Use of Transputers in a 3-D Positron Emission Tomograph

M. Stella Atkins, Donald Murray, and Ronald Harrop, *Member, IEEE*

Abstract—**The use of a VME-based transputer network as a parallel processing engine for positron volume imaging is discussed. We find that the speedups of parallel networks depend on two major factors: the ratio of computation to communication for a task, and the size of the task, and we give a simple model to explore the limits to speedups. Through actual implementation we show that real-time PVI data acquisition can be achieved with about 20 transputer nodes, and we estimate that 3-D image reconstruction can be achieved within 10 min using 200 nodes. Larger images and a larger number of histograms can readily be accommodated using the same parallel algorithms as our model places no limits to the size of the images. The versatility and scalability of transputers makes them very suitable for use in PVI tomographs in that the same transputers can be used for speeding up data acquisition, image reconstruction, and display.**

I. Introduction

In the new positron volume imaging (PVI) tomographs, considerable computation power and communication bandwidth are required to acquire, reconstruct, and display data for true 3-D images.

First, the raw events must be acquired and preprocessed in real time, to permit data compaction and to histogram the events as the first phase of image reconstruction. This requires a sustained performance of about 18–20 MFlops [1]. In 2-D tomographs, special-purpose hardware is used to sort the 2-D data into histograms, but for PVI the data acquisition should be flexible enough to accommodate different 3-D reconstruction algorithms so a software-based system is more suitable.

Second, compute-intensive image reconstruction algorithms using the resulting histograms must reconstruct the data into a 3-D image array. Algorithms for 3-D reconstruction of PET images are still being actively researched [2]–[9]. We are experimenting with suitable sample sizes and filters, as well as with the basic algorithm (X-ray or Radon-plane based) and with scatter correction techniques [10]. We find that computer memory constraints still dictate the sampling intervals and image size, and although computer memory is becoming cheaper, we will probably always need more, for, as the resolution of detectors becomes finer, finer sampling will be required. We also find that although computing power is increasing (the new Intel 80860 chip can operate at a sustained 40 MFlops), the computing requirements are increasing faster—for 3-D image reconstruction in 5 min, we estimate at least 400 MFlops are required, since using our algorithm [8] without any scatter correction, it takes around 90 000 s on a 1 MFlop machine to reconstruct a 3-D image from a file of data. We therefore take the approach that a 3-D tomograph must have a flexible system structure, so that the software can exploit new and future hardware such as fast chips and buses as they are invented.

Third, the 3-D image array must be displayed for the end-user. Ideally the image can be rotated, segmented, and sliced in arbitrary directions in real-time, requiring much I/O and computing power.

One approach is to have a single processor with the desired performance, but it does not scale and is expensive. We choose to use many smaller processors working together [1], [11], [12]. This maintains the flexibility of software control and scalability while at the same time being reasonably priced. Two classes of processors are used: a single controller processor (*CP*) which controls the data flow, and multiple transformation processors (*TP's*) which perform the actual data processing. In this way, real-time performance can be achieved by using the required number of *TP's* and, of importance in a research environment with limited funding, each extra processor can provide an incremental performance gain towards the goal of real-time reconstruction.

Our hardware platform is a SUN-4 workstation acting as a *CP*, connected via its VME-bus to a low-cost high-performance network of T800 transputers [13] acting as *TP's* as in Fig. 1. Each T800 transputer node contributes 1 MFlop of computing power, and the power scales well with the number of nodes because there are four high-speed links connecting the nodes and each link operates autonomously from the node's CPU (using direct memory access). Future versions of the transputer will operate at 10–20 MFlops [14], and the parallel processing software described in this paper will scale to future requirements. We wish to emphasize, though, that the software concepts explained here apply to a much wider context than some particular model of transputers.

Manuscript received November 16, 1990; revised April 18, 1991. This work was supported by the Natural Science and Engineering Council of Canada under Grant A8020 and by the Center for Systems Science at Simon Fraser University. This paper was presented at the 1990 IEEE Nuclear Science Symposium, Medical Imaging Conference, Crystal City, Arlington, VA, October 22–27, 1990.

The authors are with the School of Computing Science, Simon Fraser University, Burnaby, B.C., V5A 1S6, Canada.

IEEE Log Number 9101379.

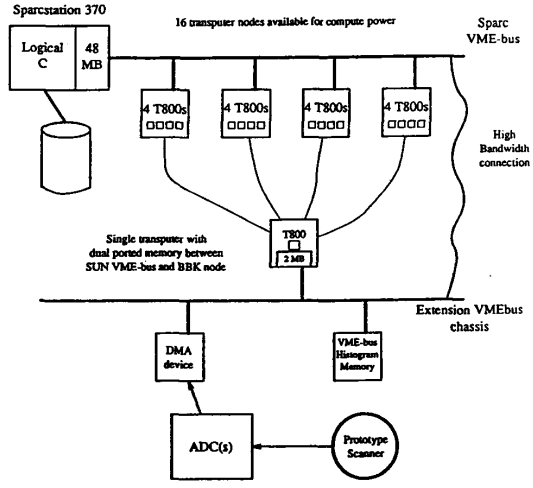

Fig. 1. Prototype architecture showing transputer boards.

