A CHARACTERIZATION OF A HYBRID AND DYNAMIC PARTITIONER FOR SAMR APPLICATIONS

Henrik Johansson
Dept. of Information Technology, Scientific Computing
Uppsala University
Box 337, S-751 05 Uppsala, Sweden
email: henrik.johansson@it.uu.se

Johan Steensland
Advanced Software Research and Development
Sandia National Laboratories
P.O. Box 969, Livermore, CA 94550, USA
email:jsteens@sandia.gov

Abstract

Significantly improving the scalability of large structured adaptive mesh refinement (SAMR) applications is challenging. It requires sophisticated capabilities for using the underlying parallel computer's resources in the most efficient way. This is non-trivial, since the basic conditions for how to allocate the resources change dramatically during run-time due to the dynamics inherent in these applications.

This paper presents a first characterization of a hybrid and dynamic partitioner for parallel SAMR applications. Specifically, we investigate parameter settings for trade-offs like communication vs. load balance and speed vs. quality. The key contribution is that the characterization shows that the partitioner is able to respond accurately to stimuli from system and application state, and hence adapt to various SAMR scenarios. This potentially reduces the run-time for large SAMR applications.

1 Introduction

Structured adaptive mesh refinement (SAMR) methods are being widely used for simulations of physical phenomena in domains like computational fluid dynamics [1, 14], numerical relativity [8, 16], astrophysics [4, 11], and subsurface modeling and oil reservoir simulation [13, 23]. Significantly improving the scalability of large SAMR applications is challenging. It requires sophisticated capabilities for using the underlying parallel computer's resources in the most efficient way. This is non-trivial, since the basic conditions for how to allocate the resources change dramatically during run-time due to the dynamics inherent in these applications.

The present contribution is a part of a larger research effort [5, 18, 19, 20, 21] that aims to improve performance for general SAMR applications executing on general parallel computers. The main motivation for our research is that *no single partitioning scheme performs the best* for all types of SAMR applications and systems. For a given application, the most suitable partitioning technique depends on input parameters and the application's run-time state [15, 18]. Adaptive management of these dynamic applications at run-time is necessary. The *meta-partitioner* [6, 7] provides such capabilities, by selecting and configuring the most suitable partitioner based on the current system and application state. Previous research has offered design, proofs-of-concepts, and evaluation of major components.

This paper provides a detailed study of `Nature+Fable` [18], a core component of the meta-partitioner. `Nature+Fable` is a partitioning framework, hosting a set of parameters. Different parameter settings lead to different behavior. The work presented here is an initial characterization of this parameter space. This characterization provides an early basis for conclusions about how to adapt `Nature+Fable` to various SAMR scenarios, by dynamic adjustment of parameters.

2 Partitioning of SAMR Applications

Dynamically adaptive mesh refinement (AMR) methods for the numerical solution to partial differential equations (PDEs) [2, 3, 17] employ locally optimal approximations, and can yield highly advantageous ratios for cost/accuracy when compared to methods based on a static uniform mesh. These techniques seek to improve the accuracy of the solution by dynamically refining the computational grid in regions with large local solution error. The resulting grid structure is a dynamic and adaptive grid hierarchy.

Most partitioners for SAMR grid hierarchies can be classified as either patch-based or domain-based. For *patch-based partitioners* [9, 24], distribution decisions are independently made for each newly created grid. These partitioners can achive a manageable load imbalance and a full re-partition at steps involving re-griding can be avoided. However, they suffer from communication serialization bottlenecks and an inability to exploit available parallelism both across grids at the same level and different levels [18].

Domain-based partitioners [12, 15, 22] partition the physical domain, rather than the grids themselves. The domain is partitioned along with all contained grids on all refinement levels. Inter-level communication is eliminated and the available parallelism is better exploited. The disadvantages are intractable load imbalance for deep hierarchies and the occurrence of "bad cuts" leading to increased overhead costs [18].

Because of their drawbacks, neither patch-based or domain-based partitioners can efficently cope with deep

grid hierarchies. Also, current methods lack the flexibility needed in order to function as a partitioner for general SAMR-applications that executes on general parallel computers.

