Online Appendix to: Algorithm 953: Parallel Library Software for the Multishift QR Algorithm with Aggressive Early Deflation —Electronic Appendix: Derivation of the Performance Model

ROBERT GRANAT and BO KÅGSTRÖM, Umeå University
DANIEL KRESSNER, EPF Lausanne
MEIYUE SHAO, Umeå University and EPF Lausanne

1. ESTIMATING T_{SWEEP}

The QR sweep is relatively simple because the computation and communication cost is well determined by n, n_{shift}, and p. Usually there are up to $\sqrt{p}$ simultaneous computational windows, one at each diagonal processor in the grid, with at most $n_b/3$ shifts in each window. If $n_{\text{shift}} > \sqrt{p}\,n_b/3$, these shifts are chased in several rounds. So we use a rough approximation $n^*_{\text{shift}} = \sqrt{p}\,n_b/3$ to represent the total amount of shifts that can be chased simultaneously in the QR sweep. Based on the assumption $\sqrt{p}\,n_b \ll n$, the overhead for the start-up and ending phases of the bulge chasing are not important. Therefore, the cost of one QR sweep is roughly

$$T_{\text{sweep}}(n, n_{\text{shift}}, p) = \frac{n_{\text{shift}} n}{n^*_{\text{shift}} n_b}(T_{\text{local}} + T_{\text{cross}}),$$

where T_{local} and T_{cross} represent the runtime for local and crossborder bulge chasing, respectively. Both parts require chasing the chain of bulges with $n_b/2$ steps inside the computational window, as well as updating the corresponding off-diagonal blocks. Hence, the runtime for arithmetic operations is $(4n_b^3 + 4n\,n_b^2/\sqrt{p})\gamma$, half of which is for accumulating the orthogonal matrix Q. The only communication cost in the local chasing phase is broadcasting the accumulated orthogonal matrix rowwise and columnwise in the processor grid, which requires $\log_2 p(\alpha + n_b^2\,\beta)$ runtime, that is,

$$T_{\text{local}} = \left(4n_b^3 + \frac{4n\,n_b^2}{\sqrt{p}}\right)\gamma + \log_2 p(\alpha + n_b^2\,\beta) \approx \frac{4n\,n_b^2}{\sqrt{p}}\gamma + \log_2 p(\alpha + n_b^2\,\beta).$$

One round crossborder chasing requires at least the same amount of communication as in one local chasing step, with some extra cost for explicitly forming the $n_b \times n_b$ computational window and exchanging data with processor neighbors for updating the off-diagonal blocks. Notice that usually there are two rounds for a crossborder chasing step; therefore, we have

$$T_{\text{cross}} = 2\left[T_{\text{local}} + 3\left(\alpha + \frac{n_b^2}{4}\beta\right) + 3\left(\alpha + \frac{n n_b}{2\sqrt{p}}\beta\right)\right],$$

and then

$$\begin{aligned} T_{\text{sweep}}(n, n_{\text{shift}}, p) &\approx \frac{12n^2 n_{\text{shift}} n_b}{\sqrt{p}\,n^*_{\text{shift}}}\gamma + \frac{3n\,n_{\text{shift}}}{n_b n^*_{\text{shift}}}(\log_2 p + 4)\alpha + \frac{3n^2 n_{\text{shift}}}{\sqrt{p}\,n^*_{\text{shift}}}\beta \\ &= \frac{36n^2 n_{\text{shift}}}{p}\gamma + \frac{9n\,n_{\text{shift}}}{\sqrt{p}\,n_b^2}(\log_2 p + 4)\alpha + \frac{9n^2 n_{\text{shift}}}{p n_b}\beta. \end{aligned}$$

DOI: http://dx.doi.org/10.1145/2699471

From this model, we can see that the cost for updating the off-diagonal blocks dominates in both the computation and communication parts, under the assumption that $\sqrt{p}\,n_b \ll n$ (or equivalently $n^*_{\text{shift}} \ll n$). As a byproduct, the performance model of a plain multishift QR algorithm without AED can also be obtained. By assuming the convergence rate as $\Theta(1)$ shifts per eigenvalue, that is, $k_{\text{sweep}} = \Theta(n/n_{\text{shift}})$ and neglecting the cost for generating shifts, the total execution time of a plain multishift QR algorithm is

$$T_{\text{new}}(n,\, p) = \Theta\left(\frac{n^3}{p}\right)\gamma + \Theta\left(\frac{n^2 \log p}{\sqrt{p}\,n_b^2}\right)\alpha + \Theta\left(\frac{n^3}{p n_b}\right)\beta.$$

