
ELSEVIER

Parallel Computing 24 (1998) 2129–2142

PARALLEL COMPUTING

Data-parallel tomographic reconstruction: A comparison of filtered backprojection and direct Fourier reconstruction

Jos B.T.M. Roerdink *, Michel A. Westenberg

Institute for Mathematics and Computing Science, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands

Received 4 October 1997; received in revised form 14 April 1998

Abstract

We consider the parallelization of two standard 2D reconstruction algorithms, filtered backprojection and direct Fourier reconstruction, using the data-parallel programming style. The algorithms are implemented on a Connection Machine CM-5 with 16 processors and a peak performance of 2 Gflop/s.

Keywords: Computerized tomography; Filtered backprojection; Direct Fourier reconstruction; Data-parallel implementation; Connection Machine CM-5

1. Introduction

Computerized tomography (CT) deals with reconstruction from projections by means of digital computers. In practice, it is a method to reconstruct cross sections of the interior structure of an object without having to cut or damage the object. One of the most prominent applications of computerized tomography occurs in diagnostic medicine, where the method is used to produce images of the interior of human organs. A related technique is positron emission tomography (PET) which uses radioactive isotopes, and can for example be used for functional brain imaging. Other applications arise in radio astronomy, 3D electron microscopy, soil science, non-destructive testing, aerodynamics and geophysics, to name a few [1–3].

* Corresponding author. E-mail: roe@cs.rug.nl

0167-8191/98/$ – see front matter
PII: S0167-8191(98)00095-7

One of the problems in CT is the fact that it is computationally intensive. A straightforward implementation for a 2D image of size $N \times N$ leads to an algorithm of time complexity $O(N^3)$. This is quite an obstacle for CT applications where real-time performance is required. This holds even more so for reconstruction of 3D images, or time sequences of images, such as is the case in cardiac imaging [4]. Therefore we investigate in this paper the possibility of reducing reconstruction time by parallel processing.

Parallel computing in tomography has previously been used for the case of algebraic methods such as the algebraic reconstruction technique (ART), where one starts from a full discretization of Radon's integral equation, leading to a large and sparse linear system of equations, usually solved by an iterative method [2]. Parallelization in this case may be based on standard parallel solution methods of sets of linear equations. Zenios and Censor [5] describe an implementation of a block-iterative version of Multiplicative ART on a Cray X-MP/48. Other parallel implementations of iterative types of algorithms in PET were reported in [6,7]. Barresi et al. [8] implemented the 3D filtered backprojection algorithm on a transputer system, using parallel FFT algorithms. In [9] an implementation of the filtered backprojection algorithm is given on a hypercube system with 32 nodes (Intel iPSC/2). Laurent et al. [10] give a comparison of reconstruction algorithms (some of them iterative, some based on analytic procedures) on MIMD computers.

Most of the implementations mentioned above have been based on *control parallelism* using explicit message passing. In this paper we study a *data-parallel* approach, which is well suited for tomographic reconstruction in view of the many array operations involved, and offers advantages in terms of ease of programming since no extra code to ensure synchronization has to be written. Although most rewards are to be expected for large 3D reconstruction problems, we have decided to test the feasibility of a data-parallel approach first for the two standard reconstruction algorithms in 2D computerized tomography: filtered backprojection and direct Fourier reconstruction. The algorithms have been implemented on a Connection Machine CM-5 distributed memory MIMD computer with 16 processors and a peak performance of 2 Gflop/s. The results are very promising, although in the case of filtered backprojection one has to use special care to achieve good speedup while keeping memory requirements per processing node limited. Reconstruction times are in the order of seconds for image sizes up to 512×512. This opens the way to fast data-parallel 3D reconstruction methods for large data sets such as occur in medical imaging and scientific visualization.

The organization of this paper is as follows. Section 2 describes the parallelization of the filtered backprojection algorithm and the same is done for direct Fourier reconstruction in Section 3. Conclusions are given in Section 4.

2. Filtered backprojection

The filtered backprojection algorithm for tomographic reconstruction [11,3] is currently the most used reconstruction method in the medical field, and has proved

to be very accurate. This section deals with the algorithm and its parallel implementation on a Connection Machine CM-5.

2.1. Mathematical basis

Let f be the density to be reconstructed from its 2D *Radon transform* $\mathscr{R}f$, i.e. from its line integrals

$$\mathscr{R}f(\theta,t) := \int_{\mathbb{R}^2} f(x,y)\,\delta(x\cos\theta + y\sin\theta - t)\,\mathrm{d}x\,\mathrm{d}y, \tag{1}$$

where θ is an angle between 0 and 2π and t a real number. In the case of a parallel sampling geometry, which is used throughout this paper, the symmetry relation $\mathscr{R}f(\theta,t) = \mathscr{R}f(\theta+\pi,-t)$ allows θ to be taken between 0 and π. The integral $\mathscr{R}f(\theta,t)$, with θ and t fixed, is called a *projection* and the function $\mathscr{R}_\theta : t \mapsto \mathscr{R}f(\theta,t)$ a *profile*.

