TIMING ANALYSIS FOR THE CDOT-PPS

Sanjay Bhatnagar

This report presents some timing analysis for the CDOT-PPS, based on the rough timing estimates that have become available for the various data paths and the MFLOPs at the Master Controller (MC) and the Processing Elements (PEs).

There are some parameters by which a parallel or concurrent machine can be evaluated and the algorithms designed for that. The communication links which connect the various nodes of a parallel machine, are the weekest part in parallel computing. Therefore, inter nodal communications must be minimized. How much of this an algorithm is able to achieve is reflected in the efficiency which is

$$\eta = \frac{t_s}{t_p * NPE}$$

where t_s is the time taken by a sequential machine to do the job and t_p is the time taken by a parallel machine (total time) with NPE nodes.

Also, only those algorithms are parallelisable on a concurrent machine (as against parallel machine) which have a good compute to communication ratio. The communications required in turn depend partly on the problem and partly on the architecture of the machine.

And lastly, and also the most useful figure from evaluation point of view is the Effective number of MFLOPS of the machine. This is the ratio of the number of operations required by a sequential machine to the total time taken by a parallel machine to do the same job. This figure can then be directly compared with the MFLOPS of a sequential machine to set the speed-up achieved. This again is problem dependent and should be worked out separately. Here are some estimations of the efficiency and the Effective MFLOPS for the CDOT-PPS for the various algorithms used in Radio Astronomy mapping.

The C-DOT PPS

There are three data paths involved in the machine which can be used by an application. Each have been treated separately here. The three paths and the variables used in the analysis are:

t_{MCMD} : time taken for data from MC to MDAM in sec./byte
t_{MDPE} : time taken for data from MDAM to PEs in sec./byte
t_{MCPE} : time taken for data from MC to PEs in sec./byte

There will be some computation done on the MC and some on the PEs. All the application computation done at the MC, will constitute the sequential part of the algorithm. The following variables are used for this:

$MFLOP_{mc}$: MFLOPs at the MC
$MFLOP_{pe}$: MFLOPs at the PEs

The programming paradigm here is generally that of data splitting (Single Algorithm and Multiple Data). The performance of algorithms in such a case depends largely on the amount of communication and the link speeds available. Since the data is split among all the available PEs, the performance is dependent on the number of PEs as well as the data size.

NPE : Number of available PEs

Using these variables for the CDOT-PPS, communication and computation times are estimated for the various algorithms used in Radio Astronomy Image Processing and an attempt is made to come out with a clear cut scheme for implementing these.

Map making in Radio Astronomy

Map making in RA involves basically 4 operations. A raw map, called the Dirty Map is made by performing a 2D Fourier transform of the raw data. For reasons of speed, the FFT algorithm is used, which requires the data to be resampled over a regular grid, called Gridding here. To remove the artifacts of the instrument and noise from the map, two iterative algorithms are used namely CLEAN and SELFCAL.

For an 10 hr. observation, with 5 sec. integration and one polarization, the raw data would be roughly 64 Mbytes. However in the worst case it will be resampled into a grid of 1024x1024 pixels of complex data.

The CLEAN algorithm

It has been shown in an earlier report (Kulkarni and G.Subramanian,1988) that the Clark CLEAN gives no significant advantage over the Hogbom CLEAN on this machine. This is because the Clark CLEAN was designed to use the computing power of an array processor by using the FFT for beam removal (de-convolution). However in our case repeated use of 2D FFT will incur communication overheads enough to account for the advantage given by the use of FFT. On the other hand, the Hogbom CLEAN wins in the complexity analysis.

Apart from the above variables defined, the performance of the CLEAN algorithm will depend on

- the size of the map and beam
- the number of CLEAN components extracted

The variables used for these are

N^2 : for a map of size N by N
C : for the number of CLEAN components extracted
NBY : number of bytes per pixel

The Hogbom CLEAN operates in the map domain and therefore, both the map and the beam will be floating point real numbers, each of which will occupy NBY bytes.

