Fault Tolerant Domain Decomposition for Parabolic Problems

Marc Garbey and Hatem Ltaief

Department of Computer Science, University of Houston, Houston, TX 77204 USA
garbey@cs.uh.edu, ltaief@cs.uh.edu

1 Introduction

The objective of this paper is to present some numerical schemes for the time integration of parabolic problems that can recover from a failure of the computer system. We construct an algorithmic solution of the problem in the context of domain decomposition and distributed computing.

Our model problem is the heat equation:

$$\frac{\partial u}{\partial t} = \Delta u + F(x,t),\ (x,t) \in \Omega \times (0,T),\ u_{|\partial\Omega} = g(x),\ u(x,0) = u_o(x). \quad (1)$$

We suppose that the time integration is done by a first order implicit Euler scheme,

$$\frac{U^{n+1} - U^n}{dt} = \Delta U^{n+1} + F(x,t^{n+1}), \quad (2)$$

and that Ω is partitioned into N subdomains $\Omega_j,\ j = 1..N$. The computation of these subdomains is distributed among N Processing Units (PUs) or computers.

We anticipate that one or several PUs may stall or get disconnected. We complement the distributed architecture of these N PUs, with S additional PUs called spare processing units.

The problem in designing a Fault Tolerant (FT) code that can survive to several failures of PUs decomposes as follows:

- (Pb 1) guaranty that if one or several PUs get disconnected the code can still be executed.
- (Pb 2) provide an algorithm that can restart the time integration from the data that are available in the distributed memory or file system.

While FT is not so critical for a standard application running for few hours on a medium scale parallel system, it becomes a real issue for long time runs on

a large system or grid computing architecture. In both cases the probability of failures of computing units becomes almost certain, and parallel input/output are not as efficient as ordinary check point procedures may require.

Pb1 is solved by middlewares, like *FT-MPI* [1, 2] for example, which ensure the application goes on while some processors have failed. On the other hand, the application should be FT as well because these middlewares do not guaranty to get the correct numerical solution after a failure.

In this paper we focus on the design of numerical algorithms that solved (Pb2) without using global check pointing.

Global checkpointing does not scale on large parallel system and is impractical on a grid of computers. Indeed, as the number of nodes and the problem size increases, the cost of checkpointing and recovery increases, while the mean time between failures decreases.

The approach we have taken is as follows: spare processors are used to efficiently store the subdomain data of the application during execution in local *asynchronous mode* in their local memory. In case of failure, a spare processor takes over for the failed processor without the entire system having to roll back to a globally consistent checkpoint.

The numerical problem that we address can be defined as follows:

- We assume that spare processors have stored copies of all subdomain data $U_j^{n(j)}, j = 1..N$, in their local memory. A priori the time step $n(j) \neq n(k)$ for $j \neq k$.
- We look for a (parallel) reconstruction process of U^M at a common time step $M \in (min_j\{n(j)\}, max_j\{n(j)\})$, from subdomains data $U_j^{n(j)}, j = 1..N$, at different but nearby time steps.

The code can then restart from U^M. Because the time interval between two asynchronous back ups is small, we are looking for a numerical procedure that is completely explicit and does not require the complexity of a standard parameter identification method.

In the next section, we will discuss several algorithmic ideas to solve this problem.

2 The Fault Tolerant algorithms

For the simplicity of the presentation, we will restrict ourselves to the one dimensional heat equation problem $\Omega = (0, 1)$, discretized on a regular Cartesian grid:

$$\frac{U_j^{n+1} - U_j^n}{dt} = \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{h^2} + F_j^{n+1}, \tag{3}$$

and we assume that $dt \sim h$.

However, most of the ideas presented here should be generalized easily to higher space dimension. We will review progressively few numerical methods

to reconstruct a uniform approximation of U^M, from disparate data $U^{n(j)}$ in each subdomain Ω_j.

