Interleaved Parallel Hybrid Arnoldi Method for a Parallel Machine and a Network of Workstations *

Guy Edjlali
UMIACS and Computer Science Department, University of Maryland,
College Park, MD 20742, USA

Serge G. Petiton
Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq Cedex, FRANCE

Nahid Emad
Laboratoire PRISM, Université de Versailles,
45 Avenue des Etats-Unis,
78035 Versailles Cedex, FRANCE

Abstract

The data-parallel paradigm which is effective for many numerical algebra computations is no more sufficient when programming heterogeneous machines. To efficiently utilize these new resources we need to combine the two paradigms: data-parallelism and task parallelism. This leads us to define new algorithms.

In this paper, throughout an example we will illustrate this statement. The chosen example is the computation of some eigenvalues of large non-structured sparse non-Hermitian matrices. We present a new algorithm, Interleaved Arnoldi, and compare its performances to the parallel Arnoldi algorithm. The different environments we consider are: a CM5, a CM5 and a Sparc 20, a CM5 and 2 Sparc 20. We show that significant results can be obtained even when the heterogeneous machine consists of a CM5 and 2 Sparc 20 workstations.

1 Introduction

The breakthrough in high speed networks and their availability allow computer centers to consider a new type of architecture: heterogeneous architectures. They are created by linking different machines with a high speed network. The heterogeneous machines have many advantages. Theoretically, they are able to reach a super-linear speed-up and they have a tremendous amount of memory and of computational power. Thus they should be able to cope with large industrial applications. Moreover, these machines will have different programming paradigms, one related to each individual component : i.e. each individual machine. This last point can be a burden for the programmer but also an optimization opportunity. One last problem with these heterogeneous environments is the availability as well as the cost of the resources.

This paper focuses on the feasibility and on the performances obtained with heterogeneous computing through the computation of the eigenvalues and eigenvectors of very large non-structured sparse non-Hermitian matrices using interleaving of computations on a parallel machine. We show that to fully utilize the capabilities of these machines, new algorithms and methods should be introduced. We present different implementations of the Arnoldi method on different environments. The environments are: a CM5, a CM5 and a Sparc 20, a CM5 and 2 Sparc 20. We illustrate the fact that significant results can be obtained even when the heterogeneous machine consists of a CM5 and 2 Sparc 20 workstations.

These results are achieved by using a task and data-parallel programming model. Integration of task and data-parallelism have received a lot attention during these last years. Some research are based on enhancements of data parallel languages or task parallel languages in order to integrate data parallelism and task parallelism. Fx [12] uses additional HPF directives to specify task parallelism. Fortran M [2] makes extensions to Fortran77 for task parallel computations, and also allows some data distribution statements. Braid [14] introduces data parallel extensions to the Mentat Programming Language [10]. Communication libraries like PVM and MPI [15] may be used by the programmer to directly transfer messages from one data parallel task to another.

*This research was supported by NID/DCAT, Census bureau and NASA under grant NASA #NAG-1-542482 (ARPA Project Number 8874), by ARPA under grant #F19628-94-C-0057.

2 Parallel implementation of the Arnoldi method

The iterative Arnoldi method computes the r algebraically largest eigenvalues and associated eigenvectors of a very large non-Hermitian matrix A of size n. These eigenpairs are obtained by an orthogonal projection of the problem onto a Krylov subspace $\mathcal{K}_{m,v} = lin(v, Av, \cdots, A^{m-1}v)$ of size $m \ll n$. This method defines a m-sized subspace, and solves the projected problem into this subspace. The targeted eigenvalues and eigenvectors, called the Ritz elements, are approximated by the ones of the projected problem in the subspace. Therefore, this method can be divided in two main phases: a projection phase and a Ritz computation phase. After each iteration, we restart the method with a linear combination of some of the Ritz vectors[11].

The projection phase is highly parallel [6] and a data-parallel implementation on a CM-5 with CM-Fortran has been effectuated. The second phase is vectorial: as we do not have an easy access to a vectorial machine, a sequential version of it has been implemented. We call *Parallel Arnoldi* this implementation.

As the eigenvalues and eigenvectors are approximated by using the ones into the Krylov subspace, the size of this subspace determines the amount of time necessary to execute each iteration. The cost of each iteration increases with the size of the subspace. The number of iterations decreases with the size of the subspace. There is then a tradeoff to find between the execution time of each iteration and the number of iterations to reach convergence [6, 3].

