Application of JavaSpaces to improve iterative reconstruction of SPECT data

P. Knoll, S. Mirzaei & H. Köhn

*L. Boltzmann Institute of Nuclear Medicine*
*Wilhelminenspital, Vienna, Austria*

**Abstract**

Iterative algorithms to reconstruct single photon emission computerized tomography (SPECT) data are based on the mathematical simulation of the acquisition process. The reconstruction times of these methods are much longer than that of routinely used reconstruction methods. Java, a platform independent programming language changed the way of software design by using Jini and JavaSpaces, new technologies that have been introduced recently. By applying JavaSpaces, a *space* is used to store objects persistently which can be used also for effective parallel processing. In this paper, we report a novel approach for iterative reconstruction of SPECT data by means of JavaSpaces, which uses only standard personal computer equipment and results in significant improvement of reconstruction time due to the fact that several layers of the object are computed in parallel.

**1 Introduction**

Java platform developed by Sun Microsystems ([http://www.sun.com](http://www.sun.com) [1]) represents a new means of computation specially designed for network computing and based on the idea that computer software should run on different available computer hardware and operating systems. Software written in Java and compiled with the freely available Java Development Kit 1.4 ([http://java.sun.com/products/jdk1.4](http://java.sun.com/products/jdk1.4) [2]) runs different kinds of computers due to the Java virtual machine which enables the Java platform to host applications on any computer without rewriting the programming code. Java applications are already used in medicine, but mainly to distribute clinical information (Malameteniou [3], Slomka [4]). However, it does not seem reasonable to translate already existing programs written in C/C++ into Java. A new
The Jini concept, which is 100% pure Java, is a brand new technology that can be used to create homogeneous distributed systems. The JavaSpaces technology kit (JSTK 1.0) ([http://www.sun.com/jini/index.html](http://www.sun.com/jini/index.html)), which is part of the Jini application programming interface (API) provides a fundamentally different programming model. This model considers application as a collection of processes cooperating via the flow of objects into and out of one or more spaces. In this context, these spaces are used as a shared and network accessible repository for data objects (Freeman [6]). Therefore, the space-based model of distributing computing can also be elegantly used to solve computing-intensive problems faster by means of parallel application patterns.

Nuclear medicine is a medical speciality that applies small amounts of radioactive materials or radiopharmaceuticals, which are attracted to specific organs, bones, or tissues to diagnose and treat disease. Nuclear medicine imaging is unique because it documents organ function and structure, in contrast to diagnostic radiology, which is based upon anatomy. Nuclear medicine imaging techniques often identify abnormalities very early in the progression of the disease – long before some medical problems are apparent with other diagnostic tests. This early detection allows a disease to be treated early in its course when there may be a more successful prognosis. A radioactive tracer is intravenously injected and transported into the organ of interest from which the emitted photons pass through the body of the patient. These photons can be detected externally by a gamma camera. The radionuclides used for gamma cameras emit only a single photon and therefore a special “lens” known as a collimator, which is placed between detector surface and the patient, has to be used to select a ray orientation. The resulting scintigram represents the two-dimensional projection of the three-dimensional activity distribution and therefore the tomographic information is lost if only a single planar acquisition is performed. Rotating gamma cameras have been developed to utilize this information and the single photon emission computed tomography (SPECT) acquisition technique is widely used. During a standard SPECT acquisition the head of the gamma camera rotates around the patient. In order to determine the three-dimensional activity distribution mathematical reconstruction algorithms are applied.

SPECT data can be reconstructed either by filtered back projection (FBP) or iterative methods. Instead of filtering and back projecting the projection data, iterative algorithms try to simulate the acquisition process itself and minimize the difference between measured and simulated data. This process is repeated until a “final” image is found. Due to the time-consuming iterative calculations, the reconstruction time of these methods are much longer than that of the FBP even if very simple acquisition models (no scatter, no attenuation, etc.) are applied. Therefore, the FBP is still the standard reconstruction method in nuclear medicine. By separating the algorithm into independent tasks, it is possible to shorten the computation time but existing methods mostly require either specialized hardware (Passieri [7], Comtat [8]) or network computing (Kontaxakis [9]). Instead of using dedicated hardware or peer-to-peer network communication we present a new approach of successful parallel implementation.
of an iterative algorithm for reconstructing of SPECT data using a truly distributed system and a space-based programming model.

2 Methods

To accelerate iterative reconstruction, it is possible to decompose this computing intensive problem into smaller independent tasks that then can be computed simultaneously.

