Modeling the LU Factorization for SMP Clusters

Emmanuel Jeannot
(joint work with Julien Langou and Jack Dongarra)

LORIA-INRIA Lorraine

Meeting on PARALLEL ROUTINES OPTIMIZATION AND APPLICATIONS

13th June 2007

Outline of the talk

Introduction: LU Factorization

- A: a (squared, no-singular) matrix,
- LU factorization computes $A = P \cdot L \cdot U$ where:
  - L is a lower triangular unit matrix,
  - U is a upper triangular matrix,
  - P is a permutation matrix.
- Used to solve the problem $Ax = b$ where A and b are given (by performing two successive triangular solves on U then L).

Introduction: context

- A: dense matrix of size $N \times N$,
- parallel factorization,
- 2-D **block-cyclic** distribution of the LU factorization as it is implemented in ScaLapack,
- $P \times Q = NP$ processor grid,
- block are squared of size NB.

Introduction: previous work

Scalapack Users Guide:

- crude model,
- shows the scalability of ScaLapack,
- Do not use the processor grid shape.

Lapack working note 43 (Dongarra, van de Geijn, Walker):

- simple model with 3 parameters:
  - the bandwidth of the network,
  - the latency of the network,
  - and the flop rate of the processor
- the value of these parameters does not depend on the subroutine of the algorithm.

Domas, Desprez, Tourancheau 96:

- Very fine model with more than 30 parameters.
- Do not show how to compute the parameters.
- Able compute the optimal grid size.

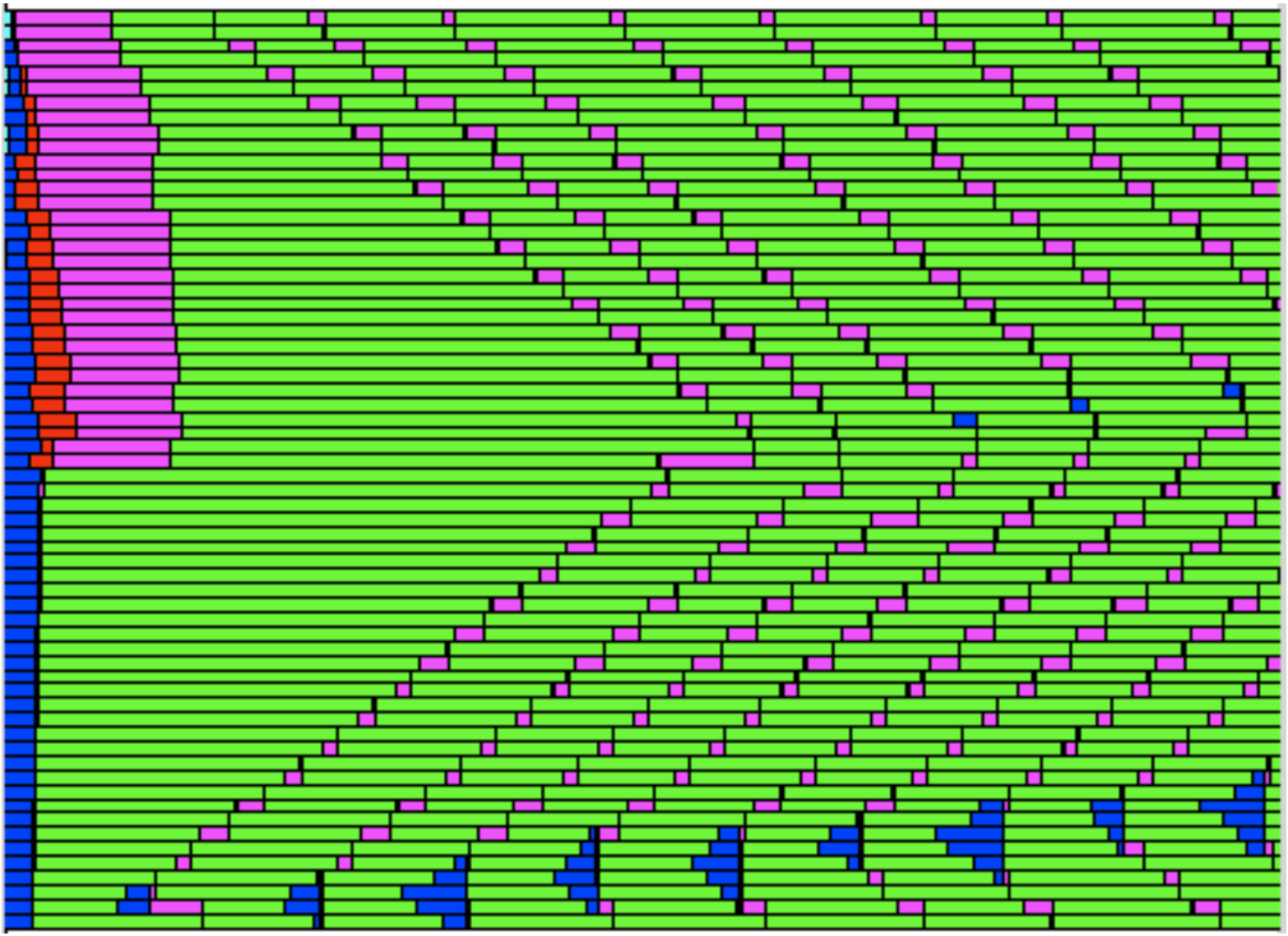
Goal of this work

A new model:

- A detailed model :
  - take into account all the phenomena
  - derive a model for each step of the factorization and for the whole factorization.
- A fine parameterization: different parameters are derived for each subroutines used.
- An automatic parameter computing. We show how to simply compute these parameters.
- An Automatic grid shape computing.
- Enhancement to SMP clusters.

