NOW Based Parallel Reconstruction of Functional Images

F. Munz[1], T. Stephan[2], U. Maier[2], T. Ludwig[2], A. Bode[2],
S. Ziegler[1], S. Nekolla[1], P. Bartenstein[1] and M. Schwaiger[1]

[1]Nuklearmedizinische Klinik und Poliklinik
des Klinikums rechts der Isar,
Technische Universität München,
81675 München, Germany
email:Munz@Informatik.TU-Muenchen.DE

[2]Lehrstuhl für Rechnertechnik
und Rechnerorganisation
Technische Universität München,
80290 München, Germany

Abstract

This paper deals with the parallel implementation of reconstruction algorithms for functional imaging on a network of workstations (NOW). Algorithms which provide the best image quality are not used in clinical routine, because they have a runtime of up to 60 hours with real clinical data sets of several hundred megabytes. After giving an overview of currently used image reconstruction algorithms, we describe a general parallel implementation of these algorithms with almost linear speedup and high efficiency which cuts down the runtime to a feasible limit. The high load which is caused by the parallel application conflicts with the predominantly interactive usage of clinical workstations, therefore we address load balancing with an application oriented, adaptive mechanism in order to preserve the ownership of workstations. Furthermore we explain how the integration of MATLAB and IDL based applications with a conventional distributed queuing system (DQS) can be achieved and why this significantly improves usage in clinical routine.

1. Introduction

Tomographic imaging has become a key technology for modern medical diagnosis. Recently technologies like positron-emission-tomography (PET) or single-photon-emission- tomography (SPECT) have been developed which enable the direct measurement of function whereas traditional technologies such as computer tomography (CT) are only capable of imaging the morphological structure. PET or SPECT images are acquired by measuring the decay of radioisotopes attached to molecules with known physiological properties such as water, sugar or molecules binding to a particular type of neuronal receptor. Reconstructing the measured projection data is a complex task and there is a lot of ongoing research in this area [2]. This text deals with the benefits and caveats of parallel algorithms for PET, however, due to the similarity most of it is applicable to SPECT as well. Most PET scanners consist of several rings of small detectors. Axial collimation is provided by retractable tungsten septa between the rings. Retracting the septa switches from 2-D to 3-D mode, see fig. 1. Then cross plane events are also detected and sensitivity but also scatter fraction is increased.

Radioactive decay of the actual tracer distribution $\lambda(x, y, z)$ is characterised by the emission of a positron, its subsequent annihilation and emission of two high energy gamma rays. These are measured in coincidence along lines of response (LOR) and stored dependant of its angle ϕ and distance from the centre of the scanner s in a data structure termed sinogram $p(s, \phi)$. Each sinogram value represents the line integral of tracer distribution with $f = \lambda(x, y, (z = fixed))$: $p(s, \phi) = \int f(s\cos\phi - t\sin\phi, s\sin\phi + t\cos\phi)\,dt$. The Radon transform of $\lambda(x, y, z)$ is $p(s, \phi)$ and calculating a PET image corresponds to inverting the transform [6].

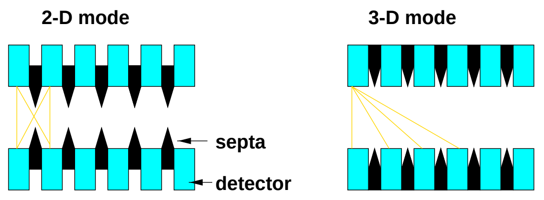

Figure 1. PET scanner operation modes (geometry not to scale)

2. Parallel Image Reconstruction from Projections

2.1. 2-D Algorithms

Ignoring the measurement of noise leads to the classical *filtered backprojection* (FBP) algorithm. The relationship between parallel projection $p(s, \phi)$ and the original tracer distribution is given by the projection-slice theorem:

$$P(\nu, \phi) \quad = \quad F(\nu \cos \phi, \nu \sin \phi) \tag{1}$$

For reconstruction, each projection $p(s, \phi)$ is convolved with a shift invariant kernel to emphasize small structures and reduce frequencies above a certain limit. Then the filtered projection value $p_F(s, \phi)$ is redistributed uniformly along the straight line (s, ϕ):

$$f_R(x, y) \quad = \quad \int_0^{\pi} p_F(x \cos \phi + y \sin \phi, \phi)\, d\phi \tag{2}$$

This method can cause strong streak artifacts and is less suitable for low count situations.