The authors introduced in [4] a new adaptation of the iterative Arnoldi method, called the *Hybrid Arnoldi* method. The main differences with the classical iterative Arnoldi are :

- We start l Arnoldi processes with different initial guesses v_i.
- Each Arnoldi process has its own subspace size $m_i, i = 1, l$.
- Restart of each Arnoldi process utilizes all the Ritz elements computed by all the Arnoldi processes.

Let m be the vector $(m_1, \cdots, m_\ell)^T$ and V the matrix whose columns are $v_1, \cdots, v_l$. The Hybrid Arnoldi(l,r,m,V) approximates one eigenpair (λ_j, w_j) of A by the Ritz eigenvalues and eigenvectors $(\lambda_j^{(m_i)}, w_j^{(m_i)})$ using Gramm-Schmidt orthogonalization process[11] ; $i = 1, l$ and $j = 1, r$.

We chose the m_i ($m_k \ll n, \forall k = 1, \ell$) with respect to memory space and computational throughput. The Arnoldi(l,r,m,V) algorithm is detailed in figure 1. Its numerical properties are described in [5].

The functions f_i are often taken as linear combination of Ritz vectors [11]. One option is to choose the new initial guess v_i as a function of the best Ritz vectors already computed. Throughout this paper, the experiments are carried with this assumption.

An important advantage of this algorithm is its large coarse granularity. Each "basic" Arnoldi process can then be executed on a different processor or computer, and the intermediate results can be merged.

A parallel algorithm for the Hybrid Arnoldi method is presented in figure 2.

In this paper, we generalize the algorithms; but on most computation centers the number of Arnoldi processes would be equal to just some unities. We will furthermore restrict ourselves to 2 processes, as our computation center has two parallel machines: a 32-Node CM5 and a 4k CM200.

The restart strategy of each Arnoldi process will include data from other processes as they are received. The restart strategy and the subspace size for each process is chosen with respect to the communication capabilities and speed of the machines involved in the computation.

3 Heterogeneous Arnoldi

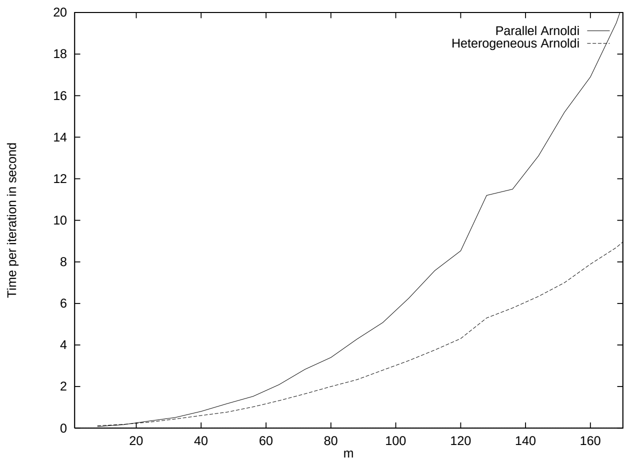

Figure 3: Time in seconds for one iteration of the Parallel Arnoldi and the Heterogeneous Arnoldi for a C-diagonal matrix when C=3

An heterogeneous environment consists of an environment with different computers linked by a network. We consider the environment consisting of a parallel machine and a fast sequential machine. The previously described Parallel Arnoldi method can be divided into two main tasks: the projection task and the Ritz computation task. These tasks can run on different machines. Consequently, the projection task, which is highly data parallel, will be mapped on the CM5, while the Ritz computation task which is vectorial, will be mapped on a fast sequential machine: a Sparc 20. We call this implementation: *Heterogeneous Arnoldi*.

The major difference between the Parallel Arnoldi implementation and the implementation of the Arnoldi method on an heterogeneous environment resides in the location where the Ritz computation phase is executed: on the Front-End of the CM5 in the first case and on the Sparc 20 on the second case.

The sparse matrix-vector multiplication operation is the most costly operation used by the projection task. In a previous work [7], we had shown that ma-

Initialization

set

a number l of different Arnoldi processes,

a set of subspace size m_i for $i = 1, l$, with $m = (m_1, \cdots, m_\ell)^T$,

a set of initial guess $v_1, \cdots, v_\ell$ with $V = (v_1, \cdots, v_l)$.

