Execution-Less Performance Modeling

Roman Iakymchuk and Paolo Bientinesi
AICES, RWTH Aachen
Schinkelstr. 2
52062 Aachen, Germany
{iakymchuk,pauldj}@aices.rwth-aachen.de

ABSTRACT

We aim at modeling the performance of linear algebra algorithms without executing either the algorithms or any parts of them. The performance of an algorithm can be expressed in terms of the time spent on CPU execution and memory-stalls. The main concern of the study is to build analytical models to accurately predict memory-stalls. We construct an analytical formula for modeling cache misses of fundamental linear algebra operations such as those included in the Basic Linear Algebra Subprograms (BLAS) library. The number of cache misses occurring in higher-level algorithms—like a matrix factorization—is then predicted by combining the models for the appropriate BLAS subroutines. As case studies, we consider the LU factorization and GER—a BLAS operation and a building block for the LU factorization. We validate the models on both Intel and AMD processors, attaining remarkably accurate performance predictions.

Categories and Subject Descriptors

B.3.2.b [**Cache Memories**]; B.3.3 [**Performance Analysis and Design Aids**]; G.4.a [**Algorithm design and analysis**]

General Terms

Measurement, Performance

Keywords

Performance prediction, Performance model, Cache misses, Memory-stalls

1. THE MODEL

Predicting the performance of an algorithm is a problem that, although widely investigated, is still far from solved. Our objective is to predict performance *without* executing either the target algorithm or parts of it. We first focus on the basic linear algebra operations like the ones included in the BLAS library. For these, we develop analytical models that predict the amount of computation and data movement to be performed. The performance of higher-level algorithms, like those included in the LAPACK library, is then built by composing the models for the BLAS subroutines used within the algorithm.

PMBS'11, November 13th, 2011, Seattle, Washington, USA.
ACM 978-1-4503-1102-1/11/11.

In general, the performance of an algorithm can be defined as a ratio between the number of floating point operations ($\#FLOPS$) performed by the algorithm and the execution time:

$$Performance = \frac{\#FLOPS}{Execution\_time}. \quad (1)$$

For direct algorithms, $\#FLOPS$ can be calculated *a priori*; the prediction of performance therefore reduces to the prediction of *Execution_time*.

Our strategy consists of exploiting detailed information about the algorithm, the CPU, and the memory hierarchy. Since memory-stalls idle the CPU and add a significant overhead to the computational time, we model *Execution_time* not only by means of the CPU execution time, but also through the time in which the CPU is idle due to memory-stalls:

$$\begin{aligned} Execution\_Time = & \quad (2)\\ & \sum_{i=1}^{n} \alpha_i \times L_i\_cache\_misses \times time(L_i\_cache\_miss) \\ & + \beta \times TLB\_misses \times time(TLB\_miss) \\ & + \gamma \times \#FLOPS \times time(FLOP). \end{aligned}$$

The memory-stalls play an especially important role in operations—like those included in the Levels 1 and 2 of BLAS, and the unblocked algorithms in the LAPACK library—that are memory-bound. Conversely, Level 3 BLAS operations are compute bound; for those, the execution time can be predicted rather accurately by a mere count of the floating point operations to be performed. For this reason, here we focus on the more challenging goal of predicting performance for memory-bound operations.

We restrict this study to a scenario where data resides in L2 cache, so only L1 data cache misses (L1 misses) occur. In our experiments, we use an unblocked variant of an LU factorization that is built on top of the BLAS routines GER and SCAL:

$$\begin{aligned} a_{21} &:= a_{21}/\alpha_{11} && \texttt{(SCAL)} \\ A_{22} &:= A_{22} - a_{21}a_{12}^T && \texttt{(GER)} \end{aligned}$$

SCAL only operates on one vector and its impact on the total number of floating point operations is negligible. Consequently, we model the cache misses for the unblocked LU factorization by modeling the misses for GER, for which we

provide an analytic formula:

$$L1\_misses = \zeta + \left\lceil \frac{p}{d} \right\rceil + \left\lceil \frac{q}{d} \right\rceil, \quad (3)$$

where d is the number of double precision floating point values in a cache line; $m \times n$ and $p \times q$ are the sizes of a general matrix A and the matrix A_{22}, respectively; $\left\lceil \frac{p}{d} \right\rceil$ and $\left\lceil \frac{q}{d} \right\rceil$ are the numbers of cache misses when reading the vectors a_{21} and a_{12}^T. The quantity

$$\zeta = \begin{cases} \left\lfloor \frac{mq}{d} \right\rfloor, & \text{if } m - \left\lfloor \frac{p}{d} \right\rfloor d < d \\ \left\lceil \frac{p}{d} \right\rceil + \sum_{i=1}^{n-1} \left\lceil \frac{p + (mi \bmod d)}{d} \right\rceil, & \text{otherwise} \end{cases}$$

indicate L1 misses when reading the matrix A_{22}.

