7-5-2012

Resilient Image Fusion

Tiranee Achalakul
Syracuse University, tachalak@syr.edu

Joohan Lee
Syracuse University

Stephen Taylor
Syracuse University, steve@scp.syr.edu

Resilient Image Fusion[1]

Tiranee Achalakul
Joohan Lee
Stephen Taylor

2-106 CST building
Syracuse University
Syracuse, NY 13244
{tiranee, jlee, steve}@scp.syr.edu

Abstract

The paper describes a distributed spectral-screening PCT algorithm for fusing hyper-spectral images in remote sensing applications. The algorithm provides intrusion tolerance from information warfare attacks using the notion of computational resiliency. This concept uses replication to achieve fault tolerance, but goes further to dynamically regenerate replication in response to an attack or failure. The concepts of resiliency are incorporated through library technology that is application independent. This library hides the details of communication protocols required to achieve dynamic replication and reconfiguration in distributed applications. The paper provides a status report on our progress in developing the concept and applying it to image fusion. In particular we examine the performance of the PCT algorithm and compare the results with and without resiliency to assess the associated overheads

1. Introduction

Any system that operates in highly adverse environments, such as battlefield command and control, must be able to tolerate attacks and failures. Many distributed systems have sought to use replication as a mechanism to provide this fault-tolerance [Agarwal 1994, Amir 1992, Budhiraja 1993]. Although this approach provides graceful degradation of system performance to the point of failure, it is clearly not sufficient to aggressively recover assured operation.

We are investigating an alternative approach, *computational resiliency*, that combines real-time attack assessment with on-the-fly replication, camouflage, and process reconfiguration to maintain and improve system capabilities. To understand how these concepts might operate, consider a distributed application, as analogous to an apartment complex inhabited by a new strain of roach (process/thread). The roaches are highly resilient: you can stamp on them, spray them, strike them with a broom but you never kill them all or prevent them from their goal of finding food (resources). To foil your eradication efforts, they use several techniques: they are *highly mobile* moving from one place in the apartment complex (network) to another with speed and agility. They continually *replicate* to ensure that it is not possible to kill them all. They *sense* (attack assessment) their environment to obtain clues that mobility is necessary: if a light is turned on, they scurry away in all directions to hide behind cupboards in places of *known safety* (secure network zones). If a new roach killer is invented they *learn* from it, and *adapt* their behavior to compensate. However, this new strain is particularly aggressive and seeks to live in the daylight (wide-area operation): thus it adopts techniques for camouflage as a form of protection and disinformation. [2]

This paper describes our progress to date in developing the ideas of computational resiliency and

[1] This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract N66001-99-1-8922.

[2] Thanks to Cathy McCullum for providing the analogy

applying them to remote sensing applications using a spectral-screening PCT algorithm. The intent is to gain a preliminary evaluation of the overheads involved in using resiliency. The PCT algorithm summarizes the information content of a hyper-spectral image into a single color-composite image using three techniques: spectral angle classification, principal component transformation, and human centered color mapping. Test data for the algorithm is supplied from the Hyper-spectral Digital Imagery Collection Experiment (HYDICE) sensor, an airborne imaging spectrometer. The algorithm is executed on a local area network containing 16 workstations connected with 100BaseT networking.

2. Computational Resiliency

There are several significant technical challenges involved in developing systems based on computational resiliency. Techniques must be developed for providing policy driven, on-the-fly replication, camouflage, and mobile threads. There are also a number of serious theoretic concerns that relate to race conditions in reconfiguration of a distributed application, resource management, and providing guarantees on message delivery:

Clustered Multiprocessing. Today's distributed computing platforms are not a simple mix of local area networking and workstations that can be utilized directly by message passing. With high-performance and wireless networking pushing from one end of the technology spectrum and low cost multiprocessor technology pushing from the other, there is now a confluence of technologies based on *heterogeneous clustered environments* that includes shared-memory multiprocessors. Theses systems are composed of machines with substantively different memory and processor characteristics, operating systems floating point representations and byte orderings.

To provide highly mobile threads with the ability to reconfigure in such an environment, it is necessary to have an explicit representation of *the communication structure* used by the application. We have developed a concurrent programming library, SCPlib, that provides this basic functionality [Taylor et al. 1995]. We implement distributed applications composed of a collection of threads that communicate and synchronize either through shared memory or by sending messages. Each thread has an associated *state*, which is operated on by application specific routines e.g. in a remote sensing application this may involve matrix algebra. A thread also has a machine independent description of its communication structure. In general these systems are *reactive* [Seitz 1985] in that the important transitions between data states occur at the receipt of messages. This provides a natural mechanism to synchronize each thread, detect and information warfare attack and initiate recovery.