Parallel computation obtains an increase in performance by using multiple processors, which are connected together in some kind of network resulting in an improvement in calculation time. In principle there are three primary classes of techniques for creating a parallel algorithm to perform a task (Newman [10]):

- Trivial parallel algorithm
- Functional decomposition
- Domain decomposition

Trivial parallel problems can be split up into separate unrelated tasks, which can be solved by separate programs running on isolated processors without the need for inter-process communication. Functional decomposition refers to the breaking up of a task into a variety of different jobs: each processor is executing a different type of work, and the results of all processors put together perform the complete task. Domain decomposition algorithms split up the task and each processor works on a different slice or on a different part of the same slice.

Recently, Sun Microsystems introduced Jini and JavaSpaces, which are new technologies that changed the way of software design fundamentally (Freeman [6], Arnold [11]). Jini, built on top of Java, provides a homogeneous view of the network and does not require any centralized administration of available services as opposed to already existing communication patterns. JavaSpaces uses the Jini programming model and provides a simple mechanism for sharing objects in network resources based on Java technology. As for any other Jini service it is necessary to run

- An HTTP server
- The Remote Method Invocation daemon (RMID)
- A lookup service
- A transaction manager (Arnold[11], Edwards [12])

when applying Java Spaces. The Jini API includes a simple HTTP server, which is necessary for exporting the code. The RMID uses a log file to keep track of the on demand activation and persistent service registration. Once a service has already been registered, it does not have to re-register after each reboot. Instead, RMID will restart it upon start-up (Edwards [12]). Lookup services keep track of all services that have joined a Jini community. Jini services broadcast its presence by dropping a multicast packet to a certain IP/Port (4160) as defined in the Jini specification. Furthermore the Jini service also establishes a TCP server socket that awaits incoming connections (Fig.1).
The multicast request relies upon the User Datagram Protocol (UDP) that allows multicast (one-to-many) networking to function properly. Lookup services monitor IP/Port 4160 for incoming requests that provide information about the Jini service such as the IP/Port number. The lookup service uses this information to contact the Jini service directly using UDP, and receives and registers a proxy object of the Jini service (Fig. 2).

The Jini client contacts the Jini lookup service, downloads the proxy object reference and uses that proxy to communicate directly with the Jini service (Fig. 3). Transactions are effective ways to organize a series of related operations resulting in only two possible outcomes: either all operations succeed or fail. Transactions, coordinated by a centralized transaction manager, are executed as if they were a single (“atomic”) operation.
The Jini API introduces a “leasing concept” that is able to clean up the whole system without human intervention if components have failed. Leasing is based on the idea that the resource is loaned to clients only for a fixed period of time. If this time interval expires, the resource is automatically removed. If all these processes (HTTP server, RMID, lookup service and transaction manager) run on the host, a space can be created. In a distributed application, JavaSpaces act as virtual spaces between providers and users of Java objects. When applying this programming model, processes are loosely coupled because in this case processes interact indirectly through a space and not directly with other processes. One of the most common application pattern used for parallel computing is the replicated worker pattern. This pattern involves a master process along with a number of workers (Freeman [6]). An iterative reconstruction algorithm based on the error back propagation method, which is used in artificial neural networks (Knoll [13, 14]) was implemented. It applies the space-based model with one master process and a number of reconstruction workers. The reconstruction software package was installed on 6 personal computers (PCs) routinely available in our department. For the trivial parallel approach we used the following strategy: The master process simply writes the slice numbers and the projection data to be reconstructed into the space using a template matching mechanism. Each of the reconstruction workers, which run on separate computer, takes one of the tasks, removes it from the space, and reconstructs only the slice that corresponds to this specific task. Since the task is removed from the space, exclusive access to this job is guaranteed. After reconstruction, the worker writes the reconstructed transaxial slice back into the space from where it will be collected by the master process. After the worker has finished the work, the space is searched again for more available tasks. If successful, the worker starts to reconstruct the next slice. At the same time, the other workers reconstruct other slices and write them into the space. When all slices are reconstructed, the master process collects and views the reconstructed slices using a self-developed Java slice viewer. As shown by various authors (Luig [15], Zeng[16]) it is possible to improve SPECT image quality further by modeling the gamma camera characteristics such as camera resolution and
scatter effects. These effects make the slices depend on each other, and each slice cannot be reconstructed independently. For the domain decomposition approach each worker applies only one iteration to calculate the correction factors for the voxel values. Again the result of this iterative procedure is written into the space from where it is removed by the master. In contrast to the trivial parallel approach the master updates all voxel. If all slices are updated, they are written back into the space. This process is repeated until the “final” image is found.

3 Results

We compared the runtime of the parallel implementation ($T_p$) with the execution time of a single processor Java installation ($T_1$), which was performed on one personal computer of our department. Each simulation was performed for 10 times. To quantify the results we computed a speedup factor, which is defined as ratio of $\frac{T_1}{T_p}$.

