MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Tuning the Blocksize for Dense Linear Algebra Factorization Routines with the Roofline Model

P. Benner[1], P. Ezzatti[2], E.S. Quintana-Ortí[1], A. Remón[3] and J.P. Silva[2]

December 15, 2016

[1]MPI for Dynamics of Complex Technical Systems, Magdeburg, Germany
[2]Universidad de la República, Montevideo, Uruguay
[3]Universidad Jaime I, Castellón, Spain

Motivation

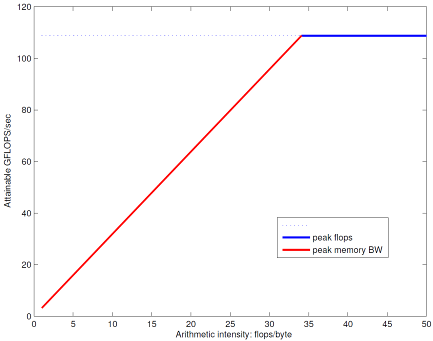
- Dense numerical linear algebra operations are crucial in many scientific and engineering applications
- Consequently, many efforts to optimize this operations can be found in the literature. A good example are the `basic numerical linear algebra subroutines` (BLAS) and LAPACK specifications
- Additionally, hardware manufacturers usually provide specific implementations of BLAS and LAPACK for their platforms

Motivation

- Tiled implementations has demonstrated to suit modern hardware architectures
- In particular, they provide a convenient memory pattern access that facilitates an efficient use of the memory hierarchy
- Unfortunately, the performance of tiled implementations highly depends on the `algorithmic blocksize` employed
- Find the optimal `algorithmic blocksize` is far from being trivial

Motivation

- The `Roofline` model is a graphical tool used to evaluate the performance of a computational kernel



- It merges in a single plot the memory-bound and compute-bound spaces of a hardware architecture

Motivation

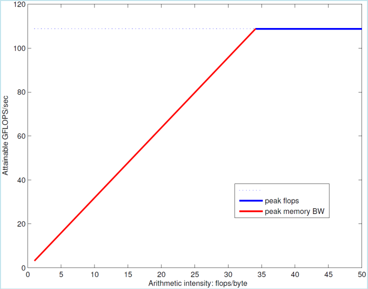
We aim to obtain the optimal `algorithmic blocksize` for a given computational kernel and hardware plaform from the related `Roofline model`

Outline

1. The `Roofline model`
2. The Gauss-Jordan alg.
3. Experimental evaluation
4. Conclusions and future work

The Roofline model

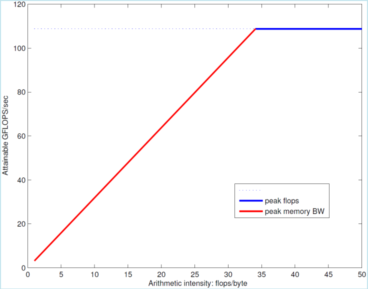
- Joints in a plot the memory and compute-bound spaces of the HW
- Note that the Roofline model is platform-specific



The Roofline model

The Roofline model

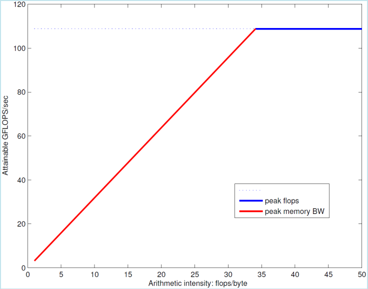
- Shows the maximal attainable performance at a given arithmetic intensity (AI)
- The AI of a kernel is computed as the ratio between: floating-point arithmetic operations (flops) and memory accesses (memops)



