Performance Prediction through Time Measurements

Roman Iakymchuk

AICES Graduate School, RWTH Aachen
iakymchuk@aices.rwth-aachen.de

International Conference on High Performance Computing
October 12-14, 2011
Kyiv, Ukraine

Objective

$$Performance = \frac{\#FLOPS}{Execution\_time}$$

#FLOPS is known **a priori**

$$Performance = \frac{\#FLOPS}{Execution\_time}$$

#FLOPS is known **a priori**

Modeling Performance

- Target—linear algebra algorithms

$$Performance = \frac{\#FLOPS}{Execution\_time}$$

#FLOPS is known **a priori**

Modeling Performance

- Target—linear algebra algorithms

LAPACK ScaLAPACK FLAME PLAPACK ATLAS

$\searrow \quad \downarrow \quad \swarrow$

BLAS

Objective

$$Performance = \frac{\#FLOPS}{Execution\_time}$$

#FLOPS is known **a priori**

Modeling Performance

- Target—linear algebra algorithms

LAPACK ScaLAPACK FLAME PLAPACK ATLAS

↘ ↓ ↙

BLAS

- Model time of the kernel operations (BLAS) by conducting only **few** measurements

Objective

$$Performance = \frac{\#FLOPS}{Execution\_time}$$

#FLOPS is known **a priori**

Modeling Performance

- Target—linear algebra algorithms

LAPACK ScaLAPACK FLAME PLAPACK ATLAS

$\searrow$ $\downarrow$ $\swarrow$

BLAS

- Model time of the kernel operations (BLAS) by conducting only **few** measurements
- Higher level algorithms are modeled without timings

Outline

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Wall time

- High resolution (cycle-accurate)

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Wall time

- High resolution (cycle-accurate)
- Includes other processes

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Wall time

- High resolution (cycle-accurate)
- Includes other processes

Improving the accuracy of timings

- Take multiple timing samples

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Wall time

- High resolution (cycle-accurate)
- Includes other processes

Improving the accuracy of timings

- Take multiple timing samples
- Select **median** timing for CPU time

Choosing a system timer

CPU time

- if machine heavily loaded and no I/O and parallelism
- **Low resolution**
- Over- and under-reports time

Wall time

- High resolution (cycle-accurate)
- Includes other processes

Improving the accuracy of timings

- Take multiple timing samples
- Select **median** timing for CPU time
- Select minimum for wall time