Resilience. To tolerate information warfare attacks, applications may choose to replicate mission critical threads as shown in Figure 1, thereby gracefully degrading to the point of failure. In any realistic system, there will never be sufficient resources to replicate all resources, therefore some policy-based methods for controlling replication are required. Although this approach provides fault-tolerance it is not resilient in that it does not assure continued operation of the system.

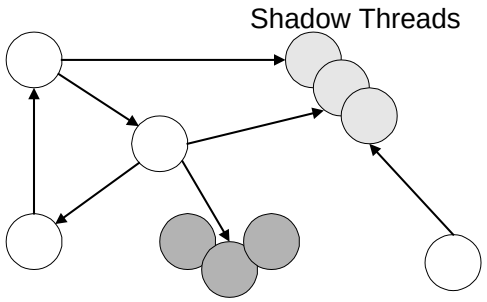

Figure 1: Replication of Threads

An alternative approach is to dynamically recreate the level of replication in the face of attack and so as to assure that operational readiness is eventually restored, subject only to the constraints imposed by the total available resources. Obviously to be successful, the replacement thread must be mapped to an alternative location in the network with sufficient resources. Protocols are required both to communicate between groups of threads and to reconfigure the groups dynamically. Prototypes for these protocols have been devised and are used to configure the concurrent computation in this paper. The protocols both recreate the thread and automatically reconfigure communication to the new location in the network. The protocols deal with race conditions inherent in reconfiguration, ensure that no communication is lost, that the integrity of the state is maintained, and that where possible locality of communication is preserved.

Resource Management. To dynamically recover, replication requires the ability to recreate a thread with the appropriate communication structure at some other location in the network. Replicating a thread at the

location of one of the backup copies, and subsequently moving the new thread to a new location can achieve this. Unfortunately, it may not be efficient with the available resources to move the thread either because of memory disparities or because of *granularity*: the ratio of computation to communication in the application. Protocols are needed to allow thread granularity to be increased, by merging, or decreased, by splitting the associated thread computation [Watts and Taylor 1998]. Armed with basic techniques for mapping and granularity control. It is then possible to build concepts for resource management [Watts et al. 1995, 1998]. There exists no general solution to the resource management problem, and thus each application must employ an appropriate technique [Bokhari 1981]. In this paper a simple Manager-Worker approach is used.

3. Concurrent Spectral-Screening PCT Algorithm

To examine how to utilize resiliency, we developed a *spectral-screening PCT algorithm* that can be used for hyper-spectral image fusion in remote sensing applications [Achalakul et al. 1999]. The algorithm combines the Principal Component Transform (PCT) [Mackiewicz 1993, Singh 1993] with spectral angle classification [Kruse et al. 1993] and human-centered color mapping [Boynton 1979, Peterson et al. 1993, Poirson and Wandell, 1993].

The PCT is used to summarize and de-correlate the images by removing redundancy and packing the residual information into a small set of images, termed *principal components*. To prevent the PCT from highlighting only the variation that dominates numerically, we augment it with spectral angle classification prior to the de-correlation process. This has the effect of reducing the importance of an object that occurs frequently in a scene. For example, the spectral signature of a mechanized vehicle embedded in a forest scene will be treated as equally important as the signature associated with trees. The final step of the algorithm is to generate a color-composite image from a collection of principal components. To achieve this, we use a human-centered approach that attempts to match the spatial-spectral content of the output image with the spatial-spectral processing capabilities of the human visual system.

The distributed version of this algorithm uses the standard manager/worker decomposition technique [Chandy and Taylor 1992]. The manager thread partitions the problem and distributes the sub-problems to worker threads. The workers solve the allocated sub-problem, send back the result, and wait for the next problem. Each sub-problem is *a sub-cube of the hyper-spectral image set* similar to the decompositions used in [Palmer et al. 1998]. To reduce communication overhead, a worker overlaps the request for its next sub-problem with the calculation associated with the current sub-problem. Using this approach the PCT algorithm is divided into 8 steps as follows:

1. ***Spectral classification:*** The manager divides an original hyper-spectral image cube into *P* parts, where *P* is the number of workers in the system. Each part, which consists of a set of pixel vectors, is sent to a worker. Each worker operates concurrently to form a unique spectral set by calculating an arccosine of dotproduct of all pixel vectors pair.

for all p=1 to P *concurrently*
for all pixels (i, j) in each p {
$\alpha(i, j) = \cos^{-1}(x \bullet y / \|x\| \bullet \|y\|)$
}

where P = number of parts
x, y = pixel vector pair

2. ***Merge unique sets*:** The P unique sets are sent back to the manager and combined. Upon completion there will be one unique set left with *K* pixel vectors.

