Meeting the Computational Demands of Nuclear Medical Imaging Using Commodity Clusters

Wolfgang Karl[1], Martin Schulz[1], Martin Völk[2], and Sibylle Ziegler[2]
{karlw, schulzm, voelk}@in.tum.de, s.ziegler@lrz.tum.de

[1] Lehrstuhl für Rechnertechnik und Rechnerorganisation, LRR–TUM
Institut für Informatik, Technische Universität München, Germany
[2] Nuklearmedizinische Klinik und Poliklinik, MRI-NM,
Klinikum Rechts der Isar, Technische Universität München, Germany

Abstract. Even though Positron Emission Tomography (PET) is a relatively young technique within Nuclear Medical Imaging, it has already reached a high level of acceptance. However, in order to fully exploit its capabilities, computational intensive transformations have to be applied to the raw data acquired from the scanners in order to reach a satisfying image quality. One way to provide the required computational power in a cost–effective and efficient way, is to use parallel processing based on commodity clusters.
These architectures are traditionally programmed using message passing. This, however, leads to a low–level style of programming not suited for the general user. In this work, a new programming environment based on a graphical representation of the application's behavior has been successfully deployed. The result is an image transformation application, which is both easy to program and fulfills the computational demands of this challenging application field.

1 Motivation

Over the last few years, Positron Emission Tomography (PET) has become a very important instrument in medical diagnosis procedures. However, in order to reach the level of image quality needed, computational intensive algorithms need to be deployed for the conversion of raw scanner data to humanly readable images, the so called PET image reconstruction. Quite a bit of work has been invested in increasing the image quality [5], but the computational demands remain high. One way to match these requirements in order to keep the time needed for image reconstruction process at an acceptable level and therefore to make the application of these improved algorithms in daily clinical routine feasible, is the deployment of parallel computing.

An attractive platform for such an approach are clusters built from commodity parts. They are cost effective and easy to build and maintain. Due to these very favorable properties, this class of architectures has recently earned a lot of attention and has started to replace traditional large–scale tightly coupled parallel systems. Clusters are generally programmed using message passing,

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 27–36, 2001.

mostly in the form of a standard library like PVM [1] or MPI [6], as this directly matches their distributed memory organization. These message passing APIs, however, are in most cases quite complex and cumbersome to apply. The user has to worry about many implementation details related to communication and data management, which do not belong to the actual problem to solve. This discourages many potential users, especially those directly from application areas without a formal computer science background, and therefore hinders the wide deployment of cluster architectures outside of computer science research.

One approach to overcome this problem of complex programmability is to deploy a programming environment which offers a high level of abstraction to the user and hides most of the low–level complexity of the underlying architecture. Such an approach has been taken within the NEPHEW project [9], which this work is part of, by applying a graphical programming environment, called PeakWare [8], for the implementation of several real–world applications including the reconstruction of PET images discussed in this paper.

In PeakWare, any communication within the whole system is abstracted into a modularized graphical representation which is easily comprehendible for the application programmer. The actual implementation of the communication, as well as the mapping of the application onto the target architecture, is automatically taken care of by the system. This creates a framework that enables an efficient implementation of a PET image reconstruction software on top of a Windows based PC cluster without having to deal with most of the complex issues normally involved in parallel and distributed processing. The resulting system is able to perform the reconstruction of a whole body scan in about 3 minutes. This is within the limits that allow an interactive on-line diagnosis by doctors while the patient remains in the clinic and within the scanner.

The remainder of this paper is organized as follows: Section 2 introduces PET imaging and its computational demands, followed by a discussion on how to match those using cluster computing in Section 3. Section 4 then presents the graphical programming environment used within this work and Section 5 provides details about the implementation of the PET reconstruction algorithm within this environment. The performance of the system is then evaluated in Section 6. The paper is rounded up by some concluding remarks and a brief outlook in Section 7.

2 Nuclear Medical Imaging Using PET

Positron Emission Tomography is a nuclear medicine technique which allows to measure quantitative activity distributions in vivo. It is based on the tracer principle: A biological substance, for instance sugar or a receptor ligand, is labeled with a positron emitter and a small amount is injected intravenously. Thus, it is possible to measure functional parameters, such as glucose metabolism, blood flow or receptor density. During radioactive decay, a positron is emitted, which annihilates with an electron. This process results in two collinear high energy gamma rays. The simultaneous detection of these gamma rays defines

lines-of-response along which the decay occurred. Typically, a positron tomograph consists of several detector rings covering an axial volume of 10 to 16 cm. The individual detectors are very small, since their size defines the spatial resolution. The raw data are the line integrals of the activity distribution along the lines-of-response. They are stored in matrices (sinograms) according to their angle and distance from the tomograph's center. Therefore, each detector plane corresponds to one sinogram. Image reconstruction algorithms are designed to retrieve the original activity distribution from the measured line integrals. From each sinogram, a transverse image is reconstructed, with the group of all images representing the data volume.

