REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	9 October 1995	Quarterly Technical 1 July 1995 - 30 Sept 1995

4. TITLE AND SUBTITLE
KQML - Accessible, High-Performance, Massive Knowledge Bases, Quarterly Technical Report

5. FUNDING NUMBERS
Grant Number
N00014-94-1-0907

6. AUTHOR(S)
James Hendler
Joel Saltz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Maryland
Department of Computer Science
College Park, Maryland 20742-3255

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Ballston Tower One
800 North Quincy Street
Arlington, Virginia 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
~~ONR, NRL~~ Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Research this quarter focused on developing generic HPC components motivated by the high performance knowledge base work. The attached extended abstract, accepted for presentation at the SIPAR Workshop on Parallel and Distributed Systems (Biel-Bienne, Switzerland, October, 1995) describes this work.

19960424 087

14. SUBJECT TERMS	15. NUMBER OF PAGES
Knowledge based systems, High performance computing, interoperability	8
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Quarterly Report: N00014-94-0907

PI: James Hendler Co-PI: Joel Saltz

University of Maryland, College Park

Contract Title: KQML-Accessible, High-Performance, Massive Knowledge Bases

Contract Number: N-00014-94-0907

Principal Investigators: James Hendler, Univ. of Maryland
Joel Saltz, Univ. of Maryland

ARPA Order: B399

1 Research Results

Research this quarter focused on developing generic HPC components motivated by the high performance knowledge base work. The attached extended abstract, accepted for presentation at the SIPAR Workshop on Parallel and Distributed Systems (Biel-Bienne, Switzerland, Oct. 1995) describes this work.

2 Grant Related Activities

2.1 Awards And Honors

Professor Hendler has been chosen as a member of the prestigious Defense Science Study Group run by the Institute for Defense Analysis. Dr. Hendler will be a member of the 1996-1997 Study Group.

2.2 Publications This Quarter

Gagan Agrawal and Joel Saltz, "Interprocedural Data Flow Based Optimizations for Compilation of Irregular Problems", *Languages and Compilers for Parallel Computing*, Aug 1995.

K. Stoffel, D. Sharma, J. Hendler, J. Saltz, Integrating task-parallel computations into data-parallel applications Proceedings of the SIPAR Workshop on Parallel and Distributed Systems, Biel-Bienne, Switzerland, October, 1995.

Manuel Ujaldon, Shamik D. Sharma, Joel Saltz, and Emilio Zapata, "Runtime Techniques for Parallelizing Sparse Matrix Applications," *Proceedings of the 1995 Workshop on Irregular Problems*, September, 1995.

J. Hendler, Artificial Intelligence: yesterday, today and tomorrow, Currents in Modern Thought, *The World & I*, July, 1995.

K. Erol, J. Hendler and D. Nau, "A Critical Look at Critics in HTN Planning", Proc. International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Aug, 1995.

V. Manikonda, J. Hendler, and P.S. Krishnaprasad "Formalizing Behavior-Based Planning for Nonholonomic Robots,"Proc. International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Aug, 1995.

V. Manikonda, P. Krishnaprasad, and J. Hendler, "A Motion Description Language and a Hybrid Architecture for Motion Planning with Nonholonomic Robots," Proc. International Conference on Robotics and Automation, Nagoya, Japan, 1995.

Integrating task-parallel computations into data-parallel applications

Kilian Stoffel Shamik D. Sharma James Hendler
Joel Saltz

Department of Computer Science
University of Maryland, College Park, MD 20742
{stoffel, shamik}@cs.umd.edu

Abstract

We examine the parallelization of data-parallel applications which have a significant task-parallel component. We describe some applications which exhibit this feature, and present the runtime techniques we used to efficiently parallelize the task-parallel component of such computations while retaining a data-parallel programming framework.

1 Introduction

Recently there has been an increased effort to integrate task and data parallelism, which are often seen as mutually exclusive approaches to parallel programming. This effort has been fueled by the need to parallelize important applications that require exploitation of both forms of parallelism.