Various transputer connection topologies and corresponding parallel algorithms are being developed to determine how best to utilize cost-effectively the transputers' CPU and I/O power.

The simplest topology treats each transputer as a separate *TP* directly accessible from the *CP*, with each transputer needing to have a direct interface to the VME-bus. This model for using the transputers matches the mode of operation of the system when standard processors such as the Intel 80860 process the events.

An alternative topology, configured as in Fig. 2, uses a single *MASTER* node connected over the VME-bus to the *CP* and the data acquisition system. The *MASTER* passes input data to the *WORKER* nodes, which process the data completely before returning the result to the *MASTER*. Such *MASTER/WORKER* configurations are called *processor farms* [15]. The advantage of this kind of topology is that only the *MASTER TP* needs an (expensive) VME-bus connection.

We found that several factors affect the performance of the transputer networks, the most important of which are the ratio of computation to communication at the transputer nodes (the comp/comm ratio), and the computation granularity, which is defined as proportional to the size of the input data packets transferred between the transputers. Another factor is the practical limit to the size of packets, for they have to fit within the memory of the node.

We developed analytical models of transputer topologies to determine the limitations to the possible speedups for different comp/comm ratios and computation granularities. As the granularity decreases, speedup with increasing numbers of processors is reduced from optimum by the computational overhead in the *MASTER* and *WORKER* nodes in distributing and receiving the data packets from their neighbor nodes. Also, as the comp/comm ratio decreases, speedup with an increasing number of processors is limited by the communication requirements causing saturation of the *MASTER's* links. We consider these problems to be of paramount importance in parallel programming, and we address these issues in this paper. We used the experimental data obtained from parallel data acquisition to derive a detailed analytic model [12]. The model has been experimentally validated with different computation granularities and different comp/comm ratios at the *WORKER* and *MASTER* nodes arranged in a multilevel hierarchy of up to 16 nodes. Nearly linear speedups over a single node are possible, provided that the comp/comm ratio is high enough, and the unit of computation performed on each node is long enough to overcome system overheads.

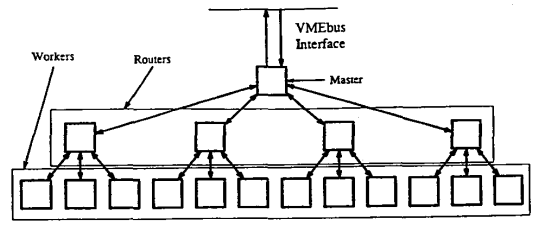

Fig. 2. A transputer processor farm of 17 nodes.

Section II describes how we configured and used the transputers to speed up data acquisition where the computation granularity is easily controlled, memory provides no major constraints, and the comp/comm ratio is relatively low. Although the use of multiple transputers for the purpose of data acquisition may not be cost-effective compared to hardware solutions [16], or with using a very high-performance chip, we show in Section III that the same transputers can also be used to speed up image reconstructions, thus providing flexible, scalable processing power to the desired performance level. Section III discusses the potential for using transputers in 3-D image reconstruction in our prototype 3-D tomograph, where memory limitation forces bounds on the computation granularity, but where the comp/comm ratio is relatively large and there is scalability to over 200 transputers. Section IV presents our conclusions—namely, that parallel processing using low-cost transputers can speed 3-D PET image reconstruction times to an acceptable time of around 5 min.

II. Evaluation of Transputers for Parallel Data Acquisition

A. Transputer Overview

The transputer T800 is a RISC processor which was introduced by INMOS. It is essentially a computer on a chip, complete with memory, four high-speed data links, central processor, system services, and memory interface [17], thus allowing parallel processing networks to be built easily and economically. Although there is no memory shared between any two nodes in a transputer network, efficient message passing is possible since the transputer allows communication and computation to occur simul-

taneously and autonomously. As each transputer has four datalinks with which to pass data to other nodes, a large number of topologies can easily be built. This connection flexibility gives the system designer many topology choices for exploiting the parallelism of a problem. The transputer contains a micro-coded scheduler which has a context switching time of less than 1 μs. [18]. This allows multiprogramming to be used with little performance degradation [19] when compared to traditional multiprocessing systems such as UNIX.

The software used for the investigation performed here was developed using Logical Systems C compiler [20]. This C system possesses a library which permits the concurrency features of the transputer to be easily exploited.

B. Transputer Topologies

We based our evaluation of transputers to fill the role required of the *TP*'s for event data acquisition by studying six topologies [12], of which two are described here. The first does not make any use of the *inter*-transputer links but rather treats each transputer as a separate *TP*. The second uses one transputer (*MASTER*) to distribute raw events to nine other transputer nodes (*WORKERS*) which translate the input data to output data and return it.