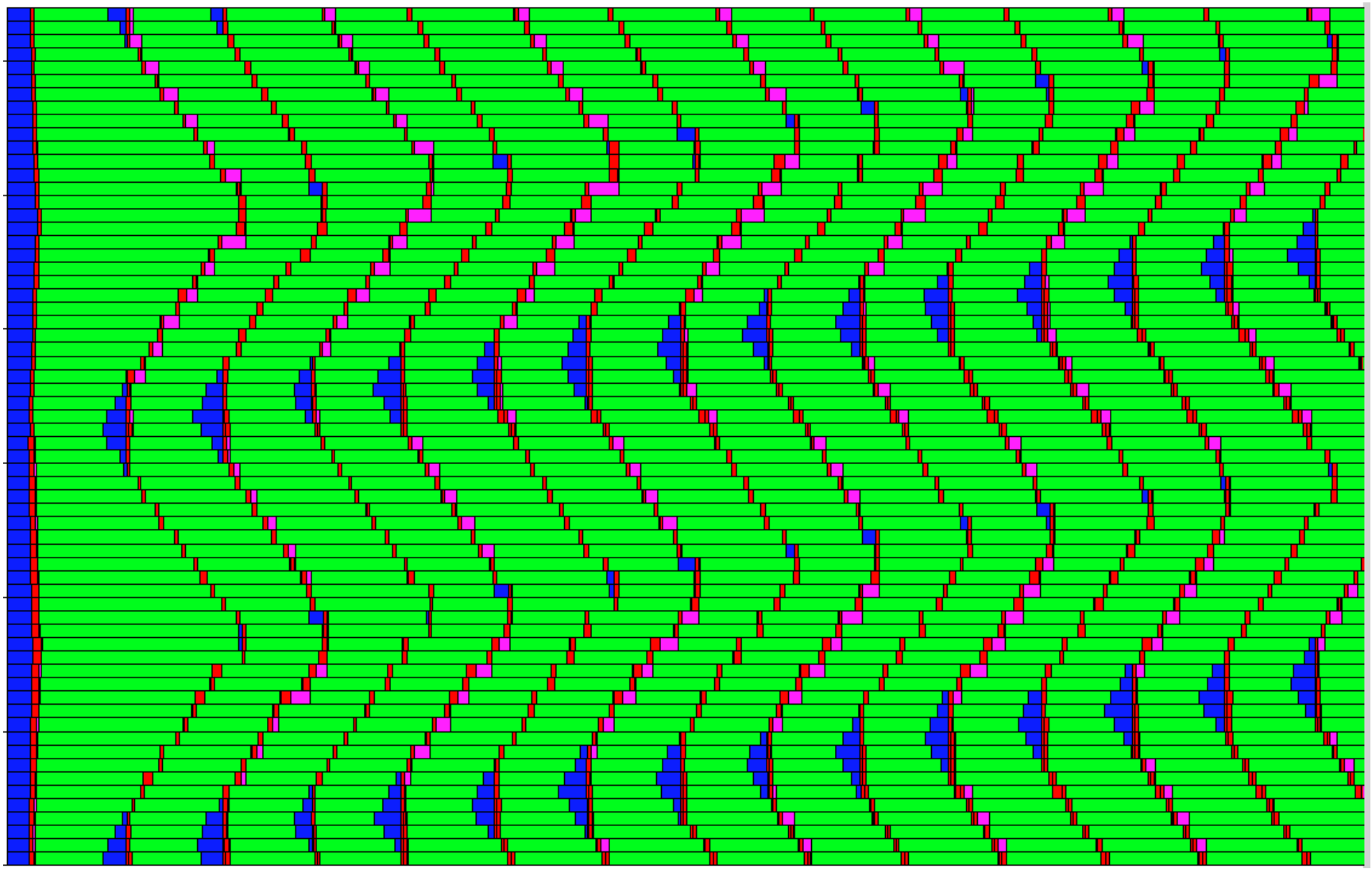
Example of what we want to model

$P = 1,\ Q = 64,\ N = 10000$ and $NB = 32$



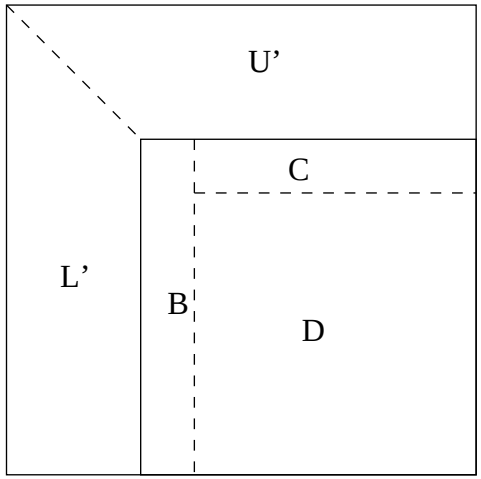
Example of what we want to model

$P = 2,\ Q = 32,\ N = 10000$ and $NB = 32$



Partial fatorization

After a partial factorization:



4 steps:

1. factorize the panel,
2. swapping row according to pivoting
3. compute block row of U
4. updating trailing submatrix

ScaLapack Code Example

```
      DO 10 J = JN+1, JA+MN-1, DESCA( NB_ )
         JB = MIN( MN-J+JA, DESCA( NB_ ) )
         I = IA + J - JA
*
* Factor diagonal and subdiagonal blocks and test for exact
* singularity.
*
         CALL PDGETF2( M-J+JA, JB, A, I, J, DESCA, IPIV, IINFO )
*
         IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $   INFO = IINFO + J - JA
*
* Apply interchanges to columns JA:J-JA.
*
         CALL PDLASWP( 'Forward', 'Rowwise', J-JA, A, IA, JA, DESCA,
     $        I, I+JB-1, IPIV )
*
         IF( J-JA+JB+1.LE.N ) THEN
*
* Apply interchanges to columns J+JB:JA+N-1.
*
            CALL PDLASWP( 'Forward', 'Rowwise', N-J-JB+JA, A, IA, J+JB,
     $           DESCA, I, I+JB-1, IPIV )
*
* Compute block row of U.
*
            CALL PDTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
     $           N-J-JB+JA, ONE, A, I, J, DESCA, A, I, J+JB,
     $           DESCA )
*           IF( J-JA+JB+1.LE.M ) THEN
*
* Update trailing submatrix.
*
               CALL PDGEMM( 'No transpose', 'No transpose', M-J-JB+JA,
     $              N-J-JB+JA, JB, -ONE, A, I+JB, J, DESCA, A,
     $              I, J+JB, DESCA, ONE, A, I+JB, J+JB, DESCA )
*
            END IF
         END IF
*
   10 CONTINUE
```

Modeling each step

Each step have different requirement in term of:

- computation,
- access pattern to the data,
- communication volume.

