Local Predecimation with Range Index Communication Parallelization Strategy for Fractal Image Compression on a Cluster of Workstations

Syed Hussain[1], Kalim Qureshi[2], Mohammad Al-Mullah[2], and Haroon Rashid[1]
[1]Department of Computer Science, Comsats Institute of IT, Pakistan
[2]Faculty of Math and Computer Science, Kuwait University, Kuwait

Abstract: *In this paper, we have implemented and evaluated the performance of local predecimation with range index communication parallelization strategy for fractal image compression on a beowulf cluster of workstations. The strategy effectively balances the load among workstations. We have evaluated the execution time of LPRI, varying the number of workstations and user-specified root mean square error. We have also reported the measured speedup and worker idle time of LPRI parallelization.*

Keywords: *Load balancing, task partitioning, parallelization, fractal image compression.*

Received October 25, 2007; accepted March 5, 2008

1. Introduction

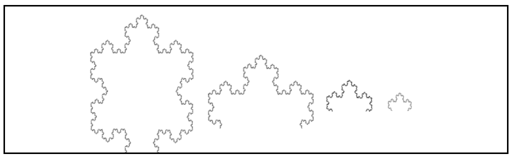
"A picture is worth hundred lines". Multimedia data, especially images contain much more information than traditional text files. With the emergence of novel applications such as telemedicine, teleconferencing and remote sensing, the need for efficient storage and fast communication of images is multifold. Image compression is an answer to this ever-increasing need. Images have much more room for compression because of two reasons namely; massive redundancy that they contain and susceptibility of human vision system to interpret images with minor data.

Fractal Image Compression (FIC) [1, 2, 3, 7, and 10] is a lossy image encoding scheme, which uses the self-similarity existing in images to compress them. Fractal is a geometrical shape which contains self similarity; more formally, "a set for which the Hausdorf dimension strictly exceeds the topological dimension" [3]. Figure 1(a) shows a snowflake image, which contains self similarity. A fractal is a geometric shape that is divisible in parts, such that each part is a reduced copy of the original image. This property provides immense opportunity of reduction in data.

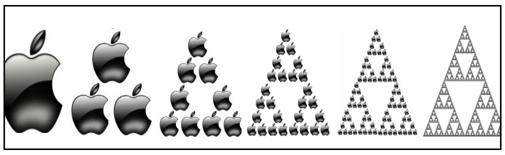
Strictly speaking, a set of contractive transforms is repeatedly applied to any image to obtain the unique attractor formally known as Fixed Point Theorem [3]. Mathematically, this transform represented in equation 1 [6].

$$w_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \quad (1)$$

where a_i, b_i, c_i, and d_i represent scaling, skew and rotation while e_i and f_i represent the translation of the image.



(a) Snow flake fractal.



(b) The fractal transform.

Figure 1. An example of fractal images.

As shown in Figure 1(b), the transform does three things; it scales the image to half, generates three copies of it and translates them in a triangle. Applying this transform on any input results the Sierpinsky triangle [3]. A transform is contractive, if it brings the points of input image closer after any iteration [6].

FIC is an asymmetric compression technique, with encoding phase taking much more time then decoding phase. To reduce the encoding time, research has leapt forward in two directions; classification techniques [4] and use of high performance distributed computing. Given the independent ranges to be searched, the FIC is a suitable candidate as a data parallel application for parallelization [6, 9]. Historically there have been efforts to parallelize FIC [5, 6, 9], but none of them focused on implementation of parallel FIC on low-cost multi-computer cluster of workstations.

We have implemented Local Predecimation with Range Index (LPRI) communication parallelization strategy for encoding phase of FIC on homogeneous cluster of workstations running Linux operating system with Message Passing Interface (MPI) [8]. We have evaluated the performance of LPRI parallelization strategy measuring execution time, speedup, and worker idle time cost.

2. Adaptive Quadtree Partitioning Algorithm

2.1. Serial Algorithm

Grey-scale images are three dimensional matrices; two spatial dimensions and one dimension to represent the grey level intensity at each pixel specified by a pair of spatial coordinates. For these images, domain pixel intensities must also be transformed making the transform equation 1 becomes [6]:

$$w_i = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{2}$$

where o_i controls brightness and s_i controls contrast of coded image.

The input image is partitioned into non-overlapping range blocks by iteratively splitting an image into its four quadrants. These partitions are stored as ranges with depth 1 in the range pool. Now each depth n range from range pool is further divided into four depth n+1 ranges. A range is not further split when its depth reaches minimum-partition threshold.

