# Tuning of BLAS level 1 and 2

Hans Henrik Brandenborg Sørensen

Section for Scientific Computing

DTU Informatics

# BLAS (Basic Linear Algebra Subprograms)

- Basic routines for numerical applications.
- Legacy software package 1979-2002 (Netlib.org).
  - C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, *Basic Linear Algebra Subprograms for FORTRAN usage*, 1979
  - J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN Basic Linear Algebra Subprograms*, 1988
  - J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, *A set of Level 3 Basic Linear Algebra Subprograms*, 1990
  - L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, *An Updated Set of Basic Linear Algebra Subprograms (BLAS)*, 2002

# BLAS (Basic Linear Algebra Subprograms)

- Level 1 BLAS: (xAXPY, xAMAX, xDOT, xNRM2, etc.)
  - Vector of length N (4xN bytes)
  - 4xN bytes : O(N) flops
  - Memory bound

# BLAS (Basic Linear Algebra Subprograms)

- Level 1 BLAS: (xAXPY, xAMAX, xDOT, xNRM2, etc.)
  - Vector of length N (4xN bytes)
  - 4xN bytes : O(N) flops
  - Memory bound

- Level 2 BLAS: (xGEMV, xSYMV, xHEMV, xTRSV, etc.)
  - Matrix of size NxN (4xNxN bytes) + Vector (4xN bytes)
  - 4xN^2 + 4xN bytes : O(N^2) flops
  - Memory bound

# BLAS (Basic Linear Algebra Subprograms)

- Level 1 BLAS: (xAXPY, xAMAX, xDOT, xNRM2, etc.)
  - Vector of length N (4xN bytes)
  - 4xN bytes : O(N) flops
  - Memory bound
- Level 2 BLAS: (xGEMV, xSYMV, xHEMV, xTRSV, etc.)
  - Matrix of size NxN (4xNxN bytes) + Vector (4xN bytes)
  - 4xN^2 + 4xN bytes : O(N^2) flops
  - Memory bound
- Level 3 BLAS: (xGEMM, xSYMM, xHEMM, xTRSM, etc.)
  - 1 or 2 matrices of size NxN (4xNxN bytes)
  - (2x)4xN^2 bytes : O(N^3) flops
  - Compute bound for large N

# BLAS (Basic Linear Algebra Subprograms)

- Level 1 BLAS: (xAXPY, xAMAX, xDOT, xNRM2, etc.)
  - Vector of length N (4xN bytes)
  - 4xN bytes : O(N) flops
  - Memory bound

- Level 2 BLAS: (xGEMV, xSYMV, xHEMV, xTRSV, etc.)
  - Matrix of size NxN (4xNxN bytes) + Vector (4xN bytes)
  - 4xN^2 + 4xN bytes : O(N^2) flops
  - Memory bound

- Level 3 BLAS: (xGEMM, xSYMM, xHEMM, xTRSM, etc.)
  - 1 or 2 m
  - (2x)4xN
  - Compute bound for large N

Whenever possible, use Level 3 BLAS in your (GPU) applications!

# BLAS (Basic Linear Algebra Subprograms)

- Level 1 BLAS: (xAXPY, xAMAX, xDOT, xNRM2, etc.)
  - Vecto
  - 4xN b
  - Memo
- Level 2 B etc.)
  - Matrix xN bytes)
  - 4xN^
  - Memory bound
- Level 3 BLAS: (xGEMM, xSYMM, xHEMM, xTRSM, etc.)
  - 1 or 2 matrices of size NxN (4xNxN bytes)
  - (2x)4xN^2 bytes : O(N^3) flops
  - Compute bound for large N

Why consider Level 1 and Level 2:

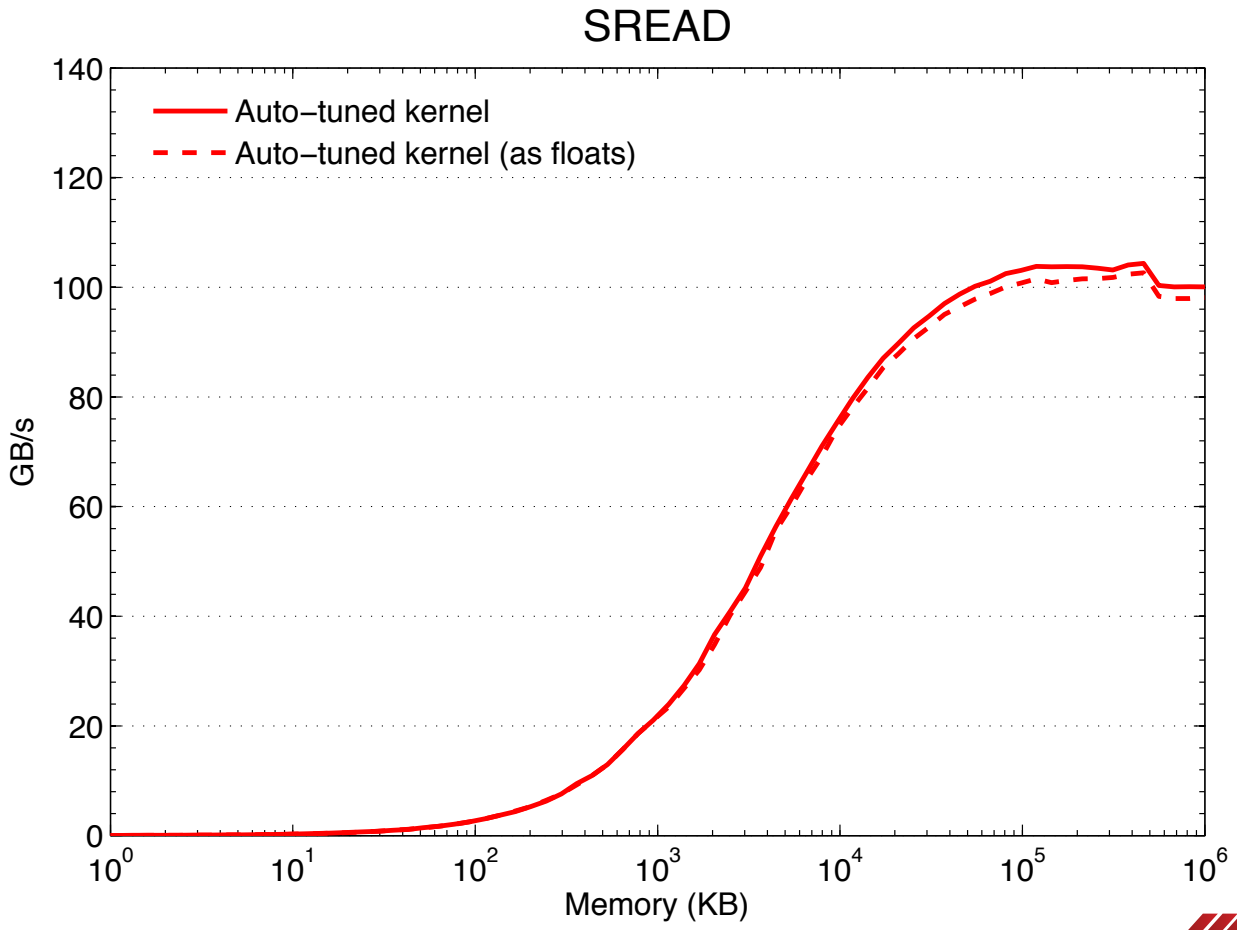
- Matvecs, orthogonalizations, norms, triangular solves, LAPACK building blocks, e.g., factorizations.