Time Measurements

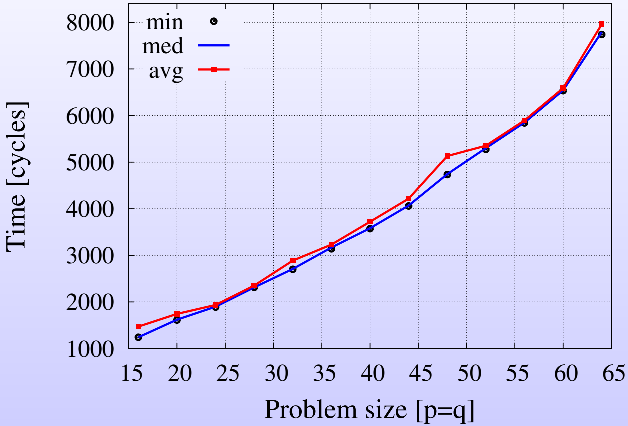


Figure: In-cache timing of GER

- GER: $A := A + \alpha x y^T$
- The cycle-accurate wall timer is used

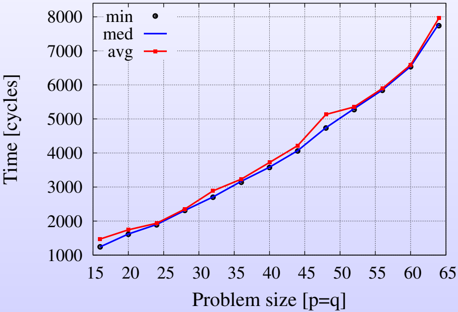


Figure: In-cache

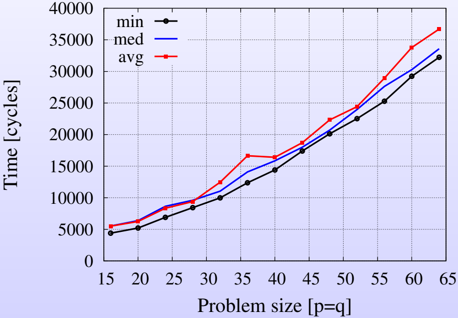
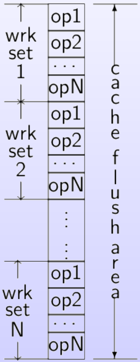
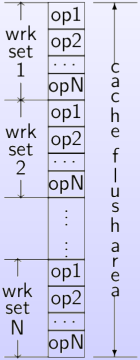


Figure: Out-of-cache



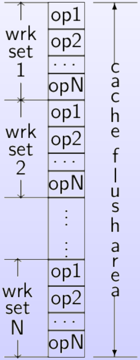
Source: R. Clint Whaley (UTSA-CS)

- $size($*flush_area*$) =$ *Associativity* $\times\ size($*cache*$)$



Source: R. Clint Whaley (UTSA-CS)

- $size($*flush_area*$) =$ *Associativity* $\times\ size($*cache*$)$
- Conduct `n_rep` timing samples of an algorithm



Source: R. Clint Whaley (UTSA-CS)

- $size(\textit{flush\_area}) = \textit{Associativity} \times size(\textit{cache})$
- Conduct `n_rep` timing samples of an algorithm
- On each iteration of `n_rep` loop **operands** are out-of-cache

BLAS subroutines

- Apply polynomial interpolation

$$\textit{Execution\_time} = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0$$

BLAS subroutines

- Apply polynomial interpolation

$$\textit{Execution\_time} = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0$$

- Complexity of a BLAS subroutine is at most $O(n^3) \rightarrow k \leq 3$

BLAS subroutines

- Apply polynomial interpolation

$$\textit{Execution\_time} = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0$$

- Complexity of a BLAS subroutine is at most $O(n^3) \rightarrow k \leq 3$
- Perform $4 - 6$ measurements on each memory level

BLAS subroutines

- Apply polynomial interpolation

$$\textit{Execution\_time} = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0$$

- Complexity of a BLAS subroutine is at most $O(n^3) \rightarrow k \leq 3$
- Perform $4 - 6$ measurements on each memory level
- Solve a linear least squares problem

Performance Prediction

BLAS subroutines

- Apply polynomial interpolation

$$\textit{Execution\_time} = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$$

- Complexity of a BLAS subroutine is at most $O(n^3) \rightarrow k \leq 3$
- Perform $4 - 6$ measurements on each memory level
- Solve a linear least squares problem

Higher level algorithms

$$\textit{Execution\_time} = \sum_{i=1}^{n-1} \texttt{Model\_subroutines\_time}(i)$$

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**

Repartition

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

where α_{11} is 1×1

$a_{21} := a_{21} / \alpha_{11}$ SCAL

$A_{22} := A_{22} - a_{21} a_{12}^T$ GER

Continue with

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

endwhile

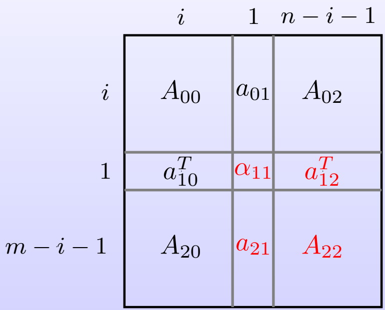


Figure: 3×3 partitioning of A.