Now ranges are taken from range pool, to match it a suitable domain is searched within the entire image. The size of domain must be double the size of range to ensure that the transform is contractive. The domains may be overlapping and are selected on a k-distance lattice [3]. To map a domain to a range one of eight isometries [5] namely; four rotations, flip and four further rotations are tried. Suitable values of the transform parameters as shown in (equation 2) are also searched for each domain-range pair. Domain-Range mapping is judged by calculating Root Mean Square (RMS) error metric calculated as equation 3 [3] suggests:

$$\text{RMS} = \sqrt{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (r(i,j) - d(i,j))^2} \tag{3}$$

where *r(i,j)* and *d(i,j)* are ranges and domains from the original N×N image. If for a given range the domain with RMS values less than a predefined tolerance factor known as Collage Error (CE) [3], the domain is selected as the best match, the range is removed from the range-pool and the calculated transform parameters are saved. If any domain is found for a range, it is said to be covered. If no domain can be found for a particular range, the range is further split into four quadrants; the depth of quadrants is set to be one more than the depth of original range. These quadrants are then added to the range-pool. The guard condition for stopping range split is maximum-partition threshold.We have parallelized the Adaptive Quadtree Algorithm [3, 5] for FIC, by applying LPRI strategy.

2.2. Local Predecimation with Range Index Communication LPRI Parallelization

MPI task-farm paradigm [8] is used to implement our strategy. In task-farm approach, the master has a pool of tasks (called task-farm) and a pool of workers. In our strategy the master node does the initialization chores, splits the input image into ranges with depth equal to minimum partition. The master nodes then takes range from its range-pool one by one, assigns it to any worker from pool of workers. Worker encodes the range, returns the transform parameters to master and waits in worker pool for next range.

Size of domains in fixed double the size of ranges in each spatial dimension. That is there are four domain pixels for one range pixel, hence the domain pixels need to be averaged before comparison to range pixels. This process is formally called as the predecimation [5]; this process is usually done before any domain-range comparisons so that each domain is predecimated once and only once. In our strategy, the predcimation of domains is done locally for each worker. Pseudocode of Figure 2, shows the detailed functionality of our strategy.

3. Experiment Results and Discussion

We have implemented LPRI on Message Passing Interface [8] task-farm approach using ANSI C programming language.The platform of our experiment was homogenous cluster of workstation with 32 PII-333 nodes having 96MB memory, running Linux 2.4, kernel connected with 100Mbps Fast Ethernet. We have conducted our experiment on 256 × 256 Lena image. Figure 3(a) shows the execution time of LPRI parallelization, changing user-specified RMS values. Lesser the RMS tolerance factor is, the algorithm looks for a closer domain-range match, which consequently increases the execution time. As RMS is increased, looser matches can also be used, and execution time decreases. The curve becomes steep as RMS is increased beyond 10.0 because domain-range matches are more or less same.Speedup [8] a metric used to measure the performance gain of parallel implementation as opposed to serial implementation calculated as:

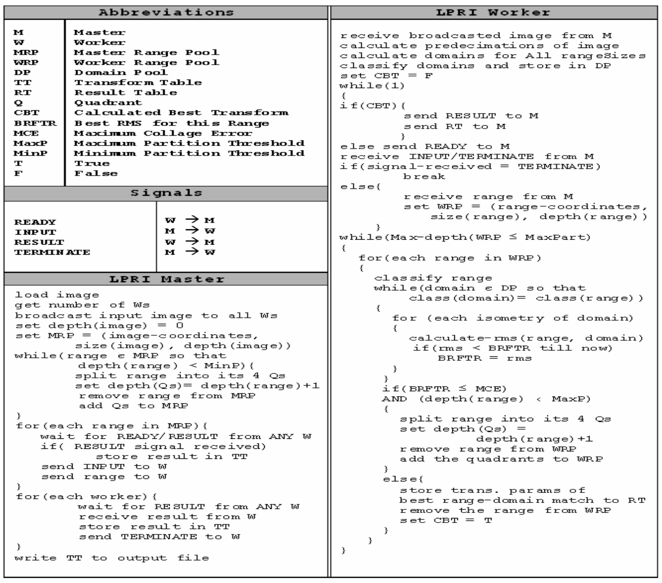
$$speedup = \frac{T_{faste}}{T_{pdp}} \qquad (4)$$

where $T_{fastest}$ is the execution time of serial algorithm on a single fastest machine, while T_{pdp} is the execution time of LPRI parallelization strategy. Figure 3(b) reveals the speedup by increasing the workers. As the workers are increased beyond 12, the speedup curve is steep because number of return and reply communication messages are increased, consequently increasing the communication overhead. Overhead time O_i, which includes communication data access to initiate a new task on worker i is calculated according to equation 5.