Iterative methods were introduced to overcome the disadvantages of FBP. They are based on the discrete nature of data and incorporate physical phenomena such as scatter or attenuation directly into the model. It is generally acknowledged that iterative methods yield images with improved diagnostic quality in low count situations. Shepp and Vardi presented an algorithm to *maximize the likelihood* of the reconstructed data (ML)[4]. They modelled the projection data as random variables with means equal to the true tracer distribution. The aim of the reconstruction algorithm is then to maximize a set of Poisson processes that give rise to the projection data.

Other researchers claim that the data is not Poisson distributed due to a precorrection done by the PET scanner. Fessler defines a *weighted least squares objective function* to describe the similarity of the forward projected iterated image compared to the measured data and a *penalty function* (PWLS) is introduced to cope with noise. Compared to the other algorithms, PWLS has the longest runtime and the highest demands for main memory, because it uses a very large system matrix describing the physical properties of the scanner. It is therefore seldom found in clinical routine, although it yields best image quality [2].

2.2. 3-D Algorithms

The most common approach to reconstruct 3-D data is the *3-D reprojection* algorithm (3DRP) which is an extension of the filtered backprojection algorithm (FBP) to 3-D that estimates data not measured by the scanner. The reconstruction is typically done on a special purpose hardware.

Another way to reconstruct 3-D data are *rebinning algorithms*. They first sort the 3-D data into ordinary two dimensional sinograms representing transaxial planes, which can then be reconstructed using e.g. FBP or PWLS. This approach is significantly faster because $O(n^2)$ oblique sinograms are reduced to $O(n)$ ordinary sinograms. Depending on the way cross ring decays are redistributed to ordinary sinograms there are various algorithms termed *single slice rebinning* (SSRB), *multislice rebinning* (MSRB) or Fourier rebinning (FORE) [1]. Functional decomposition of these algorithms cannot be easily done due to their complex numerical and statistical nature and decomposing the highly optimized system matrix for PWLS is impossible without redesigning the whole algorithm.

2.3. Related Work

Other researchers describe a simple data parallel implementation of one particular algorithm [7]. To our knowledge there was no previous work at any PET center describing a general approach that works for most available reconstruction algorithms, in particular PWLS, and deals with load balancing and queuing issues.

2.4. Design of a Data Parallel PWLS Prototype

Building a data parallel program raises the question of granularity. Possibilities are LORs, planes or frames[1]. Other partitioning schemes like assigning certain parts of sinograms to individual processors increase the complexity of algorithms. The easiest solution regarding the amount of interprocess communication needed and the complexity of algorithms, is setting granularity of a data parallel implementation to the plane level.

This approach works for FBP, iterative algorithms and it can be used for rebinned 3-D data. The remainder of this paper deals with an implementation of the PWLS algorithm, which is a state of the art iterative algorithm and explains why a straightforward data-parallel implementation is *not* feasible in clinical routine.

2.5. Implementation of the Prototype

Data parallelism as described above was exploited. Single planes were assigned to processors in a round robin way. The first implementation was straightforward without queuing, sophisticated load balancing or fault tolerance. The implementation was based on PVM 3.3.7[5]. The time needed to reconstruct one plane was about 2-3 minutes depending

[1]a set of planes acquired at one point of time during a dynamic study

on the image size and number of iterations[2] on a single CPU, provided that the workstation was equipped with sufficient memory. Main memory is a key issue because otherwise the dynamic allocation of memory for the system matrix P, typically some 50 MB, causes the machine to page virtual memory onto disk and this slows down the process by several orders of magnitude. Common setup values for the slave processes are multicast using `pvm_mcast()` to all slave processes, e.g. parameters to create the sparse system matrix P.