Iteration

do until convergence

for $i = 1$ to ℓ do

(i)- Arnoldi projection onto $\mathcal{K}_{m_i, v_i} = lin(v_i, Av_i, \cdots, A^{m_i - 1} v_i)$

(ii)- Ritz elements computation $\mathcal{R}(i) = ((\lambda_1^{(m_i)}, w_1^{(m_i)}), \cdots, (\lambda_r^{(m_i)}, w_r^{(m_i)}))$

done for i

if no convergence then

for $i = 1$ to ℓ do

(iii)- <u>Generalized Restart</u> : $v_i = f_i(\mathcal{R}(1), \cdots, \mathcal{R}(l))$

done for i

end if

end until

Figure 1: Arnoldi Algorithm

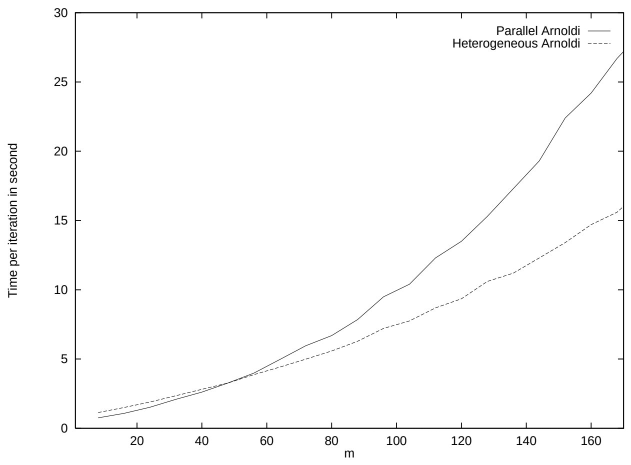

Figure 4: Time in seconds for one iteration of the Parallel Arnoldi and the Heterogeneous Arnoldi for a C-diagonal matrix when C=63

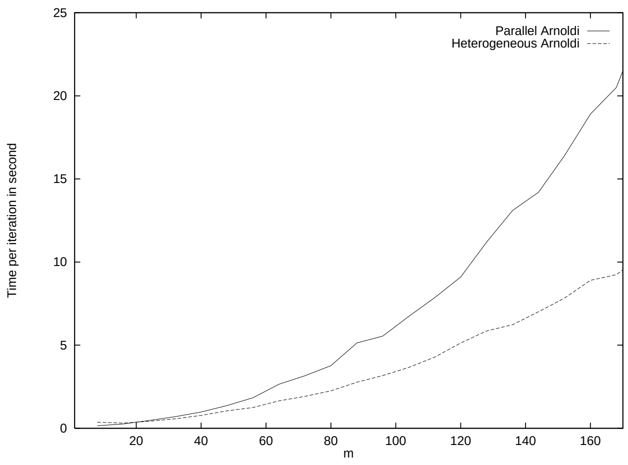

Figure 5: Time in seconds for one iteration of the Parallel Arnoldi and the Heterogeneous Arnoldi for a C-distributed matrix when C=4

trices with C-diagonal and C-distributed pattern provided an estimation of a lower and upper bounds for the execution time of the sparse matrix-vector multiplication operation on machines using data-parallel programming model. Thus, we chose matrices with C-Diagonal (resp. C-distributed) pattern in order to estimate the lower (resp. upper) bound of the execution time for the projection phase.

Figures 3, 4, 5 and 6 present the time needed to compute one iteration of the Arnoldi method as a function of the size, m, of the subspace on a parallel environment and on an heterogeneous environment. The test matrices are of size 16K and have C-diagonal or C-distributed pattern. Figures 3 and 4 illustrate the results for C-diagonal matrices when C=3 and C=63, while Figures 5 and 6 present results for C-distributed matrices when C=4 and C=64. C is the maximum number of non-zero elements per row or per column. The sparse matrices are stored using the SGP [13] or S^3 format[7].

