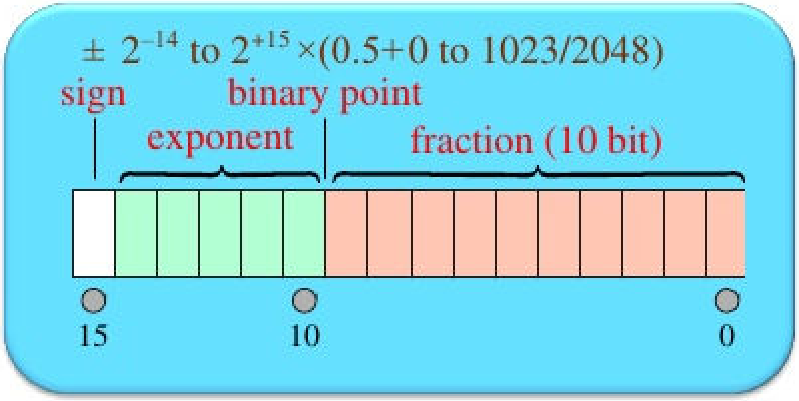
Half-precision floating-point (FP16)

- Double-precision (FP64) not needed for deep-learning
- Even FP32 is too much → introducing half-precision FP16
- Implemented in low-power devices (ARM Mali, Intel GPUs) and deep-learning specific GPUs (Tesla P100, V100)
- Potential for 2x savings in:
  bandwidth, storage, compute, energy