Therefore, for each substep `xxx`:

- γ_{xxx} : the time to perform one operation in this substep,
- β_{xxx}: the latency of the network for this substep
- α_{xxx}: the time to transfer one matrix element for this subroutine.

$\Rightarrow$ 12 parameters.

Step 1: factorize the panel

For each column of the panel:

1. determe the pivot row.
2. swap the pivot row.
3. broadcast the pivot column.
4. update the local submatrix.

The cost for factorizing panel k is:

$$\beta_{\text{tf2}}\left(NB\left(2\log_2(P)+\frac{P-1}{P}\right)+log_2(Q)\right) + \alpha_{\text{tf2}}\left(\left(\log_2(P)+\frac{P-1}{P}\right)\frac{NB^2}{2}+NB\log_2(Q)\right) + \frac{\gamma_{\text{tf2}}}{P}\left((N-(k-1)NB)\,NB^2-\frac{1}{3}NB^3\right)$$

Factorizing all the panels ($k=[1,\ldots,\lceil N/NB\rceil]$), costs:

$$\beta_{\text{tf2}}\left(N\left(2\log_2(P)+\frac{P-1}{P}\right)+\frac{N}{NB}log_2(Q)\right) + \alpha_{\text{tf2}}\left(\left(\log_2(P)+\frac{P-1}{P}\right)\frac{NNB}{2}+N\log_2(Q)\right) + \frac{\gamma_{\text{tf2}}}{P}\left(\frac{1}{2}N(N+NB)NB-\frac{1}{3}NNB^2\right)$$

Step 2: swapping row according to pivoting

There is $\frac{P-1}{P}NB$ rows to swap between different processors on the average.

The cost for panel k is:

$$\beta_{\text{swp}}\frac{P-1}{P}NB + \alpha_{\text{swp}}\lceil (N-NB)/Q \rceil \frac{P-1}{P}NB + \gamma_{\text{swp}}\frac{NB}{P}\lceil (N-NB)/Q \rceil$$

The swapping cost for all the matrix is then:

$$\beta_{\text{swp}}N\frac{P-1}{P} + \alpha_{\text{swp}}N\lceil (N-NB)/Q \rceil \frac{P-1}{P} + \gamma_{\text{swp}}\frac{N}{P}\lceil (N-NB)/Q \rceil$$

Step 3: compute block row of U

This substep is done by :

1. distributing the factored upper block of the panel
2. performing a triangular solve

For panel k the cost is:

$$\log_2(Q)(\beta_{\text{trsm}} + \alpha_{\text{trsm}} NB^2) + \gamma_{\text{trsm}} NB^2 \lceil (N - kNB)/Q \rceil$$

For all the whole factorization the total costs of this substep is:

$$\beta_{\text{trsm}} \log_2(Q) \frac{N}{NB} + \alpha_{\text{trsm}} \log_2(Q) NNB + \gamma_{\text{trsm}} \frac{N^2 NB}{2Q}$$

Step 4: updating trailing submatrix

This step consists in a parallel matrix multiplication:

- broadcasting of the rows,
- broadcasting of he columns,
- local multiplication of the blocks.

For panel k the cost is:

$$\begin{aligned} & \log_2(P)(\beta_{\text{mm}} + \alpha_{\text{mm}} NB\lceil (N-(k-1)NB)/Q\rceil) \\ + \quad & \log_2(Q)(\beta_{\text{mm}} + \alpha_{\text{mm}} NB\lceil (N-kNB)/P\rceil) \\ + \quad & 2\gamma_{\text{mm}} NB\lceil (N-kNB)/P\rceil\lceil (N-kNB)/Q\rceil \end{aligned}$$

Hence, for the whole factorization the approximate cost is:

$$\begin{aligned} & \beta_{\text{mm}} \frac{N}{NB}(\log_2(Q)+\log_2(P)) \\ + \quad & \alpha_{\text{mm}} \left(\log_2(Q)\frac{N^2}{2P} \log_2(P)(\frac{N^2}{2Q} + \frac{NNB}{Q}) \right) + \gamma_{\text{mm}} \frac{2N^3}{3NP} \end{aligned}$$

Extending the model for SMP nodes

The p processors of a given SMP node can be arranged:

1. in a row-wise manner : $P \longrightarrow P/p$, for counting the number of messages.
2. in a column-wise manner $Q \longrightarrow Q/p$, for counting the number of messages.

Parameters evaluation: time spent in each step

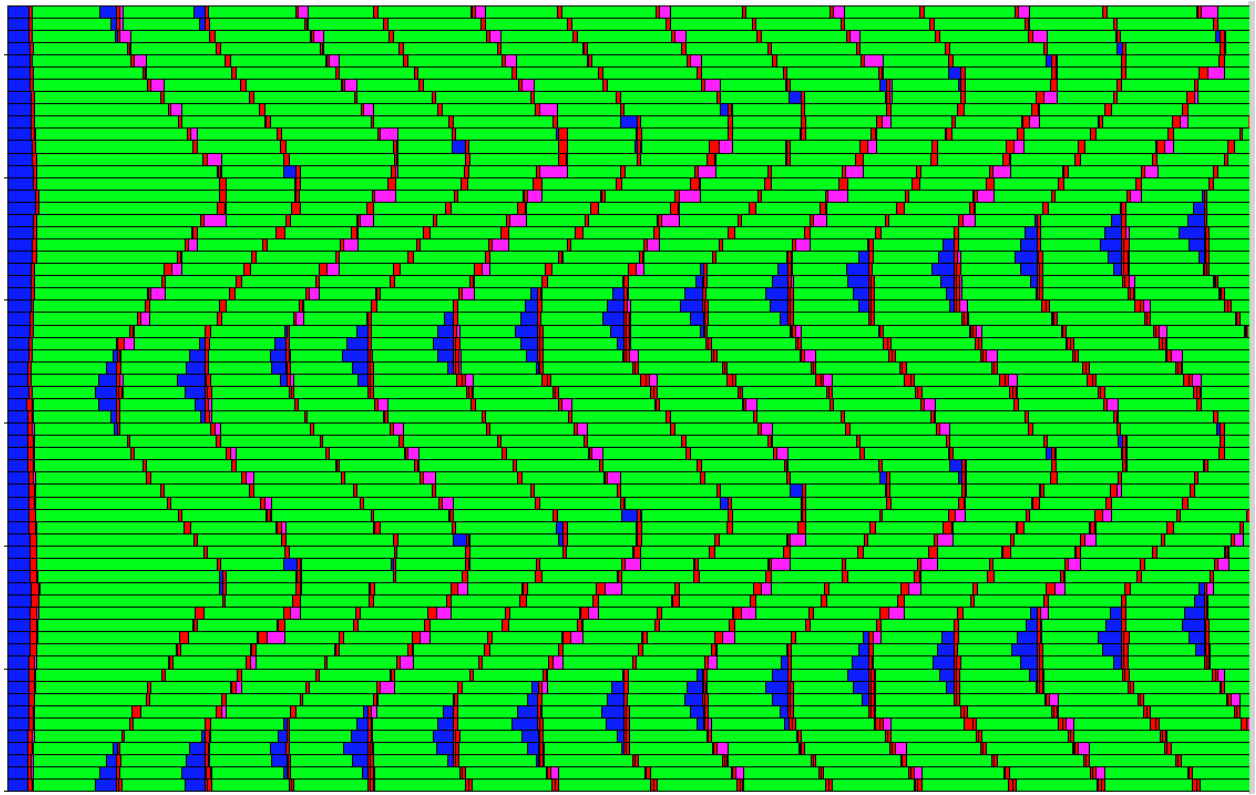
Strategy :