Although the Sparc 20 is much faster than the Front-End, the implementation requires some com-

Do in parallel for $i = 1, l$

Arnoldi projection i

Ritz elements computation $\mathcal{R}(i)$

Computation of residual $\epsilon_k^{(i)} = ||Aw_k^{(m_i)} - \lambda_k^{(m_i)} w_k^{(m_i)}||, k = 1, r$

If no convergence then

Do (*Computation process*)

do until convergence

if necessary informations are received then

Hybrid Restart

else

Simple Restart

endif

Arnoldi projection i

Ritz elements computation

Computation of $\epsilon_k^{(i)} = ||Aw_k^{(m_i)} - \lambda_k^{(m_i)} w_k^{(m_i)}||, k = 1, r$

end until

in parallel with (*Communication process*)

do until convergence

Do

send elements of $\mathcal{R}(i)$ to process $k, k = 1, l$ and $k \neq i$ with respect to the function f_k

in parallel with

received elements of $\mathcal{R}(k)$ from process $k, k = 1, l$ and $k \neq i$ with respect to the function f_i

End parallel Do

end do until

Done in parallel for i

end if

end do in parallel

Figure 2: Hybrid Arnoldi Algorithm

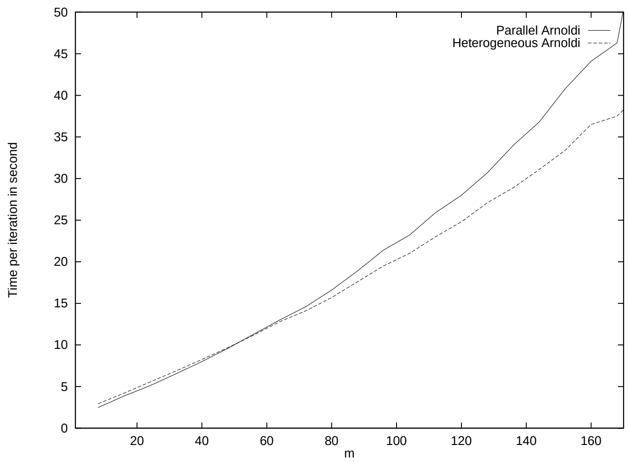

Figure 6: Time in seconds for one iteration of the Parallel Arnoldi and the Heterogeneous Arnoldi for a C-distributed matrix when C=64

munications to move the data from the CM5 to the Sparc 20. The extra communication cost generated by the Heterogeneous Arnoldi implementation cannot be compensated by the speed-up achieved on the Ritz computation when the size of the subspace is too small (e.g.: for small values of m).

The speed-up of Heterogeneous Arnoldi over Parallel Arnoldi can be greater than *two* when the cost of the projection phase is small (when the matrix is C-diagonal or C-distributed with a few number of non zero elements per row) and when the cost of the Ritz computation is high (when the size of the subspace is large). When m increases, the interest of the Heterogeneous Arnoldi over Parallel Arnoldi becomes more significant.

These results are supported by the theory which says that the number of operations required for the projection phase is approximately $n * m^2 + n * C * m$ [3], while it is $10 * m^3$ for the QR [9].

4 Interleaved Arnoldi

In the previous section, we have reduced the time of each iteration by parallelizing the Arnoldi method and by distributing each task on adapted architectures. Having done so, the parallel (resp. sequential) machine is idle when the sequential (resp. parallel) machine is computing. Some extra computation and/or communication can thus occur during this idle time. We present in this section a new implementation, *Interleaved Arnoldi* that takes advantage of these idle cycles to accelerate the convergence of the method at no cost.

This algorithm is based on the use of the Arnoldi Hybrid method in an heterogeneous programming environment.

The Hybrid Arnoldi method, presented on section 2, combines the intermediate results of different Arnoldi processes to accelerate the convergence. We implemented it [3, 8] on a network of super-computers and verified some of its numerical behaviors.

The implementation of the Hybrid Arnoldi algorithm on a parallel machine and on two sequential machines is illustrated by figure 9.

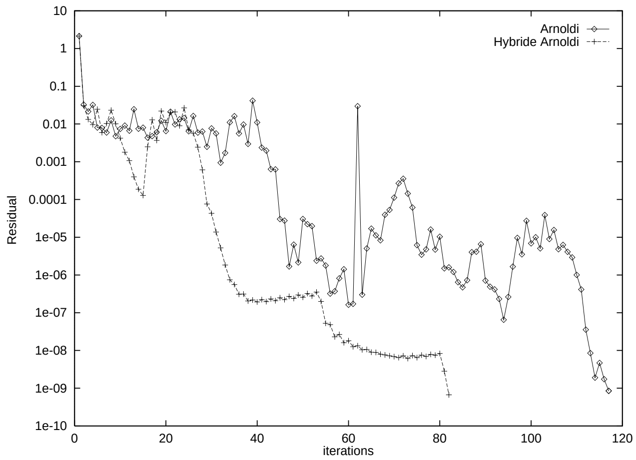

Figure 7: Residual for the Hybride Arnoldi method when N=1024 and C=3, m1=28 and m2=15