- Still very little attention from GPU community.

# Performance Prediction

C2050: Theoretical bandwidth 144 GB/s

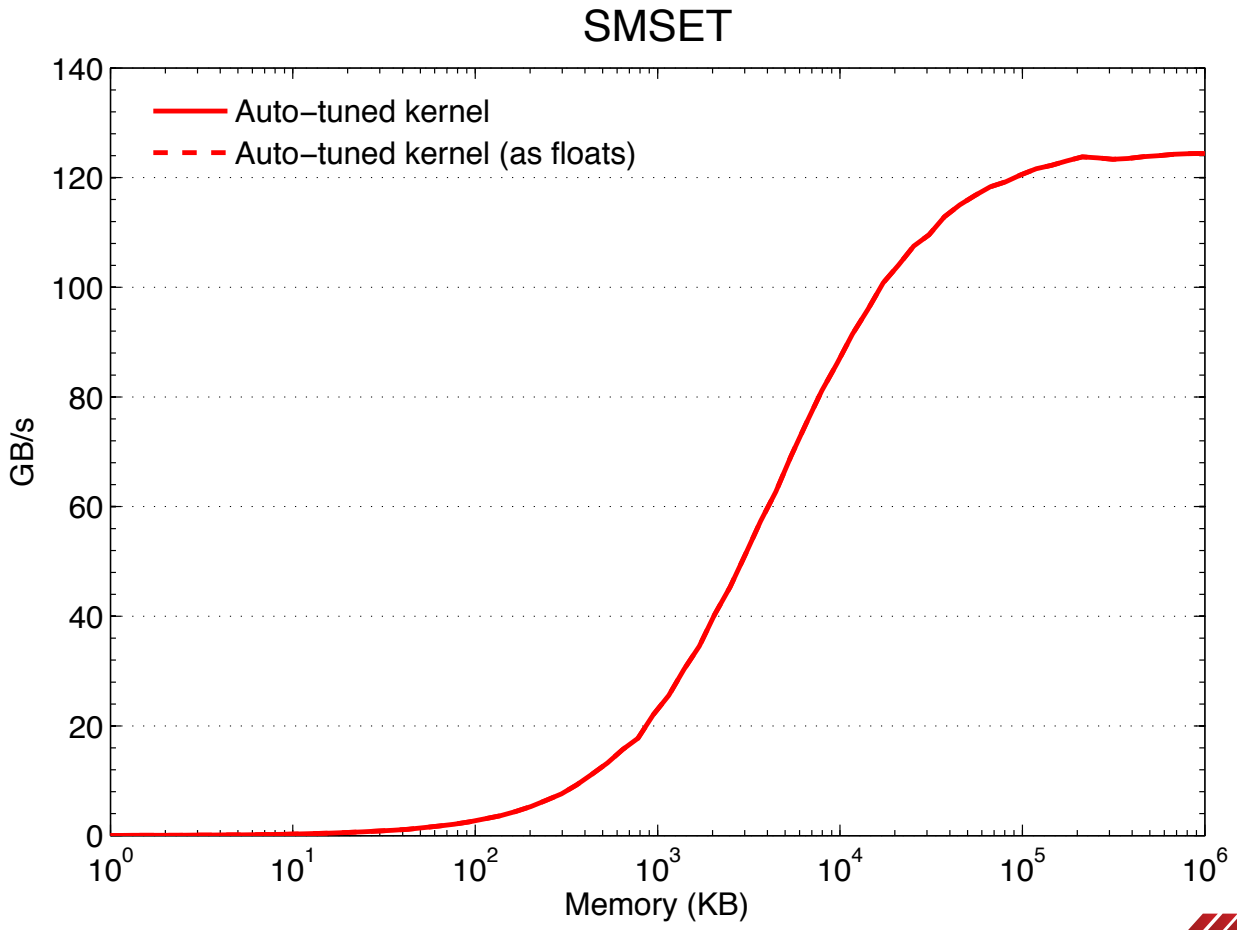
What is the effective bandwidth?