`Nature+Fable` [18] is a hybrid partitioner [10, 12, 22] and it aims to be the best possible tool for partitioning SAMR grid hierarchies. It uses a 2-step partitioning approach. The first step uses domain-based techniques to generate coarse partitions, which are mapped to a group of processors. The second step uses a combination of domain and patch based techniques to optimize the distribution of each coarse partition within its processor group. `Nature+Fable` inherits the good properties of both the domain-based and the patch-based techniques at same same time as it avoids their main drawbacks. All involved parts of `Nature+Fable` are engineered to be components of the meta-partitioner. Thus, they offer carefully designed parameters to steer its behavior, enabling adaptation to the changing partitioning requirements imposed by applications and computer systems.

`Nature+Fable` separates homogeneous, unrefined (Hue) and complex, refined (Core) domains of the grid hierarchy (Fig. 1 The Hues contain the portions of the grid hierarchy without refinements; consequently they contain only parts of the base grid (refinement level 0). The Cores contain the portions of the grid where refinements are present. The Cores are separated from the Hues in a strictly domain-based fashion, meaning that each Core contains a portion of the base grid and all its overlaid, refined grids. The Cores are then clustered the into "easy-to-block" *bi-levels* [18]. A bi-level consists of two refinement levels - a partition of grid level k together with all its superimposed refinements on the next level. Expert blocking algorithms are finally used for the Hues and the bi-levels (the Cores). To achieve an optimal partition for an arbitrary application-computer combination, the whole process is controled by a large set of parameters.

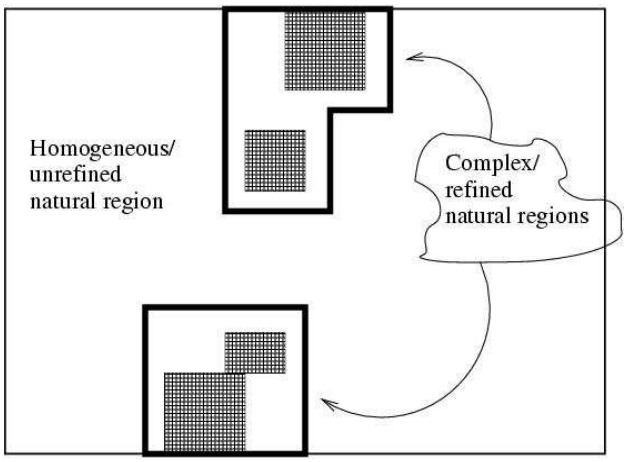

Figure 1. The division of the grid into Hues and Cores. Some unrefinement area around each Core is included. Note how both Cores consist of two separate refined areas.

To fully exploit the adaptation possibilities, the parameter space offered by `Nature+Fable` must be characterized with regards to the impacts of a given parameter — or a combination of parameters. Below, such a characterization is presented. It provides an initial basis for dynamically adapting `Nature+Fable` to various SAMR scenarios, within the concept of the meta-partitioner.

3 Experimental Setup

A suite of five "real-world" SAMR application kernels [19] was used for the characterization. Application domains included numerical relativity (Scalarwave - SC), oil reservoir simulations (Buckley-Leverett - BL), and computational fluid dynamics (compressible turbulence, Richtmyer-Meshkov - RM). These applications demonstrate different runtime behavior and adaptation patterns. The 2D (3D) applications used 5 (3) levels of factor 2 refinements in space and time. Regridding and redistribution was performed every 4 time-steps on each level.

The evaluation was performed using software that simulates the execution of the Berger-Colella SAMR algorithm for 32 processors. The performance of the partitioning configuration at each regrid step was computed using a metric [18] with the components load balance, communication, data migration and partitioning time. Here, communication was measured as the sum of the amount of interprocessor communication taken over all time-steps. Data migration was captured by the sum of the total number of data points forced to migrate as a result of re-partitioning, taken over all time-steps. Both communication and data migration were normalized with respect to grid hierarchy size and workload [20]. Partitioning times were measured as the time to partition an entire trace file, minus the time for the same program calling a dummy partitioning.