The basis of the filtered backprojection algorithm is formed by the following reconstruction formula:

$$f(x,y) = \int_0^\pi \int_{-\infty}^{\infty} \hat{\mathscr{R}}_\theta(\lambda)|\lambda| \exp\left[\mathrm{i}2\pi\lambda(x\cos\theta + y\sin\theta)\right] \mathrm{d}\lambda\,\mathrm{d}\theta, \tag{2}$$

where $\hat{\mathscr{R}}_\theta$ denotes the 1D Fourier transform of a profile $\mathscr{R}_\theta$ taken at angle θ. The multiplication with $|\lambda|$ in the Fourier domain represents a filtering operation, where the filter is commonly called the *ramp* filter. The integral over θ constitutes the so-called backprojection step.

In practice one replaces the ideal ramp filter $|\lambda|$ in the frequency domain by a function $W_b(\lambda) = |\lambda|H(\lambda)$, where the 'window function' $H(\lambda)$ approximates 1 for low frequencies but goes to zero for high frequencies, in order to suppress noise.

2.2. Algorithm

In the discrete case the Radon transform $\mathscr{R}f(\theta,t)$ is assumed to be available for (θ_j, s_l), $j = 1,\ldots,p$, $l = -q,\ldots,q$, with

$$\theta_j = \pi(j-1)/p, \quad s_l = hl, \quad h = 1/q. \tag{3}$$

Here it is assumed that the support of f is the unit disk, i.e. $|f(\vec{r})| = 0$ for $\|\vec{r}\| > 1$. The filtered backprojection algorithm in this case goes as follows [3].

Step 1: For $j = 1,\ldots,p$ carry out the convolutions

$$v_{j,k} = 2h\sum_{l=-q}^{q} w_b(s_k - s_l)\mathscr{R}_{\theta_j}(s_l), \quad k = -q,\ldots,q. \tag{4}$$

The convolution may be computed in the frequency domain using the FFT. The Fourier transform of w_b is the function W_b defined above. For W_b a Hanning window is used in this paper [12].

Step 2: For each reconstruction point $\vec{r}$, compute the discrete backprojection

$$f(\vec{r}) = \frac{\pi}{p}\sum_{j=1}^{p}\big((1-u)v_{j,k} + uv_{j,k+1}\big), \tag{5}$$

where, for each $\vec{r} = (x, y)$ and j, k and u are determined by

$$s = x\cos\theta_j + y\sin\theta_j, \quad k \leqslant \frac{s}{h} \leqslant k+1, \quad u = \frac{s}{h} - k. \tag{6}$$

Reconstruction may be performed on a $(2q+1)\times(2q+1)$ grid, which corresponds to the sampling of the profiles.

The complexity of filtered backprojection can be estimated as follows. The number of operations needed for the convolutions is $O(pq \log q)$ if an FFT is used. The backprojection operation takes $O(p)$ operations for each $\vec{r}$. If the reconstruction is performed on a $(2q+1)\times(2q+1)$ grid, this leads to $O(pq^2)$ operations. Using the optimal relation $p = \pi q$ [3], the total complexity is $O(q^3)$.

2.3. Parallel implementation

The filtered backprojection algorithm was implemented using the data-parallel programming style on a Connection Machine CM-5 with 16 processing nodes, each having four vector units, aggregate memory of 512 Mb, a RAID of 7 Gb and a peak performance of 2 Gflop/s. Program execution on this machine starts on a control processor – also called the 'front-end' – where all sequential operations execute, whereas the parallel operations are performed by the processing nodes. There are two fat-tree communication networks, one connecting the control processor to the processing nodes for synchronization and broadcasting, and a data network interconnecting the nodes for bulk data transfer. All programs were written in CM Fortran which has Fortran-90 arrays as its parallel data structures, with several extensions for data-parallel programming such as the `FORALL` statement and additional intrinsic functions.