Figure 7 and 8 present the number of iteration to reach convergence using the Arnoldi method and the one using the Hybrid Arnoldi method. The program is running on a CM5 and a CM200 interconnected through a 10Mbit/s ethernet link. Experiment on figure 7 is using a C-Diagonal matrix of size N=1024 and C=3, two different initial vectors and two subspaces of size $m1$=28 and $m2$=15. The Arnoldi method is converging in 120 iterations, while the Hybrid Arnoldi method is converging in 80 iterations. Experiment on figure 8 is using a C-Diagonal matrix of size N=1024 and C=63, two different initial vectors and two subspaces of size $m1$=26 and $m2$=26. The Arnoldi method is converging in 54 iterations, while the Hybrid Arnoldi method is converging in 42 iterations.

The speedup is equal to the time to solve the problem on a CM5 and a CM200 over the time to resolve the same problem on a CM5 alone. For the first experiment (figure 7), the speedup is equal to 1.5. For the second experiment (figure 8), the speed up is equal to 1.28. This speedup was achieved despite the difference of speed between the CM5 and the CM200.

We implemented the same Hybrid Arnoldi algorithm on one parallel machine. Until now the projection phase and the Ritz computation phase are effectuated one after the other. Thus during the Ritz computation most of the processors are idle. The Hybrid Arnoldi method can therefore be implemented by interleaving, on the parallel machine, the projection phase of two different Arnoldi processes. Two sequential machines are necessary to compute the two associated Ritz computation phases. This computation is done concurrently to the parallel projection phase. We name this implementation the Interleaved Arnoldi. Interleaved Arnoldi is implemented on a CM5 and two Sparc 20.

Interleaved Arnoldi has two projections phases, and consequently two subspaces of size $m1$ and $m2$. We present on figures 10, 11, 12 and 13 three sets of curves:

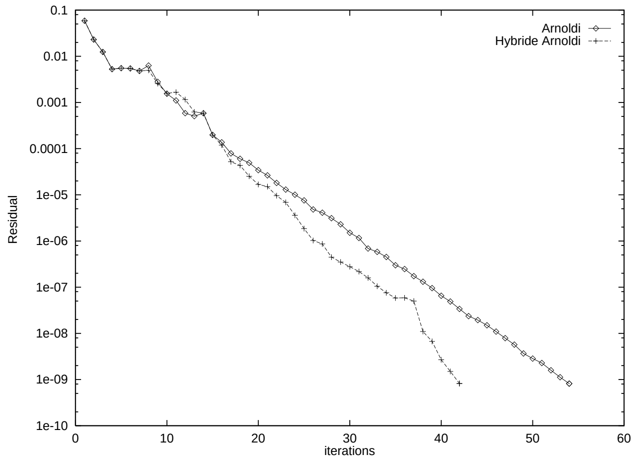

Figure 8: Residual for the Hybride Arnoldi method when N=1024 and C=63, m1=26 and m2=26

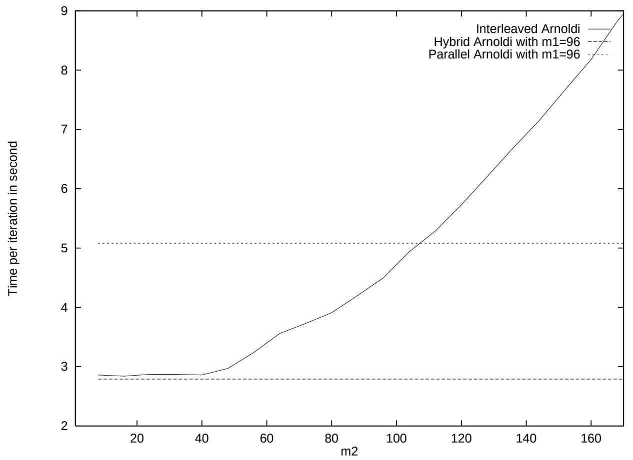

Figure 10: Comparison between the Parallel Arnoldi on the CM5, the Heterogeneous Arnoldi using the CM5 and one and the Interleaved Arnoldi using the CM5 two Sparc 20 computers, for C-diagonal matrices with $C = 3$.