Fixing the memory load per processor (that is, $n/\sqrt{p} = \text{constant}$) yields

$$T_{\text{new}}(n,\, p) = \Theta(n)\,\gamma + \Theta(n \log n)\,\alpha + \Theta(n)\,\beta.$$

2. ESTIMATING T_{AED} and T_{SHIFT}

The execution time for one step AED is modeled as

$$\begin{aligned} T_{\text{AED}}(n, n_{\text{AED}}, p) = {} & T_{\text{redist}}(n_{\text{AED}}, p, p_{\text{AED}}) + T_{\text{pipe}}(n_{\text{AED}}, p_{\text{AED}}) \\ & + T_{\text{reorder}}(n_{\text{AED}}, p) + T_{\text{Hess}}(n_{\text{AED}}, p) + T_{\text{update}}(n, n_{\text{AED}}, p), \end{aligned}$$

where the terms in the right-hand side represent the runtime for data redistribution, Schur decomposition of the AED window, deflation checking and reordering of eigenvalues, Hessenberg reduction, and updating the off-diagonal blocks corresponding to the AED window, respectively. We estimate these terms one by one using the hierarchical approach in Dackland and Kågström [1996].

—T_{redist}: The general-purpose data redistribution routine `PDGEMR2D` in ScaLAPACK uses the algorithm described in Prylli and Tourancheau [1997]. Since the scheduling part is tiny compared to the communication part, the complexity of data redistribution is provided [Prylli and Tourancheau 1997] as

$$T_{\text{redist}}(n_{\text{AED}}, p, p_{\text{AED}}) = \Theta(p)\,\alpha + \Theta\left(\frac{n_{\text{AED}}^2}{\sqrt{p\,p_{\text{AED}}}}\right)\beta.$$

—T_{pipe}: The complexity of the Schur decomposition performed by `PDLAQR1` largely depends on the property of the matrix, since AED affects the convergence rate significantly. To obtain an estimate of the complexity, we assume that AED roughly reduces the number of pipelined QR sweeps by half. According to the experimental results presented in Kågström et al. [2012], this assumption usually provides a reasonable upper bound of the runtime, although it can be overestimated. Using the model in Henry et al. [2002], we obtain an approximate execution time

$$\begin{aligned} T_{\text{pipe}}(n,\, p) &= \frac{20n^3}{p}\gamma + \frac{3n^2}{\sqrt{p}\,n_b}(\log_2 p + 2)\alpha + \left(\frac{3n^2 \log_2 p}{\sqrt{p}} + \frac{8n^3}{p n_b}\right)\beta \qquad (1) \\ &= \Theta\left(\frac{n^3}{p}\right)\gamma + \Theta\left(\frac{n^2 \log p}{\sqrt{p}\,n_b}\right)\alpha + \Theta\left(\frac{n^3}{p n_b}\right)\beta. \end{aligned}$$

If the orthogonal matrix Q is not accumulated in the calculation, the arithmetic operations are roughly halved, that is,

$$\widetilde{T}_{\text{pipe}}(n,\, p) = \frac{10n^3}{p}\gamma + \frac{3n^2}{\sqrt{p}\,n_b}(\log_2 p + 2)\alpha + \left(\frac{3n^2 \log_2 p}{\sqrt{p}} + \frac{8n^3}{p n_b}\right)\beta.$$

The model provided in Blackford et al. [1997] is similar, but with slightly different coefficients.

—T_{reorder}: Obviously, the cost for eigenvalue reordering depends on the deflation ratio. However, we can evaluate an upper bound for the cost—all eigenvalues are involved in the reordering. Then the performance model is almost the same as that of QR sweeps, since updating the off-diagonal blocks is the dominant operation. Notice that each eigenvalue needs to move $n_{\text{AED}}/2$ steps in average, so the overall cost for eigenvalue reordering inside the AED window is bounded by

$$T_{\text{reorder}}(n_{\text{AED}}, p) \approx \frac{4n_{\text{AED}}^2 n_b}{\sqrt{p}}\gamma + \frac{2n_{\text{AED}}}{n_b}(\log_2 p + 3)\alpha + \frac{3n_{\text{AED}}^2}{2\sqrt{p}}\beta.$$

As a different feature compared to QR sweeps or the performance model in Granat et al. [2009] for parallel eigenvalue reordering, the degree of concurrency here is $\Theta(\sqrt{p})$ instead of $\Theta(p)$, since usually there are at most two computational windows for the reordering phase inside the AED window.