The Roofline model

The Roofline model

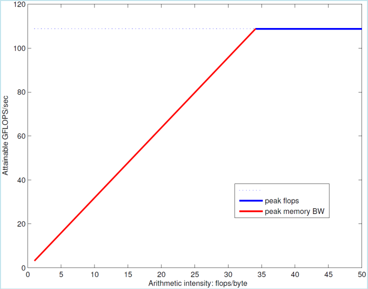
- Given the AI related to a computational kernel, the Roofline model states the maximal performance that can be expected
- In practice, Roofline model can be used to evaluate the performance of a computational kernel and identify its bottlenecks



The Roofline model

The Roofline model

- To create the model we need the peak performance and the memory bandwidth of the architecture
- These figures are provided by the HW manufacturer, or can be obtained experimentally



Outline

1. The `Roofline model`
2. The Gauss-Jordan alg.
3. Experimental evaluation
4. Conclusions and future work

The Gauss-Jordan alg.

- The Gauss-Jordan elimination (GJE) algorithm computes the inverse of a matrix
- GJE presents a computational cost and numerical properties similar to the traditional method based on the LU factorization
- It presents:
  - Fine grain parallelism computations: pivoting
  - Coarse grain parallelism computations: matrix-matrix products
- Similarities with matrix factorization methods: LU, Cholesky, QR

The Gauss-Jordan alg.

- The Gauss-Jordan elimination (GJE) algorithm computes the inverse of a matrix
- GJE presents a computational cost and numerical properties similar to the traditional method based on the LU factorization
- It presents:
  - Fine grain parallelism computations: pivoting
  - Coarse grain parallelism computations: matrix-matrix products
- Similarities with matrix factorization methods: LU, Cholesky, QR

$\Downarrow$

Results can be extrapolated to other linear algebra kernels

Algorithm: $[A] := \text{GJE\_BLK}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix} := \text{GJE\_UNB}\left(\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix}\right)$	Unblocked Gauss-Jordan
$A_{00} := A_{00} + A_{01}A_{10}$	Matrix-matrix product
$A_{20} := A_{20} + A_{21}A_{10}$	Matrix-matrix product
$A_{10} := A_{11}A_{10}$	Matrix-matrix product
$A_{02} := A_{02} + A_{01}A_{12}$	Matrix-matrix product
$A_{22} := A_{22} + A_{21}A_{12}$	Matrix-matrix product
$A_{12} := A_{11}A_{12}$	Matrix-matrix product

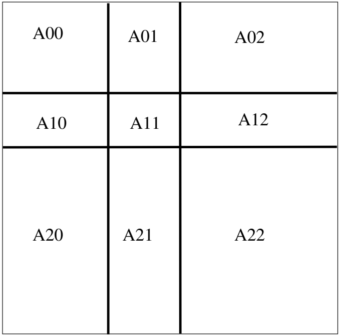
Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

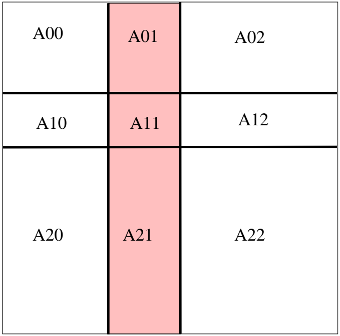
endwhile

The Gauss-Jordan alg.

At a given iteration



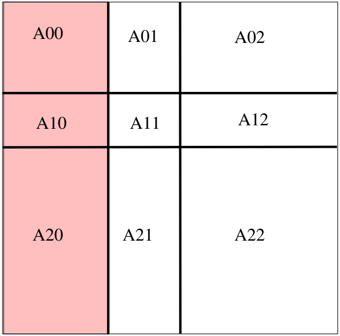
$$A_{11} \text{ is } b \times b$$



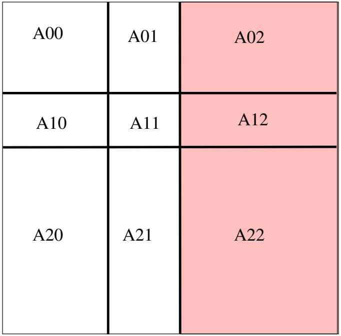
$$\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix} := GJE_{\text{UNB}}\left(\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix}\right)$$

The Gauss-Jordan alg.