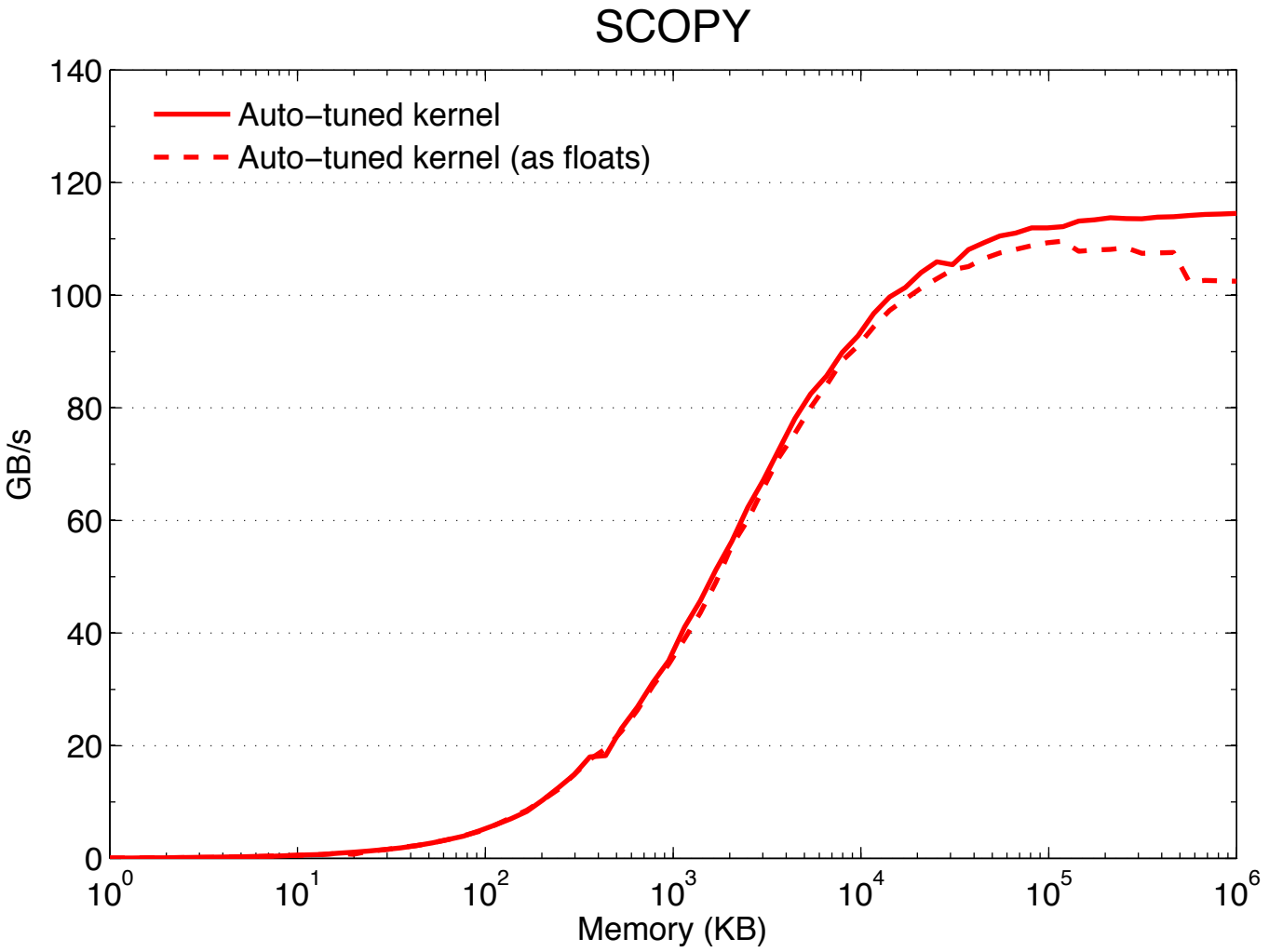
# Performance Prediction

C2050: Theoretical bandwidth 144 GB/s

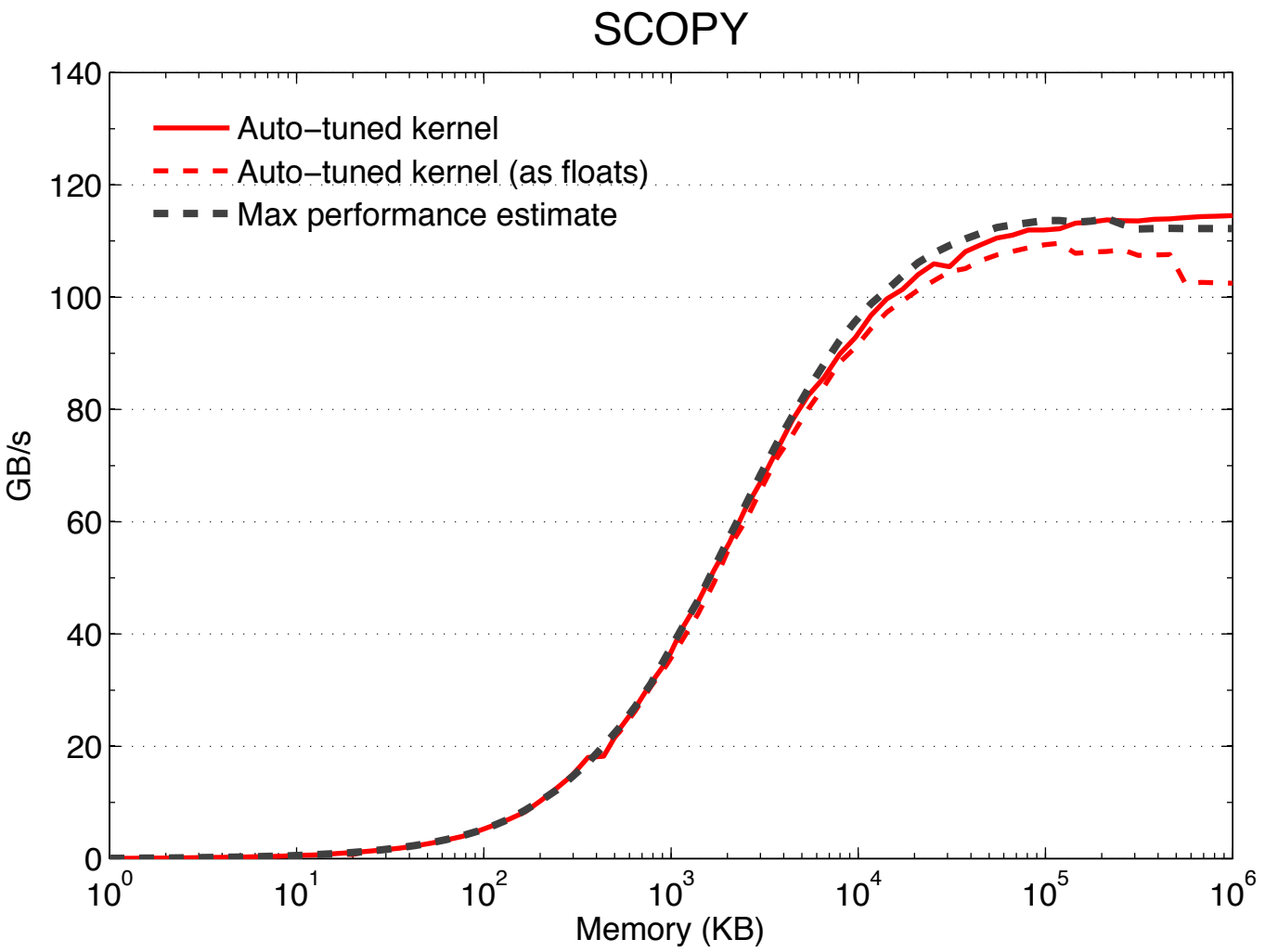
What is the effective bandwidth?