In all tests a bufferful of unprocessed events is initially located in an input buffer as would be the case in our data acquisition system. The single transputer scheme simply processes the events and places the results into the output buffer. When the events are processed by the multitransputer schemes, the events are initially located in a single input buffer, located on the *MASTER*. They are then distributed to the other nodes as small packets of events, *tasks*, until the input buffer is empty at which time an "end" message is sent to all transputers. When the transputers get this message they finish processing any remaining tasks and send results immediately back to the *MASTER* which contains the output buffer. The *MASTER* then distributes another bufferful of data, divided into many tasks, and the cycle is repeated.

C. Single Transputer Scheme

In this scheme each transputer is a separate *TP*, and must have a direct (costly) connection to the VME-bus of the host *CP*. This is analogous to the system proposed in [1] which uses 68020/68881's as *TP*'s. Overhead is minimized, but this scheme does not scale beyond 15 transputers for a 21-slot VME-bus, as the six remaining slots are occupied by other devices.

D. Master/Worker (Multilevel Hierarchy)

In this scheme, our ten-available transputers were used in a nearly balanced tree, similar to the 17-node farm shown in Fig. 2. In the data acquisition system only the *MASTER* communicates with the *CP*, this being done via the VME-bus interface which consists of dual-ported RAM enabling input and output buffers to be placed directly on the *MASTER*. Most of the work is done by *ROUTERS* and *WORKERS*. *ROUTERS* are needed to route the events to the lower levels as well as perform local processing. The *WORKER* transputers communicate only with the *ROUTER* above them. When this scheme is viewed from the *CP* the transputers appear as a single *TP*–*ROUTER's* and *WORKER's* being hidden from it. This gives the system designer more flexibility as now it is only required that the *MASTER* be mounted on the VME-bus. The *WORKERS* and *ROUTERS* can be mounted in any chassis that is capable of giving them power. In order to get the maximum performance from this scheme we need to use the transputers' feature of overlapping of communication with computation to minimize delays so that one task is being processed while the preceding task's results are being returned and the next task's data are being received, as detailed in [12]. Therefore, several processes per transputer are required, to perform the I/O on each link, and to do the actual computation on the input data.

E. Performance Results

The performance results given below were achieved with the different topologies for processing the events as described in [1]. For each test, 25 600 events were processed, this being the largest size which the memory of the test system's *MASTER* could accommodate. Using transputers of the type available to us, our aim was to design a topology appropriate for the processing of 120 000 events per second, requiring 18 MFlops performance. This is somewhat lower than the peak processing rate (often 300–500K events/s) which would be required for commercial tomograph operation [21]. On the other hand the rate is high enough for realistic consideration of the nature of the parameters involved in the topology and its application. The new H1 transputers [14] will be ten times more powerful than our current T800's, so with those transputers a processing speed of 1.2M events per second should easily be achieved.

The timing results were obtained by reading the transputers' real-time clock just before processing of the events began and again just after processing of the events finished.

1) Single Transputer: We found that 6944 events/s can be processed by a single transputer, which is almost 2.5 times faster than by a single 68020/68881. It would thus require approximately 18 transputers directly connected to the VME-bus to process 120 000 events/s ignoring additional overhead in the *CP*. As this is not feasible on a single VME-bus, additional hardware would be required to implement this approach. The overhead within a single transputer is negligible. The 6944 events/s value is used as a basis for all other topologies to measure the overhead required to distribute events between transputers.

2) Master/Worker: In multitransputer topologies the events are distributed between the nodes in packets (*tasks*) which contain anywhere from 1 to 1024 events per packet with the packet size being fixed during any single test.

TABLE I
RESULTS FOR MULTILEVEL TEN TRANSPUTERS

Task Size (# events)	(events/s)	Speedup
1	25 839	3.7
2	35 335	5.1
4	43 290	6.2
8	49 261	7.1
16	51 546	7.4
32	58 139	8.4
64	60 606	8.7
128	60 240	8.7

We found that the packet size (which is proportional to the computation task granularity) had a large impact on performance, with the optimal size depending on the number of events being processed and the number of nodes in the topology. If the granularity was too fine then the computational overhead to handle the packets degraded performance, and if it was too coarse then the communication time taken to download the initial packets was never overcome by the smaller packet overhead, again causing performance to degrade. Thus, the packet size must be chosen to balance the comp/comm ratio in order to get the best performance. We present in Table I the results for ten transputers configured in a multilevel hierarchy. The optimum task size was 64 events/task.

The observed speedup of 8.7 using ten transputers is very good. We later measured the performance on our enlarged network of 16 transputer nodes and observed a speedup of 13.3 times. Our detailed model [12] shows that with the observed comp/comm ratio of 4.15 the required event throughput can be met by 20 transputers, arranged in two trees rather than in a single tree because beyond 17 nodes arranged in a single tree the *MASTER*'s links become communication bound. Thus, the overhead to distribute events results in the requirement of two extra processors to meet the data acquisition needs when compared to the single transputer approach.