1. The time to compute one iteration of Interleaved Arnoldi when $m1$ varies while $m2$ is a constant of value 96.

2. The time to compute one iteration of Heterogeneous Arnoldi when the subspace size is 96.

3. The time to compute one iteration of Parallel Arnoldi when the subspace size is 96.

The last two curves are presented only for comparison purposes. They show the time needed to compute the Arnoldi method when one using a “classical” Arnoldi algorithm. The first curve illustrates the behavior of one implementation of the Interleaved Arnoldi algorithm compared to two possible implementations of the Arnoldi algorithm.

The matrices for experiments presented on figure 10 (resp. 12) and figure 11 (resp. 13) are C-diagonal

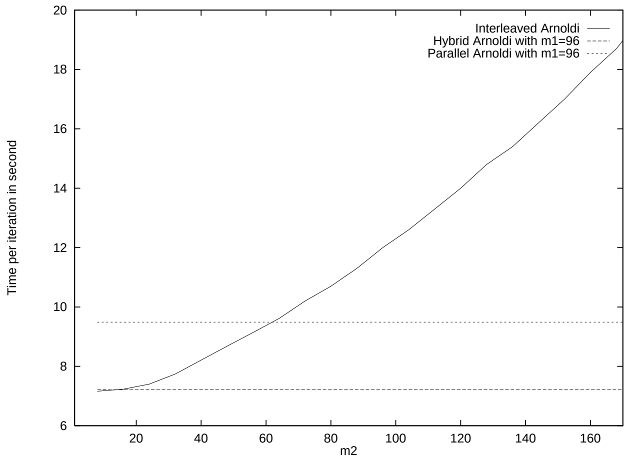

Figure 11: Comparison between the Parallel Arnoldi on the CM5, the Heterogeneous Arnoldi using the CM5 and one and the Interleaved Arnoldi using the CM5 two Sparc 20 computers, for C-diagonal matrices with $C = 63$.

(resp. C-distributed). These matrices are of size N=16384.

From the results of figure 10, we note that until $m1 < 44$, the time execution for one iteration with the Interleaved Arnoldi is experimentally identical to the time execution of the Heterogeneous Arnoldi. Thus for the same problem, we can compute at *no additional cost* a partial solution onto a subspace of dimension 44. The Hybrid Arnoldi method combines partial results from different Arnoldi processes in order to compute a more accurate restart vector. In this case, we are able to compute each restart vector by combining the partial solution given by a projection through a subspace of dimension 96 and through a subspace of dimension 44. This should reduce the number of iterations needed for the method to converge (see figures 7 and 8).

If we compare Interleaved Arnoldi to Parallel Arnoldi, then while Parallel Arnoldi performs one iteration on a subspace of dimension 96, Interleaved Arnoldi performs one iteration on a subspace of dimension 96 and one iteration on a subspace of dimension 105 and combines these two results to accelerate his convergence.

Figure 11 and 12 present the same kind of behavior for matrices with different patterns or different numbers of non zero elements.

On figure 13, the time to compute a projection is too high compared to the time to compute the Ritz computation. Thus there is not enough idle time during the Ritz computation to perform a projection.

5 Conclusion

We proposed a new method, the Hybrid Arnoldi method, and we presented one implementation of it: the Interleaved Arnoldi.

The speed up presented here with Heterogeneous Arnoldi and Interleaved Arnoldi requires only few additional resources: two Sparc 20. It allowed us to utilize more efficiently our most expensive resource: the

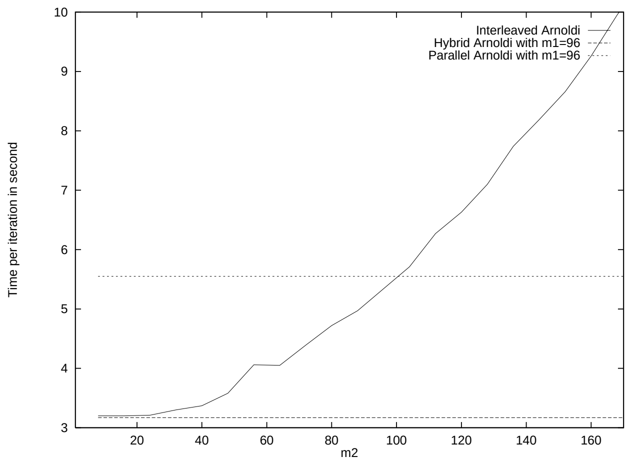

Figure 12: Comparison between the Parallel Arnoldi on the CM5, the Heterogeneous Arnoldi using the CM5 and one and the Interleaved Arnoldi using the CM5 two Sparc 20 computers, for C-distributed matrices with C=4.