1. Put timers for each routine implementing a step
2. Run the LU factorization
3. Store the timing of each step for each panel and for each processor.
4. Sum the timings for computing the amount of time is spent in each step.
5. Fit the parameters with the timings: least square method

Summing the timings

The computation is not synchronized. Summing the timing is tricky.

Example: $P = 2$, $Q = 32$, $N = 1000$ and $NB = 32$.



Fitting the parameters

For each run:

- Build a matrix where is stored the value of the coefficient of the parameters
- Build a vector where you store the timing
- apply a least square method to compute the model parameters

Template based modeling : example

$$t = \alpha_1 m + \alpha_2 n + \alpha_3 k + \alpha_4 mn + \alpha_5 mk + \alpha_6 nk + \alpha_7 mnk$$

$$m = 4601 \; n = 2417 \; k = 4563 \; \Rightarrow \text{est. t} = -1$$

$$\text{wall time} = 97.434$$

$$A = \begin{pmatrix} 4601 & 2417 & 4563 & 11.12 \cdot 10^6 & 20.99 \cdot 10^6 & 11.03 \cdot 10^6 & 50.74 \cdot 10^9 \end{pmatrix} x = \begin{pmatrix} 97.434 \end{pmatrix}$$

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	0	0	0	0	0	$1.9 \cdot 10^{-9}$

Template based modeling : example

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	0	0	0	0	0	$1.9 \cdot 10^{-9}$

$$t = \alpha_1 m + \alpha_2 n + \alpha_3 k + \alpha_4 mn + \alpha_5 mk + \alpha_6 nk + \alpha_7 mnk$$

$$m = 2695 \; n = 2479 \; k = 2295 \; \Rightarrow \text{est. t} = 29.441$$

$$\text{wall time} = 29.123$$

$$A = \begin{pmatrix} 4601 & 2417 & 4563 & 11.12 \cdot 10^6 & 20.99 \cdot 10^6 & 11.03 \cdot 10^6 & 50.74 \cdot 10^9 \\ 2695 & 2479 & 2295 & 6.68 \cdot 10^6 & 6.19 \cdot 10^6 & 5.69 \cdot 10^6 & 15.33 \cdot 10^9 \end{pmatrix} x = \begin{pmatrix} 97.434 \\ 29.123 \end{pmatrix}$$

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	0	0	0	$2.0 \cdot 10^{-6}$	0	$1.1 \cdot 10^{-9}$

Template based modeling : example

$$\begin{array}{cccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \\ 0 & 0 & 0 & 0 & 2.0 \cdot 10^{-6} & 0 & 1.1 \cdot 10^{-9} \end{array}$$

$$t = \alpha_1 m + \alpha_2 n + \alpha_3 k + \alpha_4 mn + \alpha_5 mk + \alpha_6 nk + \alpha_7 mnk$$

$$m = 1692 \; n = 2432 \; k = 2256 \; \Rightarrow \text{est. t} = 22.442$$

$$\text{wall time} = 23.663$$

$$A = \begin{pmatrix} 4601 & 2417 & 4563 & 11.12 \cdot 10^6 & 20.99 \cdot 10^6 & 11.03 \cdot 10^6 & 50.74 \cdot 10^9 \\ 2695 & 2479 & 2295 & 6.68 \cdot 10^6 & 6.19 \cdot 10^6 & 5.69 \cdot 10^6 & 15.33 \cdot 10^9 \\ 1962 & 2432 & 2456 & 4.77 \cdot 10^6 & 4.16 \cdot 10^6 & 5.97 \cdot 10^6 & 11.72 \cdot 10^9 \end{pmatrix} x = \begin{pmatrix} 97.434 \\ 29.123 \\ 23.673 \end{pmatrix}$$

$$\begin{array}{ccccccc} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \\ 0 & 0 & 7.9 \cdot 10^{-4} & 0 & 4.5 \cdot 10^{-6} & 0 & 0 \end{array}$$

Template based modeling : example

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	0	$7.9 \cdot 10^{-4}$	0	$4.5 \cdot 10^{-6}$	0	0

$$t = \alpha_1 m + \alpha_2 n + \alpha_3 k + \alpha_4 mn + \alpha_5 mk + \alpha_6 nk + \alpha_7 mnk$$

$$m = 4245 \; n = 1837 \; k = 4703 \; \Rightarrow \text{est. t} = 92.897$$

$$\text{wall time} = 70.978$$

$$A = \begin{pmatrix} 4601 & 2417 & 4563 & 11.12 \cdot 10^6 & 20.99 \cdot 10^6 & 11.03 \cdot 10^6 & 50.74 \cdot 10^9 \\ 2695 & 2479 & 2295 & 6.68 \cdot 10^6 & 6.19 \cdot 10^6 & 5.69 \cdot 10^6 & 15.33 \cdot 10^9 \\ 1962 & 2432 & 2456 & 4.77 \cdot 10^6 & 4.16 \cdot 10^6 & 5.97 \cdot 10^6 & 11.72 \cdot 10^9 \\ 4245 & 1837 & 4703 & 7.80 \cdot 10^6 & 19.96 \cdot 10^6 & 8.64 \cdot 10^6 & 36.67 \cdot 10^9 \end{pmatrix} x = \begin{pmatrix} 97.434 \\ 29.123 \\ 23.673 \\ 70.978 \end{pmatrix}$$