Since we use a simulator, it is not possible to obtain information about the real-world runtimes of our applications. This is unfortunate, but `Nature+Fable` is not at the current time fully integrated into our target framework. Even with this information, it would not be possible to compare our results with the ones obtained from other SAMR frameworks, as they depend on the application and the computer system. However, earlier research has shown that `Nature+Fable` , without an optimized parameter setting, performs as well as any other SAMR load balancing method [18].

Next, we turn to the `Nature+Fable` parameters. To isolate the effect of a given parameter, we based our experiments on a "default setting" and let only the parameter under study deviate from it. Our goal here is not to find an optimal parameter combination but to understand and exhibit how the individual parameters influence the results. Combining the parameters would also, in this context, lead to an unmanageable number of permutations. Our study included the following specific items (see [18] for a description of the parameters which are held constant):

Quantum The parameter `Q` affects the partitioning

granularity. A larger `Q` corresponds to the partitioner being able to create smaller blocks when splitting larger blocks. We studied this by varying `Q` between 2, 4, and 6.

Mapping As each bi-level is partioned individually, there is a possibilty that overlapping blocks, belonging to different bi-levels, are assigned to different processors. This leads to unneccessary inter-level communication between the processors. The mapping functionality [20] in `Nature+Fable` strives to reduce this communication by assigning overlapping boxes (with respect to different bi-levels) to the same processor. If the overlap is above a specified amount, the mapping is deemed good enough. We compared runs without mapping with runs with mapping and an overlap of 50 percent.

Blocking Mode Two blocking modes are available in `Nature+Fable` , Multiple and Fractional blocking. Multiple blocking tries to achieve load balance by creating more blocks in accordance with the idea of virtual processors. When the amount of processors is increased with a large number of virtual processors, it is easier to find a good blocking scheme. All blocks mapped to a subset of the virtual processors are then mapped onto a single physical processor.

Instead of scaling the problem, Fractional blocking admits a block to be split into a part that is smaller than a processor workload. For example, if a block has a workload suitable for 1,4 processors, it can be split into two blocks were one block gets the size of one processor workload while the remainder, 0.4 workloads, are placed into the second block. The blocking scheme then tries to find a matching block for the remainder. In this case it could e.g. be a block with a workload for 2.6 processors. This strategy thus results in much fewer blocks than Multiple blocking.

WhiteSpace When blocking a bi-level, the work load for the bi-level is estimated as if it would only consist of its highest level of refinement, since the large majority of the computations are performed on this level. However, parts of the bi-level that only has the lower level negatively influcences the workload estimation, as these areas are judged to have a higher workload than in reality (Fig. 2). Efforts are, of course, made to minimize these lower level parts during the creation of the bi-levels but the result can under certain conditions be less then satisfactory. The parameter `whiteSpace` determines how large fraction of a bi-level that is allowed to have only the lower level. A smaller value means that we cut the bi-levels into more boxes that are either strictly single- or bi-level.

4 Results

Quantum A large quantum (fine granularity) resulted in greatly improved load balance (Fig. 3). The downside was a small increase in communication, as a finer granularity means a greater number of blocks. A large number of blocks results in more and longer intersections between the blocks and hence more communication.

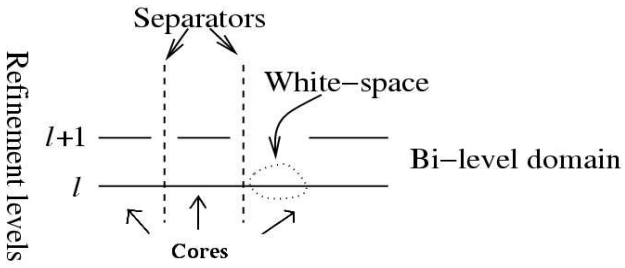

Figure 2. Illustration of the concept of WhiteSpace. The rightmost bi-level has a large fraction which only consists of the lower refinement level.

The decrease in load imbalance is at its largest when we move from Q=2 to Q=4. This indicates a pattern of diminishing returns as the granularity is made finer, at least for the applications and grid sizes studied in this paper.

Data migration exhibited a seemingly random behavior, as it decreased for some applications and increased for others. There does not seem to be any correlation between the changes in load imbalance and data migration. For the two applications with the largest decrease in load imbalance, BL2D and RM2D, the data migration has an opposite behavior, decreasing for one application and increasing for the other.