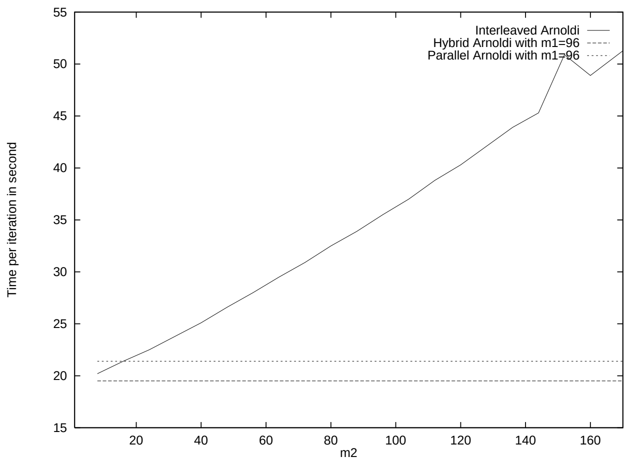

Figure 13: Comparison between the Parallel Arnoldi on the CM5, the Heterogeneous Arnoldi using the CM5 and one and the Interleaved Arnoldi using the CM5 two Sparc 20 computers, for C-distributed matrices with C=64.

CM5.

Compared to a parallel version of the Arnoldi method, the Interleaved Arnoldi method reduces the number of iterations to reach convergence while reducing the time of each iteration. This was achieved by changing our platform, our programming model and our algorithm. From a purely parallel platform we went to an heterogeneous platform. From a data parallel programming model we went to a task and data-parallel programming model. We modified the classical Arnoldi Algorithm and introduced the Hybrid Arnoldi method.

References

[1] C. Brezinski and M. R. Zaglia, *A hybrid procedure for solving linear systems*, Tech. Rep. ANO-283, USTL, September 1992.

[2] K. Chandy, I. Foster, K. Kennedy, C. Koelbel, and C. Tseng, *Integrated Support for Task and Data Parallelism.* Journal of Supercomputing Applications, 8(2), 1994.

[3] G. Edjlali, *Contribution to the parallelization of hybrid iterative methods for sparse matrices on heterogeneous architecture* (in French), PhD Thesis, University of Paris 6, December 1994.

[4] G. Edjlali, N. Emad, and S. Petiton, *Hybrid Method on Network of Heterogeneous Parallel Computers*, Proceedings of the 14th IMACS World Congress, Atlanta, July 11-15, 1994.

[5] N. Emad, S. Petiton, and G. Edjlali, *The Iterative Arnoldi Hybrid Method*, To appear.

[6] S. Petiton, *Parallel Subspace Method for non-Hermitian Eigenproblems on the Connection Machine*, Applied Numerical Mathematics Journal 10, North Holland, 1992.

[7] S. Petiton and G. Edjlali, *Data parallel structures and algorithms for sparse matrix computation*, in Advances in Parallel Computing, North-Holland, 1993.

[8] S.Petiton, G.Edjlali, *Krylov Subspace Methods on Multi-Connection Machines Environment*, Proceedings of the Second European CM Users Meeting, pp 17-28, Jean-Michel Alimi et al. Editors, Meudon, France, 1993.

[9] G. Golub and C.V. Loan, *Matrix Computations*, North Oxford Academic, 1983.

[10] A. Grimshaw, *The Mentat computation model data-driven support for object-oriented parallel processing.* Technical Report CS-93-30, University of Virginia, May 93.

[11] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1993.

[12] J. Subhlok, J. Stichnoth, D. O'Hallaron, and T. Gross, *Exploiting task and data parallelism on a multicomputer.* In Proceedings of the Fourth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming (PPOPP), May 1993.

[13] C. Weill-Duflos, *Optimisation de méthodes de résolution itératives de grand systèmes linéaire creux sur machine massivement parallèle*, PhD Thesis, University Of Paris 6,1994.