Data size

Map of size NxN in bytes = $NBY*N^2$ bytes

Beam size for CLEANing a NxN map in bytes = $\frac{NBY*N^2}{4}$ bytes

The entire map is split between the various PEs and is transfered to the PE memory via the MDAM. The Dirty Beam will be loaded in the PE memory via the serial PE-MC link. Between extraction of each CLEAN component, there will be NBY bytes comming from each PE via the MD-PE & MD-MC link. This will mean the MC to receive NBY*NPE bytes per CLEAN cycle.

$$\text{Total communication time} = NBY*\left(\frac{N^2+2*NPE*C}{t_{MCMD}}+\frac{N^2/4+C}{t_{MCPE}}+\frac{C+N^2/NPE}{t_{MDPE}}\right)$$

Computations

In each CLEAN cycle,

- each PE will search for the local peak in portion of map ($\frac{N^2}{NPE} - 1$ comparisions)
- the MC will search for the global maximum on the NPE real numbers (NPE-1 comparisions)
- Beam scaling at each PE will involve $\frac{N^2}{4*NPE}$ multiplications
- Beam removal will involve $\frac{N^2}{NPE}$ subtractions at each PE.

The last will not be strictly true but since the architecture allows no load balancing, it will amount to all PEs doing that many operations. All the PEs have to wait till the last PE finish the job.

Therefore

$$\text{Total time for computations} = \frac{NPE-1}{MFLOP_{mc}} + (2*N^2 + \frac{N^2}{4} - 1)*\frac{C}{MFLOP_{pe}*NPE}$$

The MC-PE link and MD-PE link operate at 1MByte/s. This is the raw rate. At the protocol level, let us assume that the link operate at 70% of the peak. Also let the MFLOPS at MC are 0.2 (measured) and at the PEs be 0.8 (measured).

With these figures we get for 10000 CLEAN cycles

NPE	η	t_{CPU}	t_{COM}	$MFLOPS_{eff}$	t_{CPU}/t_{COM}
64	0.65	463.31	3.92	33.66	118.15

One can clearly see that the compute to communication ratio is extremely high. Any further increase in the performance is possible only with an improvement in the $MFLOP_{pe}$.

The above analysis is done assuming only the inner quarter of the map is CLEANed. In the case where the entire map is CLEANed, the computations are going to go up further.

The 2D FFT algorithm

The 2D FFT is implemented by performing a 1D FFT of size N on N/NPE of the rows and columns of a NxN grid in parallel. The entire grid is split among the PEs, first by rows and then by columns. Between the 1D FFTs on rows and columns, there is a transpose of the grid involved. This is a costly operation with the compute to communication ratio being poor. On a machine like this where data is split among various processors, a transpose operations is almost all communication and no computation.

For a NxN grid, there will be 2*N/NPE 1D FFT of size N. A 1D FFT of size N involves $N \log N$ operations. Therefore

$$\text{Total computations} = \frac{2*N^2}{NPE} * \log N \text{ operations}$$

The grid has to be first loaded in the global memory (MDAM). This means N^2*NBY bytes on the

MC-MD link

$$\text{Time taken on MC-MD link} = N^2 * NBY * t_{MCMD}$$

And

$$\text{Time taken on MD-PE link} = N^2 * NBY * t_{MCPE} * NPE$$

The transpose of the matrix was initially envisaged to be achieved by the use of Batchers algorithms where the global memory could be accessed either as rows or as columns. In such a case, a transpose can be realised by writting data as rows in the MDAM and reading as columns. This, in principle is still possible in this machine. But the overheads are unacceptable. First for a NxN grid, there will be $\frac{2*N^2}{NPE}$ transfers on the slow serial link and on each transfer, a new switch setting will be required. Secondly, the data is scrambled arcoss al banks in such a way so that all the elements of a given row (or column) sit in different banks of the memory so as to allow parallel access. The addresses of the elements then have to be calculated for each element, and that will be done at the MC which will mean further overheads.