We conclude that a more detailed understanding of the effects of the parameter Quantum is of great importance for determining the trade-off between the different metrics. Our initial intuition told us that the main trade-off should be load imbalance versus communication but the results show that data migration can be of equal importance.

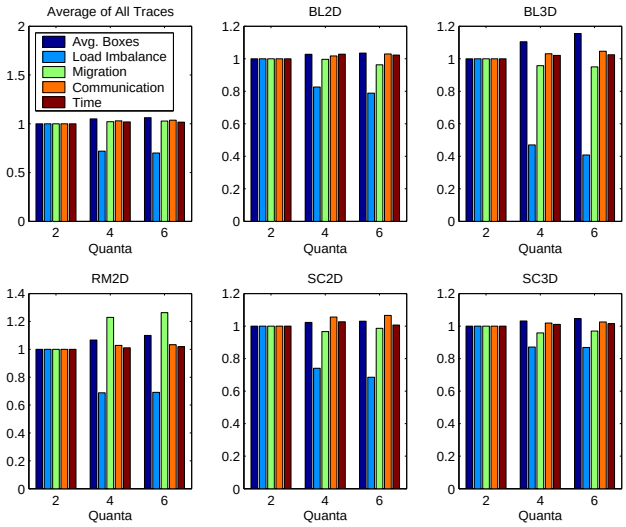

Figure 3. Results for `Q`. The scale is normalized with respect to `Q`=2. Note the large decrease in load imbalance for all applications.

Mapping The communication was reduced with at

least ten percent when mapping was turned on (Fig. 4). The amount of data migration was slightly increased, which was anticipated since a reordering is performed. The extra time invested in the mapping function was smaller than expected as it involves quite heavy computations. More research is however needed to determine the optimal use of the parameter, e.g. the impact of other amounts of mapping than the ones presented here. The load imbalance is never affected by the mapping parameter and it is not shown in the graphs.

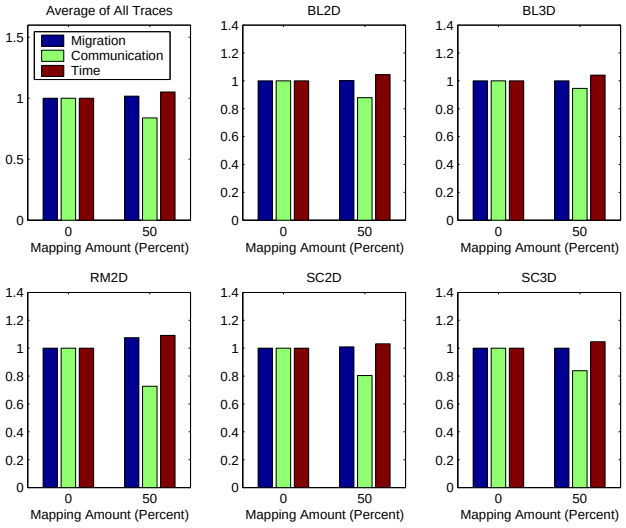

Figure 4. Results for `Mapping`. The scale is normalized with respect no Mapping. The time is only moderately increased with Mapping turned on.

Blocking Mode Multiple blocking reduced migration at the expense of longer partitioning time and more communication. (Fig. 5). Multiple blocking uses more boxes than Fractional blocking, which means that the time to order the boxes is increased. Each transfer of a block involves less data when multiple blocking is employed since these blocks are in general smaller. As the number of transfered blocks is not increased as fast as the number of created blocks, the data migration is decreased. The reduction was generally smaller than expected, but it was still substantial in some special cases (e.g. RM2D). However, the communication is higher when mutiple blocking is used, since number of blocks is increased. The load imbalance was roughly equal for both blocking modes.

WhiteSpace Varying the parameter `whiteSpace` resulted in some counter-intuitive effects. As can be seen in Fig. 6, the impact on the 2D and 3D versions of the BL application was different. A low amount of allowed whiteSpace resulted in a significant reduction of load imbalance for BL2D. The opposite was true for the 3D version (BL3D), where a larger amount of white space gave the best overall result.