Since the hardware configuration of the CM-5 at our disposal did not allow variation of the number of processors to individual users we estimated speedups by varying the size of the problem, i.e. the dimensions of the image to be reconstructed. The *scaled speedup* ([13], pp. 81–84) is the ratio between the *estimated* time it takes to run a sequential program on one processor of a parallel computer and the *actual* time it takes to run the parallel program on multiple processors of the same parallel computer. It may be difficult to estimate the time it takes to run the sequential algorithm on one processor, since one may not have a sequential implementation of the algorithm or, as mentioned above, one cannot vary the number of processors. For those cases, we define the normalized scaled speedup (NSS) to be the scaled speedup which is normalized with respect to a fixed problem size k

$$\mathrm{NSS}_k(N) = \frac{T_\mathrm{e}(N)}{T_p(N)}\frac{T_p(k)}{T_\mathrm{e}(k)}. \tag{7}$$

The estimated sequential time taken is denoted by $T_\mathrm{e}(N)$ and the time to run the parallel program on multiple processors is denoted by $T_\mathrm{p}(N)$, where N is the problem

size. The normalization factor is the ratio between $T_p(k)$ and $T_e(k)$. In this paper we will use $k = 128$. The advantage of the normalized scaled speedup is that one can simply use the time complexity of an algorithm to compute $T_e(N)$. For an optimal sequential algorithm, the normalized scaled speedup would be equal to 1, independent of the problem size N. On a parallel computer, one would expect the normalized scaled speedup to increase with N, a phenomenon which is known as the Amdahl effect ([13], pp. 81–84).

The filtering step of the algorithm is easily implemented on the CM-5 by using the specialized FFT's which are provided by the Connection Machine Scientific Software Library (CMSSL). The backprojection step is more challenging and we describe three different implementations in the next sections.

2.3.1. Straightforward method

The first implementation is a direct translation of Eq. (5) for the backprojection step into CM-Fortran code. The filtered projection data are stored in a 2D array $\mathbf{V}(j,k)$, where the elements along the first dimension are distributed blockwise over the processing nodes and the elements along the second dimension are located on the same processing node. The reconstructed image is stored in a 2D array $\mathbf{F}(x,y)$, where the elements along both dimensions are distributed blockwise over the processing nodes.

The discrete backprojection consists of three nested loops, one over the angles j and two over the elements (x,y) of the reconstruction grid. When the data-parallel programming paradigm is adopted on the CM-5, it is not possible to compute all three loops in parallel. Instead, one has to serialize one of the loops. The effect of serialization of a loop is that it is executed by the front-end. In our case, the loop over j is serialized so that for each j the front-end instructs the processing nodes to execute the loop over (x,y) in parallel. It would be possible to remove the loop over j by adding a third component to the reconstruction grid. Computations could then be carried out on all dimensions in parallel and afterwards be summed in the j-direction. The drawback of this method is that it consumes a lot of memory. In this paper, the method of serialization is adopted.

Timing results are shown in Table 1. The parameter N corresponds to the size of the reconstruction grid ($N \times N$). Projections were generated using N rays and N

Table 1
Timing results in seconds on the CM-5 for different implementations of the backprojection step of the algorithm. The last column contains the normalized scaled speedups when using lookup tables for the backprojection

N	Filtering	Backprojection			NSS
		Straight-forward	Lookup tables	One lookup table	
128	0.059	3.820	0.224	0.207	1
256	0.100	22.876	1.310	1.129	1.6
512	0.213	120.581	8.541	7.652	2.1
1024	0.754	1004.963	57.435	–	2.5

angles. For reasons of simplicity in the analysis of the algorithms we do not use the optimal relation $p = \pi q$ for the timings. As can be seen, the backprojection step consumes a lot of time and grows roughly by a factor of 6 when N is doubled. For each point in the reconstruction grid information of all angles is needed, and this information is distributed over different processors. This means that there is a maximum of $O(N^2)$ communications for each angle. It is not possible to choose the layout of the data in such a way that there will be (almost) no communication. A partial solution to this problem is given in the next section.

2.3.2. Use of lookup tables

The communicational demand can be decreased by making use of a so-called lookup table. For a given j, the filtered profile $\mathbf{V}(j,\cdot)$ is sent from the processing nodes to the front-end. The front-end then broadcasts these data to all processing nodes, the result of which is that each processing node receives a copy of these data. The local copy constitutes the lookup table. Addressing can now be done locally and thus requires no more communication. This reduces the number of communications for a given angle j from $O(N^2)$ to $O(N)$, and furthermore, the high-bandwidth communication network of the CM-5 is used more efficiently.

Table 1 shows the dramatic reduction in run-time when using lookup tables. On the average a speedup by a factor of about 16 is achieved for values of $N \geqslant 128$. Note the reduction in run-time for $N = 1024$, from almost 17 min down to 57 s, see also Fig. 1.