Broadly, two types of mixed task/data-parallel applications can be readily distinguished. The first class includes those problems where separate data-parallel computations are coupled with a task-graph specifying the interactions. Problems of this sort include multi-disciplinary simulations and image-processing pipelines. Individual tasks are coarse-grained, and the data-parallelism is contained within each task of the task-parallel computation. In the second class of problems, a data-parallel program invokes a task-parallel subcomputation. This may be required if an application that is largely data-parallel includes a phase that requires the exploitation of task-parallelism. For example, in the ETMSP [2] code for transient stability analysis of electrical power grids, the loops that perform the triangular solution of the sparse network matrices have loop-carried dependencies. These loops constitute less that 10% of the total sequential execution time, but when the rest of the code is parallelized this becomes the bottleneck. The task-parallelism in these loops must be

exploited to achieve scalability. Similar problems arise in other sparse-matrix codes, knowledge-base search-engines as well as certain video-manipulation algorithms.

A principal reason why such applications have not been successfully parallelized despite the presence of substantial parallelism is the lack of system support for low latency communication and synchronization. However, with decreasing communication latencies we show that it is now possible to exploit such parallelism by applying runtime techniques that increase the granularity and reduce the synchronization overheads. We present two applications that need such support and describe how they were successfully parallelized.

2 Applications

2.1 Knowledge Retrieval

In data mining applications a knowledge repository is searched for information with the aim of obtaining as much information as possible. Examples of data-mining applications are WWW worms and search-engines for the inheritance networks of knowledge bases. In such knowledge-bases, the networks are represented by DAGs, with the links representing paths used to inherit information. The search on such a inheritance network can be very computationally intensive and is parallelizable. Such search engines have been implemented for AI knowledge bases [5, 3] and the parallelization techniques developed for these problems can be applied for WWW searches as well. In this paper, we have used the search engine of the PARKA knowledge-base as a sample application. The knowledge-base is represented as a DAG with hierarchical links representing inheritance relationships (such as IS-A). A sample query may be "Give all animals that have horns". This search traverses the DAG and keeps track of all nodes that satisfy the constraints.

2.2 Sparse Matrix Applications

Sparse matrices are used in numerous scientific applications. Many sparse-matrix operations are data-parallel (e.g. sparse-matrix vector products, matrix-matrix multiplication) while others are more amenable to fine-grained task-parallelism (e.g Gaussian elimination of triangular matrix). Since the loops which are most computationally intensive are usually data-parallel, such codes are usually written in a data-parallel fashion (say in languages in HPF). In such cases the loops which are not data-parallel must be executed sequentially. This has two negative effects. Firstly, the sequential loop soon becomes the bottleneck if parallelism is increased. Secondly, all data required to execute the sequential loop must be brought into a single processor - this severely restricts the problem size that can be solved. This necessitates the support for task-parallelism. As

an example of such applications, we have used a sparse triangular solve kernel, similar to the one used in [1].

3 Programming Model

Our programming model attempts to have as much of a data-parallel flavor as possible. This places restrictions on the types of task-parallelism we allow. We list these restrictions below.

1. The task-graphs and static and acyclic.
2. Each data item is updated by a unique task and all data-items updated by a given task are on the same processor. This allows an owner-computes strategy.
3. The same task-graph is repetitively used for different data-sets. This allows runtime preprocessing optimizations.

These restrictions are not severe and the two motivating applications we consider satisfy these restrictions.

The nodes of the task-graph are distributed across processors based on a user-specified distribution and the computation is distributed using the owner-computes rule. Since the task-graph computation is iterative, certain optimizations can be performed once in a preprocessing step and reused. One such optimization is to perform a topological sort of the DAG, thus dividing it into levels. As shown in Figure 1, the synchronization requirements can now be confined to the levels thus increasing the computation granularity. This method has been suggested in [4], and was used to parallelize sparse matrix codes with loop carried dependencies. However global synchronization does not work very well because the synchronization constraints affect the load balancing. All processors have to wait for the slowest processor at each level. The total computation time is $\sum_{i=1}^{N} \max(Comp(level_i))$.