There are, though, two important advantages of this topology over the previous scheme: only the two *MASTER*'s need direct connection to the VME-bus, and this scheme is able to fit within a single VME-bus chassis.

III. PARALLEL 3-D IMAGE RECONSTRUCTION

A. Introduction

In this section we outline our scheme for parallel 3-D reconstruction, and estimate the performance gains for our 3-D reconstruction algorithm.

The problem of parallel 3-D image reconstruction for CT images has recently been researched by Chen *et al.* [22] using a 32-node hypercube. Like Chen, we find that node memory current limits our ability to store the complete image at each node, but unlike Chen, we do not allow this to restrict our image size (e.g., to $95 \times 95 \times 95$ voxels). Instead, our software provides for any projection size and any image size, reconstructed on any number of nodes, with only two assumptions: namely, that there is sufficient mass storage available in the transputer nodes to hold the complete image, and that there is no bottleneck in transferring the image and histogram data between the transputer network and the host *CP*. The first assumption is easily realized, as over 1 Mbyte of intermediate image voxels can be stored at each transputer node during processing, requiring only eight nodes to hold the complete image. The second assumption may be harder to meet, as our architecture uses an extendible 16 Mbyte VME-based memory board dual-ported between the *MASTER* transputer and the host *CP* for the histogram and final image. A measured maximum of 4 Mbyte/s bandwidth is available between the host *CP* and the *MASTER* transputer (via the dual-ported VME-bus memory), which is adequate for our communication needs with up to 32 nodes. However, this link could become the communication bottleneck for a system with many more nodes, i.e., when the *MASTER*'s four links each utilize 1 Mbyte/s. At least two solutions are possible. One is to have two *MASTER*'s on the VME-bus, each with many *WORKERS* below, so that the overall communication bandwidth between the *CP* and the transputers can be increased to 8 Mbytes/s[1] which is adequate for our needs. Another solution is to enhance the capacity of the transputer network with more memory or disks, so that only the initial raw data and the final image needs to be transferred to the *CP* and the display engine. We are investigating both of these alternatives for future scalability; meanwhile, we proceed with the assumption that the *MASTER*'s link to the VME-bus is not a bottleneck.

Unlike Chen, we find that our backprojection takes much more computation than the convolution of the histogrammed events, mainly due to our algorithm performing a fast convolution with a 1-D filter on data sorted into a 4-D array derived using Radon transform methods [8], rather than the CT algorithm's 2-D filter on the histogrammed 3-D X-ray projections. Using a sequential algorithm, we find that 80% of the reconstruction time is in the backprojection phase [8] and we have chosen to concentrate initially on speeding up that phase.[2]

B. Parallel Backprojection

We define two types of data parallelism, involving different types of data partitioning.

1) Divide the image space up so that any particular node calculates (and stores within its local memory) only a part of it. If we have w nodes, and if each node can store $1/w$th of the image space, and if the communication links are not saturated, then the reconstruction will take $1/w$ as long as for a single node, and the final image can be assembled from the parts by writing into appropriate parts of the system's mass memory at the end of the backprojection phase. The disadvantage of this approach is that all the projection data must be sent to every node, thus

[1]We assume that the VME-bandwidth of 40 Mbytes/s is never the communication bottleneck.

[2]The scatter correction may also prove to be computationally intensive; note that this algorithm does not make any scatter correction.

incurring high data communication costs and high software overhead (as the node's memory is certainly not large enough to hold all the projection data at once) to provide new data when required. This is a purely voxel-driven approach.

2) Partition the projection data so that each node backprojects only some of the projection planes. Each node calculates a $1/w$th of the data values, which will be spread over the whole image. Again, we cannot assume that the whole image can be stored in node memory, so the image values must be transmitted and added, either individually or in batches, to a shared image memory as they are calculated. The advantage of this approach is that it can use both voxel-driven and ray-driven backprojection algorithms, including for example, the hybrid *Incremental Algorithm* scheme for speeding up backprojection [23] for parallel beam geometries, whereas the previous scheme could not take full advantage of this.

Both approaches can take advantage of the *STRETCH* algorithm for fast backprojection [24], in which an interpolated (stretched) projection array can be used for voxel-driven backprojection, although if applied to the first scheme it would dramatically increase the communication requirements. The performance efficiency of these two schemes depends on the software overhead (which depends on the granularity of the tasks) and the comp/comm ratio.

C. Estimated Performance Gains

For calculating estimates of the speedups possible using a transputer-based *MASTER/WORKER* configuration, we need to calculate the comp/comm ratio to ensure that the *MASTER*'s links are not saturated with transferring the input and output data from the *WORKERS* below, while ensuring that the granularity of the computation phase is coarse enough so that the computation overhead in managing the task transfers is not significant. A very simple model illustrating the performance bottlenecks of tree-structured transputer networks such as described in Figs. 1 and 2 is given in Fig. 3. This model ignores all the computation overheads in initiating node-to-node communication, but it serves to illustrate the upper bound of speedups for tree-structured transputer networks.