At a given iteration



$$A_{00} := A_{00} + A_{01}A_{10}$$
$$A_{20} := A_{20} + A_{21}A_{10}$$
$$A_{10} := A_{11}A_{10}$$



$$A_{02} := A_{02} + A_{01}A_{12}$$
$$A_{22} := A_{22} + A_{21}A_{12}$$
$$A_{12} := A_{11}A_{12}$$

The Gauss-Jordan alg.

Which is the AI of GJE?

Flops and memops count (in 1 step of the algorithm)

Operation	flops	memops	BLAS level
GJE_UNB	$2nb^2$	$2nb^2$	1 - 2
MM products	$2n(n-b)b$	$2n(n-b)$	3

We asume that:

1 memop is required by 1 BLAS-1 / BLAS-2 flop

1 memop is required by b BLAS-3 flops

In summary

Total number of flops: $2n^3$ flops

Total number of memops: $2n^2(n/b-1+b)$ memops

The Gauss-Jordan alg.

Which is the AI of GJE? (cont.)

Computing optimal AI and b

$$AI_{gje} = \frac{2n^3}{2n^2(n/b - 1 + b)} \text{ flops-per-memop}$$

- We pretend to maximize AI $\rightarrow$ minize memops
- Optimal b (b_{opt}) is the one that minizes $2n^2(n/b - 1 + b)$

$$b_{opt} = \sqrt{n}$$

$$AI_{gje\,opt} = \frac{n}{2\sqrt{n}-1} \approx \frac{\sqrt{n}}{2}$$

The Gauss-Jordan alg.

The multi-block variant

- The panel factorization in GJE limits the algorithm performance
- This operation is performed via an unblocked variant of GJE, based on BLAS-1 and BLAS-2 kernels
- A variation that alleviates this problem replaces the unblocked GJE by a blocked variant of the same algorithm
- This reports a multi-blocksize variant of GJE that delivers higher performance

The Gauss-Jordan alg.

Algorithm: $[A] := \text{GJE\_BLK}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix} := \text{GJE\_UNB}\left(\begin{bmatrix} A_{01} \\ A_{11} \\ A_{21} \end{bmatrix}\right)$	Unblocked Gauss-Jordan $\leftarrow$ Blocked variant of blocksize c
$A_{00} := A_{00} + A_{01}A_{10}$	Matrix-matrix product
$A_{20} := A_{20} + A_{21}A_{10}$	Matrix-matrix product
$A_{10} := A_{11}A_{10}$	Matrix-matrix product
$A_{02} := A_{02} + A_{01}A_{12}$	Matrix-matrix product
$A_{22} := A_{22} + A_{21}A_{12}$	Matrix-matrix product
$A_{12} := A_{11}A_{12}$	Matrix-matrix product

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

The Gauss-Jordan alg.

Which is the AI of multiblock-GJE?

Flops and memops count (in 1 step of the algorithm)

Operation	flops	memops	BLAS level
GJE_UNB	$2nc^2$	$2nc^2$	1 - 2
GJE_BLK	$2n(b-c)c$	$2n(b-c)$	3
MM products	$2n(n-b)b$	$2n(n-b)$	3

Note that:

The inner and outer instances of GJE_{blk} require b/c and n/b steps

1 memop required by c/b BLAS-3 flops in inner/outer GJE_{blk}

In summary

Total number of flops: $2n^3$ flops

Total number of memops: $2n^2(n/b - 2 + b/c + c)$ memops

Which is the AI of multiblock-GJE? (cont.)

