A Parallel Implementation of 3-D CT Image Reconstruction on Hypercube Multiprocessors

C.M. Chen, S.Y. Lee and Z.H Chao

Presenter: Lutful Karim

Outline

- Introduction
- Convolution and Backprojection
- Modified 2-D Convolution Algorithm
- Incremental Backprojection Algorithm
- Parallelization
- Results
- Conclusion

Introduction

- Computerized Tomography (CT)
  - advanced x-ray and computer system that makes detailed pictures of cross-sections of the body
  - Important role in medical diagnosis (image processing)
  - Started 1970's
  - Large amount of projection data needs to be processed
    - Long time required to get an image
- Why CT?
  - A CT scan is 100 times clearer than an ordinary x-ray.
  - CT scan can detect some problems at an earlier stage than x-rays.
  - can make pictures of areas protected or surrounded by bone
- Application of CT
  - Lung examinations for obstructions and tumors
  - Heart disease and many stages of cardiac illness
  - Detection of internal organ diseases.

Introduction (contd.)

- 2 types of CT image reconstruction methods
  - Analytic
    - Filtering (Convolution)
    - Backprojection
  - Iterative
    - Starts with an assumption of the image or solution
    - Iteratively update or corrects according to the measured projection data
    - ART (algebraic reconstruction technique) algorithm developed in 1970 by Richard Gordon
- Efforts to shorten the reconstruction time
  - algorithmic improvement
    - tries to reduce iterations in iterative method
    - FIR (filter iteration reconstruction)
      - extract high frequency component
    - Fast backprojection algorithm
      - developed both for analytic and iterative method
      - example: STRETCH algorithm , approximates interpolation by using pre-computed look up tables
  - Dedicated hardware
    - dedicated backprojectors to speed up backprojection

Introduction (Contd.)

- Problem :
  - For huge computation, speed up is limited for above 2 methods
- Solution :
  - parallel processing (multiple processing elements)
- Parallel implementation of 3-D CT image reconstruction
  - on hypercube multiprocessor iPSC/2
    - Commercially available
    - One controller, 32 PE's, hypercube topology
    - Feasible and usable in future

Convolution and Backprojection

- Image reconstruction is partitioned into
  - Convolution (Filtering)
  - Backprojection
- For this, 2 stage pipelining is adopted for each data set
- Filter kernel of 3-D CT system is symmetric to horizontal and vertical axes in the projection domain
- a conventional sequential convolution algorithm is parallelized to
  - exploit the symmetry of filter kernel
  - each processing element (PE) deals with every Nth projection data (N = # of PE)

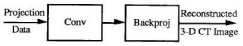


Fig. 2.1. Convolution backprojection scheme

Convolution and Backprojection (cont'd)

- Backprojection is faster than convolution
- Conventional backprojection perform on pixel by pixel basis
- Recently modified backprojection performs on a beam-by-beam basis
- Coarse-grain parallelism
  - as limited number of processors
- Each filtered projection data is partitioned into a number of sub-domains equal to the number of PE's in backprojection
- Each PE backprojects the sub-domain of projected data and stores it in a 3-D array of images
- Each PE will generate partially reconstructed 3-D image and integrate these to the PE_0, finally sends to the controller

Modified 2-D Convolution Algorithm

- 2-D convolution is performed in spatial domain
- modified to reduce computational redundancy
- Adopt the concept of generalized true three-dimensional reconstruction (GTTR) algorithm
  - GTTR uses parallel beam reconstruction technique
- our implementation uses filter kernel for the GTTR algorithm in the spatial domain

Modified 2-D Convolution Algorithm (cont'd)

- Filter kernel

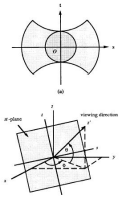
$$h_q(s,t) = S(s,t)\, h_p(s,t)$$

and

$$h_p(s,t) = \frac{-1}{(s^2+t^2)^{3/2}}$$

$$\text{where } S(s,t) = \begin{cases} 1 & \text{for } |\tan^{-1}\left(\frac{t}{s}\right)| \le g_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$$g_{\max} = \cos^{-1}\left(\frac{\cos\eta_0}{\cos\eta}\right) \text{ where, } g = \frac{\text{sequential convolution time}}{\text{sequential backprojection time}}$$



- Equation for convolution algorithm

$$g(m,n) = \sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} h_q(m-i, n-j)\; p(i,j) \quad -n/2 \le m,n \le n/2$$

Modified 2-D Convolution Algorithm (Contd.)