O_i = (*Worker's new task starting time - Worker's old task completion time*) (5)

$$Workers\ idle\ time\ cost\ W_i = \sum_j^n O_j \qquad (6)$$

Figure 3(c) shows worker idle time cost which calculated by using equation 6.



Abbreviations	
M	Master
W	Worker
MRP	Master Range Pool
WRP	Worker Range Pool
DP	Domain Pool
TT	Transform Table
RT	Result Table
Q	Quadrant
CBT	Calculated Best Transform
BRFTR	Best RMS for this Range
MCE	Maximum Collage Error
MaxP	Maximum Partition Threshold
MinP	Minimum Partition Threshold
T	True
F	False

Signals	
READY	$W \rightarrow M$
INPUT	$M \rightarrow W$
RESULT	$W \rightarrow M$
TERMINATE	$M \rightarrow W$

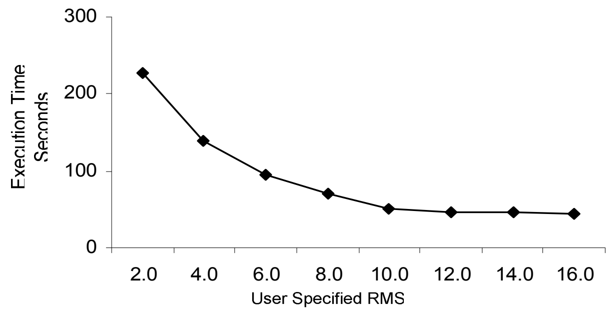
LPRI Master

```
load image
get number of Ws
broadcast input image to all Ws
set depth(image) = 0
set MRP = (image-coordinates,
      size(image), depth(image))
while(range ∈ MRP so that
       depth(range) < MinP){
      split range into its 4 Qs
      set depth(Qs)= depth(range)+1
       remove range from MRP
       add Qs to MRP
}
for(each range in MRP){
   wait for READY/RESULT from ANY W
   if( RESULT signal received)
         store result in TT
   send INPUT to W
   send range to W
}
for(each worker){
       wait for RESULT from ANY W
       receive result from W
       store result in TT
       send TERMINATE to W
}
write TT to output file
```

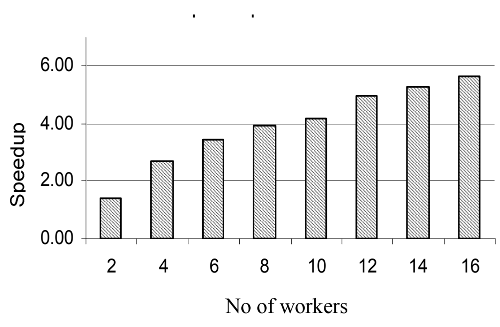
LPRI Worker

```
receive broadcasted image from M
calculate predecimations of image
calculate domains for All rangeSizes
classify domains and store in DP
set CBT = F
while(1)
{
if(CBT){
        send RESULT to M
        send RT to M
       }
else send READY to M
receive INPUT/TERMINATE from M
if(signal-received = TERMINATE)
        break
else{
        receive range from M
        set WRP = (range-coordinates,
           size(range), depth(range))
    }
while(Max-depth(WRP ≤ MaxPart)
{
  for(each range in WRP)
  {
    classify range
    while(domain ∈ DP so that
        class(domain)= class(range))
    {
      for (each isometry of domain)
      {
        calculate-rms(range, domain)
         if(rms < BRFTR till now)
            BRFTR = rms
      }
     }
     if(BRFTR ≤ MCE)
     AND (depth(range) < MaxP)
     {
       split range into its 4 Qs
       set depth(Qs) =
                 depth(range)+1
       remove range from WRP
       add the quadrants to WRP
     }
     else{
       store trans. params of
       best range-domain match to RT
       remove the range from WRP
       set CBT = T
     }
   }
}
```

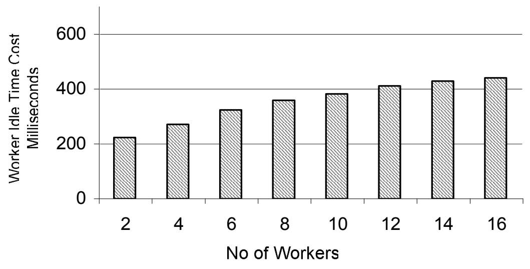
Figure 2. Communication LPRI.