2.3.3. One lookup table

The method in Section 2.3.2 loops over the angles j at which the profiles $\mathbf{V}(j,\cdot)$ are available and creates in each step a lookup table of the appropriate profile. This

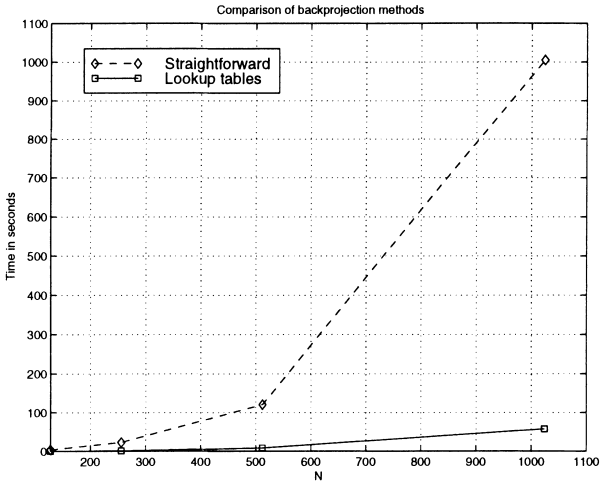

Fig. 1. Comparison of the run-time of the straightforward method and the method using lookup tables.

causes some overhead in each step for the creation and destruction of such a lookup table. An alternative is to create one large lookup table of all profiles in advance. The corresponding timing results are shown in Table 1 under 'Backprojection'. For $N = 512$ a reduction of almost a second is achieved which is only about 10%. A problem with this method is the memory usage, since a copy of all projection data is assigned to each processor. For $N = 1024$ the memory requirements are too large and the program cannot reconstruct images of this size. The results show that using one large lookup table hardly gives improvements in run-time. In view of the additional increase of memory usage, we therefore conclude that this method is less useful than the one described in Section 2.3.2.

2.3.4. Final algorithm

The final algorithm uses the lookup tables described in Section 2.3.2. A pseudo-code can be found in Algorithm 1.

Algorithm 1. Parallel implementation of filtered backprojection.

procedure FBP (**R**)
{**R** is a 2D array which contains the projection data, p denotes the number of angles and D the reconstruction grid.}
{Filter the projection data in parallel, applying a 1D FFT to each row of a matrix. The matrix $\mathbf{W}_b$ contains a copy of the filter W_b on each row.}
$\mathbf{V} = 2h \bullet \text{IFFT}(\mathbf{W}_b \bullet \text{FFT}(\mathbf{R}))$ {$\bullet$ denotes point-wise multiplication.}
for $j = 1$ to p **do** {Compute discrete backprojection.}
for all (x, y) on D **do** {Compute indices.}
$\mathbf{S}(x, y) = x \cos(\theta_j) + y \sin(\theta_j)$
$\mathbf{K}(x, y) = \lfloor \frac{1}{h} \mathbf{S}(x, y) \rfloor$
$\mathbf{U}(x, y) = \frac{1}{h} \mathbf{S}(x, y) - \mathbf{K}(x, y)$
end for
{Create a lookup table T containing the projection data for angle j, i.e. row j of matrix **V**, and assign a copy to each processor. Use this table to find the projection values $\mathbf{V}(j, k)$ and $\mathbf{V}(j, k+1)$ and store these in the matrices $\mathbf{V}_0$ and $\mathbf{V}_1$, respectively.}
$\mathbf{V}_0 = \text{TABLE\_LOOKUP}(T, k)$
$\mathbf{V}_1 = \text{TABLE\_LOOKUP}(T, k+1)$
{The assignment to $\mathbf{V}_0$ corresponds to $\mathbf{V}_0(x, y) = T(\mathbf{K}(x, y))$ and likewise for $\mathbf{V}_1$.}
for all (x, y) on D **do**
$\mathbf{F}(x, y) = \mathbf{F}(x, y) + \frac{\pi}{p}((1 - \mathbf{U}(x, y))\mathbf{V}_0(x, y) + \mathbf{U}(x, y)\mathbf{V}_1(x, y))$
end for
end for

As a measure for the quality of the parallelization of the above algorithm, we computed the normalized scaled speedup as defined in Eq. (7) for $k = 128$, based on running times for the complete algorithm (filtering and backprojection). We take $T_e(N) = cN^3$, c is a constant which cancels out when evaluating Eq. (7), as the

estimated sequential time. The results can be found in the last column of Table 1, and we see that the efficiency increases with the size N of the image. This is what one would expect, since certain overhead costs such as creating processes and loading the vector units are independent of the problem size. Communication, however, clearly influences the timing results for filtered backprojection. Partly, this problem is solved by using the lookup tables, but there remains considerable communication in the creation of the table.

3. Direct Fourier reconstruction

In this section the direct Fourier reconstruction method will be investigated [3]. This has a lower time complexity than filtered backprojection making it an interesting method to implement in parallel.

