Characterizing the relationship between ILU-type preconditioners and the storage hierarchy

Co-authored by:

D. Rivera [1]

D. Kaeli [2]

M. Kilmer [3]

ILU-type preconditioning techniques are widely recognized as being an extremely effective approach to providing efficient solvers[1]. These techniques have been used to increase the performance and reliability of Krylov subspace methods. However, a drawback of these approaches is that it is difficult to choose appropriate values for the preconditioner tuning parameters[2]. Usually, parameter selection is done through trial-and-error for a few sample matrices for a given application.

In our work we have found that the performance of these techniques and methods also depends upon the relationship between the preconditioner tuning parameters and the memory hierarchy of the machine used to carry out the computation. The parameter values used to obtain the fastest execution time, given an acceptable final error, may be different for different memory hierarchies. This occurs due to 1) the non-zero structure of the new coefficient matrix depending on the tuning parameter values and 2) the ability of the memory hierarchy to exploit the locality present in the new matrix.

The difference in performance on different memory hierarchies becomes significant when the problem's conditions make it more difficult to solve. These conditions are related to the dropping strategy adopted in the preconditioner algorithm. For example, the relation between the numerical symmetry[4] and the bandwidth (NS/B) of the coefficient matrix allows us to estimate how difficult it will be to solve the problem using the $ILUT$ preconditioner. This is because the dropping strategy for the $ILUT$ preconditioner is based on dropping elements in the Gaussian Elimination process according to their magnitude. The results shown in Figure 1 support the previous analysis. These graphs show the final error obtained by the first thirteen duples[5], ordered in increasing order by the overall execution time (i.e. the time until preconditioned GMRES reaches the tolerance). The convergence criterion is based on the residual norm; GMRES stops iterating when the relative residual norm is below a set value.

Our experiments were run on a 750MHz Sun Ultra Sparc-III system (L1D 64KB 4way, L2 8MB 2way, 1 GB RAM), and on a 3.06GHz Intel XEON system (L1D 8KB 4way, L2 512 KB 8way, L3 1 MB 8way, 2 GB RAM).

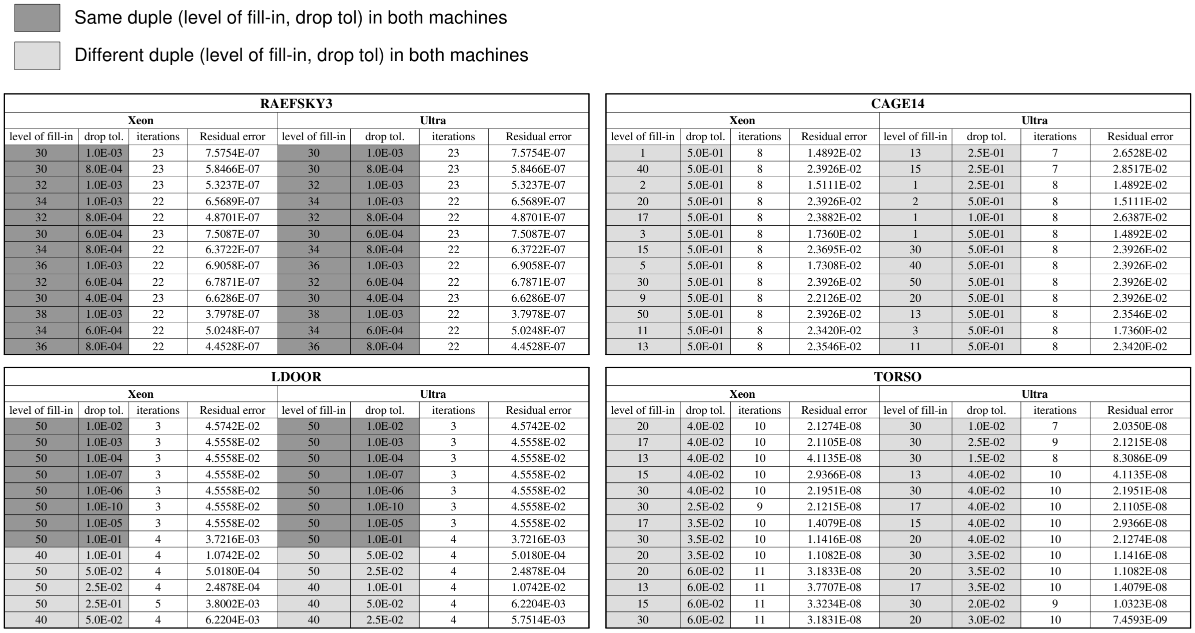
[1] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA

[2] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA

[3] Department of Mathematics, Tufts University, Medford, MA

[4] Numerical symmetry is computed as the rate between number of maches where $a_{i,j} = a_{j,i}$ *with* $i \neq j$ and the total number of offdiagonal entries

[5] A duple is a set of two parameters values, the first one specifies the level of fill-in and the other one the drop tolerance



▩ Same duple (level of fill-in, drop tol) in both machines

░ Different duple (level of fill-in, drop tol) in both machines

RAEFSKY3							
Xeon				Ultra			
level of fill-in	drop tol.	iterations	Residual error	level of fill-in	drop tol.	iterations	Residual error
30	1.0E-03	23	7.5754E-07	30	1.0E-03	23	7.5754E-07
30	8.0E-04	23	5.8466E-07	30	8.0E-04	23	5.8466E-07
32	1.0E-03	23	5.3237E-07	32	1.0E-03	23	5.3237E-07
34	1.0E-03	22	6.5689E-07	34	1.0E-03	22	6.5689E-07
32	8.0E-04	22	4.8701E-07	32	8.0E-04	22	4.8701E-07
30	6.0E-04	23	7.5087E-07	30	6.0E-04	23	7.5087E-07
34	8.0E-04	22	6.3722E-07	34	8.0E-04	22	6.3722E-07
36	1.0E-03	22	6.9058E-07	36	1.0E-03	22	6.9058E-07
32	6.0E-04	22	6.7871E-07	32	6.0E-04	22	6.7871E-07
30	4.0E-04	23	6.6286E-07	30	4.0E-04	23	6.6286E-07
38	1.0E-03	22	3.7978E-07	38	1.0E-03	22	3.7978E-07
34	6.0E-04	22	5.0248E-07	34	6.0E-04	22	5.0248E-07
36	8.0E-04	22	4.4528E-07	36	8.0E-04	22	4.4528E-07

CAGE14							
Xeon				Ultra			
level of fill-in	drop tol.	iterations	Residual error	level of fill-in	drop tol.	iterations	Residual error
1	5.0E-01	8	1.4892E-02	13	2.5E-01	7	2.6528E-02
40	5.0E-01	8	2.3926E-02	15	2.5E-01	7	2.8517E-02
2	5.0E-01	8	1.5111E-02	1	2.5E-01	8	1.4892E-02
20	5.0E-01	8	2.3926E-02	2	5.0E-01	8	1.5111E-02
17	5.0E-01	8	2.3882E-02	1	1.0E-01	8	2.6387E-02
3	5.0E-01	8	1.7360E-02	1	5.0E-01	8	1.4892E-02
15	5.0E-01	8	2.3695E-02	30	5.0E-01	8	2.3926E-02
5	5.0E-01	8	1.7308E-02	40	5.0E-01	8	2.3926E-02
30	5.0E-01	8	2.3926E-02	50	5.0E-01	8	2.3926E-02
9	5.0E-01	8	2.2126E-02	20	5.0E-01	8	2.3926E-02
50	5.0E-01	8	2.3926E-02	13	5.0E-01	8	2.3546E-02
11	5.0E-01	8	2.3420E-02	3	5.0E-01	8	1.7360E-02
13	5.0E-01	8	2.3546E-02	11	5.0E-01	8	2.3420E-02

LDOOR							
Xeon				Ultra			
level of fill-in	drop tol.	iterations	Residual error	level of fill-in	drop tol.	iterations	Residual error
50	1.0E-02	3	4.5742E-02	50	1.0E-02	3	4.5742E-02
50	1.0E-03	3	4.5558E-02	50	1.0E-03	3	4.5558E-02
50	1.0E-04	3	4.5558E-02	50	1.0E-04	3	4.5558E-02
50	1.0E-07	3	4.5558E-02	50	1.0E-07	3	4.5558E-02
50	1.0E-06	3	4.5558E-02	50	1.0E-06	3	4.5558E-02
50	1.0E-10	3	4.5558E-02	50	1.0E-10	3	4.5558E-02
50	1.0E-05	3	4.5558E-02	50	1.0E-05	3	4.5558E-02
50	1.0E-01	4	3.7216E-03	50	1.0E-01	4	3.7216E-03
40	1.0E-01	4	1.0742E-02	50	5.0E-02	4	5.0180E-04
50	5.0E-02	4	5.0180E-04	50	2.5E-02	4	2.4878E-04
50	2.5E-02	4	2.4878E-04	40	1.0E-01	4	1.0742E-02
50	2.5E-01	5	3.8002E-03	40	5.0E-02	4	6.2204E-03
40	5.0E-02	4	6.2204E-03	40	2.5E-02	4	5.7514E-03