—T_{Hess}: The Hessenberg reduction routine `PDGEHRD` uses the parallel algorithm described in Choi et al. [1995]. Almost all computations and communication are performed on matrix-vector and matrix-matrix multiplications. Therefore, we need to model these PBLAS operations first. The level 2 BLAS operations GEMV and GER require

$$T_{\text{GEMV}}(m, n, p) \approx T_{\text{GER}}(m, n, p) \approx \frac{2mn}{p}\gamma + \log_2 p\left(\alpha + \frac{m+n}{2\sqrt{p}}\beta\right),$$

where $m \times n$ is the size of the matrix. This model can be directly generalized to multiplying two $m \times k$ and $k \times n$ matrices as long as $\min\{m, n, k\} \leq n_b$, since it is merely a "fat" level 2 BLAS operation. In the Hessenberg reduction algorithm, all level 3 BLAS operations are "fat" level 2 BLAS operations, so the cost for one GEMM operation can be modeled as

$$T_{\text{GEMM}}(m, n, n_b, p) \approx T_{\text{GEMM}}(m, n_b, n, p) \approx \frac{2mnn_b}{p}\gamma + \log_2 p\left(\alpha + \frac{(m+n)n_b}{2\sqrt{p}}\beta\right). \quad (2)$$

Using these simple models of PBLAS operations, we are able to establish a model for T_{Hess}. The level 2 part consists roughly of n matrix-vector multiplications of dimension $n \times (n-j)$ (for $j = 1, 2, \ldots, n$). Therefore, the cost is

$$T_{\text{level2}} = \sum_{j=1}^{n}\left[\frac{2n(n-j)}{p}\gamma + \log_2 p\left(\alpha + \frac{2n-j}{2\sqrt{p}}\right)\right] \approx \frac{n^3}{p}\gamma + \log_2 p\left(n\alpha + \frac{3n^2}{4\sqrt{p}}\beta\right).$$

The level 3 part contains roughly n/n_b iterations with one `PDGEMM` and one `PDLARFB` per iteration. Within the jth iteration ($j = 1, 2, \ldots, n/n_b$), `PDGEMM` involves matrices of dimension $n \times n_b$ and $n_b \times (n - jn_b - n_b)$; `PDLARFB` mainly performs two parallel GEMM operations, with $\{n_b \times (n-jn_b), (n-jn_b) \times (n-jn_b)\}$ and $\{(n-jn_b) \times n_b, n_b \times (n-jn_b)\}$ matrices involved. Another sequential TRMM operation in `PDLARFB` is neglected because it only contributes lower order terms in both arithmetic and communication costs. So the cost for level 3 part is

$$\begin{aligned} T_{\text{level3}} &= \sum_{j=1}^{n/n_b}\left[\frac{2jn_b + 6(n-jn_b)}{p}n_b(n-jn_b)\gamma + \log_2 p\left(3\alpha + \frac{6n-5jn_b}{2\sqrt{p}}\beta\right)\right] \\ &\approx \frac{7n^3}{3p}\gamma + \frac{3n\log_2 p}{n_b}\alpha + \frac{7n^2\log_2 p}{4\sqrt{p}}\beta, \end{aligned}$$

and hence the execution time for Hessenberg reduction (without explicitly forming the orthogonal matrix) is

$$\widetilde{T}_{\text{Hess}}(n,p) = T_{\text{level2}} + T_{\text{level3}} \approx \frac{10n^3}{3p}\gamma + n\log_2 p\,\alpha + \frac{5n^2\log_2 p}{2\sqrt{p}}\beta. \tag{3}$$

Even if the proportion of level 3 BLAS operations is improved to 80% as suggested in Quintana-Ortí and van de Geijn [2006] but not implemented in the current PDGEHRD yet, the estimate in (3) would not change too much, since the number of messages in the level 2 part is not reduced.