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	0	$4.8 \cdot 10^{-4}$	0	0	$4.2 \cdot 10^{-9}$	$1.9 \cdot 10^{-9}$

Template based modeling : example

$$t = \alpha_1 m + \alpha_2 n + \alpha_3 k + \alpha_4 mn + \alpha_5 mk + \alpha_6 nk + \alpha_7 mnk$$

$$A = \begin{pmatrix} 4601 & 2417 & 4563 & 11.12 \cdot 10^6 & 20.99 \cdot 10^6 & 11.03 \cdot 10^6 & 50.74 \cdot 10^9 \\ 2695 & 2479 & 2295 & 6.68 \cdot 10^6 & 6.19 \cdot 10^6 & 5.69 \cdot 10^6 & 15.33 \cdot 10^9 \\ 1962 & 2432 & 2456 & 4.77 \cdot 10^6 & 4.16 \cdot 10^6 & 5.97 \cdot 10^6 & 11.72 \cdot 10^9 \\ 4245 & 1837 & 4703 & 7.80 \cdot 10^6 & 19.96 \cdot 10^6 & 8.64 \cdot 10^6 & 36.67 \cdot 10^9 \\ 1565 & 2271 & 1216 & 3.55 \cdot 10^6 & 1.90 \cdot 10^6 & 2.76 \cdot 10^6 & 4.32 \cdot 10^9 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 4227 & 3347 & 2752 & 14.15 \cdot 10^6 & 11.63 \cdot 10^6 & 9.2 \cdot 10^6 & 38.93 \cdot 10^9 \end{pmatrix} \quad x = \begin{pmatrix} 97.434 \\ 29.123 \\ 23.673 \\ 70.978 \\ 8.760 \\ \vdots \\ 74.819 \end{pmatrix}$$

α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	$2.0 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	0	$4.6 \cdot 10^{-9}$	$1.8 \cdot 10^{-7}$	$1.9 \cdot 10^{-9}$

$$m = 4245 \; n = 1837 \; k = 4703 \; \Rightarrow \text{est. t} = 70.759$$

Limiting the number of parameters

For LU, in theory 3 runs are sufficient (one for each parameters).

In practice with less than 6 runs, we obtain good results
$\Rightarrow$ Less than one hour to compute the parameter of the model of a given machine
$\Rightarrow$ Easy to build automatically the model at installation time

Experimental results

- 48 dual-processors, AMD Opteron 64 bits, Giga ethernet interconnect
- ScaLapack.
- Comparing with other models
- Predicting the block size
- Computing the grid-shape

Comparing with other models

Several models: Scalapack Users Guide (*SLUG*):

- crude model,
- shows the scalability of ScaLapack,
- Do not use the processor grid shape.

Lapack working note 43 (Dongarra, van de Geijn, Walker) – (*LAWN43*):

- simple model with 3 parameters:
  - the bandwidth of the network,
  - the latency of the network,
  - and the flop rate of the processor
- the value of these parameters does not depend on the subroutine of the algorithm.

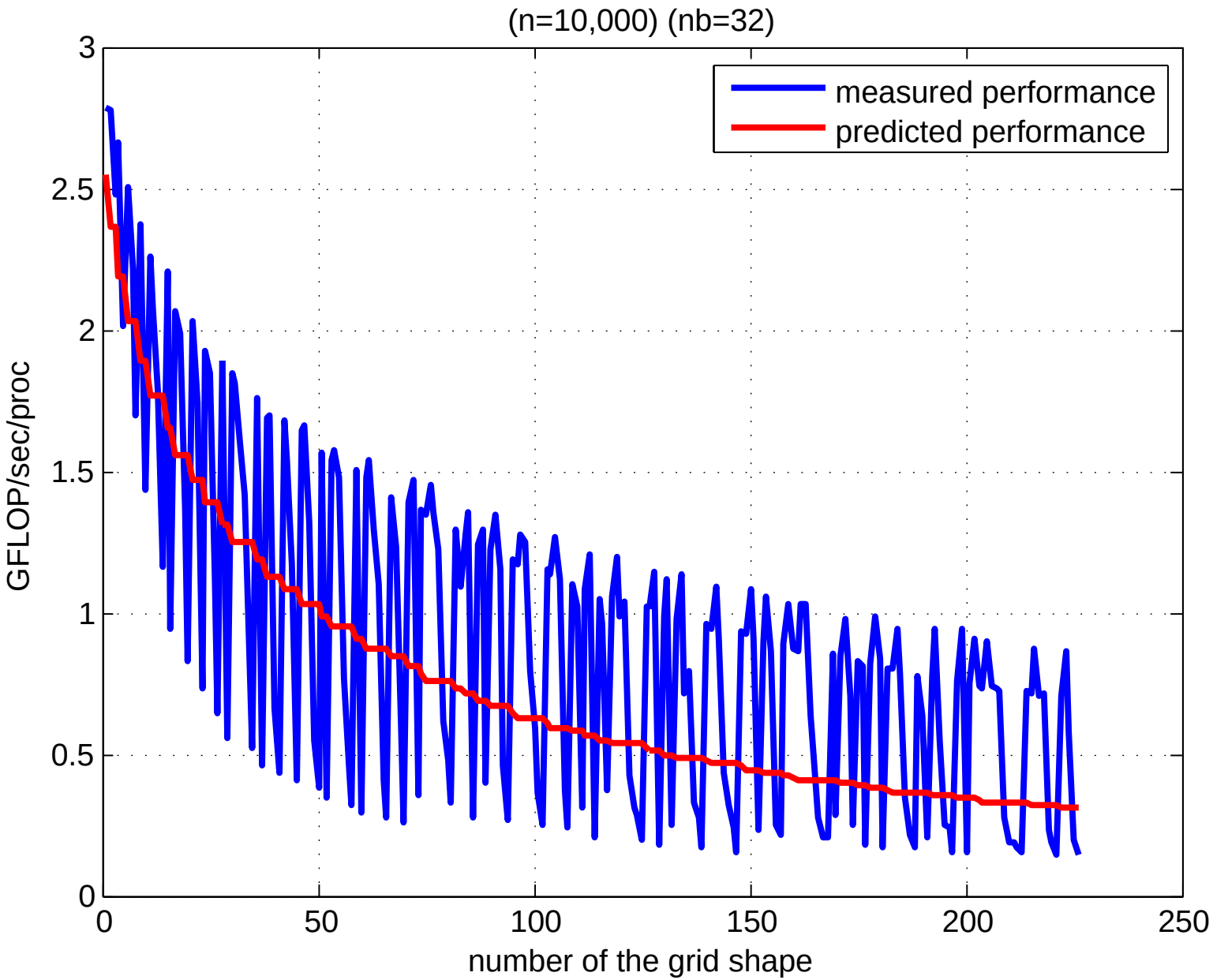
The new model (*NEW*):