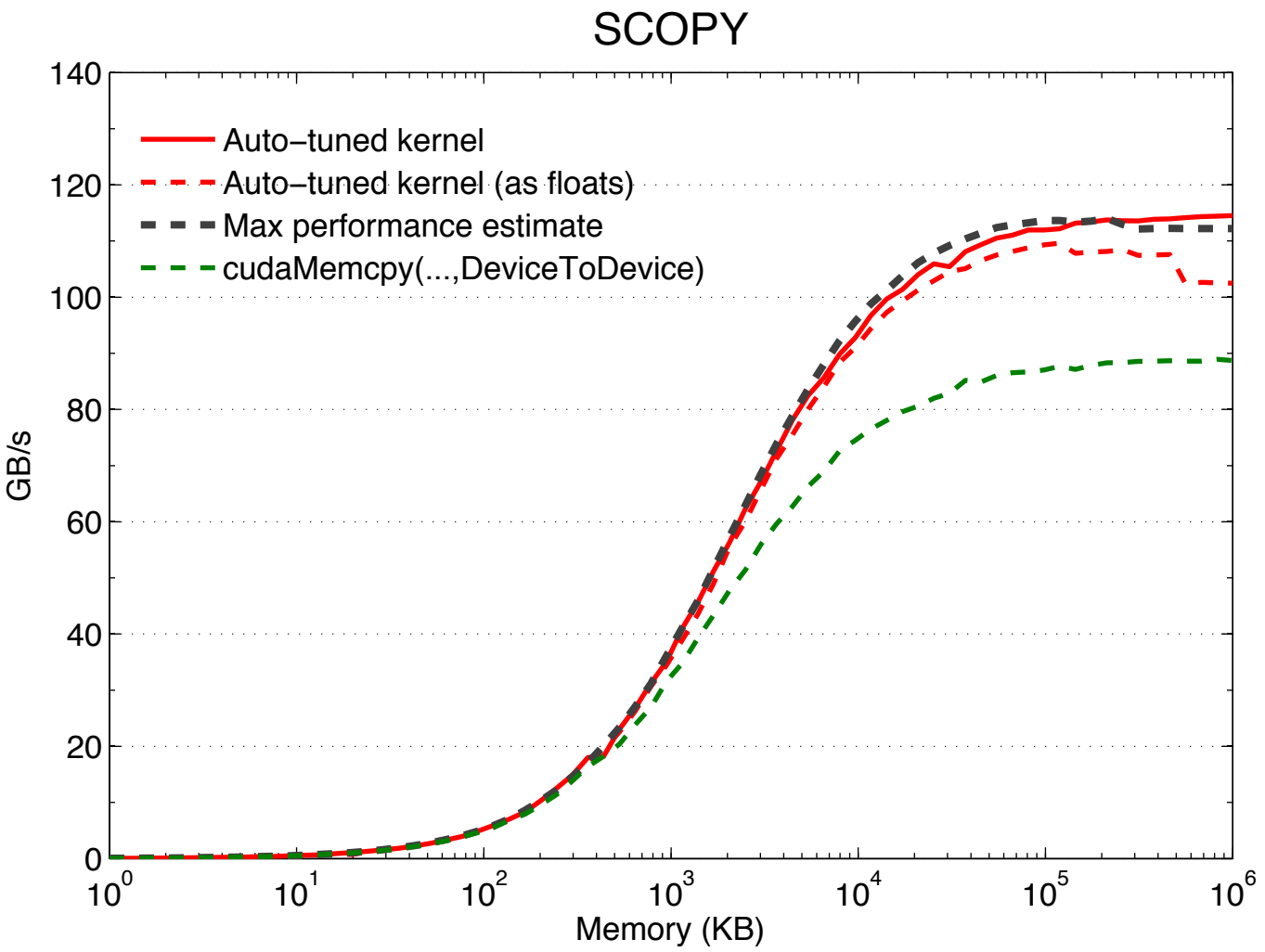
# SCOPY


SCOPY
Auto−tuned kernel
Auto−tuned kernel (as floats)
GB/s
Memory (KB)
0
20
40
60
80
100
120
140
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