2.1 Interpolation method

Let $M = \frac{1}{N} \Sigma_{j=1..N} n(j)$. We look for an approximation of U_j^M in Ω. We assume that we have at our disposal $U^{n(j)}$ and $U^{m(j)}$ at two time steps $n(j) < m(j)$, in each subdomain Ω_j.

Then, if $||U^{n(j)} - U^{m(j)}||_{\Omega_j}$ is below some tolerance number, we may use a second order interpolation/extrapolation in time to get an approximation of U^M. The numerical error should be of order $((m(j) - n(j))dt)^2$. This simple procedure reduces the accuracy of the scheme and introduces small jump at the interfaces between subdomains. This method is perfectly acceptable when one is not interested in accurately computing transient phenomena. However, this method is not numerically efficient in the general situation.

2.2 Forward Time Integration

Let us assume that for each subdomain, we have access to $U^{n(j)}$. For simplicity we suppose that $n(j)$ is a monotonically increasing sequence. We further suppose that we have stored in the memory of spare processors the time history of the artificial boundary conditions $I_j^m = \Omega_j \cap \Omega_{j+1}$ for all previous time steps $n(j) \leq m \leq n(j+1)$.

We can then reconstruct with the forward time integration of the original code, the solution $U^{n(N)}$, as follows:

- Processor one advances in time u_1^n from time step $n(1)$ to time step $n(2)$ using boundary conditions I_1^m, $n(1) \leq m \leq n(2)$.
- Then, Processors one and two advance in parallel u_1^n and u_2^n from time step $n(2)$ to $n(3)$ using neighbor's interface conditions, or the original interface solver of the numerical scheme.
- This process is repeated until the global solution $u^{n(N)}$ is obtained.

This procedure can be easily generalized in the situation of Figure 1 where we do not assume any monotonicity on the sequence $n(j)$. The thick line represents the data that needs to be stored in spare processors, and the interval with circles are the unknowns of the reconstruction process.

The advantage of this method is that we can easily reuse the same algorithm as in the standard domain decomposition method, but restricted to some specific subsets of the domain decomposition. We reproduce then the exact same information U^M as the process had no failures. The main drawback of the method is that saving at each time step the artificial boundary conditions may slow down the code execution significantly. To illustrate this difficulty, we have implemented a 3D benchmark code with a very simple explicit/Implicit domain decomposition procedure for (1). We refer to [3] for a

more sophisticated method along these lines. We use a Krylov method to implicit the time stepping per domain and impose explicitly the boundary values on the artificial boundaries. The domain of computation $\Omega = (0,1)^3$ is distributed on a two dimensional grid of $p_x \times p_y$ processors. This two dimensional grid is extended with an additional row of p_x spare processors.

We have implemented a totally asynchronous back up of the subdomains every K time steps. We have also the option to back up in addition all two dimensional artificial interfaces generated in the sequence of $K-1$ consecutive time steps in between two back ups of all subdomains. The communication between active processors and spare processors is done by non-blocking communication, and the back up data are small enough to fit in the main memory of spare processors.

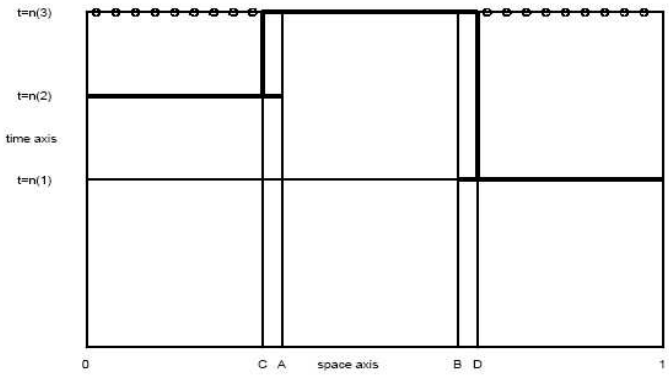

Fig. 1. Illustration of the reconstruction procedure with the forward method.