The alternative approach, considered in [1] is to use better distribution mechanisms to increase locality and then use low-latency active messages to communicate the data that does need to go off-processor. The arrival of data automatically triggers computation, thus the synchronization is implicit. The synchronization requirements of this method are very relaxed and thus the load-balance is better. The total computation time is $\max\left(\sum_{i=1}^{N} Comp(level_i)\right)$. While the computation time is better than the level-synchronized scheme, the communication/synchronization time may be worse since more messages are sent, even though the amount of data sent is the same. [1] found that high efficiencies could be achieved on the CM-5 using its very low overhead communication support. A drawback is that a dataflow programming model must be used.

```
DO K = 1, num_levels
  MPI_scatter( level(k-1))
  MPI_gather( level(k))
  COMPUTE( level(k))
END DO
```

Figure 1: Collective synchronization

```
DO K = 1, num_levels
  id = FZY_scatter( level(k-1))
  WHILE ( FZY_recv(id, data)
        == LEVEL_NOT_DONE) )
    COMPUTE( level(k))
END DO
```

Figure 2: Fuzzy synchronization

Our approach is to use the level-based data-parallel model but relax the synchronization constraints by using split-phase synchronization mechanisms. This allows processors to continue processing incoming data from the next level while waiting for the slowest processor from the previous level. Since each processor does not progresses to the next level until it has processed all nodes at the current level, the skew is limited to a maximum of one level. This is most useful when a processor can have a high load at one level and a low load at the next level. In such a case, that processor can catch up without slowing the others down.

Figure 3 shows how a DAG could be parallelized using such a *fuzzy barrier*. Each processor sends out all the outgoing data at the end of each level, but processes each incoming data-chunk as soon it arrives. The condition NOT_DONE remains true until all the incoming messages from a level have arrived, or when a termination condition has been detected. Thus even if processor A is on level X, processor B can begin to process data on level X+1 (until it eventually waits to receive A's message). Though, we do not discuss it here, the termination detection check can be incorporated to implement branch-and-bound algorithms. The computation time is thus better than that of the level-based scheme with no extra communication overhead.

4 Results

The results presented here are for the Knowledge Base application. [*The numbers for the Sparse Matrix application are similar and will be included in the extended version.*]