Memory requirements could be reduced for all involved workstations compared to the serial program. The master process does not setup the sparse system matrix, it only holds a stack of sinograms, whereas the slaves need to setup the system matrix. They only allocate memory for a single plane and later on this memory is reused. Also the system matrix is not transferred across the network, only a few parameters which describe the scanner geometry are multicasted and the system matrix is set up according to these parameters. The main memory chunks for a typical serial reconstruction process (with some 80 MB total) are sinogram data (23 %), the system matrix (76 %), other data and code area (1 %).

2.6. Performance Values

Performance measurements were done on cluster of 30 HP 9000/720 workstations running HP-UX release 9.01, interconnected via 10 MBit/s Ethernet, all machines where in the same IP subnet. Our target architecture for the implementation was SGI and SUN. Results of speedup and efficiency measurements are shown in fig. 2.

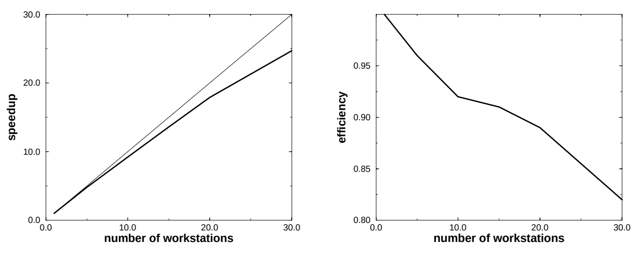

Figure 2. Speedup and Efficiency on HP Cluster

[2]the number of iterations was fixed for all further tests

3. Distributed Processing in a Clinical Environment

Speedup and efficiency values presented in the former sections show that the performance of the data parallel implementation is excellent, however, the application blocks all involved computers completely for up to one day. Hardly any clinical institute ownes a massively parallel computer, waiting queues at a computer center are typically too long and too unpredictable for clinical routine and even a dedicated NOW is quite expensive. Thus a new strategy was needed which finally made it possible to process even whole body PET scans (see fig. 3) iteratively, which consists of up to 500 transverse slices.

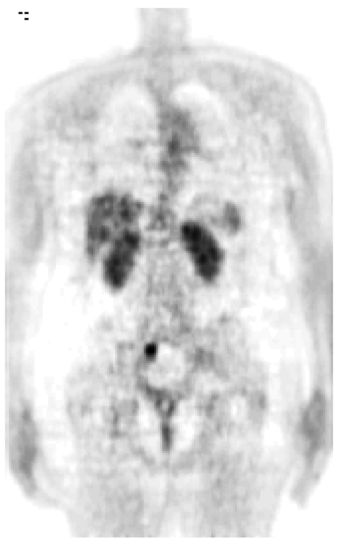

Figure 3. Whole-Body PET Scan

3.1. Batch Queuing

An interface to DQS was implemented in the program. DQS offers distributed queues using a central `qmaster` and `dqs_execd` daemons on every host. Typically medical applications are interactive with an IDL (Interactive Data Language, RSI, Colorado) or MATLAB (Math Work, Mass.) based GUI used to set up dozens of reconstruction parameters, to load sinogram data using CAPP (Clinical Application Programming Package, CTI Knoxville, TN) routines and to do some precomputations. Furthermore the interpreter for IDL and MATLAB code needs a license key during runtime so scheduling is not straightforward without expensive recoding of all IDL and MATLAB code. We took the following approach: Once the IDL bound computations are complete, a module implemented in C writes all necessary data to disk, and then creates and submits a batch job to DQS using UNIX system call `popen()` to communicate with `qsub` and releases the license key. According to its configuration `qmaster` attaches the reconstruction process to an appropriate queue at a workstation that does not exceed a predefined load threshold. Running the application can be scheduled to any particular time, e.g. non office hours. Within this context the program is treated as a serial program setting up the parallel environment itself.