3.1. Mathematical basis

The basis of this method is the *Fourier slice theorem* [11,3], which states that the 1D Fourier transform $\hat{\mathscr{R}}_\theta$ of a profile $\mathscr{R}_\theta$ of an image f taken at angle θ gives a slice of the 2D Fourier transform $F(u,v)$ of f, subtending an angle θ with the u-axis

$$\hat{\mathscr{R}}_\theta(\lambda) = F(\lambda \cos\theta, \lambda \sin\theta). \tag{8}$$

So in the continuous case, reconstruction consists of a 1D Fourier transform of the profiles according to Eq. (8) followed by a 2D inverse Fourier transform (IFFT). The problems arise in the discretization of these transforms.

3.2. Standard algorithm

In the discrete case the Radon transform $g(\theta,t) = \mathscr{R}f(\theta,t)$ is assumed to be sampled at (θ_j, s_l), $j = 1,\ldots,p$, $l = -q,\ldots,q$, with θ_j and s_l as given in Eq. (3). In the following, the largest value for l is chosen to be $q-1$ in order to have an even number of values for l. In the ideal case l has a number of values equal to a power of 2, so that base-2 FFT algorithms can be used to arrive at fast implementations. The polar coordinate grid $\mathscr{G}_{p,q}$ is given by

$$\mathscr{G}_{p,q} = \{\pi r\theta_j:\ r = -q,\ldots,q-1, \quad j = 1,\ldots,p\}. \tag{9}$$

In the standard Fourier reconstruction algorithm [3] one first computes an approximation to the 2D Fourier transform $\hat{f}$ of the object function f on radial lines in $\mathscr{G}_{p,q}$, cf. Fig. 2(a). In the next step, nearest neighbour interpolation is used to switch from the polar coordinate grid to a Cartesian grid. Finally a discrete inverse 2D FFT yields the reconstructed image. The crucial step is the interpolation from the polar grid to the Cartesian grid. Simple nearest neighbour interpolation turns out to produce severe artifacts, so several other interpolation techniques have been proposed. Natterer [3] gives two alternative methods. The first one combines

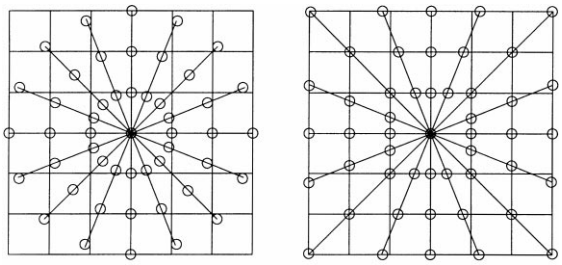

Fig. 2. (a) The representation of $\hat{g}_{jr}$ on the polar grid for $p = 8$, $q = 3$. The circles on the radial lines indicate the points on which $\hat{g}$ is known (b) Result after applying the chirp z-transform to $\hat{g}$.

oversampling of $\hat{g}$ with sinc-interpolation and the other is the so-called Pasciak algorithm [14]. This is the one also used in [15] and will be described next.

3.3. Pasciak algorithm

The idea behind the method of Pasciak [14] is to alter the first step of the standard Fourier reconstruction algorithm in such a way that the values of $\hat{g}(\theta_j, \sigma)$ lie on the vertical lines of the Cartesian grid for $|\cos\,\theta_j| \geqslant |\sin\,\theta_j|$, $\theta_j = \pi(j-1)/p$, and on the horizontal lines otherwise. The values of $\hat{g}$ so obtained can then be assigned to the grid points of the Cartesian grid by using linear interpolation. The algorithm then consists of the following steps [3].

Step 1: For $j = 1, \ldots, p$ compute approximations $\hat{g}_{jr}$ to $\hat{g}(\theta_j, \pi rc(j))$ by

$$\hat{g}_{jr} = \frac{1}{\sqrt{2\pi}} h \sum_{l=-q}^{q-1} \mathrm{e}^{-\mathrm{i}\pi lc(j)r/q} g(\theta_j, s_l), \quad r = -q, \ldots, q-1, \tag{10}$$

where $c(j) = 1/\max\,(|\sin\,\theta_j|, |\cos\,\theta_j|)$.

Step 2: Interpolate the values of $\hat{g}_{jr}$ to the Cartesian grid using linear interpolation.

Step 3: To suppress the high frequencies, filter $\hat{f}$ with the cos-filter.

Step 4: Compute the discrete inverse 2D Fourier transform f_m, $m \in \mathbb{Z}^2$, of $\hat{F}$.