- Observations
  - filter kernel $h_?$ is symmetric to *s*-axis and *t*-axis
  - $g(m, n)$, $g(2i - m, n)$, $g(m, 2j - n)$ and $g(2i - m, 2j - n)$ include the same product term $h_?(m - i, n - j)\, p(i, j)$ and form a group
  - idea of modified convolution is to utilize each product term as much as possible
  - in our implementation, if a product term is computed for one convolved projection data $g(m, n)$, it can be used to update 3 other members of the group
- Shortcomings
  - check each time, whether a product term for a member of a group already exists or computed

Modified 2-D Convolution Algorithm (Contd.)

- A scheme used to overcome
  - is to utilize geometrical symmetry in the projection circle
  - $g(m, n)$, $g(n, m)$, $g(-n, m)$, $g(-n, -m)$, $g(-m, n)$, $g(-m, -n)$, $g(m, -n)$, $g(n, -m)$ all correspond to the same group
  - *g* checking
    - a product term $h_?(m - i, n - j)\, p(i, j)$ is computed only when both of $g(m, n)$ and $p(i, j)$ are inside the projection circle
  - *k* –checking
    - a member in a *k*-group is updated by the associated product term, when the member is inside the projection circle
  - advantage
    - Information derived from *g*-checking and *k*-checking can be shared by all members in a *k*-group

Modified 2-D Convolution Algorithm (Contd.)

Algorithm MC2

```
Following the processing sequence do
  let center of filter kernel be located at (m,n)
  IF g(m,n) is inside the projection circle DO
  let h_q(m-i,n-j)p(i,j) be the product term to be computed
    IF p(i,j) is inside the projection circle DO
    compute this product term and corresponding product
      terms for the members in the g-group associated
      with g(m,n)
    update g(m,n) and all other members in the same g-
      group using corresponding product terms
    ENDIF
    FOR all three other members in the k-group
      associated with g(m,n) DO
    consider one of these three member, say g(k,l)
      IF g(k,l) is inside the projection circle DO
      update g(k,l) and all other members in the same
        g-group associated with g(k,l) by the
        corresponding product terms.
      ENDIF
    END FOR
  ENDIF
END
```

Incremental Backprojection Algorithm

- Fast CT image system
  - needs to reduce backprojection time
- Incremental backprojection
  - reduces multiplications & sine cosine computations
  - to reduce computational redundancy, a set of pixels are enclosed in a beam
  - once the value of a pixel in a beam is computed, other pixels in the same beam can be computed successively by addition only with some beam constant
- Beam constant determination
  - by the value of 2 rays enclosing the beam and
  - by positional relation of two adjacent pixels in the beam

Parallelization

- 3 types of parallelism
  - View parallelism
    - highest level
    - allows independent processing of different views
    - no interactions among views
  - Data parallelism
    - second level
    - in modified 2-D convolution, independent $g(m, n)$ are processed simultaneously
    - beams can be backprojected at the same time
  - Functional parallelism
    - convolution and backprojection executed simultaneously
    - 2 stage pipelining
    - while convolution for one view, backprojection for another view are performed

Parallelization (cont'd)

- Data parallelism is exploited by multiple PE's
- Number of PE determination
  - compute g
  - if $k < g < k+1$, then k PE's are assigned to convolution and 1 PE is assigned to backprojection

Parallelization (cont'd)

- Task Partitioning in Convolution
  - data domain are partitioned according to convolution processing sequence
  - sequence starts from centre to the boundary
  - only $g(m, n)$'s in first octant are in the processing sequence
  - all members of a group associated with a $g(m, n)$ are processed together
- Allocation of group elements to a processor
  - n_c = # of processing elements in convolution
  - $(p_1, p_2, \ldots\ldots p_t)$ = processing sequence
    where, $p_j = j$ th $g(m, n)$ in the sequence
    t = total no. of $g(m, n)$ in the 1st octant
  - $\{p(i)\} = \{p_j \mid j = n_c k + i, k = 0, 1, 2\ldots\ldots\}$
  - PE $[i]$ is responsible for every n_c*th* $g(m, n)$ in the processing sequence

Parallelization (cont'd)

- task partitioning in backprojection stage
  - when *g* is large, maximum 4 PE of iPSC/2 is assigned to backprojection
  - when # of PE = 2
    - the object is divided into four spaces by sz' and ts`plane
    - four spaces form two volume
    - each volume is assigned to each PE
    - one volume consists of two spaces containing beam in the first (+*s*, +*t*) and in the third quadrant (-*s*, -*t*)
    - other volume consists of the other two spaces

Parallelization (cont'd)