Figures 2 and 3 report on the numerical experiment with 36 PUs and 6 spare processors on two different architectures. Each processor has a block of $18 \times 18 \times 98$ grid points. The first system used in figure 2 is a beowulf cluster with dual AMD 32 bit processors and a gigabit ethernet network, while the second system used in figure 3 is a dual Itanium cluster that has a Myrinet network. While the penalty to back up the subdomain on spare processors is particularly high with the system that uses Gigabit ethernet, it can be seen that saving the artificial boundary conditions every time step significantly slows down the application on *both* computer architecture systems.

We will now discuss a reconstruction method that produces an approximate solution without using the time series of artificial boundary conditions.

2.3 Backward Integration and Space Marching

Let us suppose now that we asynchronously store only the subdomain data, and not the chronology of the artificial interface condition. To be more specific,

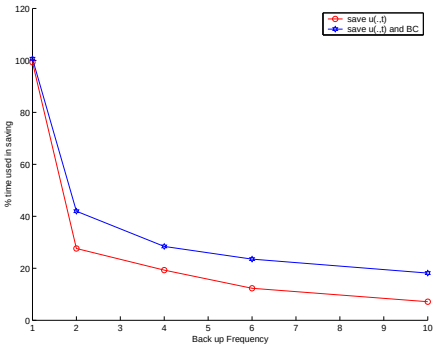

Fig. 2. Overhead with Dual AMD/Gigabit Ethernet

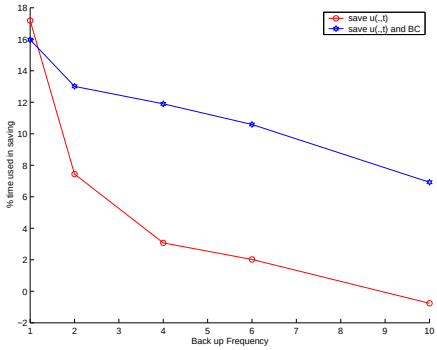

Fig. 3. Overhead with dual Itanium/Myrinet Network

we suppose that we have access for each subdomain to the solution at two different time steps $n(i), m(i)$ with $m(i) - n(i) = K >> 1$.

The Forward Implicit scheme provides an explicit formula when we go backward in time:

$$U_j^n \;=\; U_j^{n+1} \;-\; dt\,\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{h^2} \;- F_j^{n+1}, \tag{4}$$

The existence of the solution is granted by the forward integration in time. Two difficulties are first the instability of the numerical procedure and second the fact that one is restricted to the cone of dependence as shown in Figure 4.

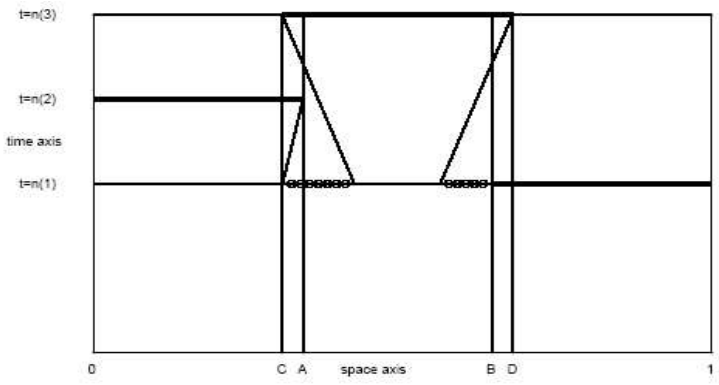

Fig. 4. Illustration in one space dimension of the problem with the third solution.