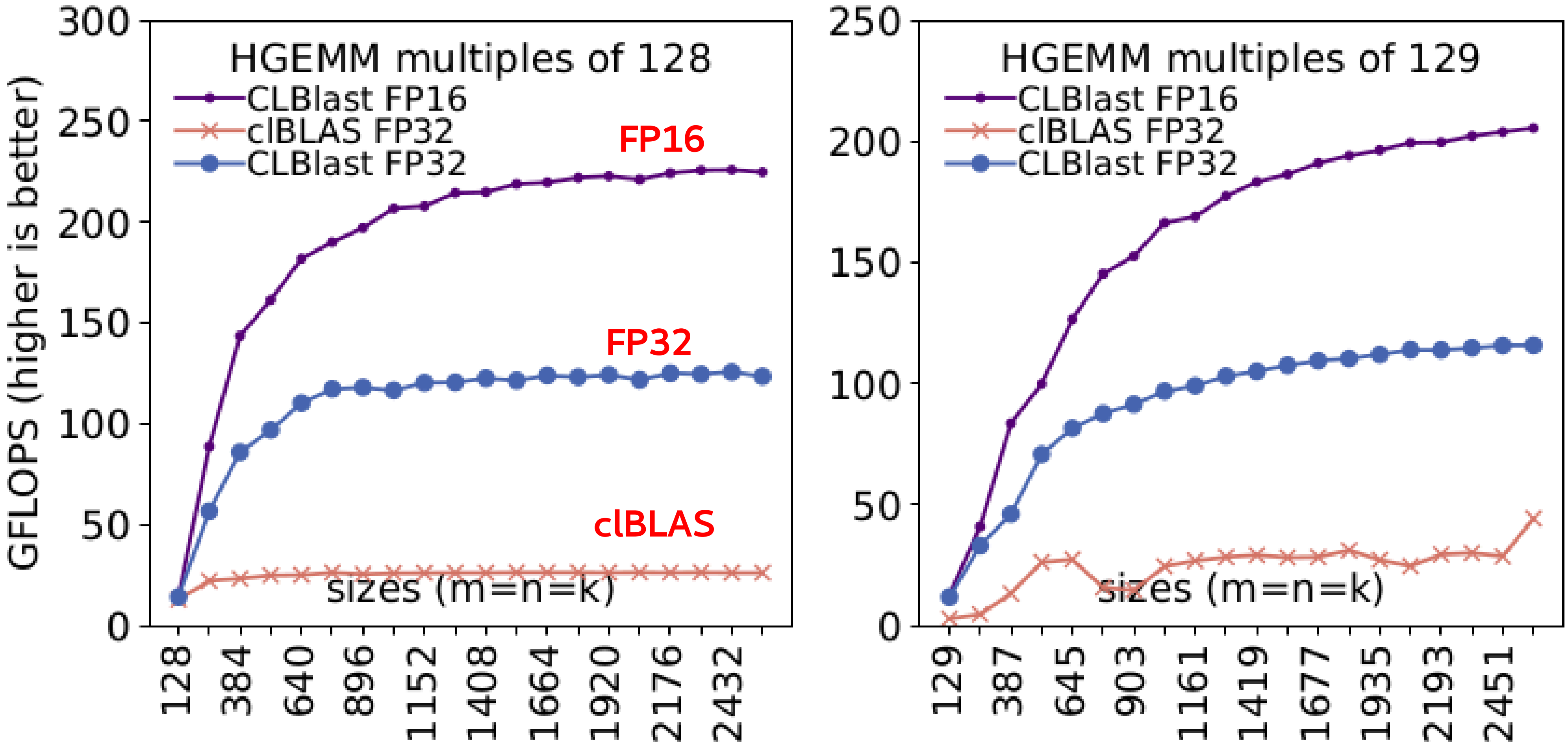


Half-precision floating-point (FP16)

- Double-precision (FP64) not needed for deep-learning
- Even FP32 is too much → introducing half-precision FP16
- Implemented in low-power devices (ARM Mali, Intel GPUs) and deep-learning specific GPUs (Tesla P100, V100)
- Potential for 2x savings in: bandwidth, storage, compute, energy
- Current FP16 support for GPUs:
  - cuBLAS: HGEMM only
  - clBLAS: no FP16 at all
  - CLBlast: all routines!