Step 1 can be done by the chirp z-transform [16]. The complexity of this algorithm applied to a sequence of length $2q$ is $\mathrm{O}(q \log q)$, i.e., the same as for the ordinary FFT. The result of this transform is shown in Fig. 2(b). As can be seen, the points of $\hat{g}$ are much closer to the grid points of $\mathscr{G}_{p,q}$ than in Fig. 2(a). Points on the diagonals are now exactly on the grid points and no interpolation is required for them.

The complexity of the Pasciak algorithm can be estimated as follows. The first step takes $\mathrm{O}(pq \log q)$ operations. The second step, the interpolation, uses $\mathrm{O}(q^2)$ operations. The filtering with the cos-filter needs $\mathrm{O}(q^2)$ multiplications and finally the 2D inverse Fourier transform can be done in $\mathrm{O}(q^2 \log q)$ steps. Using the relation $p = \pi q$ the total complexity is $\mathrm{O}(q^2 \log q)$.

3.4. Parallel implementation

The Pasciak algorithm is very suitable for parallelization on the CM-5. The chirp z-transform can be expressed as a convolution, which can be implemented by using FFT's. The distribution of the data array $\mathbf{R}(j,k)$ containing the projection data is the same as with filtered backprojection, i.e. elements along the first dimension are distributed blockwise over the processing nodes and elements along the second dimension are located on the same processing node. This is an efficient distribution for the FFT's used in the chirp z-transform, since only local data access is required during computation. The elements $\hat{f}(k_1,k_2)$ on the reconstruction grid are distributed over the processing nodes in the same way. This is done in order to take full advantage of the vector units for interpolation as well as for the final 2D FFT. Note that this choice is somewhat arbitrary, because one could also store the elements along the first dimension locally and distribute the elements along the second dimension. As for each (x,y) in the reconstruction grid the same operations are to be carried out, the interpolation step is very suitable for parallelization. The computations for the interpolation step can be easily carried out by using the available constructs of CM-Fortran like `WHERE` and `FORALL`. The `WHERE` construct corresponds more or less to a parallel `IF` statement, and `FORALL` to a parallel `DO`. So, this algorithm can be implemented using data-parallel programming in a straightforward manner. A pseudo-code can be found in Algorithm 2.

Algorithm 2. Parallel implementation of direct Fourier reconstruction.

procedure DFR (**R**)
{**R** is a 2D array containing the projection data, p is the number of angles, D the reconstruction grid.}
{Compute the chirp z-transform in parallel over the rows of **R**. For each row a different step-size is used. These are stored in the array **C**.}
$\mathbf{G} = \frac{h}{\sqrt{2\pi}} \bullet \text{chirp\_}z(\mathbf{R},\mathbf{C})$
{Interpolate the values of **G** to the Cartesian grid.}
for all (k_1,k_2) on D **do**
  Interpolate $\hat{f}(k_1,k_2)$ from **G** using linear interpolation
end for
{Filter $\hat{f}$ with the cos-filter.}
for all (k_1,k_2) on D **do**
  $\hat{f}'(k_1,k_2) = \hat{f}(k_1,k_2)\cos\left(\frac{\pi\sqrt{k_1^2+k_2^2}}{2q}\right)$
end for
{Compute the inverse 2-D Fourier transform in parallel.}
$\mathbf{F} = \frac{\pi}{2} \bullet \text{IFFT2}(\hat{f}')$

Timings on the CM-5 were carried out just as in Section 2.3, by varying the parameter N, i.e. the number of rays which equals the number of angles. The size of the reconstruction grid is $N \times N$. The results are shown in Table 2 for different values of N. Again we also give normalized scaled speedups in the last column. We take $T_e(N) = cN^2 \log N$, c a constant, as the estimated sequential time.

Table 2
Timing results in seconds on the CM-5 for the Pasciak algorithm. The last column contains normalized scaled speedups

N	Chirp z	Interpolate	Filter and IFFT	Total	NSS
128	0.100	0.042	0.028	0.173	1
256	0.216	0.127	0.090	0.438	1.8
512	0.402	0.405	0.341	1.158	3.1
1024	1.521	1.917	1.320	4.786	3.3

The chirp z-transform is computed in double precision to prevent round-off errors in an early stage. The interpolation step and inverse Fourier transform are computed in single precision. This is the reason why the chirp z-transform and the inverse 2D Fourier transform consume almost equal time. In the interpolation step there is some communication between processors when at a certain point of the reconstruction grid the actual interpolation is performed. The values needed in this step are distributed over different processing nodes and, just as in the case of filtered backprojection, it is not possible to select the distribution of the arrays over the processing nodes in such a way that no communication is involved. The communication burden, however, is not as strong as for filtered backprojection. The latter algorithm needs information of all angles for a single point in the reconstruction grid, whereas the Pasciak algorithm only needs information of at most two angles.