We have in Fourier modes

$$\hat{U}_k^n \;=\; \delta_k\,\hat{U}_k^{n+1},$$

with

$$\delta_k \sim -\frac{2}{h}(\cos(k\ 2\ \pi\ h) - 1),\ |k| \leq \frac{N}{2}.$$

The expected error is at most in the order $\frac{\nu}{h^K}$ where ν is the machine precision and K the time step. Therefore, the backward time integration is still accurate up to time step K with

$$\frac{\nu}{h^K} \sim h^2.$$

To stabilize the scheme, one can use the Telegraph equation that is a perturbation of the heat equation:

$$\epsilon\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial t} = F(x,t), x \in (0,1), t \in (0,T) \tag{5}$$

The asymptotic convergence can be derived from [4] after time rescaling. The general idea is then to use the previous scheme (4) for few time steps and pursue the time integration with the following one

$$\epsilon\frac{U_j^{n+1} - 2\ U_j^n + U_j^{n-1}}{\tilde{dt}^2} - \frac{U_{j+1}^n - 2\ U_j^n + u_{j-1}^n}{h^2}\ +\ \frac{U_j^{n+1} - U_j^n}{\tilde{dt}} = F_j^n. \tag{6}$$

Let us notice that the time step $\tilde{dt}$ should satisfy the stability condition $\tilde{dt} < \epsilon^{1/2}h$. We take in practice $\tilde{dt} = dt/p$ where p is an integer. The smaller is ϵ, the more unstable is the scheme (6) and the flatter is the cone of dependence. The smaller is ϵ, the better is the asymptotic approximation. We have done a Fourier analysis of the scheme and Figure 5 shows that there is a best compromise for ϵ to balance the error that comes from the instability of the scheme and the error that comes from the perturbation term in the telegraph equation. We have obtained a similar result in our numerical experiments.

To construct the solution outside the cone of dependencies we have used a standard procedure in inverse heat problem that is the so called space marching method [5]. This method may require a regularization procedure of the solution obtained inside the cone using the product of convolution

$$\rho_\delta\ *\ u(x\ ,\ t),$$

where

$$\rho_\delta = \frac{1}{\delta\sqrt{\pi}}\exp(-\frac{t^2}{\delta^2}).$$

The following space marching scheme:

$$\frac{U_{j+1}^n\ -\ 2\ U_j^n\ +\ U_{j-1}^n}{h^2} = \frac{U_j^{n+1} - U_j^{n-1}}{2\ dt} + F_j^n, \tag{7}$$

is unconditionally stable, provided $\delta \geq \sqrt{\frac{2\ dt}{\pi}}$.

The last time step $U^{n(i)+1}$ to be reconstructed uses the average

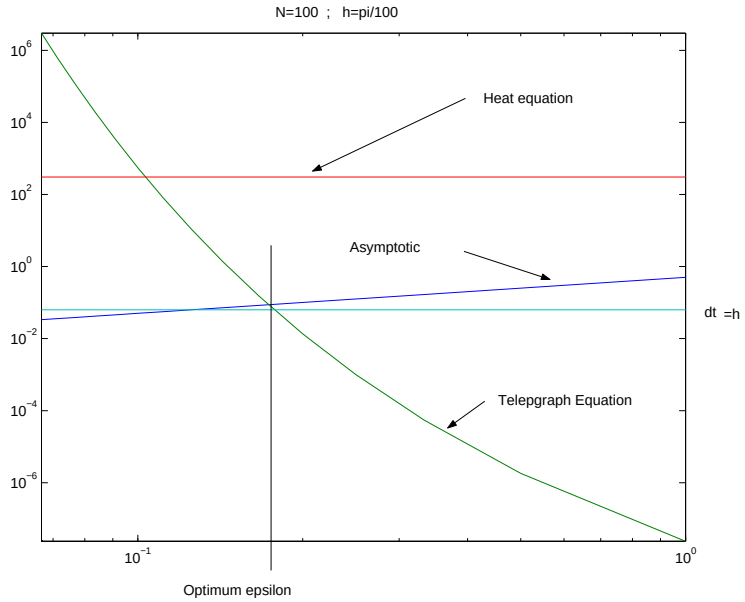

Fig. 5. Stability and error analysis with Fourier

$$U^{n(i)+1} = \frac{U^{n(i)} + U^{n(i)+2}}{2}.$$

We have observed that filtering as suggested in [5] is not necessary in our reconstruction process.