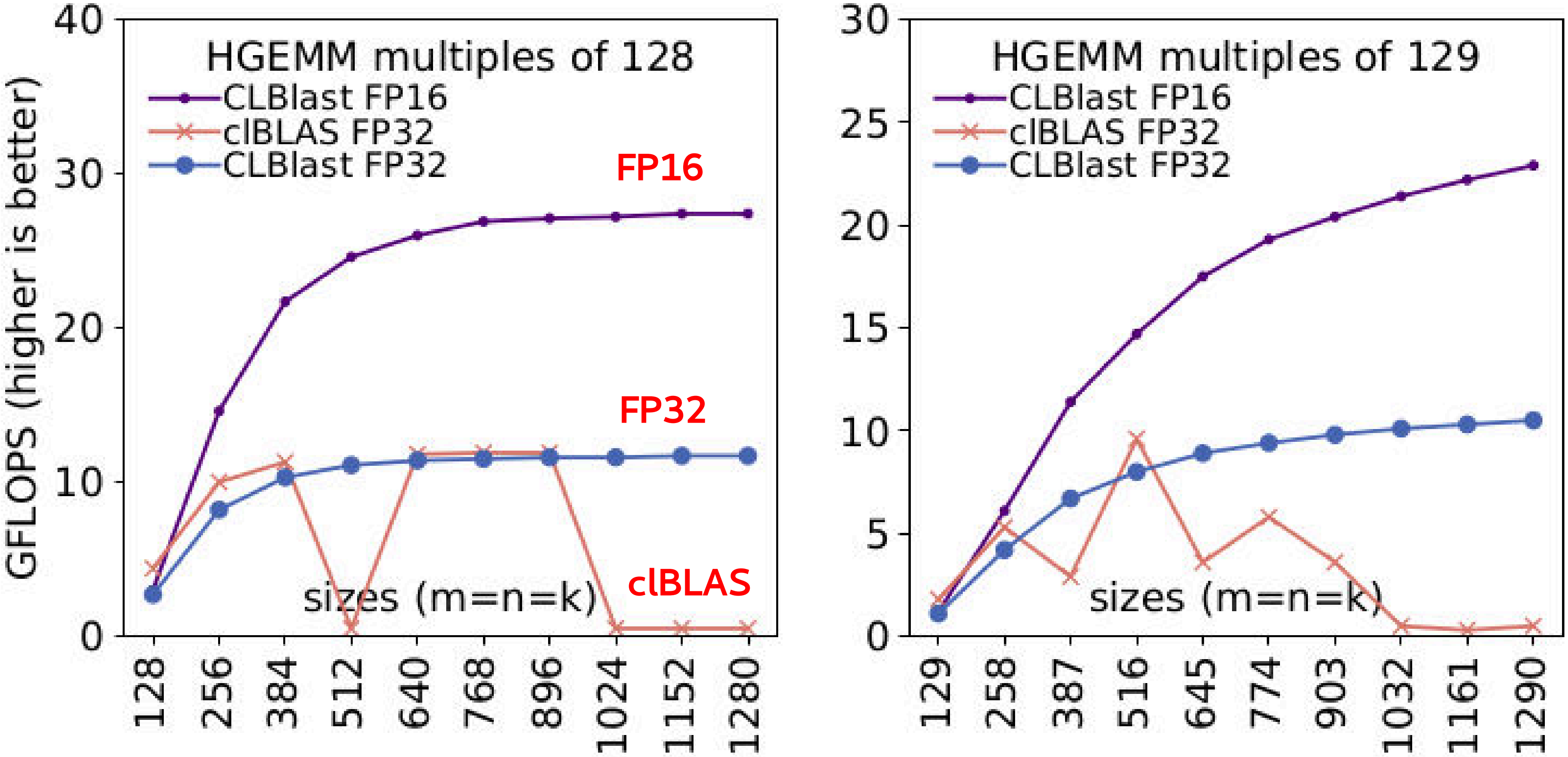


Half-precision FP16 on Intel Skylake GPU



- FP16 ~1.9x faster across the board!

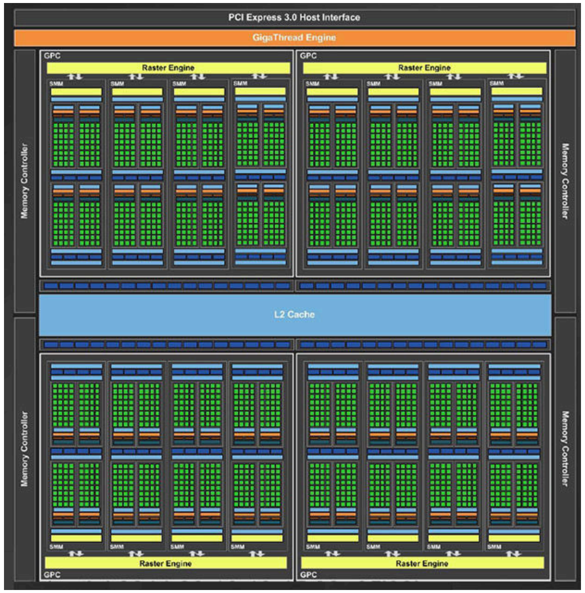
Half-precision FP16 on ARM Mali T628



- FP16 2+ times faster across the board!

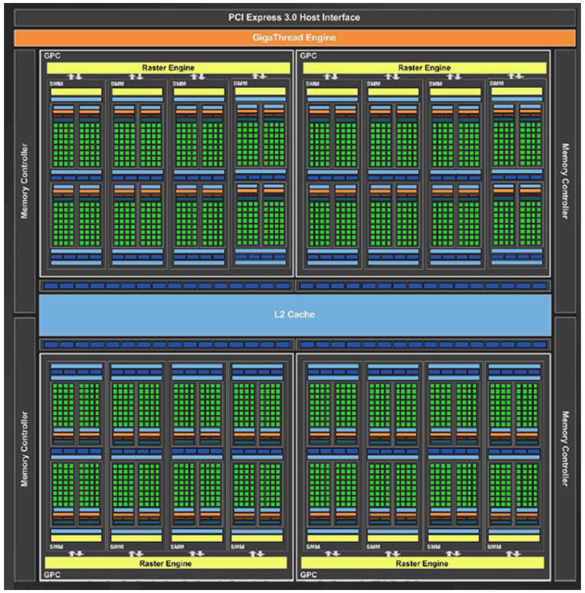
Batching BLAS routines

- Small-sized GEMM is super slow
  - Not enough work-groups
  - Not enough threads