Another less expensive method is to read the data into PEs as rows. If a transpose is attempted at the PEs themselves, all the PEs will transfer data to most others and will also receive from most others. This however is possible and for a NxN matrix there will be only $\frac{N^2}{NPE}$ transfer on the serial link (a factor of 2 less). This also will require NPE switch settings. This is however, not difficult as the switch can at any given reset, hold 256 different switching patterns and can be switched to a new pattern fast. Since each of the PEs have to receive data from all other PEs as well as it has to send data to all others, these two transfers must go on simultaneously. This is possible as each of the PEs have two communication links, both operating at 1Mbytes per second.

Thus

$$\text{Communications for transpose} = \frac{N^2}{NPE}$$

For the above mentioned parameters for the PPS, following are the figures for a 2D FFT algorithm

NPE	η	t_{CPU}	t_{COM}	$MFLOPS_{eff}$	t_{CPU}/t_{COM}
64	0.64	2.45	1.33	33.22	1.84

The efficiency of the algorithms is fine but compute to communication ratio is adverse. This ratio is greater than one and the through put of the algorithm will improve if there is a improvement in the $MFLOP_{pe}$ relative to the link speed.

Gridding

Gridding involves, say a 3x3 convolution with the raw data. This by far is the most time consuming operation in mapping. The raw data is however is too large to be held in the PEs (64MB). If the raw data is split among the PEs, it will require to be sorted first with u and v. This is because, to reduce the inter PE communication (which also are on the slow serial links), all the V(u,v) data corresponding to the portion of the grid at a PE must be at that PE only. If this partioning of raw data is done, then with 64 PEs or more, gridding can be parallelised.

The NxN grid will again be distributed among the PEs and the so will be the raw data. Explicit

convolution will then be done in parallel at all the PEs. The minimum raw data for this out of 10 hr. observation and 5 sec. integration will be roughly 64 Mbytes. This when divided among, say 64 PEs will require roughly 1 Mbytes of RAM at the PEs. Another 200 KBytes will be required to hold the complex grid and and integer grid for uniform weighting of the same size as the map.

This will still be a bad operation as the communication time will be high. First all the ~ 100 MB of data will travel on the MC-MD link. Then 100MB /NPE data will go on the slow MD-PE link. Therefore

$$\text{Communication time} = \frac{T}{t_{MCMD}} + \frac{T}{NPE * t_{MDPE}}$$

where

T = Total raw data

while the computations will be

$$\text{Computation time} = \frac{1}{NPE} * \left(\frac{3 * N^2 * m^2 * w}{MFLOP_{pe}} + \frac{N^2 * 2}{MFLOP_{pe}}\right)$$

where

$$w = 3.23x10^{6} * \frac{m^2}{N^2}$$

mxm = size of the gridding convolution function

The number of computations required for convolution is estimated as the ratio of the grid area to the area of the convolving function multiplied by the total number of visibility samples. This is the number which will be true if the visibility data was uniformly distributed in the uv-plane.

Again for the same parameters for the PPS we get the following figures

NPE	η	t_{CPU}	t_{COM}	$MFLOPS_{eff}$	t_{CPU}/t_{COM}
64	0.31	15.37	33.57	16.08	0.46

The efficiency of the algorithm is again poor as was expected because of the huge communications in loading the data in the PE memory. The compute to communication ratio is again less than 1, requiring improvement in the link speeds.

In the above calculations, it is assumed that no load balancing is possible. This puts restrictions on the best results achiveable. The efficiency of the algorithm will depend on the time taken by the PE which does the maximum work (all the PEs operate synchronously). In the case of gridding however, some load balancing can be achieved. This is by recognizing the fact that the number of visibility data towards the edge of uv-field is lesser than towards the origin. Therefore, if the uv-data is divided according to the density of the uv-data, significant load balancing and thus improvement in time can be achieved.

The various proposed schemes for mapping

Three schemes have been suggested and analysed in the Kulkarni and G.Subs. report.

I. All PEs work on different maps

II. All PEs work on the same map
III. Group of PEs work on one map

I. In the calculations above, this case corresponds to the case with NPE=1. The entire PPS is used as a collection of NPE computers working in unison but without interaction on completely different data sets. The time for the entire operation therefore will again be decided by the PE which completes the last.

