CATEGORY: **DEVELOPER - PERFORMANCE OPTIMIZATION - D004**

POSTER **P5274** | CONTACT NAME **Cedric Nugteren: cedric.nugteren@surfsara.nl**

GPU TECHNOLOGY CONFERENCE

Cedric Nugteren

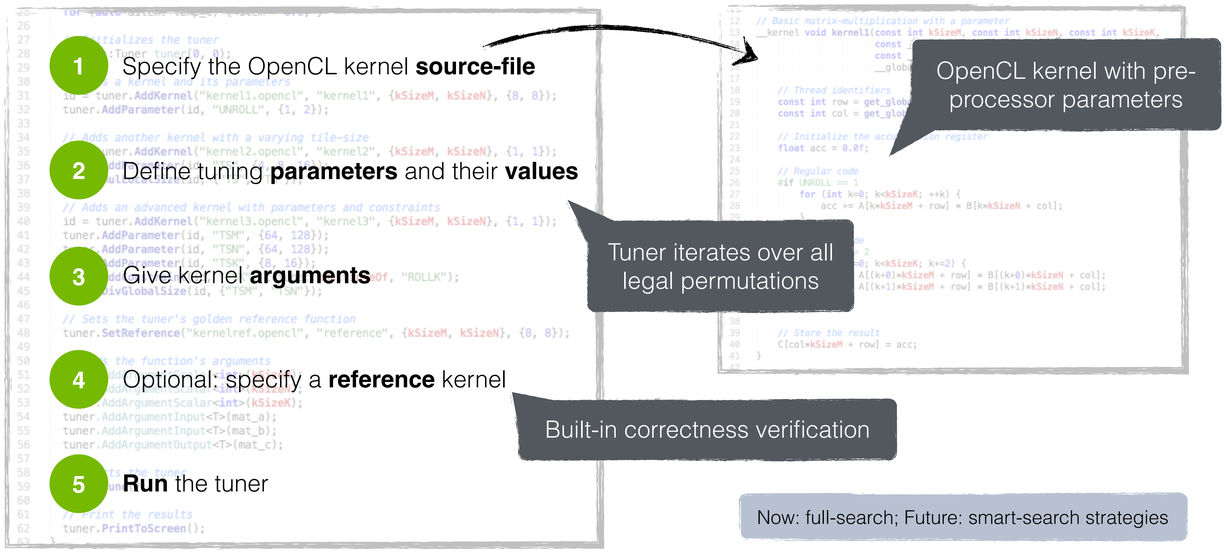
cedric.nugteren@surfsara.nl
SURFsara Supercomputer Centre
Amsterdam, The Netherlands

Auto-Tuning OpenCL Matrix-Multiplication: K40 versus K80

GPU access made possible by:

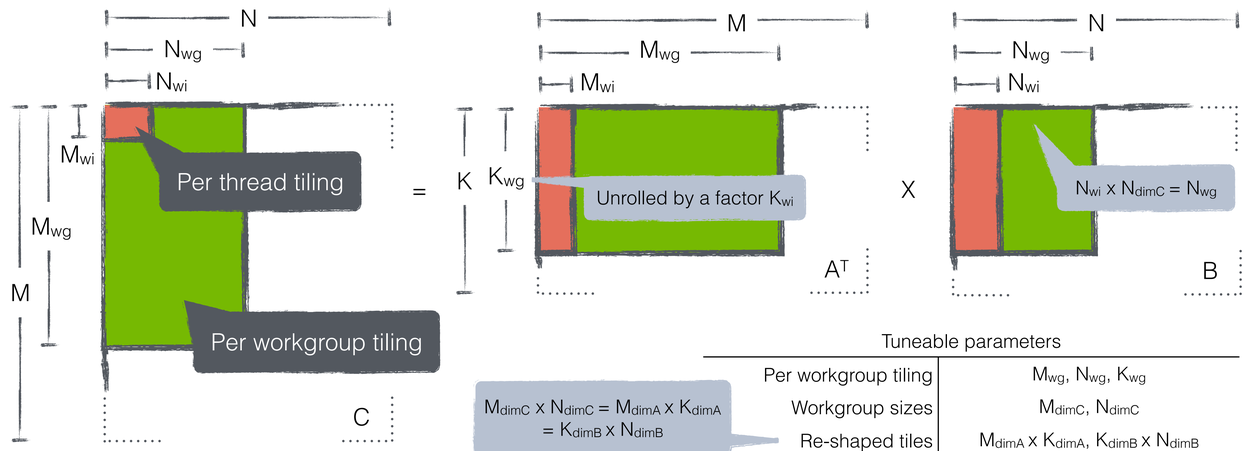
NVIDIA and DAS-4

Generic OpenCL Auto-Tuning Framework



Tuner available on GitHub (open-source): https://github.com/cnugteren/cltune

OpenCL Matrix-Multiplication (SGEMM/DGEMM)



$M_{dimC} \times N_{dimC} = M_{dimA} \times K_{dimA} = K_{dimB} \times N_{dimB}$

Tuneable parameters

Per workgroup tiling	M_{wg}, N_{wg}, K_{wg}
Workgroup sizes	M_{dimC}, N_{dimC}
Re-shaped tiles	$M_{dimA} \times K_{dimA}$, $K_{dimB} \times N_{dimB}$
Tile in local memory	$\$_{localA}$, $\$_{localB}$
Stride within a thread	M_{stride} (A&C), N_{stride} (B)
Vector widths	M_{vec} (A&C), N_{vec} (B)
Unroll factor K_{wg} loop	K_{wi}

SGEMM/DGEMM code and parameters based on:

[1] Performance Tuning of Matrix Multiplication in OpenCL on Different GPUs and CPUs. K. Matsumoto, N. Nakasato, S.G. Sedukhin. In: *SC-Companion '12*. IEEE.

[2] OpenCL SGEMM tuning tutorial. C. Nugteren. http://www.cedricnugteren.nl/tutorial.php

Best-Case Matrix-Multiplication Parameters

Tested ~100K permutations!

Dual GPU board

Same chip as K40m but twice the number of registers and L1 cache

		NVIDIA Tesla K40m		½ NVIDIA Tesla K80		AMD Radeon HD7970	
		SGEMM	DGEMM	SGEMM	DGEMM	SGEMM	DGEMM
Tile sizes	M_{wg}, N_{wg}, K_{wg}	128, 128, 16	64, 64, 8	128, 128, 8	64, 128, 8	128, 128, 16	128, 64, 8
Threads per workgroup	M_{dimC}, N_{dimC}	16, 16	8, 16	8, 16	8, 16	16, 16	16, 16
	M_{dimA}, N_{dimB}	16, 16	32, 16	32, 16	16, 16	16, 16	32, 32
	$\$_{localA}$, $\$_{localB}$	yes, yes	yes, yes	yes, yes	yes, yes	yes, yes	no, no
	M_{stride}, N_{stride}	yes, yes	yes, yes	yes, yes	yes, yes	yes, yes	yes, no
Vector widths	M_{vec}, N_{vec}	2, 2	1, 1	4, 4	2, 2	2, 8	2, 1
Unroll factor	K_{wi}	8	8	8	8	16	4

Best vector width **x2** for K80

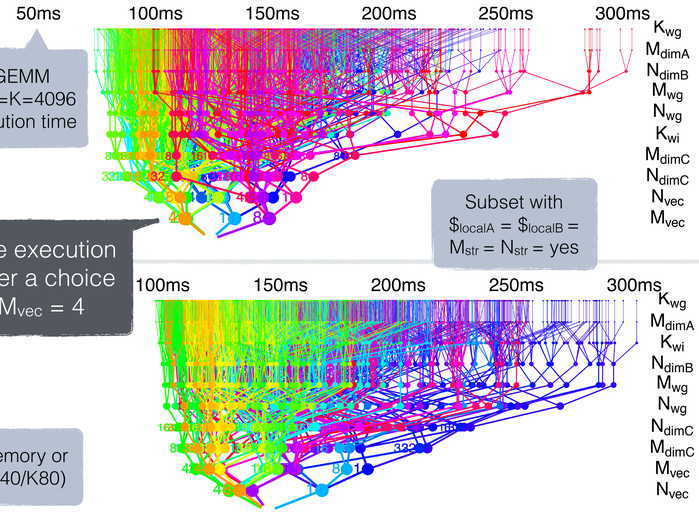
Best vector width **x½** for doubles

Only slightly slower without local memory or strides (definitely not the case for K40/K80)