Ignoring the measurement of noise leads to the classical filtered backprojection (FBP) algorithm [3]. Reconstruction with FBP is done in two steps: Each projection is convolved with a shift invariant kernel to emphasize small structures but reduce frequencies above a certain limit. Typically, a Hamming filter is used for PET reconstruction. Then the filtered projection value is redistributed uniformly along the straight line. This approach has several disadvantages: Due to the filtering step it yields negative values, particular if the data is noisy, although intensity is known to be non-negative. Also the method causes streak artifacts and high frequency noise is accentuated during the filtering step.

Iterative methods were introduced to overcome the disadvantages of FBP. They are based on the discrete nature of data and try to improve image quality step by step after starting with an estimate. It is possible to incorporate physical phenomena such as scatter or attenuation directly into the models. On the down side, however, these iterative methods are very computational intensive. For a long time, this was the major drawback for clinical use of these methods, although they yield improved image quality.

One of the steps in iterative image reconstruction is the projection of measured PET data, just as in FBP. Expectation Maximization (EM) algorithms, however, forward project the images to compare the generated to the measured data. The result of this comparison is used to continually update the estimation of the image. The vector to update the image is calculated by using Poisson variables modeling the pixel emissions. With the Likelihood Function, an image is estimated for which the measured PET data would have been most likely to occur.

Following scheme shows one iteration step of an Expectation Maximization (EM) algorithm:

1. Estimation of distribution
2. Forward-projection
3. Compare estimated projection with measured projection
4. End loop when error estimate under predefined value
5. Back-projection of error estimate
6. New estimation of distribution

Since this method converges very slowly, this iteration step has to be performed many times (e.g. 100) for each pixel in each plane to converge. To reduce the number of iterations, a "divide and conquer" strategy is used. With

OSEM (Ordered Subset Expectation Maximization) [4] the events registered by the PET-scanner are divided into subsets. In one OSEM-iteration the typical steps of projection and back projection are done for each of theses subsets. The start-value for an iteration for each subset is gained from the result of the back projection of the previous subset. With ordering the subsets in a way that there is a maximum of information between each of them, the speed is further improved. Still, due to the iterative nature a substantial amount of time is required for the reconstruction of one image plane.

3 Meeting the Computational Demands with the Help of Clusters

In order to meet these computational demands for PET image reconstruction and still keep the total reconstruction time at an acceptable level, it is necessary to apply parallel processing. It is not only suited to speed-up the iterative reconstruction, but is also likely to guarantee image reconstruction times useful for a semi real–time, on–line diagnosis of a patients PET scan results while the patient is still within the PET scanner. This increases the medical accuracy as scans can easily be repeated without extra scanner time and therefore shortens the turnaround time for medical treatment. A typical upper bound for the reconstruction of a PET image that allows such guarantees is about four to five minutes.

An architecture well suited for this endeavor are clusters of commodity PCs. They provide an excellent price/performance ratio, are easy to build and maintain, and easily scalable to match the concrete computational demands. They are already used in many real–world application scenarios, including Nuclear Medical Imaging [7]. The main problem connected with this approach, however, is their difficult programmability. The most common programming approaches for clusters are based on message passing libraries, like MPI [6] or PVM [1]. With these libraries, however, the user is forced to a very low–level style of programming and is required to take care of additional tasks including the partitioning of code and data, the mapping of code onto individual nodes within the cluster, and the complete communication setup and management. This introduces a significant amount of additional complexity, which is not related to the concrete problem.

In order to make cluster architectures attractive to users not especially trained in parallel processing, a different programming environment has to be provided at a much higher level of abstraction. Such an environment has to be capable to hide the implementation complexity of the underlying architecture without sacrificing performance and/or functionality.

4 Easing the Programmability Using a Visual Approach

In order to establish such an environment for the implementation of the PET image reconstruction, discussed in this work, a graphical tool, called PeakWare [8],

is used. This tool was originally developed by Matra Systems & Information for real-time multiprocessor systems and has been adopted for Windows 2000 based clusters within the NEPHEW project. It completely hides the implementation of the communication framework from the user by automatically generating it from a graphical representation. This includes any startup procedures, notification mechanisms, as well as the actual data transport itself. The user only has to code the actual application functionality and PeakWare then automatically combines the individual parts into a full application.