(a) LPRI execution time vs user specified RMS.



(b) LPRI speedup vs no workers.



(c) LPRI worker idle time cost vs no of workers.

Figure 3. LPRI results.

4. Conclusion

We have implemented and evaluated Local Predecimation with Range Index communication LPRI parallelization strategy for fractal image compression on a cluster of workstations. Our results reveal speedup gain up to a multiple of 5 on serial FIC. Worker idle time cost due to communication is on average 300 milliseconds. It can be reduced by applying an enhanced load balancing strategy in future.

Acknowledgment

I would like to thank COMSATS University Campus, Abbottabad, Pakistan for providing the excellent research environment. I am grateful to my MS supervisor Dr. Kalim Qureshi and co-supervisor Dr. Haroon for their guidelines and supports. I would like to thank Dr. Mohammand Al-Mullah for his suggestions.

References

[1] Chang-man X. and Zhao-yang Z., "A Fast Fractal Image Compression Coding Method," *Journal of Shanghai University (English Edition)*, vol. 5, no. 1, pp 57-59, 2002.

[2] Dasgupta D., Hernandez G., and Niño F., "An Evolutionary Algorithm for Fractal Coding of Binary Images," *IEEE Computer Journal of Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 172-181, 2004.

[3] Fisher Y., *Fractal Image Compression: Theory and Application*, Springer-Verlag, NewYork, 2000.

[4] Fisher Y., Jacobs E., and Boss R., "Fractal Image Compression Using Iterated Transforms," Technical Report 1408, NOSC, SanDiego, CA, 2000.

[5] Hämmerle J. and Hl A., "Fractal Image Compression on MIMD Architecture: Basic Algorithms," *Computer Journal Parallel Algorithms and Applications*, vol. 11, no. 34, pp. 187-204, 2004.

[6] Jackson D. and Tinney G., "Performance Analysis of Distributed Implementations of a Fractal Image Compression Algorithm," *Computer Journal of Concurrency Practice and Experience*, vol. 8, no. 5, pp 357-386, 2000.

[7] Ong G., Chew C., and Cao Y., "A Simple Partitioning Approach to Fractal Image Compression," *in Proceedings of ACM Symposium on Applied Computing*, New York, pp. 301-305, 2004.

[8] Wilkinson B. and Michael Allen., *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall, New Jersey, 2006.

[9] Wu P., "Distributed Fractal Image Compression on PVM for Million-Pixel Images," *International Conference on Information Networking*, Japan, pp. 393-398, 2004.

[10] Zhao E. and Liu D., "Fractal Image Compression Methods: A Review," *in Proceedings of Information Technology and Applications*, USA, pp. 756-759, 2006.

[11] Hasan Y., Hassan M., and Ridley M., "Incremental Transitivity Applied to Cluster Retrieval," *in International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 311-319, 2008.

Syed Hussain received his BS in computer science from COMSATS Institute of Information Technology CIIT, Abbottabad, Pakistan. He did his MS in Computer Science from CIIT. Currently, he is perusing his PhD at Carl Duisberg Cnetrun, Dortmund, Germany. His research interests include parallel and distributed computing.

Kalim Qureshi is a member of Math and Computer Science Department in Kuwait University. He is an approved supervisor for MS and PhD thesis by high education commission, Islamabad, Pakistan. His research interests include network parallel distributed computing, thread programming, concurrent algorithms designing, task scheduling, and performance measurement. He is a member of IEE Japan and IEEE Computer Society.

Mohammad Al-Mullah is an associate professor in Math and Computer Science Department, Kuwait University. He did his MS and PhD from McGill University, Canada. His research interests include artificial Intelligence, programming languages, and parallel and concurrent systems.

Haroon Rashid is director and faculty member of the Department of Computer Science at COMSATS Institute of Information Technology, Abbattabad Campus, Pakistan. His research interests include parallel computing, distributed systems, high speed networks, multimedia, and network performance optimization.