Our intutition tells us that that when the amount of whiteSpace is decreased, the number of boxes should rise since we have separated the parts of the bi-level that only

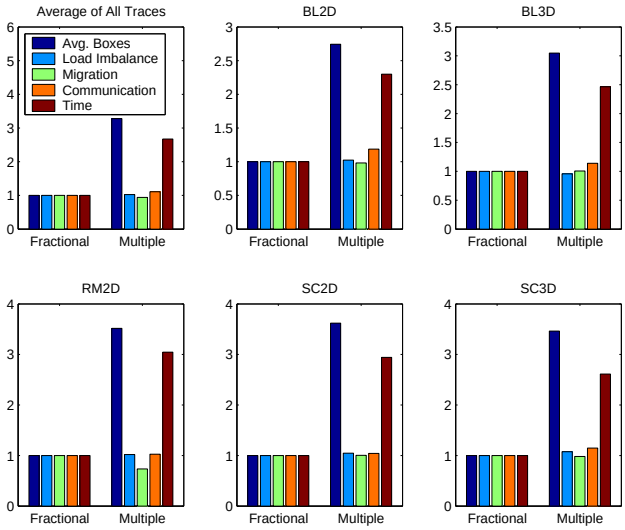

Figure 5. Results for Fractional versus Multiple blocking. The scale is normalized with respect to Fractional blocking. Note the difference in time and number of boxes

consist of the lower level. This is the case for RM2D, BL3D and SC2D, but for the first two of these three applications, the load imbalance fails to decrease as we would expect. It is only for SC2D that we can see the predicted behavior, a slight increase in communication and the number of boxes at the same time as the load imbalance is decreased.

The effects on data migration and partitioning time were of minor significant.

The cause behind the fluctuating behavior is not yet understood. Due to the large reduction in load imbalance for BL2D, it is obvious that a better knowledge about this parameter is vital.

5 Conclusions and Future Work

By varying one partitioning parameter at a time, we show that it is possible to achieve a significant improvement for every one of or metrics.

Ultimately, the presented research strives to describe how the parameter space relates to the general partitioning trade-offs being (1) communication vs. load balance, (2) speed vs. overall quality, and (3) data migration. These trade-offs constitute the basis for our classification space [21].

We found that trade-off (1) was heavily influenced by `Quantum`, trade-off (2) by `Mapping` and `Blocking Mode`, and trade-off (3) by `Quantum`, `Mapping` and `Blocking Mode`. The parameter `whiteSpace` affected all metrics. We conclude that an optimal parameter setting will lead to significant run-time reductions for arbitrary SAMR applications. The research presented here has given valuable insight of how to adjust some of those parameters.

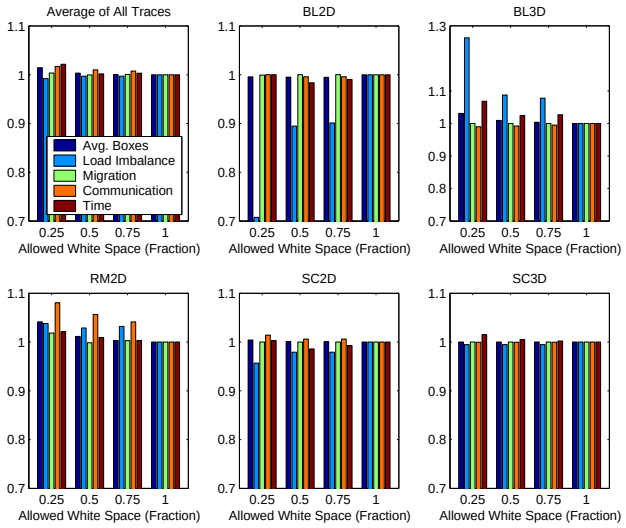

Figure 6. Results for `whiteSpace`. The scale is normalized with respect to `whiteSpace` 1.00. The behavior of BL2D and BL3D is significantly different.