Since the Householder reflections are stored in a compact form in the lower triangular part of the upper Hessenberg matrix, formulating the orthogonal matrix after Hessenberg reduction is another necessary step. This step is done by the ScaLAPACK routine PDORMHR, which is mainly a series of calls to PDLARFB. Similar to the discussion above, we obtain

$$T_{\text{ORMHR}} \approx \frac{2n^3}{p}\gamma + \frac{3n\log_2 p}{n_b}\alpha + \frac{7n^2}{4\sqrt{p}}\beta.$$

Therefore, the total runtime for the Hessenberg reduction process including formulating the orthogonal matrix is

$$T_{\text{Hess}}(n,p) = \widetilde{T}_{\text{Hess}} + T_{\text{ORMHR}} \approx \frac{16n^3}{3p}\gamma + n\log_2 p\,\alpha + \frac{17n^2\log_2 p}{4\sqrt{p}}\beta. \tag{4}$$

—T_{update}: The cost for updating the off-diagonal blocks with respect to the AED window is simple to analyze since it merely contains three GEMM operations. Since these GEMM operations are not "fat" level 2 BLAS operations, we need to use a model different to (2). According to van de Geijn and Watts [1997], the execution time for a GEMM operation on a $\sqrt{p}\times\sqrt{p}$ processor grid with $m\times k$ and $k\times n$ matrices involved is

$$T_{\text{GEMM}}(m,n,k,p) \approx \frac{2mnk}{p}\gamma + \left(\frac{k}{n_b} + 2\sqrt{p}\right)\left(2\alpha + \frac{(m+n)n_b}{\sqrt{p}}\beta\right)$$

if $\min\{m,n,k\} = k \gg n_b$. Then we conclude that

$$T_{\text{update}}(n, n_{\text{AED}}, p) \approx \frac{2nn_{\text{AED}}^2}{p}\gamma + \frac{n_{\text{AED}}}{n_b}\left(6\alpha + \frac{2nn_b}{\sqrt{p}}\beta\right).$$

Now we are ready to estimate the overall runtime T_{AED} by substituting n with n_{AED} in (1) and (4). We can see that T_{redist} is always negligible compared to other components. Reordering contributes with only marginal communication costs also. By merging all these estimates together, we eventually obtain

$$\begin{aligned}
T_{\text{AED}}(n, n_{\text{AED}}, p) \;\approx\;& T_{\text{pipe}}(n_{\text{AED}}, p_{\text{AED}}) + T_{\text{reorder}}(n_{\text{AED}}, p) + T_{\text{Hess}}(n_{\text{AED}}, p) \\
&+ T_{\text{update}}(n, n_{\text{AED}}, p) \\
\approx\;& \left(\frac{20n_{\text{AED}}}{p_{\text{AED}}} + \frac{4\sqrt{p}\,n_b + 16n_{\text{AED}} + 2n}{p}\right)n_{\text{AED}}^2\gamma \\
&+ \frac{n_{\text{AED}}^2}{n_b}\left(\frac{3(\log_2 p_{\text{AED}} + 2)}{\sqrt{p_{\text{AED}}}} + \frac{n_b\log_2 p}{n_{\text{AED}}}\right)\alpha \\
&+ \frac{n_{\text{AED}}^2}{n_b}\left(\frac{3n_b\log_2 p_{\text{AED}}}{\sqrt{p_{\text{AED}}}} + \frac{8n_{\text{AED}}}{p_{\text{AED}}} + \frac{3n_b}{2\sqrt{p}} + \frac{17n_b\log_2 p}{4\sqrt{p}} + \frac{2nn_b}{n_{\text{AED}}\sqrt{p}}\right)\beta
\end{aligned}$$

$$\approx \left[\frac{30C_2^2 n}{C_1} + \frac{9n^2 n_b}{C_1^2\sqrt{p}} + \frac{9(C_1+6)n^3}{2C_1^3 p}\right]\gamma + \left(\frac{9C_2 n}{C_1 n_b}\log_2\frac{3n}{2C_1C_2} + \frac{3n}{2C_1}\log_2 p\right)\alpha + \left[\frac{9C_2 n}{C_1}\log_2\frac{3n}{2C_1C_2} + \frac{12C_2^2 n}{C_1 n_b} + \frac{3n^2(18+C_1+51\log_2 p)}{16C_1^2\sqrt{p}}\right]\beta.$$