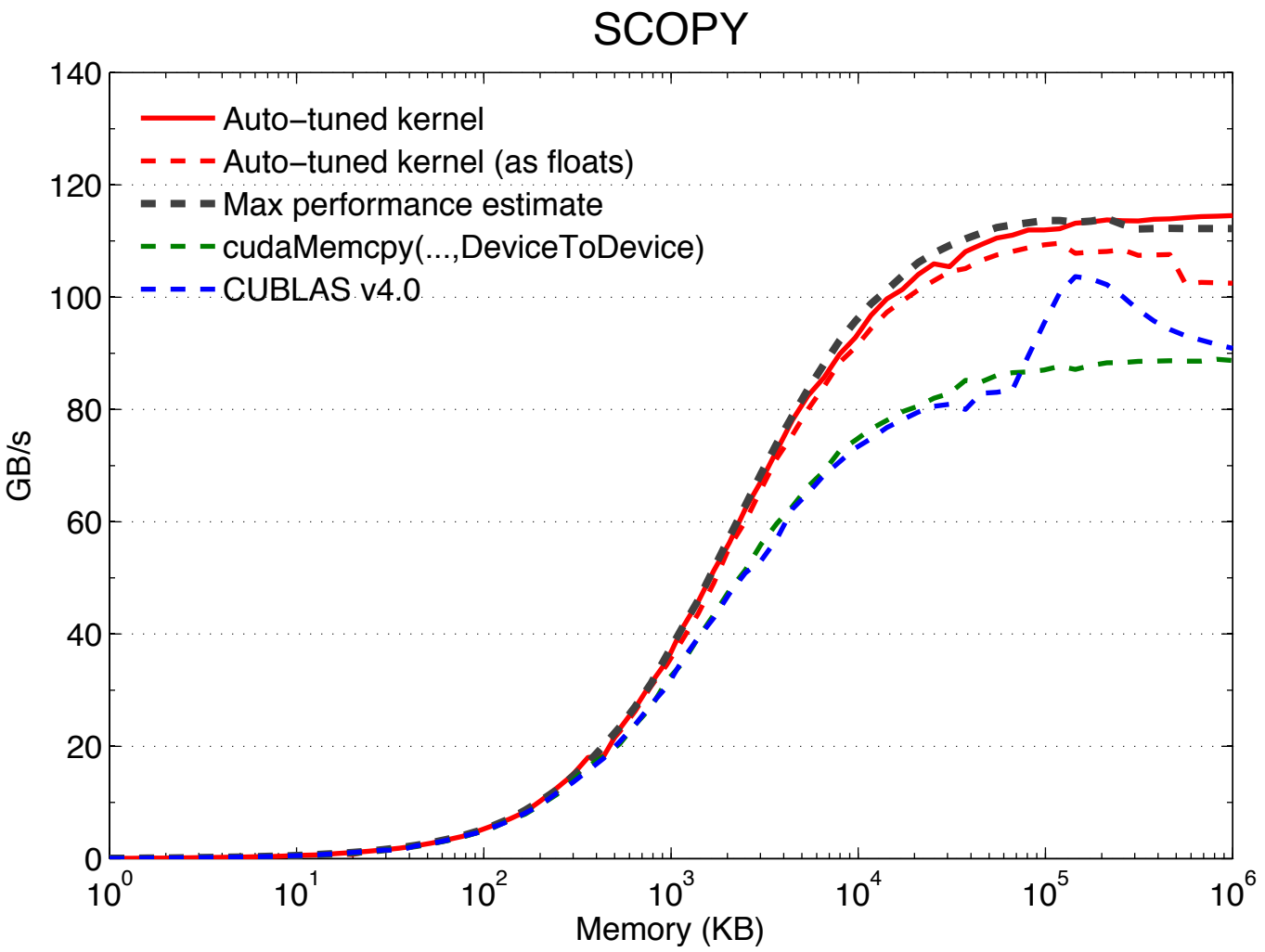
# SCOPY


SCOPY
Auto−tuned kernel
Auto−tuned kernel (as floats)
Max performance estimate
GB/s
Memory (KB)
0
20
40
60
80
100
120
140
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

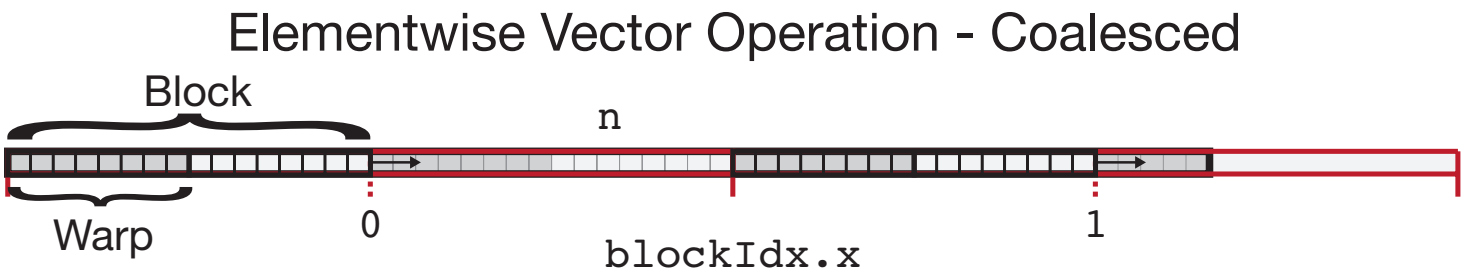
# SCOPY

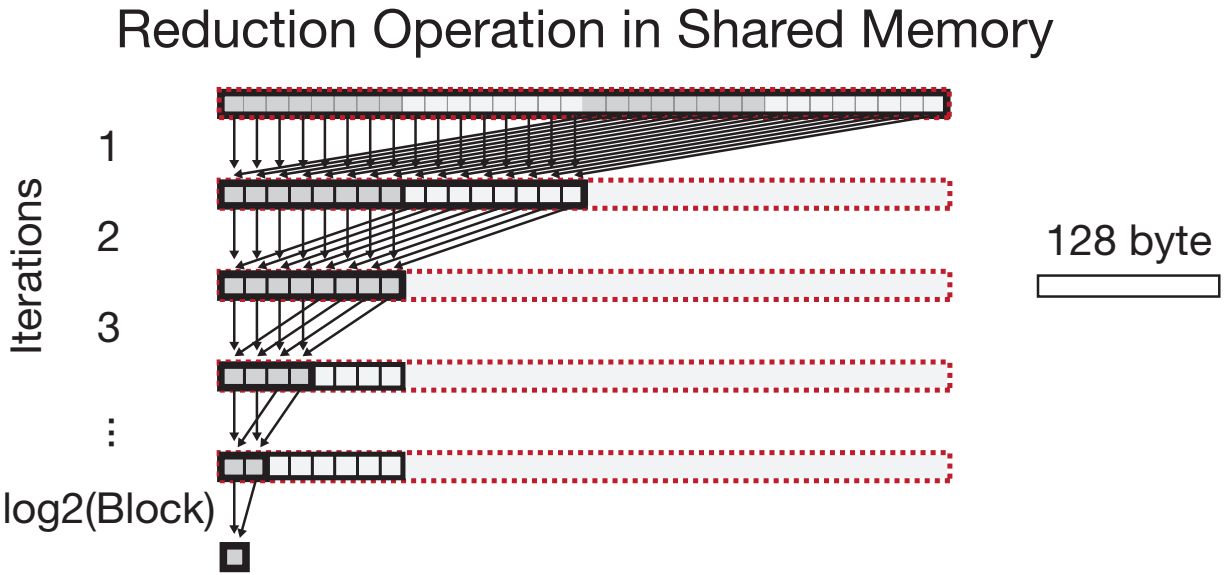

SCOPY
Auto–tuned kernel
Auto–tuned kernel (as floats)
Max performance estimate
cudaMemcpy(...,DeviceToDevice)
GB/s
Memory (KB)
0
20
40
60
80
100
120
140
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

# SCOPY


SCOPY
Auto−tuned kernel
Auto−tuned kernel (as floats)
Max performance estimate
cudaMemcpy(...,DeviceToDevice)
CUBLAS v4.0
GB/s
Memory (KB)
0
20
40
60
80
100
120
140
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