Visualization of Tested Parameter Permutations

SGEMM M=N=K=4096 execution time

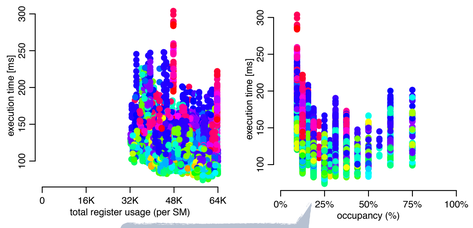
Average execution time after a choice e.g. M_{vec} = 4



corresponding colours per GPU

Subset with $\$_{localA} = \$_{localB} = M_{str} = N_{str}$ = yes

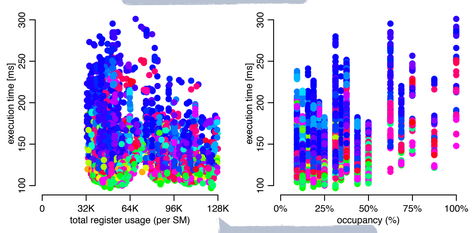
top: **Tesla K40m**



Best performance at low occupancy

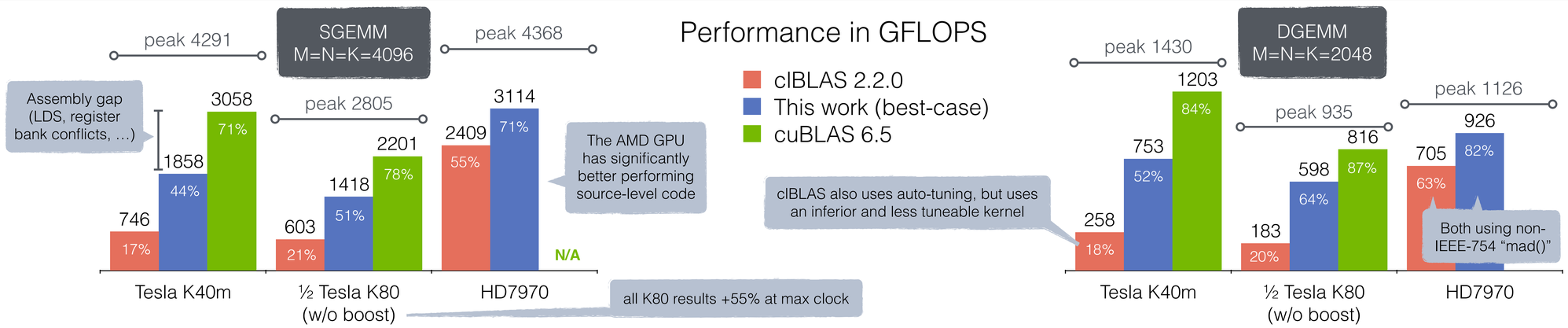
bottom: **½ Tesla K80** (w/o boost)

Parameters with highest impact at the bottom



More registers

Performance in GFLOPS



- clBLAS 2.2.0
- This work (best-case)
- cuBLAS 6.5

clBLAS also uses auto-tuning, but uses an inferior and less tuneable kernel

Conclusions

1. Generic **OpenCL auto-tuning** framework ensures **portability**
2. **Fastest** source-level **matrix-multiplication** implementation
3. Tree **visualization** shows architectural and algorithm properties
4. **K80**'s extra registers and cache yield **higher efficiency** (vs. K40)