The meta-partitioner [18] is our ultimate goal and it uses application and system state to determine the best combination of the trade-offs to decrease execution times. To function as a component of the meta-partitioner, `Nature+Fable` must be able to adapt to these recommendations. Our future research task is therefore to create an *abstract classification space layer* on top of the lower level function call to `Nature+Fable` . This abstraction layer should conform to the classification space described briefly above. The layer should host a front-end, providing input in form of a point in the classification space, and a back-end that calls `Nature+Fable` . In between the front and back ends, there should be an "engine" implementing translations based on the findings of the research.

References

[1] M. Berger, et al. Adaptive mesh refinement for 1-dimensional gas dynamics. *Scientific Computing*, 17:43–47, 1983.

[2] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82, 1989.

[3] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[4] G. Bryan. Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, pages 46–53, 1999.

[5] S. Chandra and M. Parashar. An evaluation of partitioners for parallel SAMR applications. *Lecture Notes in Computer Science*, 2150:171–174, 2001. Euro-Par 2001.

[6] S. Chandra, J. Steensland, and M. Parashar. An experimental study of adaptive application sensitive partitioning strategies for SAMR applications, 2001. Research poster presentation at Supercomputing Conference, November 2001.

[7] Sumir Chandra. ARMaDA: a framework for adaptive application-sensitive runtime management of dynamic applications. Master's Thesis, Graduate School, Rutgers University, NJ, USA, 2002.

[8] Mattew W. Choptuik. Experiences with an adaptive mesh refinement algorithm in numerical relativity. *Frontiers in Numerical Relativity*, pages 206–221, 1989.

[9] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing for structured adaptive mesh refinement applications. In *Proceedings of ICPP 2001*, 2001.

[10] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In *Proceedings of Supercomputing 2001*, 2001.

[11] M. Norman and G. Bryan. Cosmological adaptive mesh refinement. *Numerical Astrophysics*, 1999.

[12] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, 1996.

[13] M. Parashar, J.A. Wheeler, G. Pope, K.Wang, and P. Wang. A new generation EOS compositional reservoir simulator: Part II - framework and multiprocessing. *Proceedings of the Society of Petroleum Engineerings Reservoir Simulation Symposium, Dallas, TX*, June 1997.

[14] R. Pember, J. Bell, P. Colella, W. Crutchfield, and M. Welcome. Adaptive cartesian grid methods for representing geometry in inviscid compressible flow, 1993. *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 6-9.

[15] Jarmo Rantakokko. *Data Partitioning Methods and Parallel Block-Oriented PDE Solvers*. PhD thesis, Uppsala University, 1998.

[16] Hawley S. and Choptuic M. Boson stars driven to the brink of black hole formation. *Physic Rev*, D 62:104024, 2000.

[17] Jeffrey Saltzman. Patched based methods for adaptive mesh refinement solutions of partial differential equations, 1997. Lecture notes.

[18] Johan Steensland. *Efficient partitioning of dynamic structured grid hierarchies*. PhD thesis, Uppsala University, 2002.

[19] Johan Steensland, Sumir Chandra, and Manish Parashar. An application-centric characterization of domain-based SFC partitioners for parallel SAMR. *IEEE Transactions on Parallel and Distributed Systems*, December:1275–1289, 2002.

[20] Johan Steensland and Jaideep Ray. A heuristic remapping algorithm reducing inter-level communication in SAMR applications. In *Proceedings of The 15th IASTED International Conference on Parallel and distributed computing and systems PDCS03*, volume 2, pages 707–712. ACTA PRESS, 2003.

[21] Johan Steensland and Jaideep Ray. A partitioner-centric classification space for SAMR partitioning trade-offs : Part I. In *Proceedings of LACSI 2003, Los Alamos computer science symposium on CD-ROM*, 2003.

[22] M. Thuné. Partitioning strategies for composite grids. *Parallel Algorithms and Applications*, 11:325–348, 1997.

[23] P. Wang, I. Yotov, T. Arbogast, C. Dawson, M. Parashar, and K. Sepehrnoori. A new generation EOS compositional reservoir simulator: Part I - formulation and discretization. *Proceedings of the Society of Petroleum Engineerings Reservoir Simulation Symposium, Dallas, TX*, June 1997.

[24] Andrew M. Wissink et al. Large scale parallel structured AMR calculations using the SAMRAI framework. *In proceedings of Supercomputing 2001*, 2001.