Batching BLAS routines

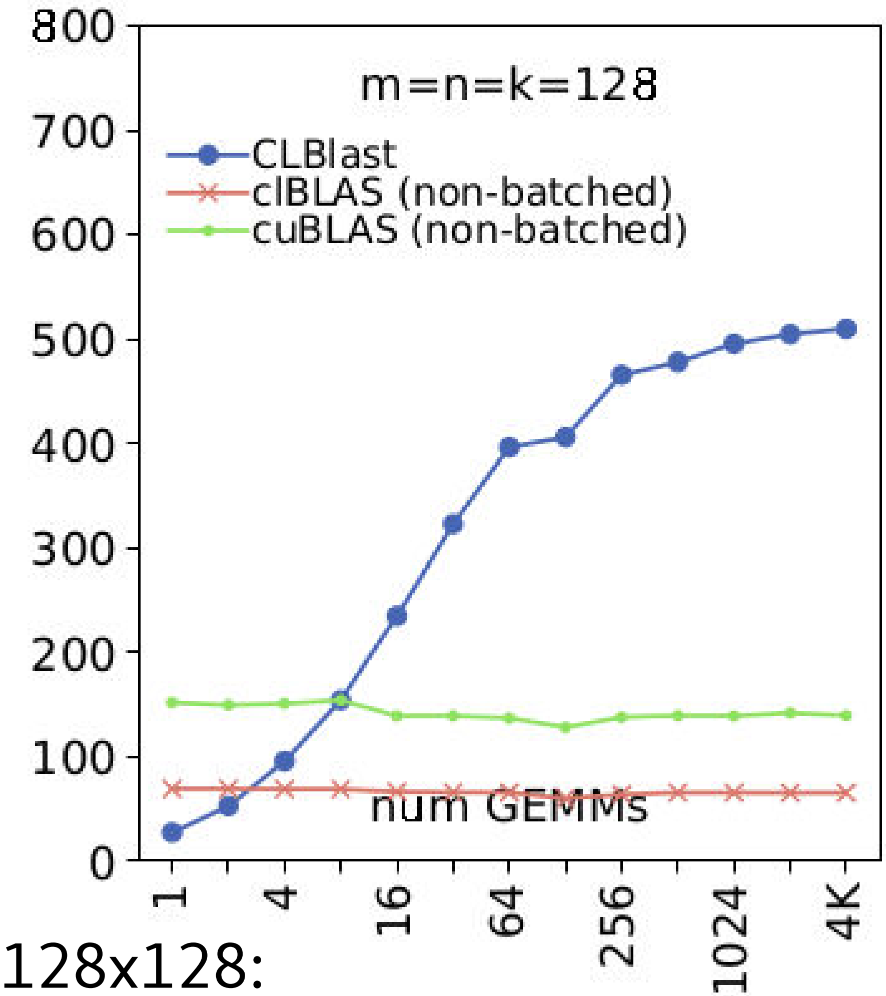
- Small-sized GEMM is super slow
  - Not enough work-groups
  - Not enough threads



- Let’s make it fast again:

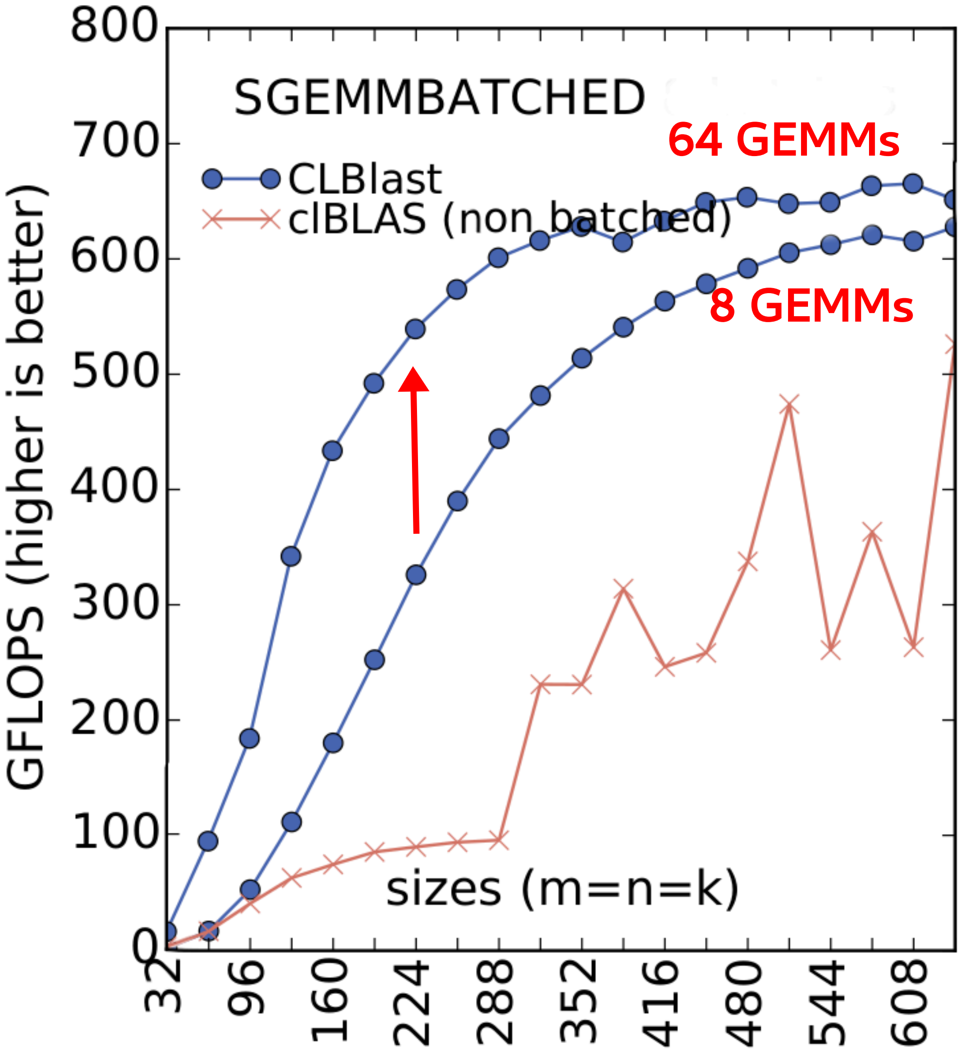
  - Combine multiple small GEMM operations into a single kernel
  - Use offsets to indicate where the next matrices start