- Extension of the LAWN43.
- 12 parameters (one per sub step)

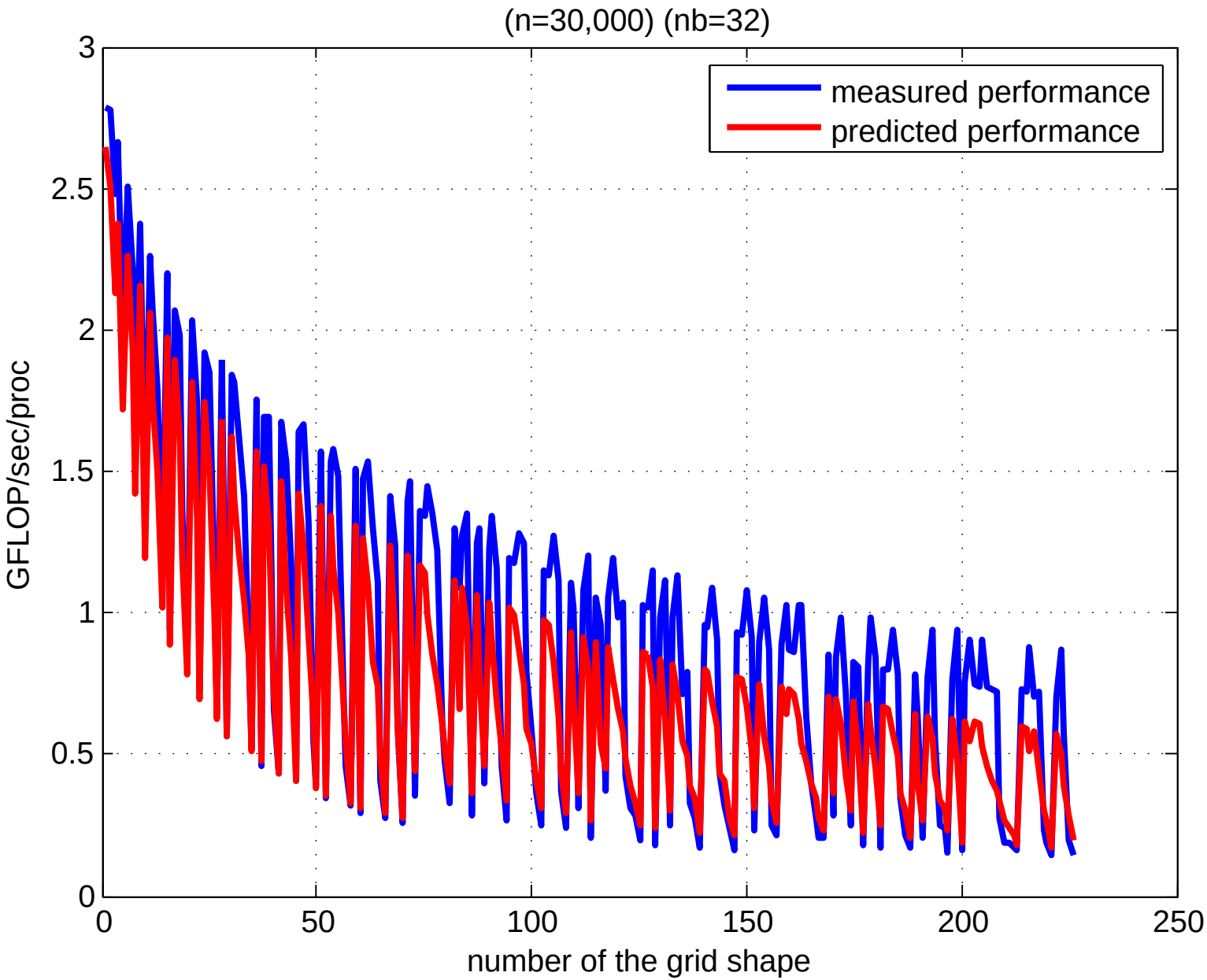
Question: how do they compare to each other?

Methodology: We compute the parameters through linear regression of the data on a parallel computers