3.1 Trivial parallel algorithm

Different matrix size

We performed this study using a fixed number of worker (5) but different matrix sizes (32x32, 64x64, 128x128 and 256x256). The corresponding runtimes vs. the matrix size are shown in Fig. 4. The figure illustrates a nearly linear speed up for all matrix sizes.

![Fig. 4: Trivial Parallel Algorithm I: Fixed number of workers (5), different matrix size (32x32, 64x64, 128x128, 256x256)](image)

Different number of worker

In this experiment we applied a 128x128 matrix but varied the number of worker (1-6). The resulting runtime curve shows that a speed up increases with the number of worker (Fig. 5).
Fig. 5: Trivial Parallel Algorithm II: Different number of workers (1-6), constant matrix size (128x128)

If the slices of a 128x128 matrix can be computed independently of each other it is already faster to use the parallel JavaSpaces reconstruction algorithm if two workers are used

3.2 Domain decomposition

Different matrix size

Again we performed this simulation using a fixed number of workers (5), but different matrix sizes (32x32, 64x64, 128x128, 256x256). The resulting ntime curve, shown in Fig. 6, illustrates the advantage of the parallel JavaSpaces implementation if the matrix size is larger than 64x64.

Fig. 6: Domain Decomposition Algorithm I: Constant number of workers, different matrix size (32x32, 64x64, 128x128, 256x256)

Different number of worker

To study the influence of the different number of workers we applied a constant lattice size (128x128) and varied the number of worker (1-6). The resulting
runtime curve shows an acceleration of processing time if the number of workers is larger > 2 (Fig. 7)

Fig. 7: Domain Decomposition Algorithm II:
Different number of worker (1-6),
constant matrix size (128x128)

4 Discussion

Distributed applications are - despite their advantages - difficult to design due to the complexity of the distributed environment which does not concern when writing standalone applications (Farley [17]). In the past, the variety of existing machine architectures and software platforms has prevented the development and proliferation of distributed applications (Arnold [11]). JavaSpaces is a high-level coordination tool for combining processes into a distributed application by means of a network-accessible "space". This recently introduced technology (http://www.sun.com/jini/index.html [5], Freeman [6]) can also be elegantly used to solve computational intensive tasks. In this paper, a new approach of the parallel implementation of an iterative reconstruction algorithm using Jini and JavaSpaces is presented, which reduces the computation time by several orders of magnitude. Although other attempts of parallel implementation for tomographic reconstruction methods using networking and/or dedicated hardware has been described before (Passieri [7], Comtat [8], Kontaxakis [9]), a truly distributed system applying only standard PC equipment to reconstruct SPECT data has not been presented before. Applying JavaSpaces, the SPECT acquisition data are distributed by a master process between five reconstruction workers, which reside on different computers of our department’s Intranet. Using the trivial parallel approach each reconstruction worker computes specific layers of the object. After successful computation of all the layers, the master process collects and views the reconstructed slices. Using five reconstruction workers the resulting speedup increases nearly linear for matrix sizes ≤ 128x128. Using a 128x128 matrix and the trivial parallel approach we obtained a speedup of 2.2 (Fig. 4). If we used the domain decomposition algorithm we measured a speedup
of 1.67 (Fig. 6). If we increase the size of the matrix to 256x256 we found a deviation form the linear behavior of the speedup factor as found if smaller matrix sizes are used. This behavior might occur because we used for our experiments computers with 128 MB RAM, which are too small for this large matrix and our software package.

If we used a fixed matrix size (128x128) and varied the number of reconstruction worker (1-6) we found for the trivial parallel approach (Fig. 5) but also for the domain decomposition (Fig. 7) a linear dependence.

The significant acceleration of processing time is due to:

- Parallel implementation
- Loading of auxiliary data
- Correct balance between computation and communication
- Processing and transfer time

The parallel implementation of the iterative algorithm enables reconstruction of several layers of the object at the same time. If it is for the reconstruction workers possible to load auxiliary data, needed for iterative reconstruction, in advance the speedup factor is much higher (approximately 20 if 5 computers are involved). A correct balance between computation and communication is difficult to achieve with other software design models (Cimpat [8]). Using a JavaSpaces implementation, the reconstruction workers stay busy and compute tasks in relation to their availability and ability to work. Consideration of both, the processing and transfer time is important for application in a clinical routine environment. The applied platform independent technology allows the use of hardware, which already exists in any department, thus avoiding communication bottlenecks since transfer time over the network connection is minimized. JavaSpaces technology does not only improve tomographic reconstruction but may also be used as a platform-independent, cost-effective means to shorten the computation time of other time-consuming computational tasks.

5 References

Applications of High-Performance Computing in Engineering VII