- Communication Patten
  - Broadcasting
    - a projection data set and a filter kernel associated with a particular ?is sent to PE_0 from cubemanager
    - PE_0 broadcasts to all other PE's
  - Integration
    - the convolved projection data is integrated to a PE called gate node
    - this integrated data is again broadcasted
    - the partially reconstructed target images are integrated into a PE called return node
    - the final reconstructed target image is sent back to cubemanager from return node
  - Pseudo binary tree structure for communication
    - each PE represents more nodes
    - provides high utilization of PE
    - requires more communication

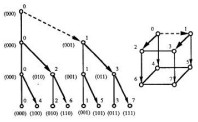


Fig. 3.1. A 4-level pseudo binary embedded in a 3-cube

Result

Table 4.1. Overall implementation results before load balancing

N	n_c/n_b	Ta	Sp	Ef	Um	Us
31	5/1	6	5.13	85.51%	98.77%	0.85%
31	6/1	7	6.16	87.95%	98.41%	0.93%
31	7/1	8	6.76	84.46%	98.33%	5.20%
31	12/2	14	11.49	81.45%	92.85%	5.54%
31	13/2	15	11.53	76.87%	89.80%	7.93%
31	14/2	16	12.21	76.29%	89.55%	8.14%
31	26/4	30	18.94	63.12%	83.29%	12.25%
31	27/4	31	19.66	63.42%	82.85%	11.03%
31	28/4	32	19.93	62.39%	79.53%	12.86%
63	14/1	15	13.33	88.86%	96.84%	2.61%
63	15/1	16	14.26	89.09%	97.30%	1.93%
63	16/1	17	15.25	89.73%	96.10%	3.84%
63	28/2	30	24.29	80.97%	92.56%	4.60%
63	29/2	31	25.02	80.71%	92.55%	5.68%
63	30/2	32	26.03	81.35%	92.69%	4.35%
95	23/1	24	21.20	88.35%	95.40%	2.37%
95	24/1	25	22.39	89.56%	96.56%	2.55%
95	25/1	26	23.06	88.70%	97.56%	2.36%

- Observations
  - For larger image size larger speed up and efficiency
  - For image size 31, the speed up for $T_n = 32$ is lower than that of $T_n = 31$

N : image size, i.e., size of N x N x N,
n_c : number of convolution PE's,
n_b : number of backprojection PE's,
Ta : $t_c + t_{bc}$,
Sp : speed-up, defined as the ratio of sequential processing time to parallel processing time,
Ef : efficiency, defined as Sp/Ta,
Um : mean value of utilization for all convolution PE's, where utilization for a convolution PE is defined as the ratio of the net computation time to the total processing time of the PE,
Us : standard deviation of utilization for all convolution PE's.

Result (Cont'd)

Table 4.2. Load distribution for N=31 and $(n_c, n_b) = (28, 4)$ before load balancing

node	1st brdcst	brdcst	integrate	I/O	compute	node time	utilize
0	104	711	43	33	6718	7257	92.57%
1	104	26	329	0	7940	8303	94.42%
2	104	302	41	0	8647	9098	95.06%
3	104	26	1449	0	7542	9125	82.65%
4	104	305	42	0	7231	7686	94.08%
5	104	25	329	0	9676	10138	93.44%
6	104	349	42	0	8593	9096	94.49%
7	104	26	2056	0	7975	10165	78.46%
8	104	300	40	0	9510	9959	95.49%
9	104	29	1173	0	8664	9973	86.87%
10	104	2767	43	0	6925	9861	70.23%
11	104	23	1424	0	8430	9987	84.41%
12	104	1626	44	0	8118	9935	82.04%
13	104	561	329	0	8951	9948	89.98%
14	104	2281	44	0	7431	9864	75.33%
15	104	27	2952	0	7092	10178	69.68%
16	104	304	40	0	8528	8981	94.96%
17	104	23	982	0	7841	8995	87.62%
18	104	2054	43	0	6810	9015	75.54%
19	104	845	2436	0	5654	9063	62.52%
20	104	3861	41	0	4801	8812	54.48%
21	104	2818	331	0	5607	8862	63.27%
22	104	2300	42	0	6555	9004	72.80%
23	104	539	1311	0	7109	9068	78.40%
24	101	3797	41	0	6803	9857	69.01%
25	101	2496	1870	0	5380	9852	54.61%
26	101	3032	40	0	6626	9825	67.44%
27	101	2199	1279	0	6282	9865	63.68%
31	614	1832	43	0	8120	10613	76.51%
30	614	1287	43	0	8691	10638	81.70%
29	614	1871	596	0	8080	11165	72.37%
28	410	1349	1091	45	8438	11337	74.43%