From a comparison of the timing results in Table 2 with the timings for filtered backprojection (Table 1), it can be verified that the Pasciak method is indeed faster than filtered backprojection. Compared to filtered backprojection, the normalized scaled speedup is also larger and it is clear that direct Fourier reconstruction has a better performance, see also Fig. 3.

3.5. Improvement of the direct Fourier algorithm

As is well known from the literature, the quality of direct Fourier reconstruction is not as good as that of filtered backprojection. In view of the advantage in speed of direct Fourier reconstruction as compared to filtered backprojection there is some room to improve the quality of direct Fourier reconstruction by more advanced interpolation methods such as proposed in [17,15]. Most of these methods involve zero-padding of the original projection data (oversampling of $\hat{g}$). Magnusson [17] employs an 8-point interpolation filter in radial direction and a linear filter in angular direction. Such rather large interpolation filters will not only cause an increase in the number of computations, but lead to an increase in communication between processing nodes as well. This may reduce the performance quite severely when implemented on a parallel system, such as the CM-5.

We therefore propose a modified version of the Pasciak algorithm involving oversampling of $\hat{g}$ and $\hat{f}$ by a factor of 2. Suppose the projection data are taken using N rays and N angles, and are stored in a 2D array of size $N \times N$. In the first step of the algorithm, the projection data are zero-padded resulting in an array $\mathbf{R}(j,k)$ of

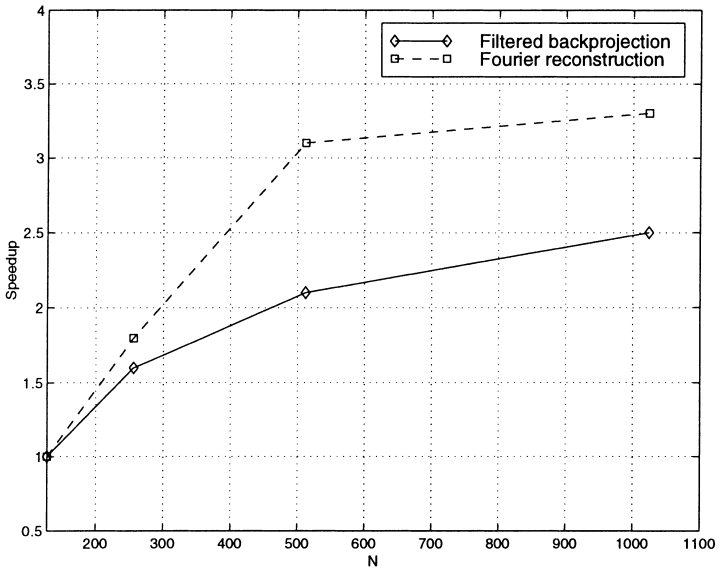

Fig. 3. Normalized scaled speedups of filtered backprojection and direct Fourier reconstruction.

size $N \times 2N$. Then the standard Pasciak algorithm is applied and a reconstructed image of size $2N \times 2N$ is obtained. Afterwards, the extra zeros introduced by zero-padding in the first step of the algorithm are discarded. Timing results are shown in Table 3 for different values of N. The accuracy of this method is comparable to that of filtered backprojection. However, as can be concluded from the table, it is much faster than the latter method. It also seems better adapted to parallelization, since the normalized scaled speedup shows a better scaling.

4. Summary and conclusions

We have addressed the parallelization of two standard reconstruction algorithms, filtered backprojection and direct Fourier reconstruction, using the data-parallel programming style. Implementation was performed on a Connection Machine CM-5

Table 3
Timing results in seconds on the CM-5 for the modified Pasciak algorithm. The last column contains normalized scaled speedups

N	Chirp z	Interpolate	Filter and IFFT	Total	NSS
128	0.199	0.239	0.145	0.592	1
256	0.414	0.545	0.291	1.267	2.1
512	0.847	1.793	1.129	3.777	3.1
1024	3.196	7.932	4.551	15.706	3.3

MIMD computer with 16 processors and a peak performance of 2 Gflop/s. The filtered backprojection algorithm turns out to be somewhat more difficult to parallelize than direct Fourier reconstruction. This is mainly due to the distributed memory of the CM-5, which is the source of considerable communication between the processors. This communication problem is greatly alleviated by using lookup tables for the filtered projections, leading to a fast reconstruction algorithm. Nevertheless, most of the computing time is still spent on communication.

Direct Fourier reconstruction using the Pasciak algorithm is easy to parallelize on the CM-5. The problem of communication is still present, but the burden is not as strong. The time needed for reconstruction is much less than for filtered backprojection, not only expressed in time complexity, but also in actual run-time. The modified Pasciak algorithm has a reconstruction accuracy which is comparable to that of filtered backprojection. Although more time is needed in comparison to the standard Pasciak algorithm, it is still much faster than filtered backprojection, especially for large N. Let us emphasize again however, that the main purpose of this paper has not been to maximize speed, but to show the feasibility of a data-parallel approach, with all its associated advantages.