Any development in PeakWare is generally done in five steps. First the application has to be decomposed into individual functional units, called modules, one of the central concepts of PeakWare. The separation into modules has to be done in a way that all communication between them can cleanly be specified in the form of a data flow graph.

The information gained through this analysis is then used to graphically describe the communication behavior in the so called software graph. An example with two modules and bidirectional communication is shown in Figure 1. PeakWare also offers the ability to scale individual modules, i.e. to replicate and distribute them among the cluster. This concept offers an easy way to introduce data parallelism into an application and allows the easy scaling to potentially arbitrary numbers of nodes.

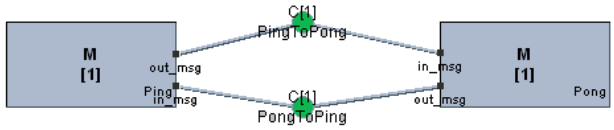

Fig. 1. PeakWare software graph (simple ping-pong communication)

Each module consists of several functions with the global communication channels as input and output arguments. The implementation of the functions themselves is done in external source files using conventional sequential programming in C. In the final application, these functions are then triggered automatically by PeakWare at corresponding communication events without requiring any further user intervention. This has to be seen in contrast to typical message passing models which require explicit receive calls and an explicit binding of incoming messages to their processing functions within the code.

The next step is the definition of the hardware that is supposed to be used for the application. This is again done with a graphical description, the hardware graph. An example of such a graph can be seen in Figure 2. It shows a small cluster of two compute and one host node connected by Fast Ethernet. This concept of an independent hardware graph enables the user to change the hardware in terms of node description and/or number of nodes without requiring changes in the software graph or the application itself.

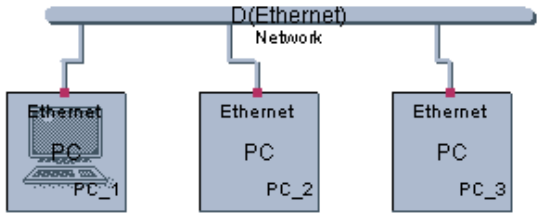

Fig. 2. PeakWare hardware graph (2 node cluster, with external development host)

Once the hard- and software graph have been completed, PeakWare gives the user the option to specify a mapping between modules (from the software graph) and nodes (as specified in the hardware graph). This mapping defines which module is executed on which node and hence represents the connection between the two graphs. It also allows the easy retargeting of applications to new hardware configurations as well as simple mechanisms for static load balancing.

The last step, after the mapping has been done and all routines have been implemented in external source files, is the code generation. In this process PeakWare uses the information from the graphical description of the software and hardware graphs and generates C source code that includes all communication and data distribution primitives. This code can then be compiled with conventional compilers resulting in a final executable and a shell script to start the application on all specified nodes.

5 PET Image Reconstruction Using PeakWare

Due to their regular data layout, the decomposition for the PET image reconstruction can be achieved in a quite straightforward manner. Each input volume consists of a number of image planes (typically 47 or 63 per scan, depending on the scanner type). The reconstruction for each of these planes can be done independently. Therefore, a parallelization at the granularity of individual image planes is most promising.

This basic scheme was implemented in PeakWare using three different modules: a sender-module reads the raw input data and distributes the read image planes to a scaled consumer module in a round-robin fashion, one plane at a time. The consumer performs the actual reconstruction and after its completion forwards the resulting data to a receiver module, which is responsible for storing the final image. In addition the sender is informed, that a subsequent plane can be sent. The sender distributes the planes of the image until the reconstruction of all image planes has been acknowledged by the receiver. If no planes are left, planes might be resubmitted to idle consumers resulting in an easy, yet efficient fault tolerance scheme with respect to the consumer modules.

The software graph based on this design is depicted in Figure 3. It shows the three modules, with the consumer module scaled to $\$(SCALE)$ instances.

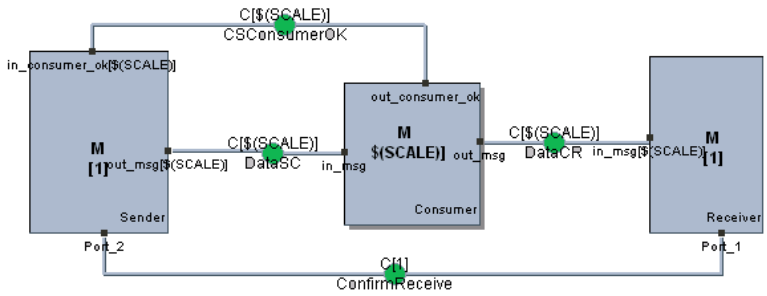

Fig. 3. Software graph for the PET reconstruction application.