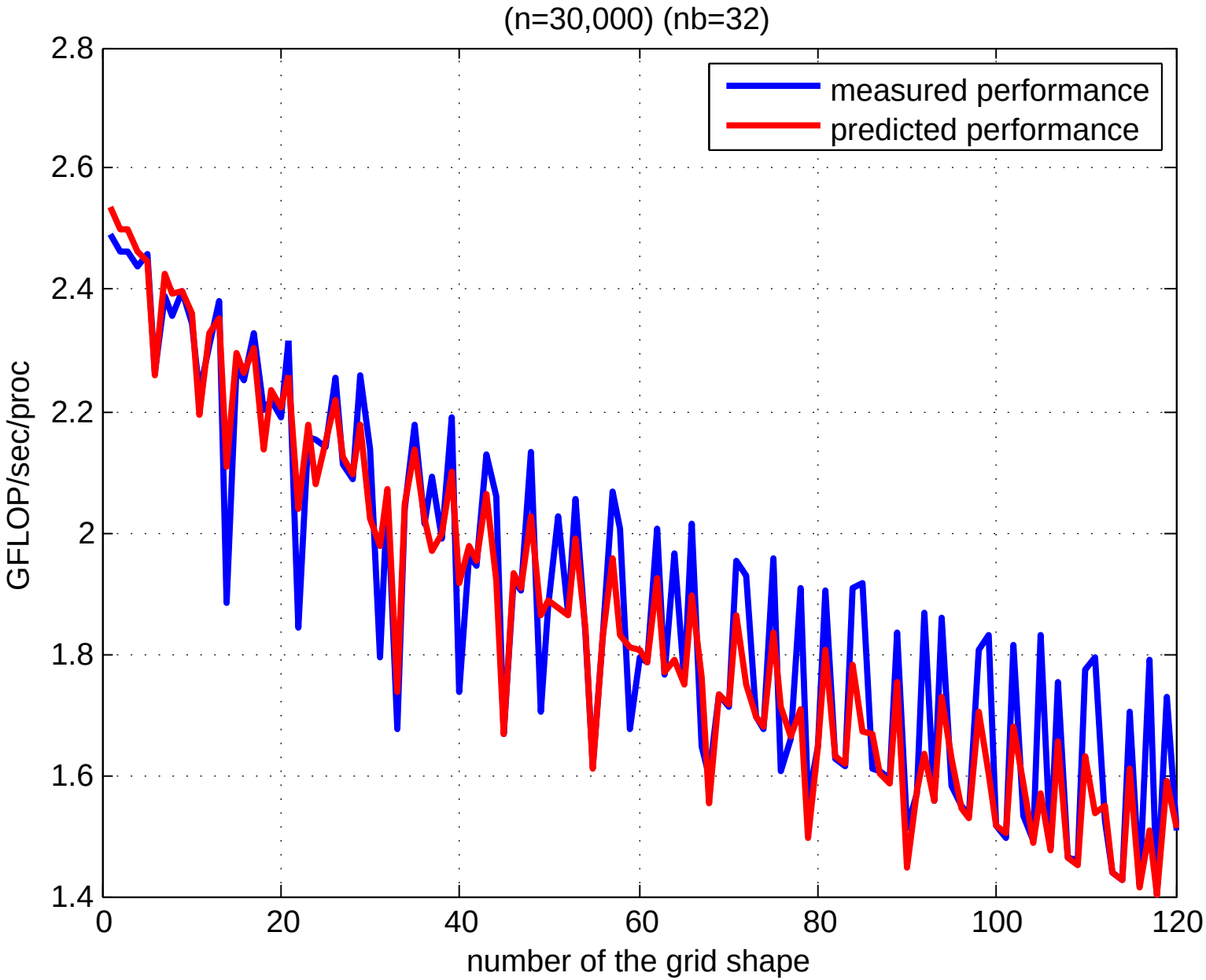
Gigaglops/proc (SLUG)



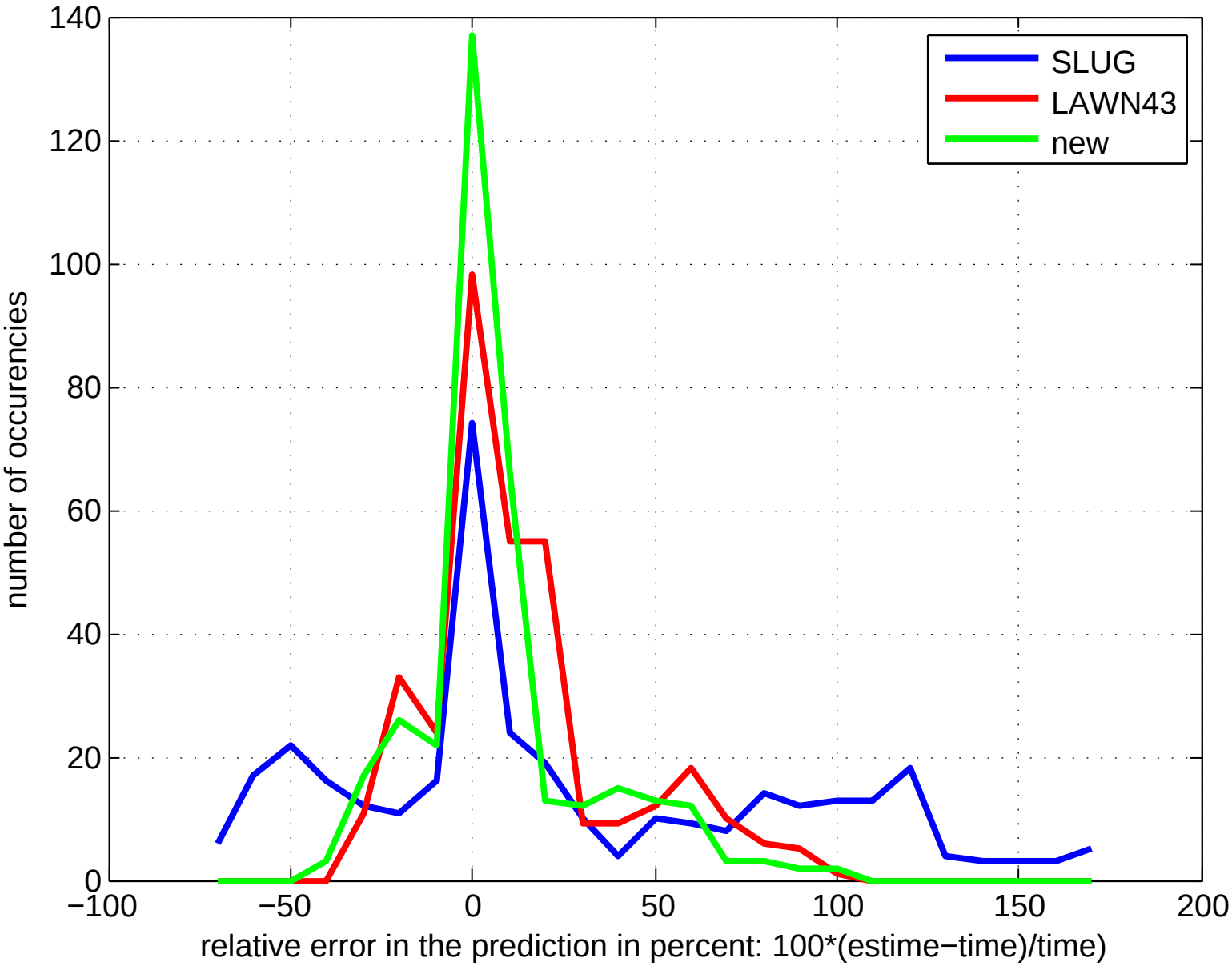
Gigaglops/proc (LAWN43)