Figures 3 and 4 shows the results of using fuzzy synchronization and the

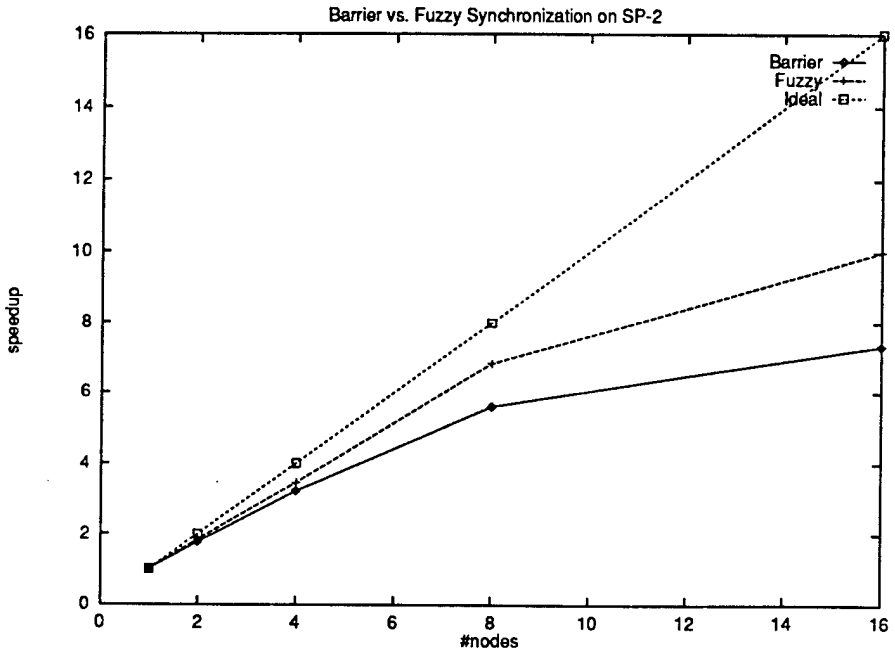

Figure 3: Barrier vs. Fuzzy Synchronization

level-based runtime techniques on the inheritance network code. These experiments used using an inheritance network with 500,000 classes and instances (i.e the DAG has 500K nodes). As can be seen from 3, fuzzy synchronization provides significant benefits over using collective synchronization. Figure 4 shows that high efficiencies can be obtained over a wide range of platforms using these techniques. The efficiencies are around 70%.

Acknowledgements

This research was supported in part by grants from ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), the ARPA I3 Initiative (N00014-94-10907) and ARPA contract DAST-95-C0037. Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384).

References

[1] Fredric T. Chong, Shamik D. Sharma, Eric Brewer, and Joel Saltz. Multi-

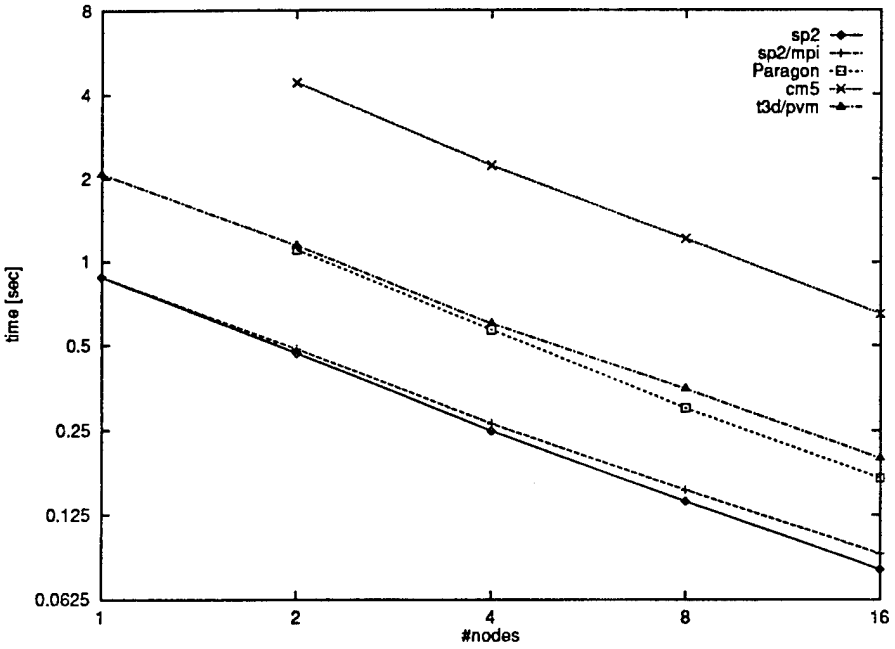

Figure 4: Performance of parallelized Inheritance Network code

processor runtime support for fine-grained, irregular DAGs. Technical Report CS-TR-3266, University of Maryland, Mar 1994. To appear in Parallel Processing Letters.

[2] Analysis of performance accelerator running ETMSP. Technical Report TR-102856, Performance Processors, Inc., Palo Alto, California 94301, October 1993. Research Project 8010-31.

[3] Douglas B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems.* Addision-Wesley, 1991.

[4] Joel H. Saltz, Ravi Mirchandaney, and Kay Crowley. Run-time parallelization and scheduling of loops. *IEEE Transactions on Computers*, pages 603–611, 1991.

[5] Kilian Stoffel, William A. Anderson, and James Hendler. Parka: Support for extremely large knowledge bases. In *Proceedings of KRUSE Symposium: Knowledge Retrieval, Use and Storage for Efficiency*, 1995.