In addition, the main data paths form the sender through the consumer to the receiver is visible in the middle augmented by two acknowledgment paths leading back to the sender.

The software-graph with its three modules can be mapped onto arbitrary hardware graphs. This enables the easy distribution of consumer module instances across arbitrarily scaled clusters. It is also possible to change the number of nodes without changing the software-graph by simply remapping it to a different hardware-graph. This independence of hardware and software description drastically eases the port of application between different hardware configuration and allows for an easy-to-handle scalability for the PET reconstruction application without the need for any code modifications.

6 Experimental Setup and Results

For the evaluation of the approach presented here, a research cluster consisting of four equally configured Dual processor PC nodes has been used. Each node is based on Intel's Xeon processors running at 450 MHz and is equipped with 512 MB main memory each. The interconnection fabric between the nodes is based on switched Fast Ethernet, which is used for both the external connection to the campus backbone, as well as for inter–node communication during the PET image reconstruction. All nodes run Windows 2000 enhanced only by a separate remote shell daemon.

For the evaluation, two different data sets have been used: one small data set of a human lung and one large data set of a whole human body. While the first one is acquired with a single scan, the latter resembles several consecutive scans of the different body sections, which are then merged into one full image. The complete size and resolution data for both data sets is shown in Table 1 and some examples of resulting image slices for the whole body case are shown in Figure 4. These two sets represent the two extreme cases of data sets currently available in the clinical routine. With improving scanner quality and availability of increased processing power, however, the trend is certainly towards larger data set sizes.

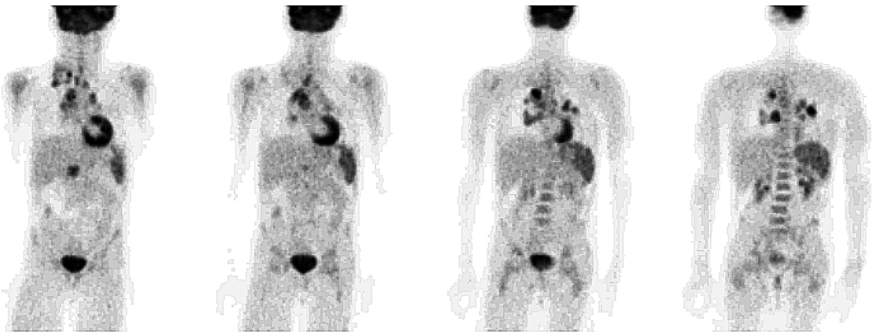

Fig. 4. Reconstructed and post–processed images of a human body (coronal slices, Nuklearmedizin TUM).

Description	Parameters					Execution times		
	Size	Planes	Scan Res.	Image Res.	Scans	Seq.	4 CPUs	8 CPUs
Human lung	14 MB	31	256x192	128x128	1	2m 54s s	1m 13s	1m 2s
Whole body	130 MB	282	256x192	128x128	6	15m 41s	4m 41s	3m 0s

Table 1. Evaluation data sets and the absolut exection times in various configurations

Table 1 also includes the absolute execution times of the code using the two data sets on various numbers of CPUs and Figure 5 (left) shows the same data transformed into speed-ups over the runtime time of a sequential execution without PeakWare involved. While the smaller data set performs rather purely, not exceeding a speed-up of 2.8 on 8 CPUs, the larger data set exhibits a much better performance with a speed-up of over 5.2. More important for the usability of the system in daily clinical routine is that the absolute execution times for any data set does not exceed the 3 minutes on 8 CPUs. This is short enough to allow a medical diagnosis based on a scan while the patient is still inside the PET scanner. Scans can therefore be repeated or adjusted to new target areas immediately without having to reschedule the patient for additional scans.

For a closer evaluation of the performance details, Figure 5 (middle and right) shows the aggregated absolute execution times over all CPUs in a breakdown into the different program phases: the startup time needed by the PeakWare environment to launch the program on all nodes, the file I/O time needed to read the initial data set and to store the final image volume, the time spent in the communication subroutines, and the time needed for the actual image reconstruction. The latter one is further broken down in a phase called *Weights*, a sequential preprocessing step, and *Rekon* the execution of the iterative reconstruction process. In this kind of graph bars of equal heights indicate perfect scalability, as the total execution time or work spent on any number of nodes is