- 1st brdcst: time for each PE to acquire data after synchronization
- brdcst : total broadcasting time except the first data
- integrate : total integration time
- node time: total processing time for a PE
- compute : net time for computation
- utilize : compute/node time
- Observations
  - computational load not well balanced
  - for some PE_5 load is as high as 9676
  - for some PE_{20} load is as low as 4801
  - load distribution in backprojection is relatively balanced

Results (Cont'd)

Table 4.3. Overall implementation results after load balancing

N	n_g/n_b	Tn	Sp	Ef	Um	Us
31	5/1	6	5.38	89.59%	98.83%	0.37%
31	6/1	7	6.42	91.68%	98.34%	0.32%
31	7/1	8	7.26	90.79%	98.15%	0.69%
31	12/2	14	11.57	82.61%	96.32%	1.65%
31	13/2	15	12.52	83.45%	95.04%	2.11%
31	14/2	16	13.10	81.89%	93.33%	1.60%
31	26/4	30	20.59	68.63%	92.73%	2.25%
31	27/4	31	20.85	67.27%	91.80%	2.80%
31	28/4	32	21.15	66.08%	90.43%	3.42%
63	14/1	15	13.85	92.35%	98.57%	0.56%
63	15/1	16	14.78	92.35%	98.41%	0.62%
63	16/1	17	15.53	91.33%	98.48%	0.57%
63	28/2	30	25.57	85.24%	96.35%	0.99%
63	29/2	31	26.60	85.80%	96.92%	1.21%
63	30/2	32	27.04	84.49%	96.02%	1.40%
95	23/1	24	22.28	92.82%	98.51%	0.34%
95	24/1	25	23.06	92.24%	98.86%	0.38%
95	25/1	26	23.96	92.16%	98.95%	0.31%

- Load balancing scheme
  - N_c load queues are created
  - Each queue keeps a list of *g*-groups to be processed by PE
  - Initially queues are empty
  - A *g*-group with heaviest load is assigned to the queue with the least total load and delete the *g* group from sorted processing sequence
- Observations:
  - all performance parameters E_f, S_p, U_m, Us have been improved
  - For image size 31, the speed up for $T_n = 32$ is higher than that for $T_n = 31$

Results (cont'd)

Table 4.5. Load distribution for N=31 and $(n_c, n_b) = (28, 4)$ after load balancing

node	in hedcm	hedcm	integrate	I/O	compute	node time	utilize
0	104	758	42	33	7300	7878	92.66%
1	104	101	604	0	7079	7892	89.70%
2	104	731	43	0	6983	7864	88.80%
3	104	27	796	0	6974	7906	88.21%
4	105	853	42	0	6855	7859	87.22%
5	105	550	428	0	6786	7873	86.19%
6	104	1011	43	0	6690	7851	85.21%
7	105	26	1080	0	6758	7921	85.32%
8	104	989	43	0	6713	7852	85.49%
9	105	689	441	0	6629	7862	84.30%
10	104	367	41	0	7518	7933	94.74%
11	104	26	618	0	7392	8143	90.78%
12	104	502	42	0	7187	7941	90.51%
13	104	26	626	0	7195	7955	90.45%
14	104	145	50	0	7692	7995	96.21%
15	105	28	1177	0	7856	9170	85.67%
16	105	328	43	0	7783	8263	94.19%
17	105	31	429	0	7708	8278	93.11%
18	105	324	42	0	7793	8269	94.26%
19	104	29	630	0	7703	8469	90.96%
20	105	658	43	0	7648	8257	92.62%
21	104	126	330	0	7710	8274	93.18%
22	104	649	43	0	7453	8253	90.31%
23	105	32	905	0	7759	8782	88.12%
24	101	398	41	0	7711	8256	93.40%
25	102	102	502	0	7561	8271	91.42%
26	102	450	44	0	7669	8259	92.74%
27	102	27	614	0	7811	8558	91.27%
31	535	305	43	0	8117	9604	84.52%
30	535	364	43	0	8682	9629	90.17%
29	535	944	597	0	8076	10153	79.53%
28	494	607	1091	45	8451	10668	79.20%

- Observations
  - Computational load is well balanced
  - Computation time for PE_5 = 6786 and PE_{20} = 7648
  - Here lowest utilization is 84.26% whereas before load balancing it was 62.52%

Conclusion

- parallelization of CT image reconstruction in message passing multiprocessors are described
- simple task partitioning scheme for convolution stage works well both for small and large image size
- it is seen that parallelization of 3-D CT image achieves high speed up close to the number of PE's
- the computing system iPSC/2 hypercube is not appropriate for this task
- dedicated computing system should be designed for 3-D image reconstruction

Thanks

?