Fault Tolerant Algorithms for Parallel 3D Heat Transfer Problems

Hatem Ltaief, Marc Garbey* and Edgar Gabriel
Department of Computer Science, University of Houston
4800 Calhoun Road, Houston, TX 77204, USA
{ltaief, garbey, gabriel}@cs.uh.edu

Key words: Parallel Computing, Fault Tolerant Algorithms

ABSTRACT

With the emerging of new massively parallel systems such as the IBM *BlueGene* with tens of thousands of processors, the mean time between machine failures is considerably decreasing. Therefore, the reliability of the machine becomes a very important issue. The goal of this paper is to present new fault tolerant approach based on numerical explicit schemes for the time integration of parabolic problems. This technique allows the application to recover from process failures and to retrieve mathematically the lost data on the failed process(es) avoiding the roll-back operation required in most checkpoint-restart schemes [1, 2]. While for low time consuming applications requiring small amount of resources, aborting after a failure does not really matter. However for critical applications running for several weeks on hundreds of processors, the application shutdown appears to be a major concern.
We construct an algorithmic solution of the problem in the context of domain decomposition and distributed computing. The model application used throughout the paper to highlight our results is the three dimensional heat equation as given by

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

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

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

and that Ω is partitioned into N subdomains $\Omega_j,\ j = 1..N$.
The work presented in the paper is based on the Fault Tolerant MPI (FT-MPI) [3, 4] framework developed at the University of Tennessee. While FT-MPI is capable of surviving the simultaneous failing of $n-1$ processes in an n processes job, it remains up to the application developer to

recover the user data, since FT-MPI does not perform any (transparent) checkpointing of user-level data items.
As a matter of fact, we present two parallel fault tolerant algorithms to solve the 3D reconstruction problem based on checkpointed data. For the first approach, the application process j has to store every K time steps its current solution $U_j^{n(j)}$. Additionally, the artificial boundary conditions $I_j^m = \Omega_j \cap \Omega_{j+1}$ have to be stored for all time steps $m < M$ since the last checkpoint. The solution U_j^M can then be reconstructed on the failed process with the forward time integration (2). Figure 1 demonstrates how the recovery works in one dimensional space. The vertical thick lines represent the boundary data that need to be stored, and the intervals with circles are the unknowns of the reconstruction process.

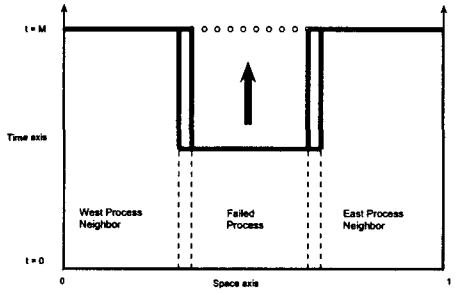

Figure 1: Reconstruction procedure in one dimension using forward time integration.

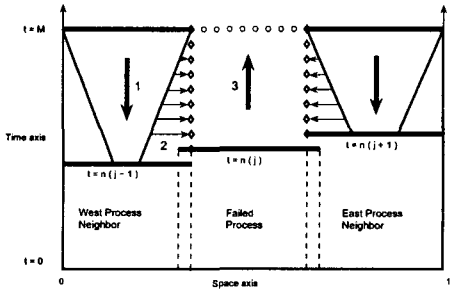

Figure 2: Reconstruction procedure in one dimension using explicit backward time stepping.

The major advantage of this method is that it is using the same algorithm as in the standard domain decomposition method. The only difference is, that it is restricted to some specific subsets of the domain. Thus, the identical solution U_j^M as if the process had no failures can be reconstructed. The major disadvantage of this approach is the increased communication volume and frequency. While checkpointing the current solution $U_j^{n(j)}$ is done every K time steps, the boundary values have to be saved each time step for being able to reconstruct the solution of the failed process(es).

The second approach does not require the storage of the boundary conditions of each subdomain at each time step, but it allows to retrieve the interface data by computation instead. It is divided in three steps: we first compute backward in time from the available data on main memory using the explicit formula provided by 2 (Step 1 in figure 2). Since this numerical procedure blows up after few time steps, one can use an hyperbolic regularization such as on the telegraph equation [5, 6] to stabilize the scheme. Then, to construct the solution outside the cone of dependencies and therefore to determine the solution at the subdomain interface, we used a standard procedure in inverse heat problem, the so-called space marching method [7] (Step 2 in figure 2). Finally, the failed process has its boundaries retrieved and is able to rebuild its lost data using the forward time integration (Step 3 in figure 2).
For the performance comparison between both methods, we have evaluated the two-dimensional version of (1) with three different problem sizes per processor ($50 * 50$, $100 * 100$ and $200 * 200$)

on four different processor configurations (9, 16, 25 and 36 processes). Figure (3-4-5-6) prove that saving the local solution each 9 time steps or saving additionally the interface do not make so much difference and bring only a small overhead (between 5% and 15%) on the overall execution time. Moreover, we simulated a process failure and measured the execution time required for respawning the failed process and to reconstruct the data of this process using the backward explicit scheme. The first results with 9 and 16 processes showed that the recovery time is in the order of 2% of the overall execution time which is quite promising for our future work in this area.
However, figure (7-8) show some performance got with the 3D version of the heat transfer problem and here, we clearly see the overhead of saving additionally each time step the interface especially with the large problem size. Thus, these results make us believe the most efficient reconstruction algorithm would be the backward explicit scheme which needs only the local solution to be checkpointed.
Therefore, we will present on the final paper a complete parallel fault tolerant **3D** heat transfer code using the explicit backward in time as the method to retrieve the lost data on the failed process(es). We will also simulate failures and show some timing measurements on how long it took the application to recover and to compute the lost solution.