For smaller numbers of processors, the speedup is limited to that attainable through the computation power available. For larger numbers, communication becomes the bottleneck because the *MASTER*'s four links connected to the four major sub-trees become saturated.[3] In Fig. 3, as the comp/comm ratio R increases, the maximum limit to speedup increases, and when the bottleneck is in the *MASTER*'s four transputer links the limit is simply $4 \times R$. Thus, for $R = 50$, the maximum speedup increases linearly with the number of processors for up to 200 processors, then remains at the maximum 200 as the *MASTER*'s four links are communication bound. For such

[3]Again, we assume that the *MASTER*'s link to the VME-bus is not a bottleneck.

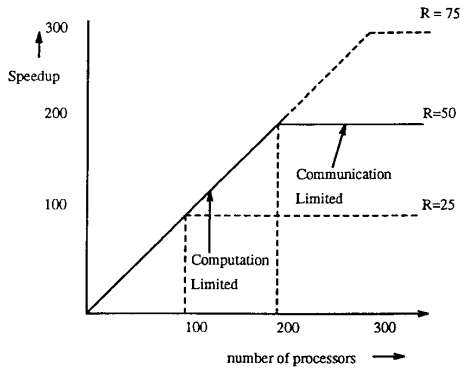

Fig. 3. Upper limits to speedup for tree-structured transputer networks with different comp/comm ratios.

a large number of processors the computation overheads in initiating node-to-node communication and the startup delays become significant; our detailed model [12] takes these into account, but for now we restrict our analysis to the upper bound given in this simplified model.

We also assume in these analyses that the computation load is evenly distributed over all the transputers; implications of this assumption are discussed later Section III-D.

Fig. 3 illustrates the effect of adding processors to the system, when it is assumed that the comp/comm ratio at *WORKER* nodes remains constant. Note that because of the tree-structured architecture, *WORKER* nodes nearer the root *MASTER* also act as *ROUTERS* for data and results between the host *CP* and the nodes below them. We assume that we can ignore this extra communication at each *ROUTER* node, because the communication bottleneck due to this extra routing appears only in the *MASTER*'s four links. Future versions of the transputer will employ hardware through-routing [14] which justify this assumption. The simplified model also extends to the case where the comp/comm ratio R depends on the number of workers w. This occurs when, for example, the total computation is to be divided among w workers, but the communication requirements for a worker remain fixed, whether that worker does only a little or a lot of computation. The comp/comm ratio R is then expressible as k/w where k is some constant. This scenario is explored later in this section, and is illustrated in Fig. 4.

We used this model to examine the maximum performance gains possible for the two alternative parallelizations of the backprojection phase of one of our 3-D reconstruction algorithms [8]. For both approaches to partitioning the data, we make the following assumptions: the image is $200 \times 200 \times 60$ ($= 2.4 \times 10^6$) voxels of four bytes each = 9.6 Mbytes; the backprojection proceeds from the histogrammed data which has been integrated over two of the four histogrammed coordinates, that is, it uses conventional 2-D X-ray projections, and there are 420 projection planes with 5600 4-byte values in each such plane so that there are $420 \times 5600 \approx 2.35$

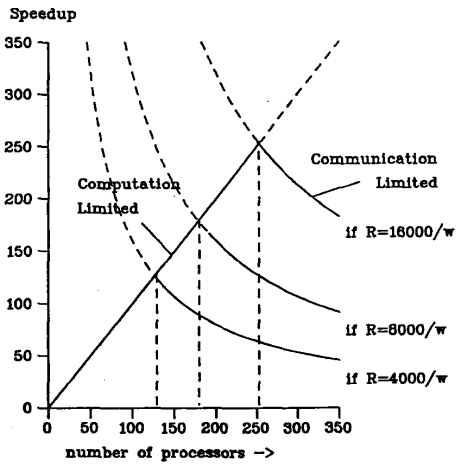

Fig. 4. Upper limits to speedup for varying comp/comm ratios.

$\times 10^6$ lines of response (LOR's) so that the histogrammed raw data occupies $\approx$ 9.4 Mbytes. Since the link communication bandwidth of the transputer links is 1.8 Mbytes/s, transmission of the histogrammed raw data takes $(9.4 \times 10^6)/(1.8 \times 10^6) \approx 5.2$ s. Transmission of the complete image takes $(9.6 \times 10^6)/(1.8 \times 10^6) \approx 5.3$ s.

We measured the time to filter and backproject a plane using a purely voxel-driven approach on a single BBK transputer node connected over a VME-bus directly to a 16 Mbyte shared memory module as in Fig. 1; it was 190 s. Therefore, the computation time on a single transputer node to backproject a complete image over all the voxels would be $190 \times 420 \approx 80\,000$ s.