Batched GEMM on GeForce GTX 750Ti



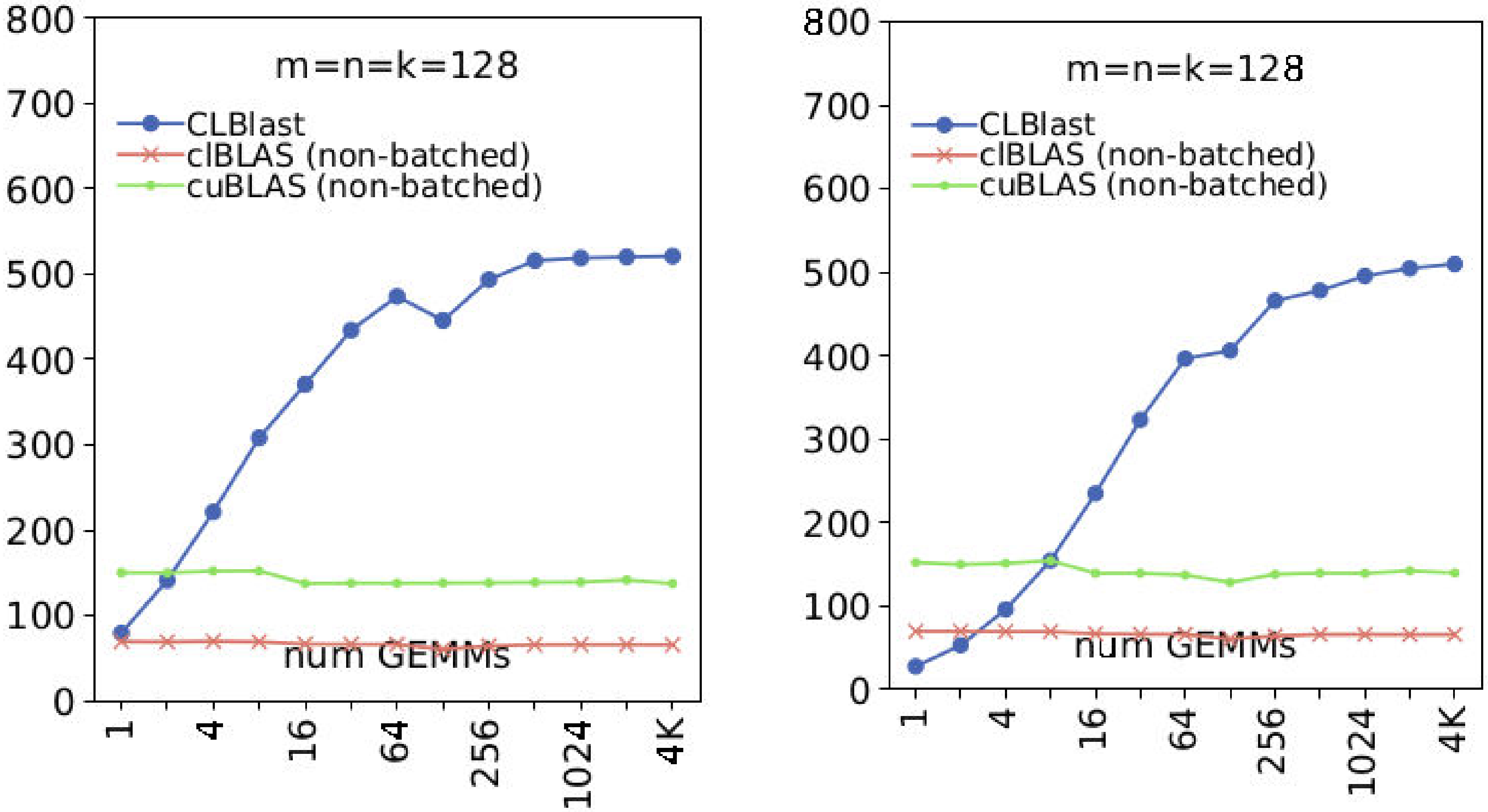
- SGEMM 128x128x128:
  - Regular: ~60 GFLOPS
  - Batched: ~20 GFLOPS (1 GEMM) up to ~500 GFLOPS (4K)!

Batched GEMM on GeForce GTX 750Ti



- Significant benefits for larger sizes as well
  - mostly beneficial in the range n=64 till 512

Batched strided GEMM on GeForce GTX 750Ti



- Strided (left) versus regular (right):
  - More assumptions, smaller overhead
  - Better performance for smaller batches

What’s next?

- More features for deep learning (convolution as GEMM, ...)
- Auto-tuning using learned models:
  - Similar to the ‘ISAAC’ library
  - Cooperation with:
    - Rafael Ballester-Ripoll (University of Zürich)
      - Based on tensor trains
    - Flavio Vella (dividiti)
      - Matrix-size aware tuning
- Suggestions?

Conclusion

- CLBlast: a modern C++11 OpenCL BLAS library
- Performance portable thanks to generic kernels and auto-tuning
- Has important features to accelerate deep-learning:
  - Problem-size specific tuning:
    - Up to 2x in an example experiment
  - Half-precision FP16 support:
    - Up to 2x benefit in speed and memory savings
  - Batched GEMM routines:
    - Order of magnitude benefit depending on the use-case

CLBlast

- Time to checkout the album.... Or clone the repository!

CLBlast: A Tuned BLAS Library

Cedric Nugteren

May 16, 2018

http://github.com/cnugteren/clblast
http://cnugteren.github.io/clblast