The results show that the data-parallel programming approach on a distributed memory system such as the CM-5 is suited for both filtered backprojection and direct Fourier reconstruction. One possibility for further research is to implement these algorithms on a shared memory computer to see whether better speedups can be achieved. However, communication with the memory of such a computer would go via a switching network and it is not a priori clear if the algorithm would gain in speed. Another area of future work is the extension to three dimensions of data-parallel reconstruction methods, so that they become viable for large data sets such as occur in medical imaging and scientific visualization. A method which seems to be suitable for a data-parallel implementation is the method of Grangeat [18], which has complexity $O(N^4)$. The input to the algorithm consists of a set of 2D cone-beam projection images from which, in the first step, the derivative of the Radon transform on vertical planes is computed. These operations can be done in parallel, operating locally on each image. When the images are distributed over the processing nodes, no communication is involved. The final reconstruction from the derivative is computed in two steps: application of filtered backprojection to each vertical plane and backprojection in horizontal planes. To implement the first backprojection step, we could extend our method using lookup tables to two dimensions. The second backprojection step can be implemented as a simple slice-wise operation in such a way that no communication is involved.

References

[1] S.R. Deans, The Radon Transform and Some of its Applications, Wiley, New York, 1983.
[2] G.T. Herman, Image Reconstruction from Projections: the Fundamentals of Computerized Tomography, Academic Press, New York, 1980.
[3] F. Natterer, The Mathematics of Computerized Tomography, Teubner, Stuttgart, Wiley, New York, 1986.

[4] J.B.T.M. Roerdink, M. Zwaan, 1993, Cardiac magnetic resonance imaging by retrospective gating: mathematical modelling and reconstruction algorithms, Euro. J. Appl. Math. 4 (1993) 241–270.
[5] S.A. Zenios, Y. Censor, Parallel computing with block-iterative image reconstruction algorithms, Appl. Numer. Math. 7 (1991) 399–415.
[6] M.S. Atkins, D. Murray, R. Harrop, Use of transputers in a 3D Positron Emission Tomograph, IEEE Trans. Med. Imaging 10 (3) (1991) 276–283.
[7] J. Llacer, J. Meng, Matrix based image reconstruction methods for tomography, IEEE Trans. Nucl. Sci. 32 (1) (1985) 855–864.
[8] S. Barresi, D. Bollini, A.D. Guerra, Use of a transputer system for fast 3D image reconstruction in 3D PET, IEEE Trans. Nucl. Sci. 37 (2) (1990) 812–816.
[9] C.M. Chen, S.Y. Lee, Z.H. Cho, A parallel implementation of 3D Computerized Tomography image reconstruction on hypercube multiprocessor, IEEE Trans. Nucl. Sci. 37 (3) (1990) 1333–1346.
[10] C. Laurent, F. Peyrin, J.-M. Chassery, Comparison of Parallel Reconstruction Algorithms for 3D X-RAY Tomography on MIMD Computers, High Performance Computing Symposium 95 (HPCS 95), Montréal, Canada, 1995.
[11] A.C. Kak, M. Slaney, Principles of Computerized Tomographic Imaging, IEEE Press, New York 1988.
[12] S.W. Rowland, Computer Implementation of Image Reconstruction Formulas, in: G.T. Herman (ed.), Image Reconstruction from Projections: Implementation and Applications, Vol. 32, Topics in Appl. Phys., Springer, Berlin, 1979, pp. 9–79.
[13] M.J. Quinn, Parallel Computing, Theory and Practice, McGraw-Hill, New York, 1994.
[14] J.E. Pasciak, A Note on the Fourier Algorithm for Image Reconstruction, preprint, Applied Mathematics Department, Brookhaven National Laboratory, Upton, New York, 1973.
[15] J. Schulte, Fourierrekonstruktion in der Computer-Tomographie, Master's Thesis, Westfälischen Wilhelms-Universität Münster, 1994.
[16] H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, Springer, Berlin, 1982.
[17] M. Magnusson, Linogram and Other Direct Fourier Methods for Tomographic Reconstruction, PhD Thesis, Department of Electrical Engineering, Linköping University, 1993.
[18] P. Grangeat, Mathematical framework of cone beam 3D reconstruction via the first derivative of the Radon transform, in: G.T. Herman, A.K. Louis, F. Natterer (Eds.), Mathematical Methods in Tomography, vol. 1497 of Lecture Notes in Mathematics, Springer, Berlin, 1991.