# Level 1 BLAS: Vector Operations




TUNING PARAMETERS: Block Size, Work Size per Thread, Unroll level

# Auto-tuning for High Performance

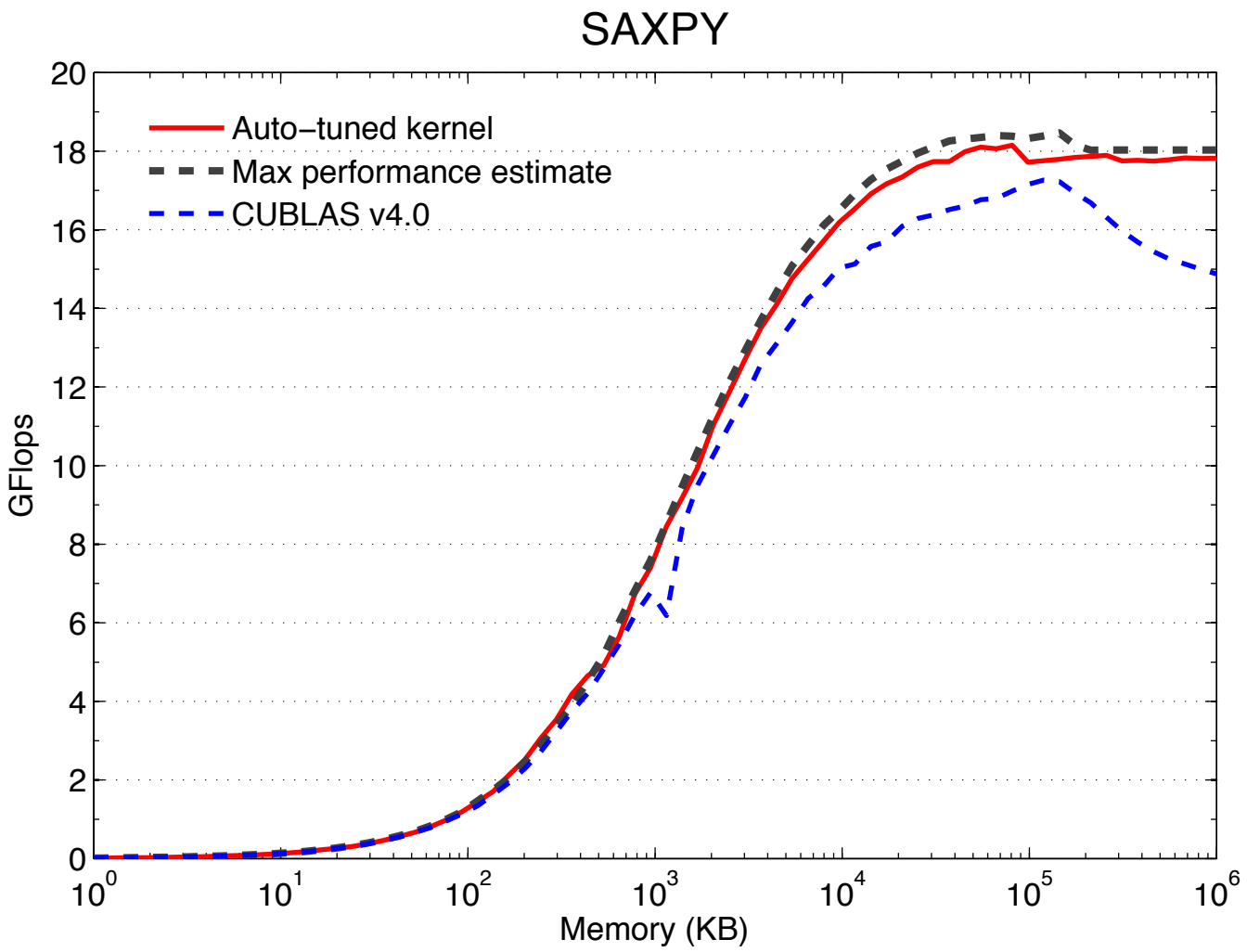
- Heuristic search of parameter space:

$$\texttt{BLOCKSIZE} \in \{32, 64, 96, 128, 160, 192, 224, 256\}$$

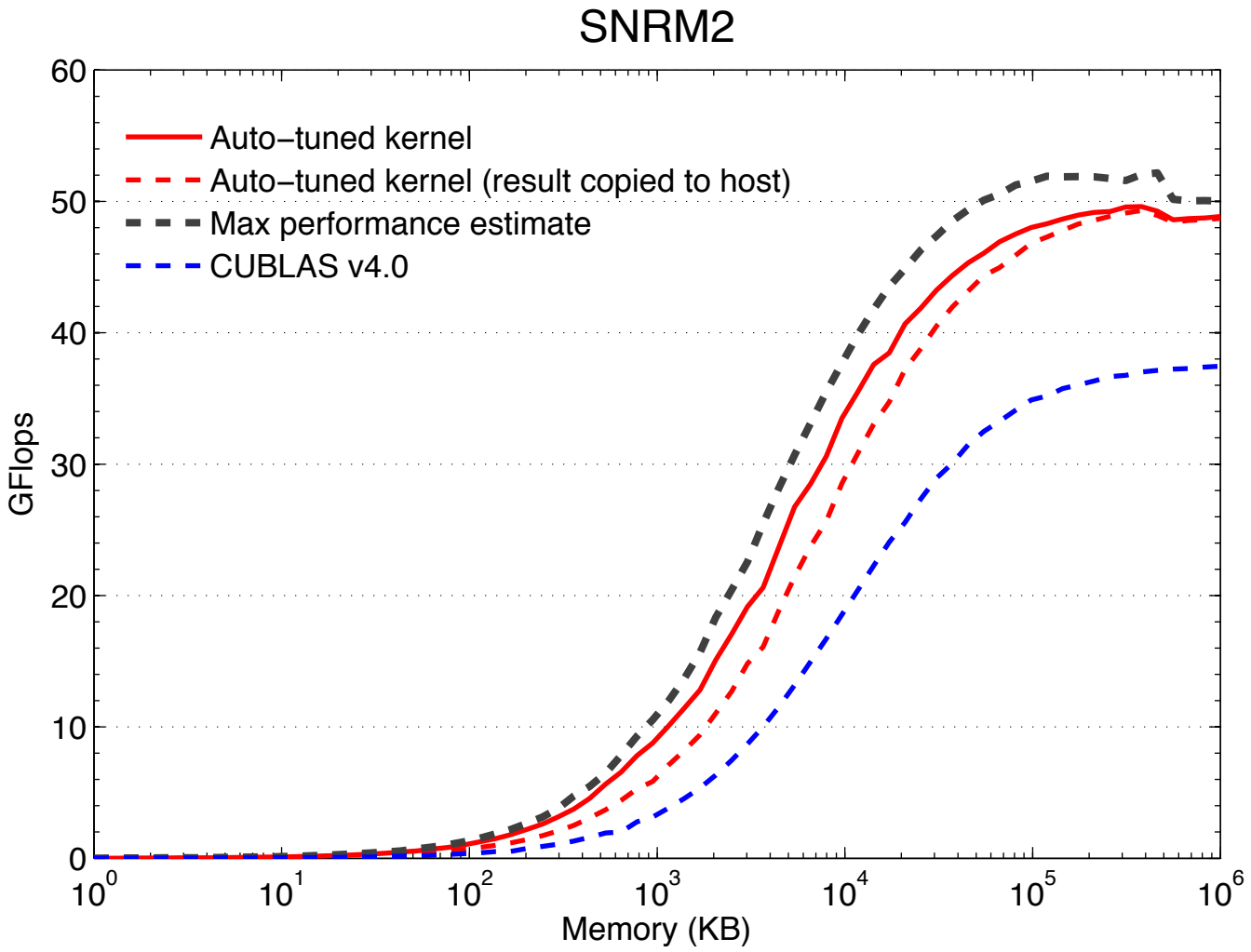
$$\texttt{WORKSIZE\_n} \in \{1, 2, 3, 4, 5, 6, 7, 8\} \times \texttt{BLOCKSIZE}$$