3. ***Mean vector:*** Each component of the mean vector, m, is the average of the pixel values of each spectral band of the unique set. The n-band hyper-spectral image produces a mean vector of n elements where each element can be computed as follows:

for all i =1 to n *concurrently*

$$m[i] = \frac{1}{K}\sum_{k=1}^{K} I_k[i]$$

where K = number of pixel vectors in a unique set
I_k = pixel vectors in the unique set

4. ***Covariance Sum:*** All the pixel vectors in a unique set are divided into *P* parts, and sent to *P*

workers. Each worker then executes the following code to form a covariance sum:

for all p = 1 to P *concurrently*{
  $sum_p = 0$
  for all pixels (i, j) in each p {
    $C_{ij} = I_{ij} I_{ij}^T - mm^T$
    $sum_p = sum_p + C_{ij}$
  }
}
where P = number of parts
  sum_p = the matrix sum of the covariance in each p
  I_{ij} = pixel vectors in the unique set

5. ***Covariance matrix:*** The covariance matrix is the average of all the matrices calculated in step 4, and is calculated sequentially by the manager since its complexity is related only to the number of workers rather than the image size.

6. ***Transformation matrix:*** The eigenvectors of the covariance matrix are calculated and sorted according to their corresponding eigenvalues which provide a measure of their variances. As a result, the high spectral content is forced into the front components. Since the degree of data dependency of the calculation is high, but its complexity is related to the number of spectral bands rather than the image size, this step is also done sequentially.

7. ***Transformation of the data:*** Each pixel vector, Is_{ij}, in the original hyper-spectral image, can be transformed independently. Therefore, all workers transform their portions of the data concurrently as follows:

for all p = 1 to P *concurrently*
  for all pixels (i, j) in each p
    $Cs_{ij} = A(Is_{ij} - m)$

where P = number of parts
  Is_{ij} = pixel vectors in the original image
  Cs_{ij} = transformed pixel vectors

8. ***Color mapping:*** Each worker performs the human-centered color mapping using the first three resulting components of step 7 to generate a portion of the final color image.

for all p=1 to P *concurrently*
  for all pixels (i, j) in each p

$$R_{ij} = (128 + (\begin{bmatrix} 0.4387 & 0.4972 & 0.0641 \\ 0.4972 & -0.1403 & -0.0795 \\ -0.1355 & 0.0116 & 0.4972 \end{bmatrix} * (C_{ij} - 128))) / 256$$

where C_{ij} = the first three pixels of the transformed pixel vectors
  R_{ij} = the resulting pixel vectors

The algorithm was tested using a 210-channel hyper-spectral image collected with the Hyper-spectral Digital Imagery Collection Experiment (HYDICE) sensor, an airborne imaging spectrometer. These images correspond to foliated scenes taken from an altitude of 2000 to 7500 meters at wavelengths between 400nm and 2.5 micron. The scenes contain mechanized vehicles sitting in open fields as well as under camouflage. Figure 2 shows two frames picked from the 210 spectral bands.

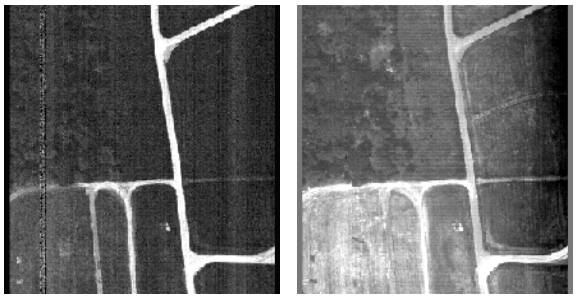

Figure 2: 400 and 1998 nm

Figure 3 shows the resulting image after applying the concurrent spectral-screening PCT to the full 210 frames data set. The color-mapping scheme maps the first principal component to achromatic, the second to red-green opponency, and the third to blue-yellow opponency. The result, when viewed on a high-quality monitor, shows significantly improved contrast levels. The forested areas show significantly improved detail and the camouflaged vehicle in the lower left corner is significantly enhanced against its background. Postprocessing steps can subsequently be applied to

detect edges in the image and use structural information to detect and classify the vehicles.