[14] E .West and A. Grishaw, *Braid: Integrating Task and Data Parallelism.* In Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation. IEEE Computer Society Press, February 1995.

[15] *Message Passing Interface Forum. Document for a Standard Message-Passing Interface.* Technical Report CS-93-214, University of Tennessee, November 1993.

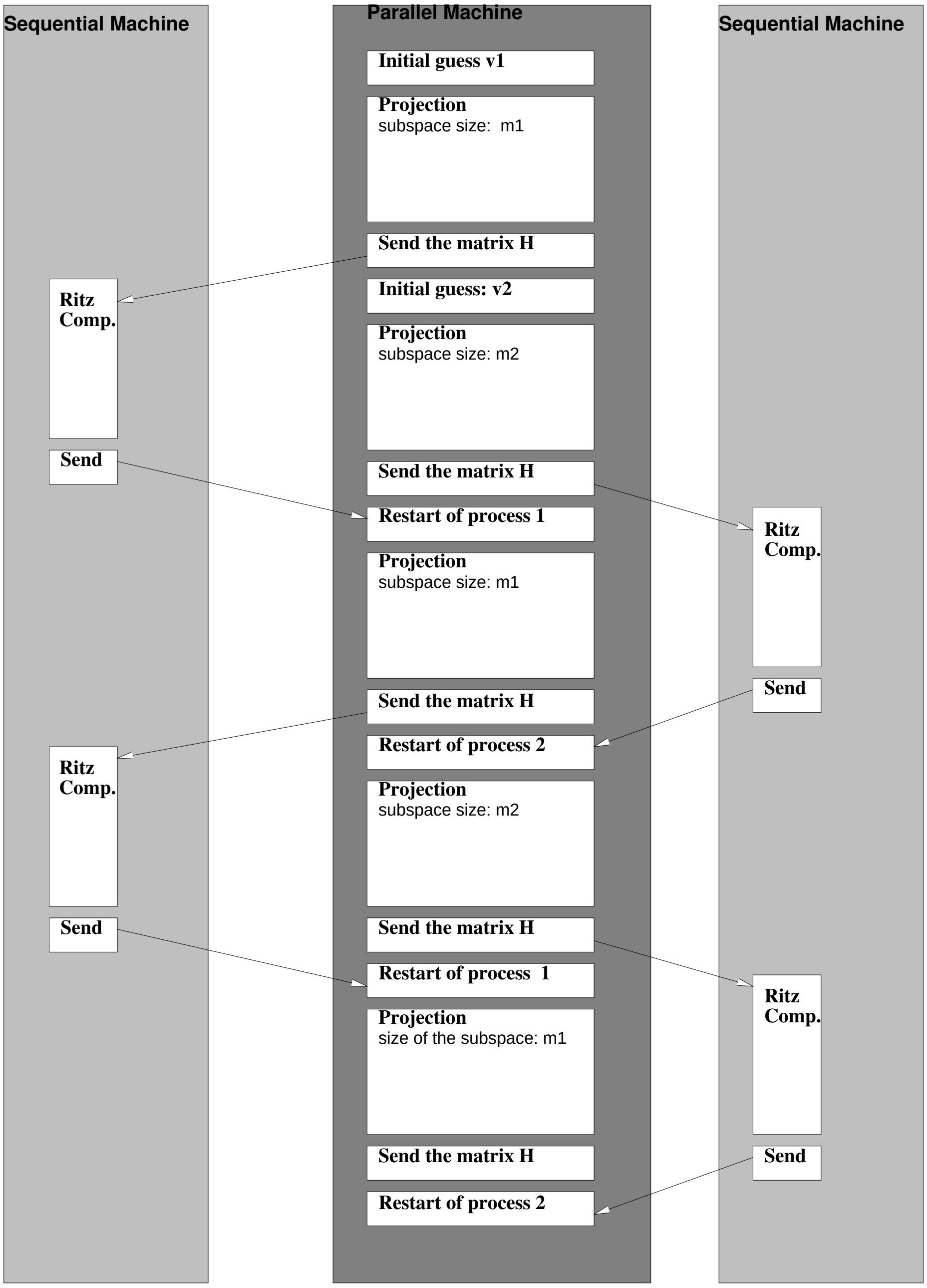

Figure 9: Mapping of the tasks for the hybrid Arnoldi Algorithm on a parallel machine and on two sequential machines