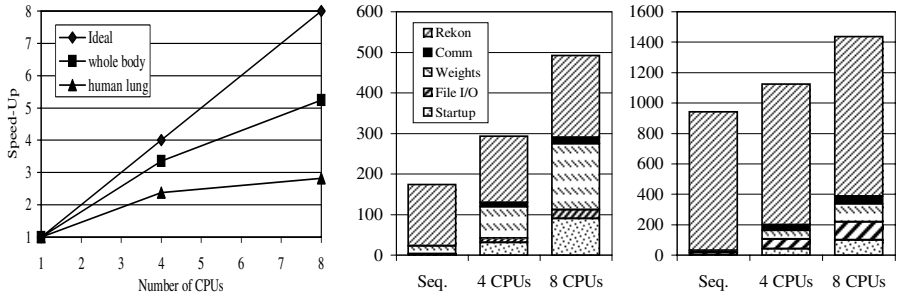

Fig. 5. Speedup (left) and Aggregated execution times (in sec) — middle: data set human lung, right: data set whole body.

equal showing that no additional overhead is introduced during the parallelization.

This behavior is clearly visible for the actual reconstruction phase for both data sets. In the small data set, however, where the total execution time on larger number of nodes is dominated by the sequential and therefore not scalable phases of the code, especially the preprocessing phase *Weights*, the overall speed-up is severely limited. In the case of the larger data set, the reconstruction time clearly outweighs any other phase, which translated into the high speed-up values shown above. It can be noted, however, that with increasing data set sizes, the I/O phase becomes more relevant demanding a new parallel I/O solution for the future with even larger data sets envisioned.

7 Conclusions and Future work

Nuclear imaging using Positron Emission Tomography is establishing itself as an important and very useful method for medical diagnosis. It is, however, connected with large computational demands for the image preparation. In order to match those, parallel processing is required and clusters built from commodity components provide a very cost–effective platform for this application domain. The problem connected with this approach is its complex and cumbersome programmability, making it difficult for scientists to use them at their full capacity.

In order to overcome this problem, the approach presented here deploys a graphical tool, which allows the specification of independent modules along with the communication between them. This raises the level of abstraction significantly and therefore eases the implementation process. The result is a very efficient and easy–to–implement and –use PET image reconstruction system, which satisfies the requirements for a use in daily clinical routine. It is already used in a production environment and has found acceptance with both doctors and medical personnel.

Based on this success, the approach is likely to be widened to more applications within the domain of nuclear medical imaging, like the spectral analysis

for the evaluation of tracer concentrations in the human body over time and the correlation of PET images to guarantee a clean overlay. In addition, further optimizations can be made in the area of parallel I/O to provide more scalability in this phase and by applying modern high–speed interconnection technologies for applications with a high communication demand. All this will lead to an integrated cluster based solution of image processing of nuclear medical data and will allow easier, faster, and more accurate utilization of this rising field within medicine.

Acknowledgments

This work was supported by the European Commission in the Fourth Framework Programme in the context of the ESPRIT Project 29907, NEPHEW (*NE*twork of *P*Cs *HE*terogeneous *W*indows-NT Engineering Toolset). The reconstruction algorithms have been implemented at the University of Michigan by Prof. Fessler and are distributed in the form of a library, called ASPIRE [2].

References

1. A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. *A User' s Guide to PVM Parallel Virtual Machine*. Oak Ridge National Laboratory, Oak Ridge, TN 37831-8083, July 1991.
2. J. Fessler. Aspire 3.0 user's guide: A sparse iterative reconstruction library. Technical Report TR–95–293, Communications & Signal Processing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan Ann Arbor, Michigan 48109-2122, November 2000. Revised version.
3. G.T. Herman. *Image Reconstruction from Projections*. Springer-Verlag, Berlin Heidelberg New York, 1979.
4. H. Hudson and R. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Transactions on Medical Imaging*, 13:601–609, 1994.
5. R. Leahy and C. Byrne. Recent developments in iterative image reconstruction for PET and SPECT. *IEEE Transactions on Nuclear Sciences*, 19:257–260, 2000.
6. Message Passing Interface Forum (MPIF). MPI: A Message-Passing Interface Standard. Technical Report, University of Tennessee, Knoxville, June 1995. http://www.mpi-forum.org/.
7. S. Vollmar, M. Lercher, C. Knöss, C. Michael, K. Wienhard, and W. Heiss. BeeHive: Cluster Reconstruction of 3-D PET Data in a Windows NT network using FORE. In *In proceedings of the Nuclear Science Symposium and Medical Imaging Conference*, October 2000.
8. WWW:. Peakware — Matra Systems & Information. http://www.matra-msi.com/ang/savoir_infor_peakware_d.htm, January 2000.
9. WWW:. SMiLE: Nephew (Esprit project 29907). http://wwwbode.in.tum.de/Par/arch/smile/nephew, June 2000.