Appl. Math. Lett. Vol. 3, No. 3, pp. 43–46, 1990
Printed in Great Britain.

0893-9659/90 $3.00 + 0.00

Balanced Computation of Two-Dimensional Transforms on a Tree Machine

[1]A.L. Gorin, [2]L. Auslander, and [2]A. Silberger

[1]AT&T Bell Laboratories
[2]CUNY Center for Large-Scale Computation

(Received September 1989)

1. Introduction

This paper describes the idea of a balanced parallel algorithm, and then applies it to the design and analysis of 2D transforms on a tree-structured parallel computer. A balanced algorithm is one that is written as a sequence of separate computation and communication stages, which is furthermore sized so that these stages require equal time. An advantage of writing an algorithm this way is simplicity of representation, which implies ease of programming. This representation furthermore allows one to analyze separately the complexity of computation and interprocessor communications in a parallel algorithm, greatly simplifying the performance modeling problem. If, based on such performance models, the stages are sized to require equal time, then one can pipeline and overlap multiple tasks to achieve (theoretical) 100% utilization of both the processors and communications network.

We apply these ideas to the design and analysis of 2D transforms on a tree machine. In particular, we consider transforms on square matrices which are defined via application of an identical operator to each row, followed by application of the same operator to each of the resultant columns. A classic example is the 2D Fast Fourier Transform (FFT), which finds application in image processing [6] and radar [1]. We also consider a binary tree of processors, in which each processing element (PE) is connected to a parent and two children, excepting for the root and leaves of the tree [2], and whose external communications is through the root PE. Such an interconnection is scalable and has low overhead communications, which yields implementations well suited to VLSI and advanced packaging [5, 3]. A tree is also universal, in the sense that it can be embedded within any lattice of PEs in reconfigurable networks [7, 9], such that the communications diameter of the network is preserved.

This paper proceeds as follows. We first describe the computing paradigm and software constructs which underly balanced 2D transform algorithms on a tree machine. We then describe the parallel algorithm and derive formulas expressing processing and communication times as a function of problem and device parameters. We then derive a criterion for a balanced transform, and as an example predict execution time for a balanced 1024×1024 2D FFT on a tree with a 20 Mhz communications skeleton that compares favorably to measured performance on a CRAY Y-MP. Although this is a comparison of projected performance on a theoretical machine with measured performance on a real machine, it illustrates that supercomputer speeds are achievable with intelligent exploitation of parallel processing.

2. Computing Paradigm

Consider parallel algorithms that can be written as a sequence of separate processing and communication stages. Such a parallel program is host-orchestrated, with a barrier synchronization between each stage. Further consider a processing paradigm in which identical

programs are executed simultaneously in each PE on different data sets, denoted a *sliced procedure*. The multiple executions of this single program can follow different instruction streams (still within the program, of course) depending on the data. These potentially different streams are initiated by broadcasting a jump address to all PEs, which are forced to converge and synchronize at the completion of the sliced procedure. This concept can be described as single program multiple data (SPMD), which can be alternatively viewed as coarse-grain SIMD or data-driven MIMD. We define two global burst communication functions which will support the 2D transform algorithm. *Broadcast* transmits data from the host to enabled PEs, and *Report* transmits data from a distinguished PE to the host. For a tree machine with K PEs (of depth $\log_2 K$) in which interprocessor communications between adjacent PEs requires time c per data element, then a broadcast or report of D data elements requires time $cD + \log_2 K$. For large data blocks, this simplifies to time cD.

3. PARALLEL ALGORITHM

Consider a 2D transform on $N \times N$ points, where the array of values is initially stored row by row in the host memory. After processing, the tree will return to the host the transformed array of values in which column follows column. We assume a tree with K PEs, where if $L = N/K$ then each PE has memory sufficient to store and compute L one-dimensional transforms. The parallel algorithm is as follows.

(1) Load the N rows into the machine, putting the Kth set of L rows into the PE $\#K$. This can be accomplished via the proposed software architecture by, for each K, broadcasting the block of LN data elements to PE $\#K$.
(2) Execute a sliced procedure concurrently in each PE that computes the N-point transform on each of the L rows resident in that PE.
(3) Perform a global transposition so that the Kth set of L columns is now in the Kth PE. This can be accomplished via the proposed software architecture by, for each K, first reporting the transformed rows from PE $\#K$ to the host; second broadcasting this block of LN data elements to all PEs, whence each PE then selects and saves the relevant partial columns from that block.
(4) Execute a sliced procedure concurrently in each PE that computes the N-point transform on each of the L columns resident in that PE.
(5) Transmit the transformed columns back to the host by sequentially reporting blocks from each PE.

4. PERFORMANCE MODELS

We treat separately processing (steps 2 and 4) and communications (1, 3, and 5). If a one-dimensional transform on N points requires time $F(N)$, then steps 2 and 4 each require time $LF(N)$. In particular, if the FFT on N points requires time $pN \log_2 N$, where p is the butterfly rate of the PE, then steps 2 and 4 each require time $p(N^2/K) \log_2 N$. For the analysis of communications complexity, we assume that block sizes (N^2/K) are sufficiently large that startup and shutdown times are negligible in comparison to the block transfer times. Step 1 requires time cN^2, as does step 5. Step 3 requires twice that, time $2cN^2$, since the data is first reported and then broadcast. Thus, total execution time is $2F(N)N/K + 4cN^2$ for a generic 2D transform, and time $2p(N^2/K) \log_2 N + 4cN^2$ for the 2D FFT.