A Case Study: LU Factorization

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**

Repartition

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

where α_{11} **is** 1×1

$a_{21} := a_{21} / \alpha_{11}$ SCAL

$A_{22} := A_{22} - a_{21} a_{12}^T$ GER

Continue with

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

endwhile

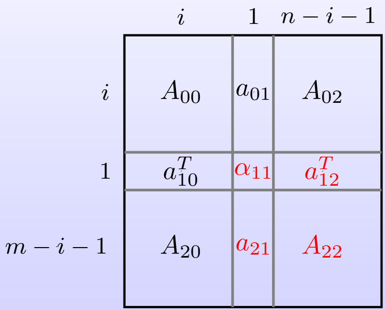


Figure: 3×3 partitioning of A.

GER performs more than 96 % of the *#FLOPS* in the LU

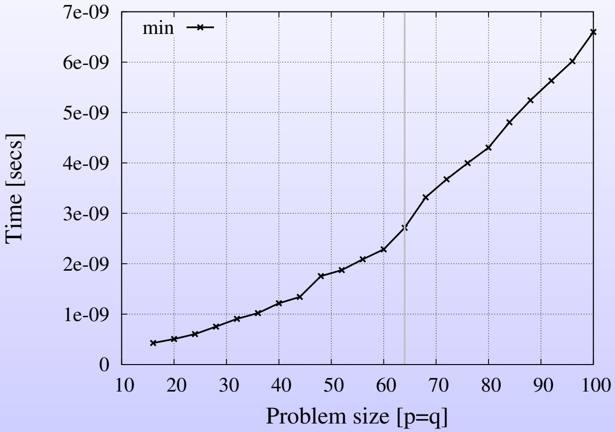


Figure: Piecewise-parabolic behavior of GER

- Intel Harpertown @3.0 GHz
- L1 (32 KB) and L2 (6 MB) caches
- Apply parabolic interpolation on L1 & L2 caches

Performance Prediction: GER & LU

GER

Complexity of GER is $O(n^2) \rightarrow$

$$\textit{Execution\_time} = a_2 n^2 + a_1 n + a_0$$

Performance Prediction: GER & LU

RWTH AACHEN UNIVERSITY

GER

Complexity of GER is $O(n^2) \rightarrow$

$$\textit{Execution\_time} = a_2 n^2 + a_1 n + a_0$$

The unblocked LU

- Prediction:

$$\textit{Execution\_time} = \sum_{i=1}^{n-1} \texttt{Model\_GER\_time}(i)$$

AI ces

GER

Complexity of GER is $O(n^2) \rightarrow$

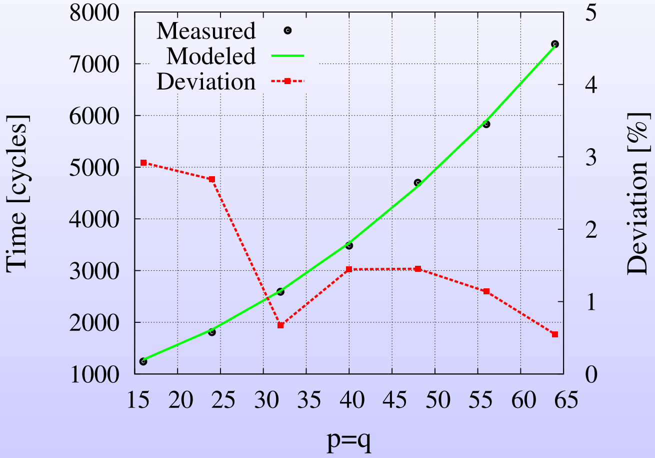
$$\textit{Execution\_time} = a_2 n^2 + a_1 n + a_0$$