When n is extremely large (that is, C_1, C_2 and n_b are all tiny enough compared to n) and $n/\sqrt{p} = \text{constant}$, we have

$$\begin{aligned} T_{\text{AED}} &= \Theta\left(n + \frac{n^2}{\sqrt{p}} + \frac{n^3}{p}\right)\gamma + \Theta(n\log n + n\log p)\,\alpha + \Theta\left(n\log n + \frac{n^2}{\sqrt{p}}\log p\right)\beta \\ &= \Theta(n)\,\gamma + \Theta(n\log n)\,\alpha + \Theta(n\log n)\,\beta. \end{aligned}$$

Asymptotically AED only has slightly larger message sizes by a $\Theta(\log n)$ factor compared to QR sweeps and is hence not much more expensive. However, in practice, we still need to handle AED very carefully because large leading factors in lower-order terms have significant impact on the performance if the matrix is not large enough. Similar to the analysis for T_{AED}, the cost for computing shifts can be estimated by

$$\begin{aligned} T_{\text{shift}}(n, n_{\text{shift}}, p) &\approx \widetilde{T}_{\text{pipe}}(n_{\text{shift}}, p_{\text{shift}}) \\ &\approx \frac{10n_{\text{shift}}^3}{p_{\text{shift}}}\gamma + \frac{3n_{\text{shift}}^2}{\sqrt{p_{\text{shift}}}\,n_b}(\log_2 p_{\text{shift}} + 2)\,\alpha \\ &\quad + \left(\frac{3n_{\text{shift}}^2\log_2 p_{\text{shift}}}{\sqrt{p_{\text{shift}}}} + \frac{8n_{\text{shift}}^3}{p_{\text{shift}} n_b}\right)\beta \\ &\approx \frac{10C_2^2 n}{C_1}\gamma + \frac{6C_2 n}{C_1 n_b}\log_2\frac{n}{C_1C_2}\,\alpha + \left(\frac{6C_2 n}{C_1}\log_2\frac{n}{C_1C_2} + \frac{8C_2^2 n}{C_1 n_b}\right)\beta. \end{aligned}$$

Asymptotically T_{shift} is not so important in the scalability analysis because it can never be larger than T_{AED}.

REFERENCES

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. W. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. 1997. *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

J. Choi, J. J. Dongarra, and D. W. Walker. 1995. The design of a parallel dense linear algebra software library: Reduction to Hessenberg, tridiagonal, and bidiagonal form. *Numer. Algorithms* 10, 2, 379–399.

K. Dackland and B. Kågström. 1996. An hierarchical approach for performance analysis of ScaLAPACK-based routines using the distributed linear algebra machine. In *Applied Parallel Computing Industrial Computation and Optimization (PARA 1996)*, J. Waśniewski, J. J. Dongarra, K. Madsen, and D. Olesen, Eds. Lecture Notes in Computer Science, Vol. 1184. Springer-Verlag, Berlin, 186–195.

R. Granat, B. Kågström, and D. Kressner. 2009. Parallel eigenvalue reordering in real Schur forms. *Concurrency and Computat. Pract. Exper.* 21, 9, 1225–1250.

G. Henry, D. S. Watkins, and J. J. Dongarra. 2002. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. *SIAM J. Sci. Comput.* 24, 1, 284–311.

B. Kågström, D. Kressner, and M. Shao. 2012. On aggressive early deflation in parallel variants of the QR algorithm. In *Applied Parallel and Scientific Computing (PARA 2010)*, K. Jónasson, Ed. Lecture Notes in Computer Science, Vol. 7133. Springer-Verlag, Berlin, 1–10.

L. Prylli and B. Tourancheau. 1997. Fast runtime block cyclic data redistribution on multiprocessors. *J. Parallel Distr. Comput.* 45, 1, 63–72.

G. Quintana-Ortí and R. A. van de Geijn. 2006. Improving the performance of reduction to Hessenberg form. *ACM Trans. Math. Software* 32, 2, 180–194.

R. A. van de Geijn and J. Watts. 1997. SUMMA: Scalable universal matrix multiplication algorithm. *Concurrency and Computat. Pract. Exper.* 9, 4, 255–274. Also as LAPACK Working Note 96.