Data size

4 MB of map
4 MB of beam
8 MB of gridded raw data (complex data required by SELFCAL)

Therefore 16 MB of RAM required at each of the PEs. The total RAM at PEs available now is 4 MB! For small map (as small as 256x256, the total data requirement will be 4 MB!

This is only for the data. Since some RAM is occupied by the local OS a large chuck by the PE application program, the memory requirement will be much more than this for this scheme.

If the large chunk for the gridded visibility data is dropped, then the SELFCAL has to run either on the MC or the data has to be redistributed among the PE for all the map in process. This is equivalent of running massive sequential code and huge communication overheads, both a curse for parallelism.

Above all, use of gridded data in SELFCAL will not be acceptable most often even for normal dynamic range mapping.

II. This is the scheme which has been discussed in detail above. The effective MFLOPS that can be extracted are listed in Table 1. One can clearly see two things

- CLEAN is computationally the most expensive algorithm
- the computation to communication ratio is high

The first observation indicates that the time taken by the entire mapping process is by far determined by the time taken by CLEAN. The compute to communication ratio being high, points to the fact that increasing the computation power at the PEs is going to help greatly.

III. This scheme is similar to the second scheme, but at a coarser level. Here the governing factor will again be the memory available at the PEs and will be possible on after a threashold of PEs are available.

It needs to be pointed here that in the previous analysis of this scheme, it was assumed that the computation and communication at the PEs can be overlapped. This however is NOT possible.

Each of the PEs have a X.25 protocol communication handler, a 68010 which acts as the controller and a T800 Transputer which does the computing. The communication handler is controlled by the 68010 and talks with it via a shared memory. The data (application related data) sits in the 4 MB memory which is addressable by the T800 alone. For communication (to MC or MDAM) the T800 transfers the data from its memory to the memory of the 68010 from where the XPC actually transfers. Since the whole process involves the T800, the above overlap is not possible. This is possible only if the T800's memory is addressable by the 68010 also.

Conclusions

1. As indicated above CLEAN is the most time consuming operation in case of PPS. Gridding which is generally the costlier operation on sequential machines scales well on PPS because of no communications involved once the data is in the PE memory.

2. In the implementation discussed above for CLEAN, the percentage of effective sequential processing done is considerably reduced, though at the price of increasing the complexity of the algorithm and increase in the number of communications. The later is however, well compensated for by the increase in the over all efficiency of the algorithm.

3. The maximum memory required for gridding on raw data for 10 hr. observation with 5 sec. integration is roughly 2.5 Mbytes at each PE. The total memory available at each PE is 4 Mbyte. The total memory required by any other application will be less than this. Therefore, the sequence of Gridding, 2D FFT, CLEAN, 2D Inverse FFT and Ungridding can be done with the data being loaded once for gridding. This will further decrease the communication time in CLEAN and increase the overall efficiency.

4. If the PEs, which are the T800 processor at the moment, are replaced by the Intel's i860 CPU, we get the following figures for CLEAN. The i860 is rated for 80 MFLOPS peak and we take 20% of the peak value i.e. 20 MFLOPS.

NPE	η	t_{CPU}	t_{COM}	$MFLOPS_{eff}$	t_{CPU}/t_{COM}
16	0.65	295.51	3.03	52.86	97.47

It needs to be mentioned that the measurements done on a i860 machine with Multi Bus II from WIPRO for the MFLOPS and the bus transfer rates gave 5 MFLOPS for the processor and 5 Mbytes/sec transfer rate on the bus. The low bus rate at least was because of the bad memory utilization by the locally developed system software and is expected to improve upon optimization.

The improvement, relative to T800 PEs is significant in the through put. This however will only be in case of CLEAN as the compute to communication ratio in other applications require improvement in the link speeds. But since CLEAN becomes the deciding factor, overall improvement will be significant. Secondly, in absolute terms, there will be an improvement in the through put because of the use of sheer faster CPU.