We thus assume:

Image Size = 9.6 Mbytes

Histogram Size = 9.4 Mbytes

Communication time for complete image transfer = 5.3 s.

Communication time for complete histogram transfer = 5.2 s.

Computation time to backproject complete image = 80 000 s.

Computation time to backproject one plane = 190 s.

1) We examine the communication requirements of the first approach, whereby we partition the image space among the w nodes and send all the projection data to each node as requested. Each voxel may be incremented by a value in each X-ray projection (assuming an unoptimized conventional backprojection algorithm). Since the image is $2{,}4 \times 10^6$ voxels, at least 10 workers are needed to divide the image into small enough units (2.4×10^5) of 4-byte voxels for each *WORKER* to be able to hold its part completely in its 2 Mbyte memory. In this case, it takes $80\,000/10 = 8000$ s for each *WORKER* to complete the backprojection over its voxels. So the comp/comm ratio[4] R for ten *WORKERS* is $\approx 8000/5.2 \approx 1600$. This ratio decreases with the number of *WORKERS*, because the communication remains constant whereas the computation is reduced and in general, for w *WORKERS* the ratio R is $16\,000/w$. The simple transputer performance model shown in Fig. 4 reflects this situation, where the total computation load needs to be distributed over w *WORKERS*, so the comp/comm ratio R is inversely proportional to w. When the communication links become saturated, $R = w/4$. Thus, the maximum speedup occurs when $16\,000/w = w/4$, i.e., when $w = \sqrt{(4 \times 16\,000)} \approx 252$, and $R = 63$. This is illustrated in Fig. 4, where the effects of limits to speedup are illustrated for different total computation times which in turn affect the comp/comm ratios. This very simple model indicates that for our case there is a maximum possible linear speedup for around 252 processors. Beyond that number, the communication bottleneck dominates performance.

2) In the second scheme each node receives one or more projection planes and calculates the contribution these planes make to the whole image. Now it takes 190 s to backproject a plane over all the voxels. The 4-byte value in each voxel must then be sent to the shared image store.[5] Note that the communication of the input projection data is dominated by the transfer of the output image which takes ≈ 5.3 s. The comp/comm ratio is $\approx 190/5.3 \approx 36$, giving rise to a maximum possible linear speedup for up to 144 processors. It is tempting to try to amortize the communication-intensive image transfer over a longer computation involving more than one projection plane, hence increasing the comp/comm ratio and the maximum possible speedup. However, this is not readily achieved, because the voxel driven backprojection only works on one plane at a time. We are investigating algorithms to utilize more than one projection plane at a time, but this is outside the scope of this paper.

We also considered another type of parallelism, *event parallelism*, which relies on the independence of individual detected events to allow for event-by-event reconstruction [25]. We intend to experiment with this approach, but since the computational requirements are of a much greater order of magnitude than for histogrammed event reconstruction algorithms, we are not proceeding with this approach at this stage.

D. Effects of Load Balancing on Performance

In the previous analyses, we assumed that the computation load was evenly distributed over all the transputers. However, the *WORKERS* near the *MASTER* will incur overhead in routing packets to/from *WORKERS* below, so some kind of load-balancing scheme may be necessary. Both techniques can be optimized for load balancing, by dividing the computation task into small *work tokens (tasks)* which the *WORKER* nodes can request from a *MASTER* node when the *WORKERS* are free.

[4]We ignore the cost of transmitting the final image (once at the end).

[5]We ignore the computation time it takes to update the shared image store.

1) In the first method each work token could represent some image voxels for the planes to be backprojected over. However, load balancing using tokens consisting of image voxels affects the maximum speedup here, because the entire projection data needs to be transmitted for each token, resulting in prohibitive communication requirements. Hence, for a balanced load, the image voxels must be statically preallocated among the nodes such that each node finishes at approximately the same time. This is feasible but requires adjustment for every different topology.

2) Load balancing is easier here than in method 1) because each work token can be a projection plane to be backprojected over all the voxels, and the comp/comm ratio is not affected by this distribution of tasks.

E. Effects of Image Size on Performance

Smaller image sizes incur both less computation cost and less communication cost; larger images have the opposite effect. To assess the effects on performance, consider a smaller image size of $128 \times 128 \times 128 \approx 2.1 \times 10^6$ voxels (as for the HISPET tomograph [9]). The computation requirements are reduced by $\approx 2.1/2.4 \approx 0.87$, giving a backprojection time of $\approx 0.87 \times 80\,000 \approx 70\,000$ s.

1) In this method the communication requirements per worker are the same, as they depend only on the size of the projection data. Thus, the comp/comm ratio for w *WORKERS* is reduced to $70\,000/(5 \times w) = 14\,000/w$. From Fig. 4 we see that the limit to the maximum speedup occurs with $w = 236$ and $R = 59$. This is still good enough for achieving significant speedups with up to 236 workers.

For smaller images still, R and hence the maximum speedup reduces more; however, as the total computation time is also decreasing, images can still be produced within an acceptable time of less than 10 min.