We validate our approach on two architectures with different processor types and memory systems. By working on two different architectures, we show that the basic formula needs to be tailored according to a number of parameters. Nevertheless, the advantage of the formula is that it can be used for other algorithms with the same structure.

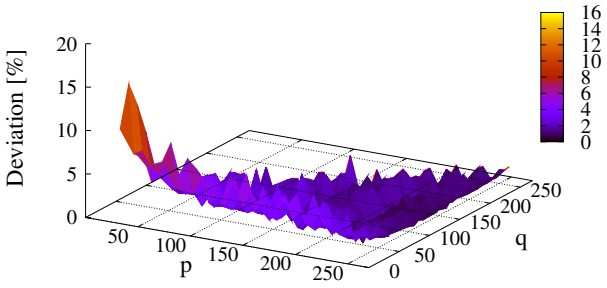

Figure 1: Deviation between predicted and measured L1 misses for GER from the GotoBLAS2 library on Intel Core 2; $p \geq q$, $LDA = 512$.

2. RESULTS

Fig. 1 presents the results of modeling L1 misses of GER in the general scenario $p \geq q$ ($LDA = 512$). The figure indicates that in the area close to the origin the deviation is slightly higher than for the rest of the spectrum. However, such deviations do not affect the accuracy of the model for the whole LU factorization.

We also look at the scenario in which the unblocked algorithm is a building block for a blocked variant of an LU factorization; in this case, q is small, $p \geq q$, and LDA might be much greater than p. To model the L1 misses of the unblocked algorithm, we aggregate the models for GER from of size $p \times q$, $p-1 \times q-1$, down to $p-q+1 \times 1$. Fig. 2 shows the modeled and measured L1 misses for $LDA = 2048$ and fixed $q = 128$. Since q is fixed and only p varies, the modeled and measured misses demonstrate linear behavior. In summary, the results are very accurate—the model is within 2-4% of the measurements.

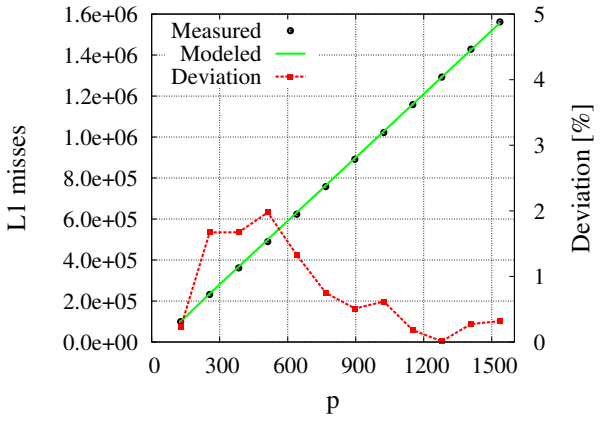

Figure 2: Deviation between predicted and measured L1 misses for the unblocked LU factorization when used as part of a blocked algorithm on Intel Core 2; $q = 128$; $LDA = 2048$.

3. CONCLUSIONS

We set out to predict the performance of linear algebra algorithms without any actual code execution. In fact, we proposed an analytical model solely based on detailed knowledge of the algorithm as well as the CPU and the memory hierarchy. As it was shown, modeling performance is equivalent to modeling both the CPU execution time and the time spent on memory-stalls. Our main focus is mainly on memory-stalls. We considered the scenario in which the input data resides in the L2 cache, and built an analytical formula for modeling L1 cache misses. As target algorithms, we considered kernel linear algebra operations like those included in the BLAS library. To verify the model, we conducted a set of experiments using GER from the reference BLAS and the highly optimized GotoBLAS2 libraries. By working on two architectures, we tailored the basic formula for different processor types and memory systems. In all cases, the model resulted extremely accurate, with deviations normally lower than 2%. We chose the GER kernel because it is responsible for most of the computation of an unblocked variant of an LU factorization. By composing modeled L1 misses of GER, we then predicted the number of misses for the LU factorization. A comparison between the model and the actual measurements yielded deviations below 3 %. The full paper will be published in [1].

Acknowledgments

We would like to thank Prof. Enrique S. Quintana Ortí and Diego Fabregat Traver for their advices and many useful discussions.

The authors gratefully acknowledge the support received from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111.

4. REFERENCES

[1] Roman Iakymchuk and Paolo Bientinesi. Modeling Performance through Memory-Stalls. *SIGMETRICS Performance Evaluation Review*, 40(2), 2012.