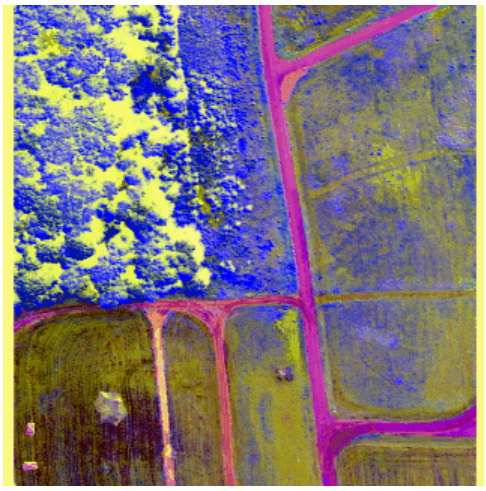

Figure 3: Color-Composite Image

4. Performance Evaluation

The performance of the algorithm was measured on a distributed environment consisting of 16 Sun Solaris 300MHz.workstations connected with 100BaseT networking technology. The same experiment was conducted with all workers replicated to a level of two; the manager, which represents the sensor itself was not replicated.

The expected result was that performance would decrease by a factor of two since the replicated processes require both memory and processor resources. We expect the performance to decrease more than a factor two due to the increased overhead of communication associated with the more complex communication protocols required to achieve redundancy.

Figure 4 shows the speed up gained as a function of the number of computers both with and without resiliency. Notice that the overhead caused by resiliency is approximately 10% plus the cost of replication uniformly. The concurrent algorithm operates within 20% of linear speedup in both cases. There are several aspects of the algorithm that cause it to deviate from ideal speed up: The problem size required communication overhead in transferring sub-problems between the manager and worker, Steps 2 and 6 in the algorithm currently operate sequentially and the code used for finding the eigenvalues (step 6) dominates the sequential time. Although the algorithm has a complexity of $O(n^3)$, at the typical problem size of 210 frames, the time used for Step 6 does not dominate the overall performance.

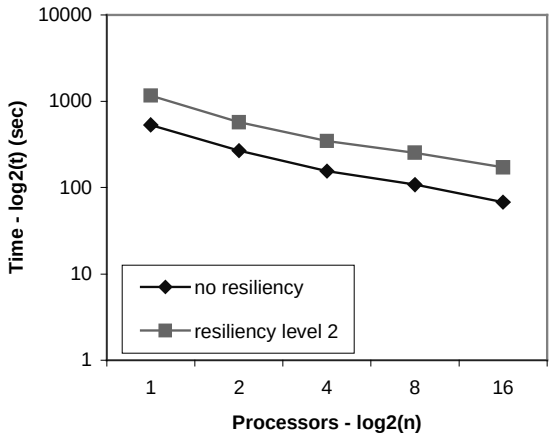

Figure 4: Performance Chart

The overhead in transferring a sub-problem is a function of the granularity of the decomposition. Figure 5 examines performance using different granularity decompositions. The results show that dividing an image cube into a considerably larger number of sub-cubes than the number of processors enables computation and communication overlapping. The overlapping reduces the communication overhead, and thus, increases overall performance. When the granularity is too fine, the computation on each sub-

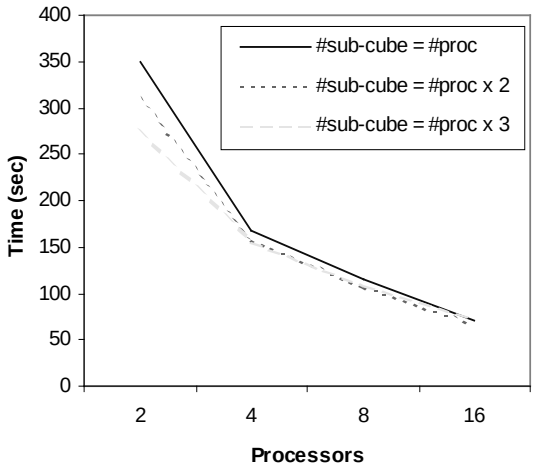

Figure 5: Granularity control

cube becomes too small, and communication overhead dominated. In the experiment the initial cube size was 320x320x105. The performance tailed off when the

problem was spilt into more than n = 32 sub-cubes. This indicates that, for this problem size, using more than 16 computers will not buy substantial performance improvement. The general effect would be more pronounced in larger image sets. For example, a real remote sensing application could have 210 frames rather than 105, and use 1024x1024 resolution rather than 320x320. Unfortunately, this data set size could not be used due to memory constraints in our available network.

The results shown here do not include multiprocessors in the network. On a shared memory system, the concurrent algorithm presented here operates within 5% of linear speedup on a wide range of problem sizes and machine sizes [Achalakul 2000]. The advantage of using share-memory multiprocessors is that no communication overhead involved in the algorithm.