3.2. Load Balancing

Using a clinical NOW successfully for parallel processing is tightly linked with the preservation of ownership of all workstations. We evaluated the benefits of system oriented load balancing and checkpointing but implemented an application oriented adaptive load balancing mechanism because of several reasons[3]:

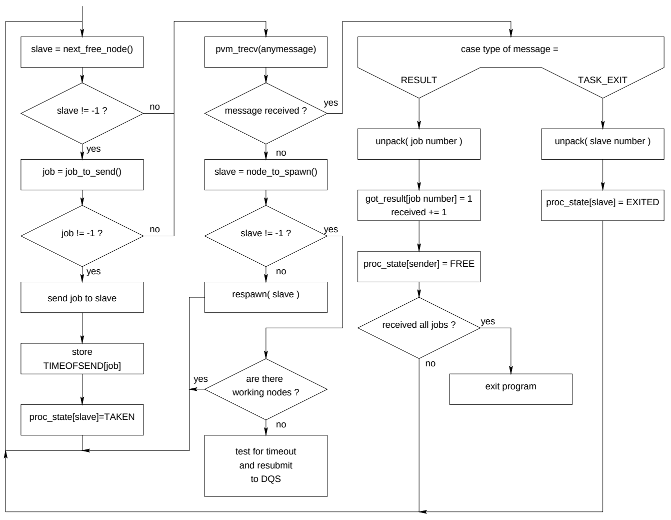

Figure 4. Master Process Flow Chart

A key issue was to gain the user acceptance by releasing used machines as quickly as possible. Writing checkpoints and the migration of processes means saving the process context to disk. The required time for this grows linearly with the size of the process context and would need more than one minute in our case. Saving process context and migrating processes must be done when a load peak occurs, but writing to disk via NFS increases the load even further. Killing the process can be done immediately and reconstructing this plane later will take less time than migrating the process.

After setting up the parallel virtual machine the master process checks for the next free node. It keeps a list with the status of each node (`FREE`, `TAKEN` or `EXITED`). The sinograms are sent with the following heuristic: first all sinograms which were never sent before, then it resends the oldest ones without answer from the remote process, but only if they are older than a heuristic threshold depending on the number of iterations. This is done to avoid resending sinogram data if a slow node has almost finished the reconstruction and to avoid waiting infinitely for a reconstructed image. Figure 4 shows a more detailed flow chart of the master process.

3.3. Load monitoring

We identified CPU and interactive usage as useful load indices for our environment of SGI workstations. These parameters are monitored by the `loadw` process. Figure 5 shows a flow chart of `loadw`.

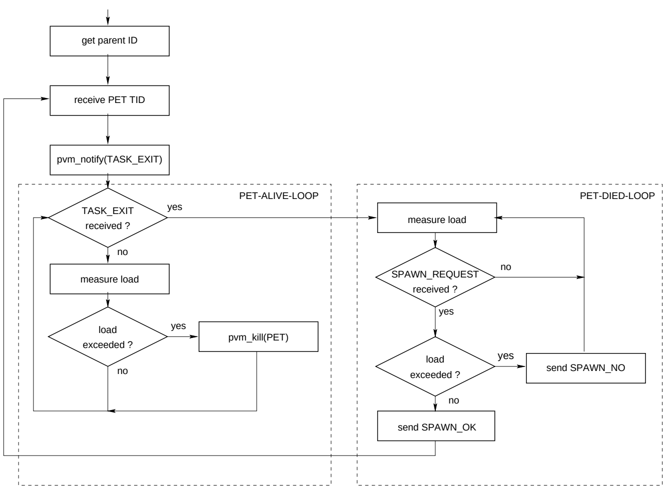

Figure 5. loadw Process Flow Chart

We neglected capturing mouse and other events of the X server under root privileges, because of strict security policies, but the load monitoring process `loadw` would react of course to X applications that cause a high CPU load. For interactivity, we looked at the idle times of the user terminals. Scanning through `/etc/utmp` gives information about logged in users, user processes marked with `USER_PROCESS` and their open terminals which are found as entries in `/proc`. Modification time of the terminal entries equals the last keyboard event which is compared to the system time.

To avoid disturbing other programs running on a particular machine without user interaction we looked at *CPU usage* as another load index. Each process has an entry in `/proc` with its process id (PID) and users are only allowed to read entries belonging to their own processes for security reasons. However under IRIX there is a second source of information readable for everyone: `/proc/pinfo`[3]. It

[3]the name may differ on other OS

provides `loadw` with the name of the program being executed by the process and a `pr_cpu` termed value representing the recent CPU usage - a value which is normally used by the kernel for scheduling. `pr_cpu` is incremented each time the system clock ticks and the process is found to be executing. Every second `pr_cpu` is adjusted by a digital decay filter.