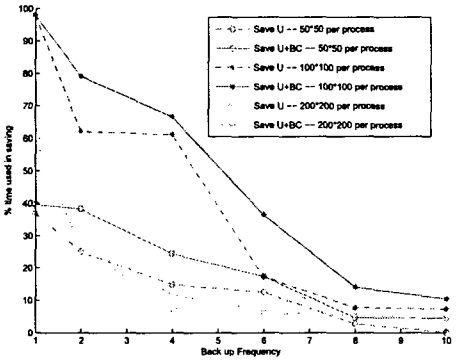

Figure 3: Asynchronous checkpointing overhead with 9 processes.

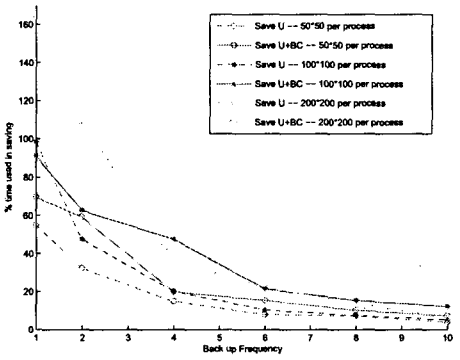

Figure 4: Asynchronous checkpointing overhead with 16 processes.

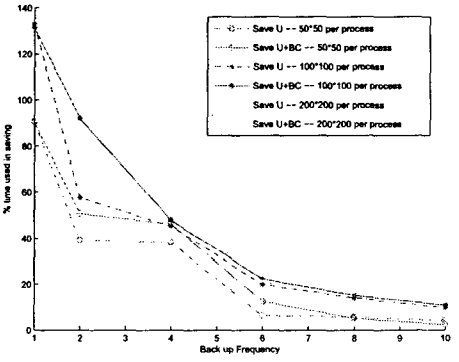

Figure 5: Asynchronous checkpointing overhead with 25 processes.

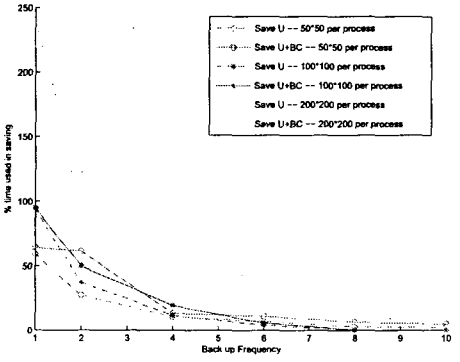

Figure 6: Asynchronous checkpointing overhead with 36 processes.

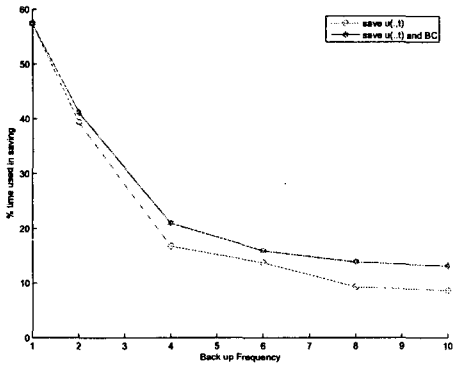

Figure 7: Asynchronous checkpointing overhead for the small 3-D test case.

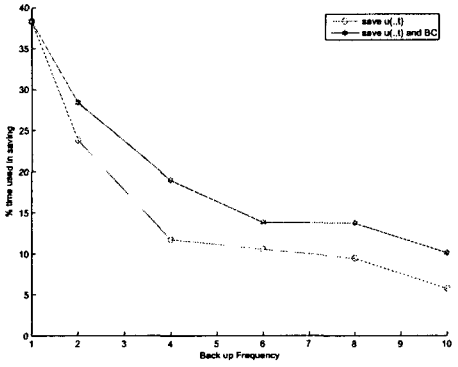

Figure 8: Asynchronous checkpointing overhead for the large 3-D test case.

References

[1] Sriram Sankaran, Jeffrey M. Squyres, Brian Barrett, Andrew Lumsdaine, Jason Duell, Paul Hargrove, and Eric Roman, The LAM/MPI Checkpoint/Restart Framework: System-Initiated Checkpointing, in *International Journal of High Performance Computing Applications*, 2004.

[2] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov, MPICH-V: Toward a Scalable Fault Tolerant MPI for Volatile Nodes, in SC'2002 Conference CD, IEEE/ACM SIGARCH, Baltimore, MD, 2002.

[3] Graham E. Fagg, Edgar Gabriel, Zizhong Chen, Thara Angskun, George Bosilca, Jelena Pjesivac-Grbovic, and Jack J. Dongarra, 'Process Fault-Tolerance: Semantics, Design and Applications for High Performance Computing', in 'International Journal of High Performance Computing Applications', Volume 19, No. 4, pp. 465-477, Sage Publications 2005.

[4] Beck, Dongarra, Fagg, Geist, Gray, Kohl, Migliardi, K. Moore, T. Moore, Papadopoulous, Scott, and Sunderam, 'HARNESS: a next generation distributed virtual machine', Future Generation Computer Systems, 15, 1999.

[5] M. Garbey, H. Ltaief, *Fault Tolerant Domain Decomposition for Parabolic Problems*, Domain Decomposition 16, New York University, January 2005. To appear.

[6] 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).

[7] D.A. Murio, The Mollification Method and the Numerical Solution of Ill-posed Problems, Wiley, New York (1993).