5. Conclusion

This paper has described an intrusion tolerant version of our concurrent spectral-screening PCT algorithm. The algorithm operates on a network of single- and multi-processor PC's/ workstations. It combines three basic image processing concepts: spectral angle classification, principal component transform, and human centered color mapping and is representative of typical techniques used in remote sensing applications.

The algorithm was implemented using the resilient concurrent programming technology and the impact on the algorithm of this technology is described in terms of a simple manager-worker partitioning and mapping scheme. The preliminary performance evaluation shows that the overheads associated with the more complex communication protocols required to provide resiliency account for approximated a 10% reduction in overall performance above that expected by the cost of replication. The protocols are as yet in an early stage of development and are not optimized. Considerable research remains to examine the concept further.

6. References

1. Achalakul T., Haaland P. D.,Taylor S., "Mathweb: A Concurrent Image Analysis Tool Suite for Multi-spectral Data Fusion", SPIE vol. 3719 Sensor Fusion: Architectures, Algorithms, and Applications III, 1999, pp351-358.
2. Achalakul T., Taylor S., A Concurrent Spectral-Screening PCT Algorithm for Remote Sensing Applications, Submitted to Journal of Information Fusion, 2000.
3. Agarwal D.A., *"Totem : A reliable ordered delivery protocol for interconnected local-area networks"*, Ph.d dissertation. Dept. of Electrical and Computer Engineering. University of California, Santa Barbara, 1994.
4. Amir Y., Dolev, Kramer S., and Malki D., *"Transis : A communication sub-system for high availability"*, Proc. of the 22nd Annual Internaltional Symposium on Fault-Tolerant Computing, pp76-84, July, 1992.
5. Bokhari, S.H. "On the Mapping Problem", IEEE Transactions on Computers Vol C-30, No 3, March 1981, pp 207-14.
6. Boynton T. M., *Human Color Vision*, Rinehart, and Winston, New York, 1979.
7. Budhiraja N., Marzullo K., Schneider F. B., Toueg S., *"The Primary-Backup Approach"*, Distributed Systems, ACM Press, 1993, pp199-216
8. Chandy L. M., *Taylor S., An Introduction to Parallel Programming*, Jones and Bartlett publishers, Boston, 1992.
9. Guerraoui R., Schiper A., *"Software-based Replication for Fault Tolerance"*, IIEEE Computer, April, 1997.
10. Kruse F. A., Lefkoff A. B., Boardman J. W., Heidebrecht K. B., Shapiro A. T., Barloon P. J., and Goetz F. H., "The spectral Image Processing System (SIPS) – Interactive Visualization and Analysis of Imaging Spectrometer Data", Remote Sensing Environment vol 44, 1993, pp 145-163.
11. Mackiewicz A. and Ratajczak W., "Principal Components Analysis (PCA)", Computers & Geosciences, vol.19, 1993, pp303-342.
12. Palmer M., Totty B., Taylor S., "Ray Casting on shared-Memory Architectures: Efficient Exploitation of the Memory Hierarchy", IEEE Concurrency, Vol 6, No. 1, pp 20-36, 1998.
13. Peterson H. A., Ahumada A. J., and Watson A. B., "An Improved Detection Model for DCT Coefficient Quantization", SPIE, vol. 1913, 1993, pp. 191-201.
14. Poirson A. B., Wandell B. A., "Appearance of Colored Patterns: Pattern-color Separability", Journal of the Optical Society of America, vol. 10, 1993, pp 2458-2470.
15. Seitz C.L., "The Cosmic Cube", CACM, vol 28, No. 1, pp. 22-33, 1985.
16. Singh A. and Eklundh L., "A Comparative Analysis of Standardised and Unstandardised Principal Components Analysis in Remote Sensing", International Journal of Remote Sensing, vol. 14, 1993, pp1359-1370.
17. Taylor S., Watts J., Rieffel M., and Palmer M., *"The Concurrent Graph: Basic Technology for Irregular Problems"*, IEEE Parallel and Distributed Technology, vol. 4, pp. 15-25, 1995.
18. Watts J., and Taylor S., "A Practical Approach to Dynamic Load Balancing", IEEE Transactions on Parallel and Distributed Systems, vol 9, pp 235-248, 1998.
19. Watts J., Taylor S., and Nilpanich S., *"SCPlib : A Concurrent Programming Library for Programming Heterogeneous Networks of Computers"*, IEEE Information Technology Conference, EX 228, pp. 153-6, 1998.