TORSO							
Xeon				Ultra			
level of fill-in	drop tol.	iterations	Residual error	level of fill-in	drop tol.	iterations	Residual error
20	4.0E-02	10	2.1274E-08	30	1.0E-02	7	2.0350E-08
17	4.0E-02	10	2.1105E-08	30	2.5E-02	9	2.1215E-08
13	4.0E-02	10	4.1135E-08	30	1.5E-02	8	8.3086E-09
15	4.0E-02	10	2.9366E-08	13	4.0E-02	10	4.1135E-08
30	4.0E-02	10	2.1951E-08	30	4.0E-02	10	2.1951E-08
30	2.5E-02	9	2.1215E-08	17	4.0E-02	10	2.1105E-08
17	3.5E-02	10	1.4079E-08	15	4.0E-02	10	2.9366E-08
30	3.5E-02	10	1.1416E-08	20	4.0E-02	10	2.1274E-08
20	3.5E-02	10	1.1082E-08	30	3.5E-02	10	1.1416E-08
20	6.0E-02	11	3.1833E-08	20	3.5E-02	10	1.1082E-08
13	6.0E-02	11	3.7707E-08	17	3.5E-02	10	1.4079E-08
15	6.0E-02	11	3.3234E-08	30	2.0E-02	9	1.0323E-08
30	6.0E-02	11	3.1831E-08	20	3.0E-02	10	7.4593E-09

Figure 1: Error norm vs. duples ordered by minimum execution time

Name	Non-zero elements	Rows	NS	B	NS/B
Raefsky3	1,488,768	21,200	48%	0.0596	8.05
Ldoor	42,493,817	952,203	100%	0.7215	1.39
Cage14	27,130,349	1,505,785	21%	0.4490	0.47
Torso3	4,429,042	259,156	0%	0.8191	0

Table 1: Description of matrices evaluated

As shown in Table 1 and Figure 1, the difference between these machines in terms of performance and the parameter values turns out to be significant when the rate between the numerical symmetry and the bandwidth decreases.

To illustrate that the overall execution time can be reduced because the memory hierarchy has the ability to exploit the locality in the new matrix, we use the *PIN* tool to capture cache events[5]. Our results show a high correlation among the execution time, memory accesses and cache misses.

We show that on one memory hierarchy, a greater level of fill-in can be used than on other hierarchies. For instance, the fastest execution time on the Ultra Sparc-III system was obtained for the duple(30, 0.01), whereas the fastest execution time on the Intel XEON system was obtained for the duple(20, 0.04).

We developed an algorithmic approach to: 1) extract the problem's conditions related to the dropping strategies adopted in the preconditioner, 2) detect if the computation of a solution depends upon the relationship between the preconditioner's parameters and the memory hierarchy of the machine used and 3) suggest values of the preconditioner's parameters which can help to reduce the time required to compute the preconditioner and the solution for matrices with similar characteristics.

We evaluated over 110 matrices from the MatrixMarket[3] and University of Florida[4] repositories. In addition to these, we created several slightly different matrices by adding random values with a controlled standard deviation and by zeroing several of the non-zero elements. This gave us a complete set of 200 matrices to be tested. The ILU-type preconditioning algorithms included in the SPARSKIT[6] library were used together with restarted GMRES with restart values equal to 30 and 40.

Our experimental results show that 78.4% of the time, the suggested values of the preconditioner's parameters were appropriate in reducing the overall execution time.

We will explore more sophisticated heuristics for our algorithmic approach in order to increase the percentage of suggested values of the preconditioner's parameters appropriated. In addition, we will extend our study to multilevel preconditioners based on ILU factorization.

Acknowledgment

Many thanks to Michele Benzi for his help and insights during the early stages of this work.

This project is supported by the National Science Foundations Computing and Communication Foundations Division, grant number CCF-0342555 and the Institute of Complex Scientific Software.

References

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[2] Michele Benzi. Preconditioning techniques for large linear systems:a survey. *J. Comput. Phys.*, 182(2):418–477, 2002.

[3] Matrix Market. http://math.nist.gov/MatrixMarket.

[4] University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices/.

[5] Pin. http://rogue.colorado.edu/Pin/index.html.

[6] Y. Saad. *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation.* http://www-users.cs.umn.edu/saad/software/SPARSKIT/sparskit.html.