5. BALANCED EXECUTION

If the algorithm is sized, via adjusting the number of PEs, so that $2cN^2 = p(N^2/K) \log_2 N$, then the five stages require times in ratio 1:2:2:2:1. This balancing criteria implies that the number of PEs is, for the 2D FFT, $K = (p/2c) \log_2 N$. Given a continuous channel which provides a sequence of transforms to be computed, then we have produced a balanced algorithm with equally sized processing and communication stages, (the final communication stage of one block plus the first communication stage of the subsequent are abutted.) This one channel utilizes both the processing and communications at 50%

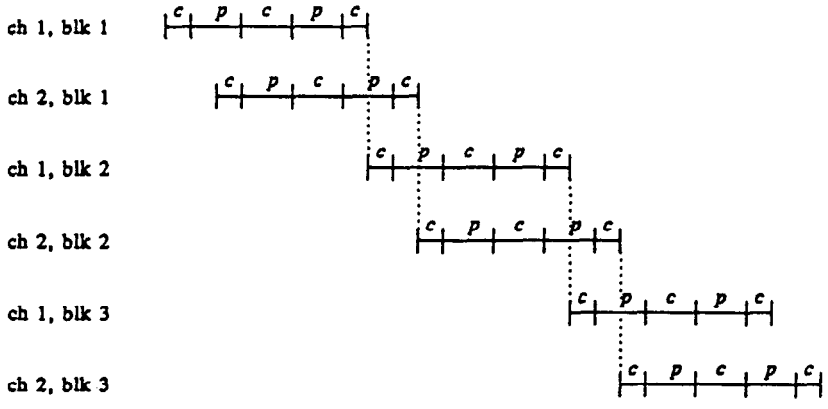

Figure 1. Pipelined execution of two-dimensional FFT yielding 100% utilization of processing and communications. Notation: c denotes a communications stage; p denotes a processing stage; (ch i, blk j) denotes block j of channel i.

capacity in a highly regular manner, computing a new transform at time intervals $8cN^2$. Thus, as shown in Figure 1, the processing of two such channels can be overlapped to achieve 100% utilization of both the processing and communications network, computing a new transform at time intervals $4cN^2$. This assumes that each PE can simultaneously process and communicate.

6. An Example

Consider a tree machine whose PE comprise a typical high-performance microprocessor such as the AT&T DSP32C, which is a C-programmable digital signal processor. As reported by Shear [8], the time to execute a 1024-point complex FFT is 3.2 ms. Assume a moderate-bandwidth communications skeleton that achieves interprocessor communication times of $c = 100$ ns per complex data element, (e.g., a 32-bit wide port clocked at 20 Mhz.) Based on the balanced algorithm analysis, such a machine could compute a stream of 1024 x 1024 2D FFTs at time intervals of 400 ms using 16 PEs. For reference, it is worth comparing this predicted time to measured experimental performance on a CRAY Y-MP8/832, which is 484 ms [4]. Although this is a comparison of projected performance on a theoretical machine with measured performance on a real machine, it does serve to emphasize that supercomputer speeds are achievable with intelligent exploitation of parallel processing.

7. Conclusions

We have described the idea of a balanced parallel algorithm and applied it to the design and analysis of a 2D transforms on a tree machine. The analysis shows that intelligent use of parallel processing can provide supercomputer performance, predicting execution time for a 1024 x 1024 2D FFT comparable to that of a CRAY Y-MP.

References

1. D.A. Ausherman, et al., Developments in Radar Imaging, *IEEE Trans. on Aerospace and Electronic Systems* **AES-20(4)**, 363–400 (July 1984).
2. S.A. Browning, *The Tree Machine: A Highly Concurrent Computing Environment*, Ph.D. Thesis, CalTech, (Jan. 1980).

3. A.L. Gorin and R.R. Shively, The ASPEN Parallel Computer, Speech Recognition, and Parallel Dynamic Programming, In Proceedings of The International Conference on Acoustics, Speech and Signal Processing (ICASSP), (April 1987).
4. C.M. Grassl and C. Kerr, private communication.
5. C.E. Leiserson, *Area-Efficient VLSI Computation*, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, (1981).
6. W.K. Pratt, *Digital Image Processing*, Wiley, (1978).
7. D.K. Pradhan, Dynamically Restructurable Fault-tolerant Processor Network Architectures, *IEEE Trans. on Computers* c-34 (5) (May 1985).
8. D. Shear, EDN's DSP Benchmarks, *EDN*, 126–148 (Sept. 1988).
9. R.R. Shively and A.L. Gorin, A Reconfigurable, Fault-Tolerant Systolic Signal Processor, *Proc. ICASSP* (May 1989).

[1]AT&T Bell Laboratories, Murray Hill, New Jersey, USA
[2]CUNY Center for Large-Scale Computation, New York, New York, USA