$$\texttt{UNROLL\_LEVEL} \in \{1, 2, 3, 4, 5, 6, 7, 8\}$$

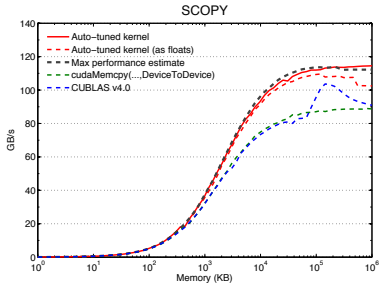
# SAXPY

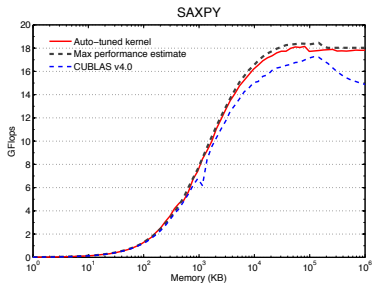

SAXPY
Auto−tuned kernel
Max performance estimate
CUBLAS v4.0
GFlops
Memory (KB)
0
2
4
6
8
10
12
14
16
18
20
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

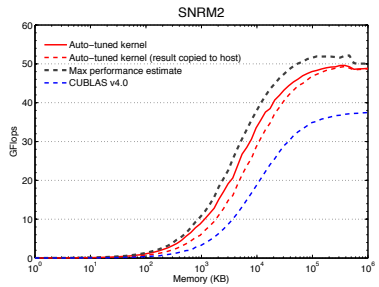
# SNRM2

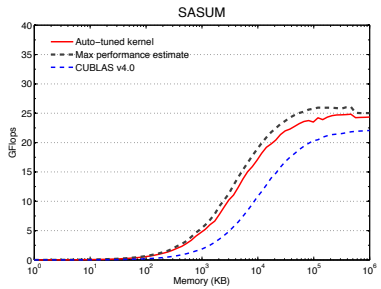

SNRM2
Auto−tuned kernel
Auto−tuned kernel (result copied to host)
Max performance estimate
CUBLAS v4.0
GFlops
Memory (KB)
0
10
20
30
40
50
60
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