The unblocked LU

- Prediction:

$$\textit{Execution\_time} = \sum_{i=1}^{n-1} \texttt{Model\_GER\_time}(i)$$

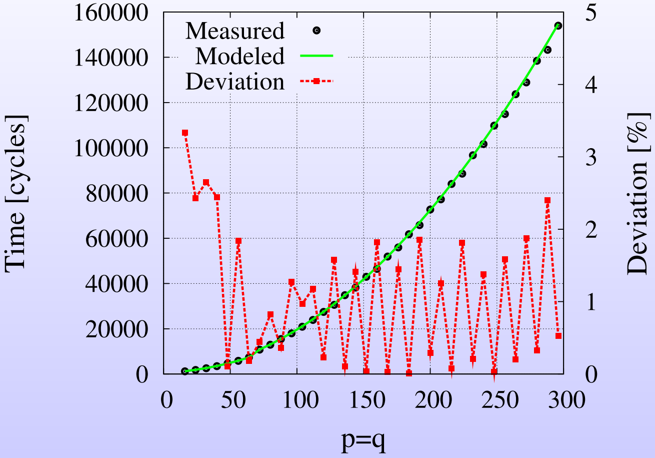
- In total, GER is measured only 8-12 times



- GER from the GotoBLAS library is used
- $p \leq 64$ fit in the L1 cache
- The deviation decreases; it is less than 3 %

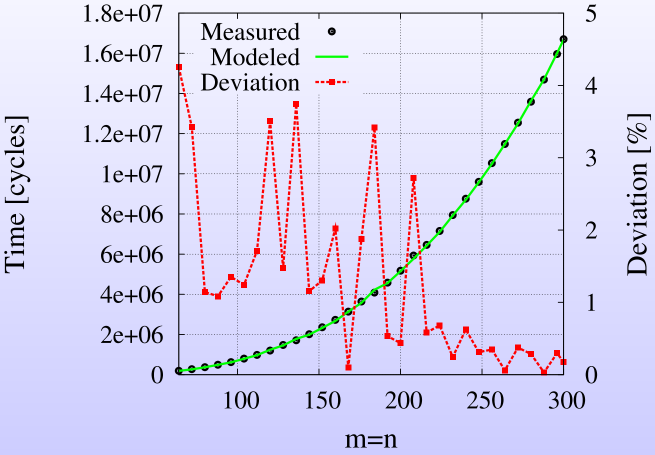
Figure: Predicting the execution time of GER on Harpertown

Evaluation: GER



- $p < 300$ fit in the L1 and L2
- The deviation is less than 2 %

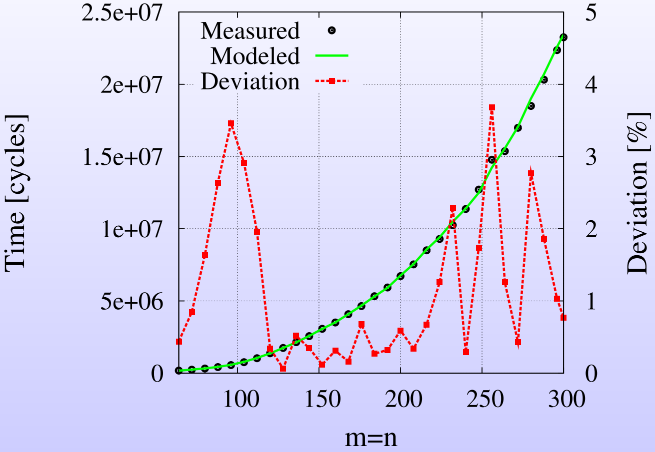
Figure: Predicting the execution time of GER on Harpertown



- Closer to origin the deviation is higher
- When $m = n$ increases the deviation $\to 0$

Figure: Modeling the execution time of the LU on **Harpertown**

Evaluation: LU



- Each core has L1(64 KB), L2(512 KB), and L3(2 MB)
- The results have higher variance
- The deviation is less than 3 %

Figure: Modeling the execution time of the LU on **Barcelona**

Conclusions

- The **approach** was validated by modeling the execution time of GER and the LU factorization

Conclusions

- The **approach** was validated by modeling the execution time of GER and the LU factorization
- The experiments were conducted on two **different architectures**

Conclusions

- The **approach** was validated by modeling the execution time of GER and the LU factorization
- The experiments were conducted on two **different architectures**
- The **deviation** is mostly less than 2-3 %

Financial support from the
Deutsche Forschungsgemeinschaft
through grant GSC 111
is gratefully acknowledged

DFG Deutsche Forschungsgemeinschaft