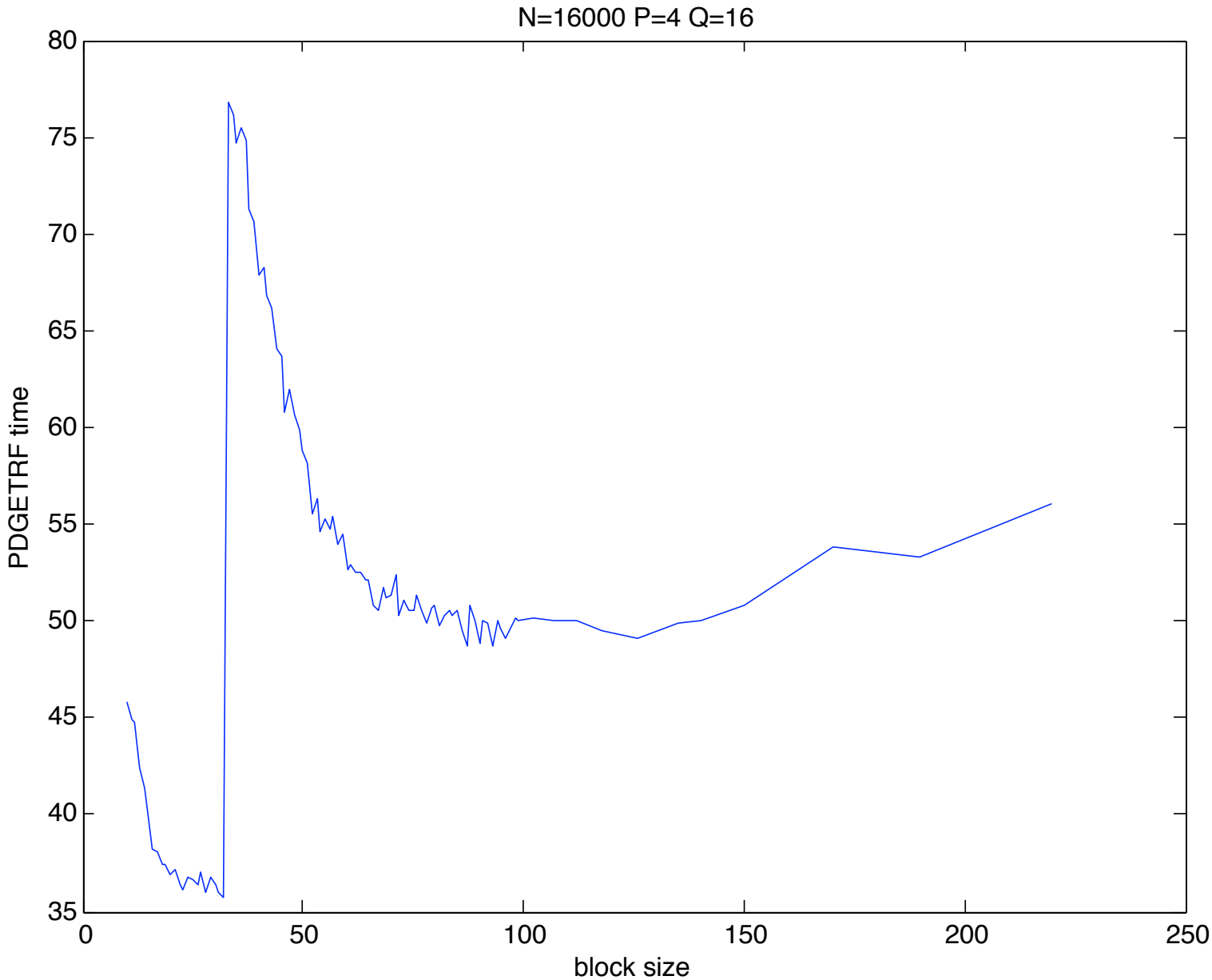
Gigaglops/proc (NEW)



Comparison of the models



Runtime Vs. Block size



Predicting the grid shape: $N = 30000$ and $P \leq 64$

NP	bestP	bestQ	time	estP	estQ	time	diff	error
12	2	6	609.956	1	12	615.227	-5.27	-0.86%
14	2	7	536.245	1	14	545.136	-8.89	-1.63%
16	2	8	472.505	1	16	487.092	-14.59	-2.99%
21	2	10	388.700	1	21	397.544	-8.84	-2.22%
23	2	11	362.586	1	23	370.557	-7.97	-2.15%
39	3	13	235.064	2	19	236.332	-1.27	-0.54%
48	3	16	195.734	2	24	196.408	-0.67	-0.34%
49	3	16	195.734	2	24	196.408	-0.67	-0.34%
50	3	16	195.734	2	25	196.212	-0.48	-0.24%
51	3	17	188.982	2	25	196.212	-7.23	-3.68%
54	3	18	182.037	2	27	184.544	-2.51	-1.36%
55	3	18	182.037	2	27	184.544	-2.51	-1.36%
57	3	19	172.508	2	28	177.105	-4.60	-2.60%
58	3	19	172.508	2	29	177.021	-4.51	-2.55%
59	3	19	172.508	2	29	177.021	-4.51	-2.55%
60	3	20	167.131	2	30	169.264	-2.13	-1.26%
61	3	20	167.131	2	30	169.264	-2.13	-1.26%
62	3	20	167.131	2	31	170.350	-3.22	-1.89%
63	3	21	159.817	2	31	170.350	-10.53	-6.18%
64	3	21	159.817	2	32	162.626	-2.81	-1.73%

Find optimal grid-size in 68% of the cases.

Worst case: 6.18%..

Conclusion

- Provided a model for LU factorization
- Presented how to compute the parameters
- Results show that the model is precise and is guess very good grid-shape