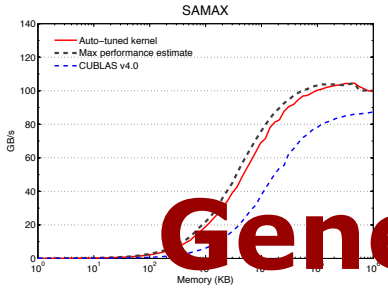
# GLAS on Nvidia C2050

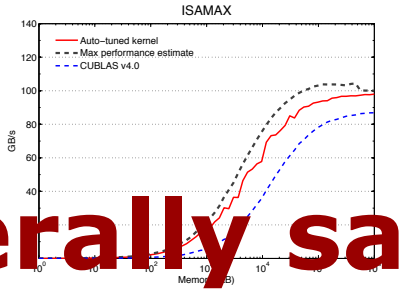


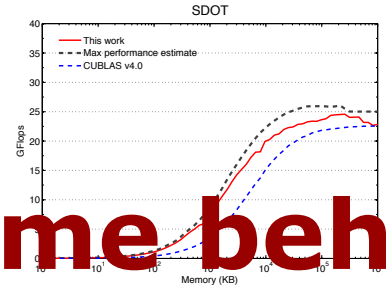



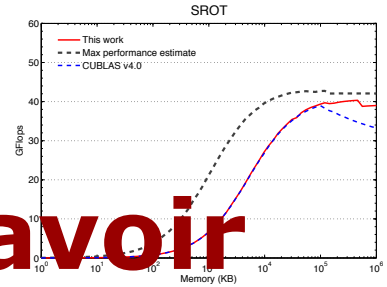



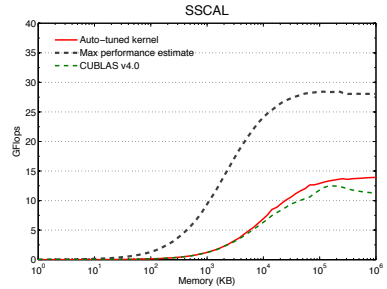



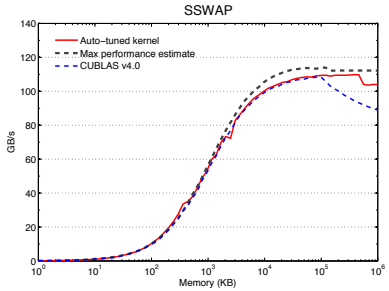



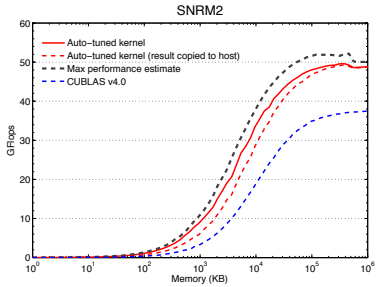



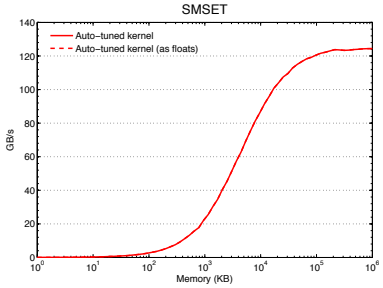



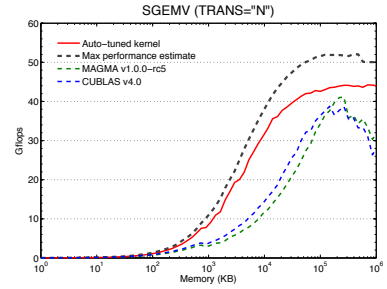

**Generally same behavoir**

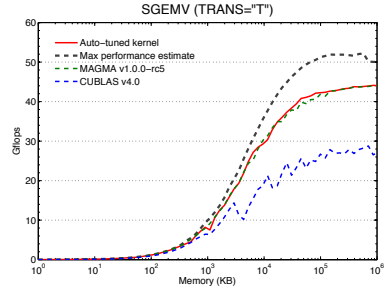


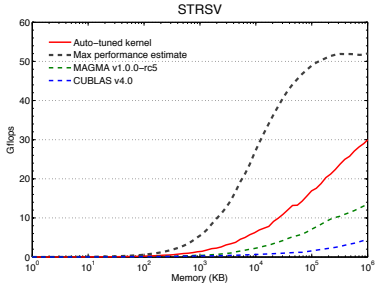



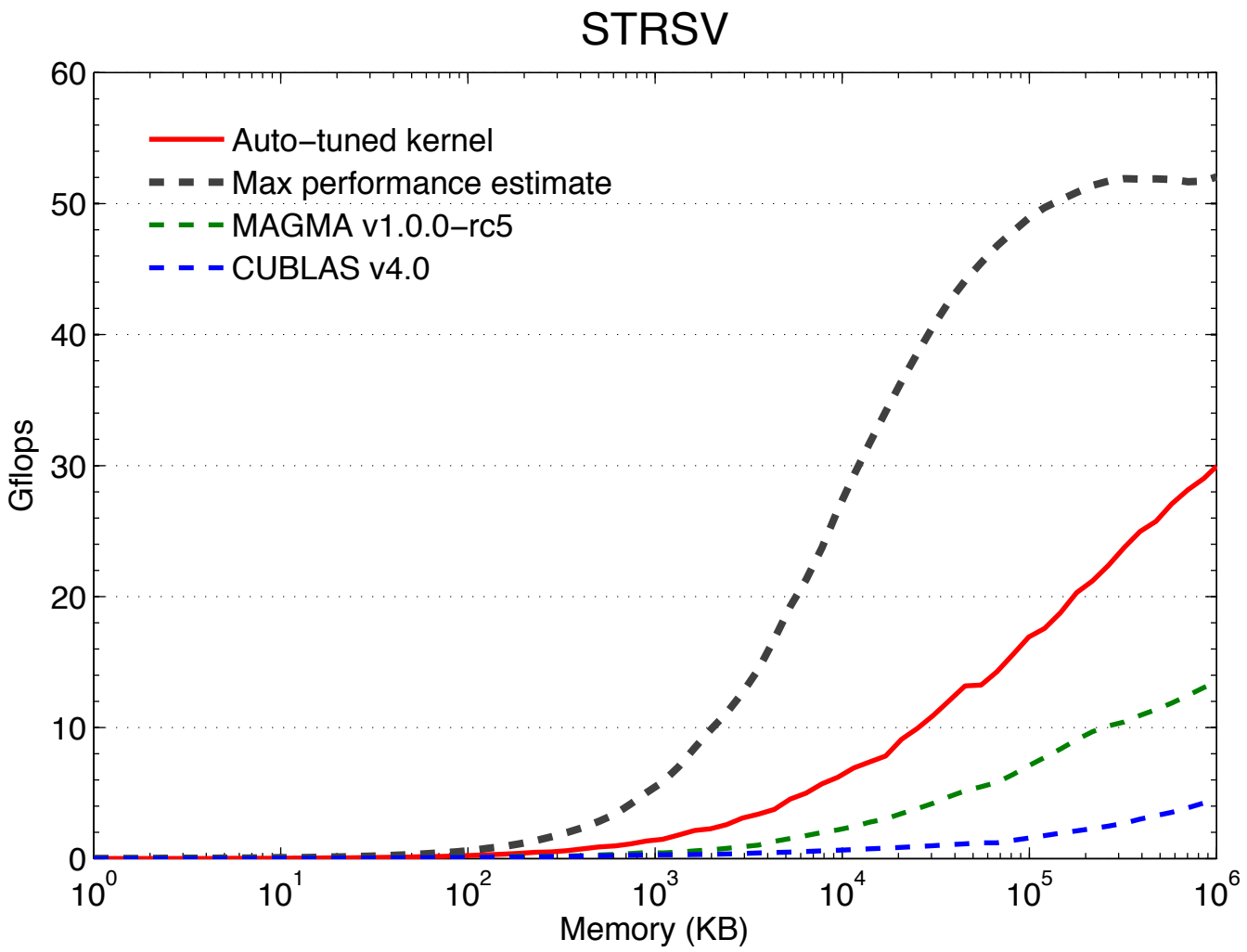




...



GPUlab DTU Informatics

# STRSV


STRSV
Auto−tuned kernel
Max performance estimate
MAGMA v1.0.0−rc5
CUBLAS v4.0
Gflops
Memory (KB)
0
10
20
30
40
50
60
$10^0$
$10^1$
$10^2$
$10^3$
$10^4$
$10^5$
$10^6$

# Summary

- BLAS level 1 and 2 kernels are totally memory bound.
- GPU's effective bandwidth sets the max performance.
- Simple auto-tuning can facilitate high-performance BLAS kernels for all input sizes and shapes.

- GLAS for single precision is available for download at `gpulab.imm.dtu.dk` (Open Source MIT License):
  - `glas_v0.2_C2050_cuda_4.0_linux.tar.gz`
  - `glas_v0.2_GTX590_cuda_4.0_linux.tar.gz`



Autotuning: Scam or artistic instrument?