Figure 6 illustrates the numerical accuracy of the overall reconstruction scheme that combines (4) and (7) for $\Omega = (-\pi, \pi)$, $dt = h = 0.0314$, $K = 7$ and F such that the exact analytical solution is $cos(q_1\ x)(sin(q_2\ t) + \frac{1}{2}cos(q_2\ t))$, $q_1 = 2.35, q_2 = 1.37$.

In this specific example our method gives better results than the interpolation scheme provided that $K \leq 7$. For larger K we can use the scheme (6) for time steps below $m(i) - 7$. However the precision may deteriorate rapidly in time.

3 Conclusion

We have presented the problem of FT algorithm for a parabolic operator. We have reviewed several procedures to reconstruct the solution in each subdomain from a set of subdomain solutions given at disparate time steps. This problem is quite challenging because it is very ill posed. We found a satisfactory solution by combining explicit reconstruction techniques that are a backward integration with some stabilization terms and space marching. We are currently applying these ideas to multi-dimensional parabolic problems.

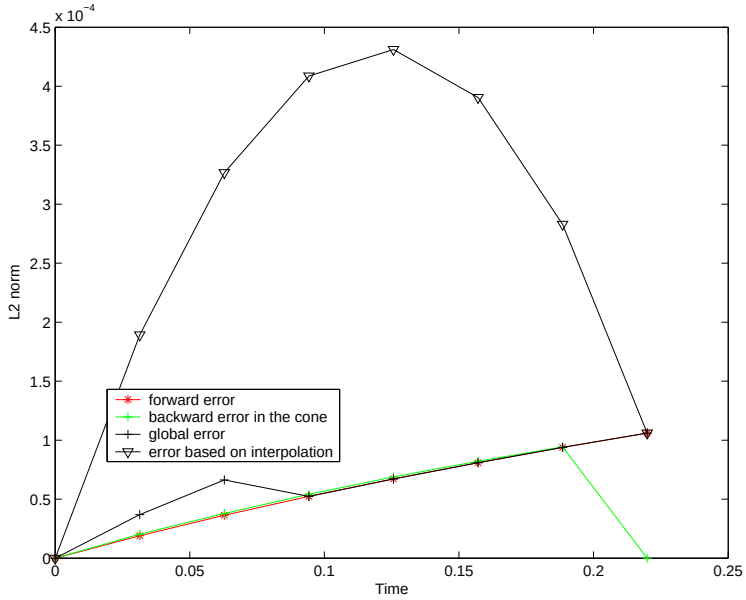

Fig. 6. Numerical accuracy of the overall reconstruction scheme

Acknowledgement Research reported here was supported by Award 0305405 from the National Science Foundation.

References

1. Gabriel, E., Fagg, G., Bukovsky, A., Angskun, T., Dongarra, J., A Fault-Tolerant Communication Library for Grid Environments, 17th Annual ACM International Conference on Supercomputing (ICS'03) International Workshop on Grid Computing and e-Science, San Francisco (2003)
2. Gropp and Lusk, Fault Tolerance in Message Passing Interface Programs, International Journal of High Performance Computing Applications, 18: 363-372 (2004)
3. C.Dawson and T.Dupont, Explicit/Implicit, Conservative Domain Decomposition Procedures for Parabolic Problems Based on Block-Centered Finite Differences, Sinum Vol 31, Issue 4, pp 1045-1061 (1994)
4. W.Eckhaus and M.Garbey, Asymptotic analysis on large time scales for singular perturbation problems of hyperbolic type, SIAM J. Math. Anal. Vol 21, No 4, pp867-883 (1990)
5. D.A.Murio, The Mollification Method and the Numerical Solution of Ill-posed Problems, Wiley, New York (1993)