Heuristics have shown that `pr_cpu` is smaller than 80 for single process being permanently on the run queue, `pr_cpu` is between 30 and 40 for two processes simultaneously on the run queue and `pr_cpu` is less than ten for system processes which are mostly idle.

`loadw` is operating in two different modes which could be described by an deterministic finite automata with two states. In state `PET_ALIVE` it tries to determine whether to stop the reconstruction process because load exceeds the threshold or a keyboard event was registered. In state `PET_DIED` it responds to `SPAWN_REQUEST` from the master process and acknowledges or denies to restart a reconstruction process depending on the load and time since the last keyboard event. If the last keyboard event was more than t_i minutes ago it is assumed that there is no interactive usage at the moment. The value for t_i is currently set to two minutes, which is roughly the time to reconstruct one plane, so even short times of noninteractivity, e.g. during a phone call, are used for image reconstruction. If no reconstruction process can be run due to extreme load conditions on all machines, then after a period of time the application reschedules itself to execute some hours laters and terminates.

4. Future Work

The application described in this paper is used routinely in our clinic with great success. There are more time consuming tasks in functional imaging, e.g. voxel by voxel modelling of neuroreceptors in human brains using PET tracers such as Diprenorphine, so image reconstruction was only the first step. We currently evaluate the design of a multithreaded version of our application oriented load leveling system, in order to simplify the flow of control and the integration into other applications by running different tasks concurrently.

5. Conclusions

In this paper we presesented a general way to implement parallel reconstruction algorithms on a NOW. We have shown that a data parallel implementation is only acceptable if the ownership of workstations is preserved. Therefore we defined the crucial load parameters for such an application and described a suitable load balancing mechanism which is easy to implement. Performance measurements have shown high efficiency and almost linear speedup. Our implementation reduces the vast CPU times, so even very sophisticated algorithms like PWLS, which provide much better image quality, are feasible for routine now. Furthermore, the parallel implementation cuts down the requirements for main memory.

6. Acknowledgement

We would like to thank R. Schaller, S. Becker, D. Schad, W. Weber, and E. Dickmann for their help. This work is part of an ongoing collaboration between the Lehrstuhl für Rechnertechnik und Rechnerorganisation and the Klinik und Poliklinik für Nuklearmedizin rechts der Isar. This study was partly supported by the Deutsche Forschungsgemeinschaft SFB 462, Sensomotorik.

References

[1] M. Defrise, P. E. Kinahan, D. W. Townsend, C. Michel, M. Sibomana, and D. F. Newport. Exact and Approximate Rebinning Algorithms for 3-d PET Data. *IEEE Transaction on Medical Imaging*, 16, April 1997.

[2] J. A. Fessler. Improved PET Quantification Using Penalized Weighted Least-Squares Image Reconstruction. *IEEE Transaction on Medical Imaging*, 1992.

[3] U. Maier and G. Stellner. Distributed Resource Management for Parallel Applications in Networks of Workstations. In *HPCN Europe 1997*, volume 1225 of *Lecture Notes in Computer Science*, pages 462–471. Springer-Verlag, 1997.

[4] L. Shepp and Y. Vardi. Maximum Likelihood Reconstruction for Emission Tomography. *IEEE Transaction on Medical Imaging*, 1982.

[5] V. S. Sunderam, G. A. Geist, J. Dongarra, and R. Manchek. The PVM Concurrent Computing System: Evolution, Experiences, and Trends. *Parallel Computing, Vol. 20 (4)*, 1993.

[6] D. W. Townsend and M. Defrise. Image Reconstruction Methods in Positron Tomography. Technical report, CERN European Organization for Nuclear Research, 1993.

[7] J. Zaers, J. Doll, P. Schmidlin, G.Brix, and W. Lorenz. Parallele Implementation iterativer Rekonstruktionsverfahren für die PET auf einem heterogenen Workstation-Cluster. *Nuklearmedizin*, 36:76–109, April 1997.