Computing optimal AI and b

- Optimal c (c_{opt}) is $\sqrt(n)$

$$AI_{mbgje} = \frac{2n^3}{2n^2(n/b - 2 + 2\sqrt(b))} \text{ flops-per-memop}$$

$$b_{opt} = \sqrt{n} \qquad c_{opt} = \sqrt{b}$$

$$AI_{mbgje_{opt}} = \frac{n}{3\sqrt[3]{n} - 2} \approx \frac{(\sqrt[3]{n})^2}{3}$$

Outline

1. The `Roofline model`
2. The Gauss-Jordan alg.
3. Experimental evaluation
4. Conclusions and future work

Experimental evaluation

Hardware

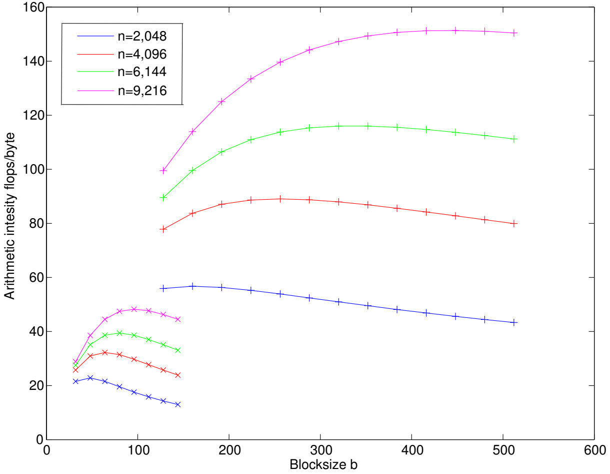
Processor	#cores	frequency	Bandwidth	Peak (DP)
INTEL i7-4770	4	3.4 GHz	25.5 GB/s	108.8 GFlops

All implementations

- Rely on kernels in the Intel MKL v.11.1 library
- Use the four cores in the platform

Experimental evaluation

Impact of *b* on the AI



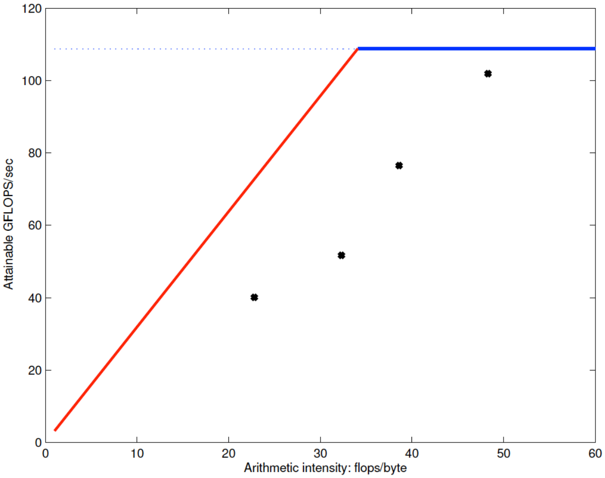
Experimental evaluation

Impact of b on the performance

Matrix dimension	b_{opt}	b	GFLOPS
2,048	45	32	38
		48	40
		64	40
4,096	64	32	45
		64	52
		96	51
6,144	78	64	62
		96	76
		128	75
9,216	96	64	78
		96	102
		128	98

Experimental evaluation

Impact of *b* on the performance



Experimental evaluation

Impact of *b* on the performance

Matrix dimension	b_{opt}-c_{opt}	b-c	GFLOPS	Arith. intensity
2,048	161-12	160-16	62	56
4,096	256-16	256-16	69	89
6,144	335-18	320-16	82	115
9,216	406-20	384-16	94	149

Outline

1. The `Roofline model`
2. The Gauss-Jordan alg.
3. Experimental evaluation
4. Conclusions and future work

Conclusions and future work

Conclusions

- we have presented simple yet accurate models to determine the blocksize in GJE implementations
- we have extended the approach to multi-block algorithms
- the experiments demonstrate that imrpovements in the AI report gains in performance
- the results can be extended to other kernels like i.e., dense matrix factorization

Future work

- apply same technique to other linear algebra kernels
- develop more accurate models

Thanks.