Larger images present larger comp/comm ratios with one problem: namely, that the minimum number of nodes may be higher, to accommodate the total image space in the transputer nodes' memory.

2) In the second method, a slightly smaller image size means that the communication and computation costs are reduced in the same ratio, as the communication cost is dominated by the transfer of the complete image after each plane has been backprojected. Slightly different image sizes therefore have no impact on the maximum attainable speedup. However, if the whole image can be held in a single node's memory, as is assumed by others in [9] and [22], the communication requirements drop considerably and this method can scale well beyond 250 workers.

F. Effects of the Number of Projection Planes on Performance

As stated in Section III-C our algorithm uses 420 projection planes, each with 5600 lines of response (LOR). It may be required to process many more than this; for example, in [16] a hardware backprojector is proposed to deal with 4096 2-D projections, each from 192×32 LOR's. Hence, we must consider the effects of ten times more projection planes. For both methods, the computation time is ten times more, as in our voxel-driven approach the total computation time is proportional to the number of projection planes as well as the image size.

1) There are no extra problems due to memory constraints on the nodes, as the projections are requested as needed from the *MASTER*. Computation time per voxel increases by ten times; however, the total communication also increases by ten times, as the number of projections is ten times more. Therefore, the comp/comm ratio remains the same, so almost linear speedups with up to 250 nodes can be expected; however, reconstruction time would increase by ten times, to $\approx 800\,000/250 \approx 3200$ s. This is well outside the desirable time limit of 5–10 min.

2) Computation time per plane remains the same, as does the communication per plane (our algorithm's 5600 LOR's to Jones' 192×32 LOR's) so the comp/comm ratio remains the same at ≈ 36. Hence, the limits to speedup remain the same, at around 140 nodes. The only difference is that load balancing becomes even better, as there are ten times more task tokens to be allocated among the same number of nodes. But again, as the overall computation takes ten times longer than before, the reconstruction time with equipment of the type being considered cannot be reduced much below $\approx 800\,000/140 \approx 5700$ s.

The number of projection planes affects the performance of both approaches in the same way, and has no impact on the limits to speedup. It would therefore appear that the first approach offers the best possibilities for speedup given our image size and number of projection planes, although the second approach would be best for networks of transputers with large enough memories to hold a complete image at each node, or for small images where this condition holds.

IV. Summary

We have presented aspects of the design of a data acquisition system meeting the needs of a next generation 3-D PET scanner, using a multiprocessing approach with inexpensive microprocessors, thus maintaining flexibility of software control with scalable performance.

We investigated the applicability of the transputer to fill the role of the transformation processor (*TP*) for 3-D PET data acquisition, by examining the performance of several different topologies. Although there is less than linear speedup due to communication overhead, with 16 transputers an improvement of 13.3 times the speed of a single transputer is possible. When these topologies are integrated into the data acquisition system a major cost is the

VME-bus interface logic, required for every *MASTER* processor. With the tree-based transputer topology described, only two processors need this expensive interface. Such a tree-based topology is the most economical alternative of those investigated. Our detailed analytic model showed that speedups of 18 times are feasible with 20 transputers, allowing for data acquisition in real time.

Real-time 3-D image reconstruction requires another order of magnitude of computation power, and we are designing parallel algorithms and transputer topologies to meet these requirements where both memory and computation constraints are present. We presented a simple analytic model to deal with different computation granularities and different comp/comm ratios, and used this model to illustrate the limits to the scalability of two alternative, unoptimized, voxel-driven schemes of implementing the backprojection phase in parallel.

We find that dividing the image space among the transputers offers the best possibilities for scalability, and reconstruction times of less than 10 min should be feasible with 200 processors. This scheme also works well for varying image sizes, provided the total memory on the transputer nodes is sufficient for distributing the whole image (around 10 Mbytes). However, if there are many more projection planes than our algorithm uses, the reconstruction time will be reducible to the desired level only by using the next generation of transputers which are ten times more powerful than the current nodes, or by using an optimized algorithm for fast backprojection.

Dividing the work equally among the nodes is more easily achieved using the second approach whereby each node computes the whole image from a projection plane. This approach also offers major advantages for smaller images.

A. Future Work

We are currently implementing our parallel 3-D image reconstruction algorithm on our network of transputers using both data-partitioning approaches. We are also considering ways to optimize the backprojection algorithm.

We plan to use the performance data obtained to develop a realistic analytic model incorporating load balancing, task distribution overheads, and task size which will accurately reflect the limits to speedup.

Finally, we hope to develop software tools which will enable the programs to be efficiently ported to other parallel architectures.

ACKNOWLEDGMENT

The authors wish to thank F. Olariu at Triumf for preparing the figures.

REFERENCES

[1] N. A. Wilkinson, M. S. Atkins, and J. G. Rogers, "A real time parallel processing data acquisition system," in *Proc. IEEE 9th Real-Time Syst. Symp.*, Dec. 1988, pp. 54–59.

[2] J. G. Rogers, R. Harrop, and P. Kinahan, "The theory of 3D image reconstruction for PET," *IEEE Trans. Med. Imaging*, vol. MI-6, pp. 239–243, June 1987.

[3] J. G. Rogers, R. Harrop, G. H. Coombes, N. A. Wilkinson, M. S. Atkins, B. D. Pate, K. S. Morrison, M. Stazyk, C. J. Dykstra, J. S. Barney, P. W. Doherty, and D. P. Saylor, "Design of a volume-imaging positron emission tomograph," *IEEE Trans. Nucl. Sci.*, vol. 36, pp. 993–997, Feb. 1989.

[4] P. E. Kinahen, "Analytic three-dimensional image reconstruction from projections," *M.A.Sc. thesis, Dep. Phys. Univ. British Columbia*, 1988.

[5] J. G. Rogers, M. Stazyk, R. Harrop, C. J. Dykstra, J. S. Barney, M. S. Atkins, and P. E. Kinahan, "Toward the design of a positron volume imaging (PVI) camera," *IEEE Trans. Nucl. Sci.*, vol. 37, Apr. 1990.

[6] D. W. Townsend, T. Spinks, T. Jones, A. Geissbuhler, M. Defrise, M. C. Gilardi, and J. Heather, "Three-dimensional reconstruction of PET data from a multi-ring camera," *IEEE Trans. Nucl. Sci.*, vol. 36, pp. 1056–1065, Feb. 1989.

[7] C. Dykstra, "The use of radon transforms in full 3D positron volume imaging—A feasibility study," M.Sc. thesis, School Comput. Sci., Simon Fraser Univ., Apr. 1989.

[8] M. Stazyk, "The radon transform in 3-D image reconstruction from projections," MASc. thesis, Dep. Phys., Univ. British Columbia, Oct. 1990.

[9] S. Barresi, D. Bollini, and D. Guerra, "Use of a transputer system for fast 3-D image reconstruction in 3-D PET," *IEEE Trans. Nucl. Sci.*, vol. 37, pp. 812–816, Apr. 1990.

[10] S. Barney, J. G. Rogers, R. Harrop, and H. Hoverath, "An object shape dependent scatter correction for PET," in *Proc. IEEE Nucl. Sci. Symp.*, Oct. 1990, pp. 1280–1287.

[11] N. A. Wilkinson, M. S. Atkins, and J. G. Rogers, "A tomograph parallel processing data acquisition system," *IEEE Trans. Nucl. Sci.*, vol. 36, pp. 1047–1051, Feb. 1989.

[12] D. Murray, "A real-time transputer-based data acquisition system," M.Sc. thesis, School Comput. Sci., Simon Fraser Univ., Canada, Apr. 1990.

[13] M. Homewood, D. May, D. Shepherd, and R. Shepherd, "The IN-MOS T800 transputer," *IEEE Micro*, pp. 10–26, Oct. 1987.

[14] D. Pountain, "Features of the new H1 transputer," *Byte Mag., E&W ed.*, pp. 3–10, Apr. 1990.

[15] A. Hey, D. J. Pritchard, and C. Whitby-Strevens, "Multi-paradigm parallel programming," in *Proc. 22nd Ann. Hawaii Int. Conf. Syst. Sci.*, Jan. 1989, pp. 716–725.

[16] W. F. Jones, L. G. Byars, and M. E. Casey, "Design of a super fast 3-D projection system for positron emission tomography," *IEEE Trans. Nucl. Sci.*, vol. 37, pp. 800–804, Apr. 1990.

[17] R. M. Stein, "T800 and counting," *Byte Mag.*, pp. 287–296, Nov. 1988.

[18] INMOS, *The Transputer Data Book*, 1987.

[19] P. Wilson, "Parallel processing comes to PC's," *Byte Mag.*, pp. 213–218, Nov. 1988.

[20] Logical Systems, *Transputer Toolset Version 88.3*, June 1988.

[21] P. E. Kinahen, *Private Communication*, Feb. 1991.

[22] C. M. Chen, S.-Y. Lee, and Z. H. Cho, "A parallel implementation of 3-D CT image reconstruction on hypercube multiprocessor," *IEEE Trans. Nucl. Sci.*, vol. 37, pp. 1333–1346, June 1990.

[23] Z. H. Cho, C. M. Chen, and S.-Y. Lee, "Incremental Algorithm—A new fast backprojection scheme for parallel beam geometries," *IEEE Trans. Med. Imaging*, vol. 9, pp. 207–217, June 1990.

[24] T. M. Peters, "Algorithms for fast back-and-reprojection in computed tomography," *IEEE Trans. Nucl. Sci.*, vol. 28, pp. 3641–3647, Aug. 1981.

[25] J. A. McIntyre, "Computer assisted tomography without a computer," *IEEE Trans. Nucl. Sci.*, vol. 28, pp. 171–173, Feb. 1981.