BIG DATA PROCESSING USING HPC FOR REMOTE SENSING DISASTER DATA

Ujwala M. Bhangale[1], Kuldeep R. Kurte[1], Surya S. Durbha[1], Roger L. King[2], Nicolas H. Younan[2]

[1]Centre of Studies in Resources Engineering, IIT Bombay, Powai, Mumbai - 400076, India,
[2]Department of Electrical and Computer Engineering, Mississippi State University, Starkville, USA
[1]{ujwala.bhangale, kuldeep.iitb, sdurbha}@iitb.ac.in, [2]{rking,younan}@ece.msstate.edu

ABSTRACT

Voluminous data (Multispectral, Hyperspectral) from **V**ariety of sensors (Airborne sensors, space borne sensors) with **V**elocity (high temporal resolution) when used for decision making to support natural disasters such as earthquakes, floods, oil-spills etc., for near real time accurate responses, is a problem that needs Big Data Analytics. To gain rapid insight from this big data, high performance computing (HPC) with some scalable solution that reduces the execution time are in extreme demand. To serve this real time need, scalable hybrid parallelism approach based on state of art multi-core GPUs and Message Passing Interface (MPI) is explored for analyzing remote sensing disaster data. Spatio-temporal remote sensing data of oil-spill at Gulf of Mexico captured by LANDSAT 7 ETM+ is considered for analysis. The core objective includes performance evaluation of the analysis process across various parallel implementation platforms.

Index Terms**—** CUDA, GPU, MPI, HPC, Big Data Analytics

1. INTRODUCTION

Tremendous amounts of availability of Remote Sensing (RS) data, complexity involved in analysis i.e. multistep process (segmentation, feature extraction, classification etc.) and extensive utilization of this data for decision making needs the entire system to be based on big data analytics framework. Voigt S. et al. [1] have discussed many disaster response applications such as Indian Ocean Tsunami, landslide extent mapping for Philippines, forest fire mapping for Portugal etc., which are based on RS data analysis. Sophisticated analysis process consists of multiple computationally expensive steps to generate more accurate results. Each of these steps may require several hours of computations. One of the most important and time consuming step of this analysis is feature extraction that extracts the attributes or properties from a scene. Various types of features are extracted from images such as, texture features, contrast features, geometric features etc. These features are extracted to serve different analysis purposes such as Image retrieval (IR) [2][3], image registration [4], object detection [5], disaster event detection [6], disaster related change /damage detection [7][8] etc. The entire analysis workflow for these type of object based classification techniques are time intensive, usually takes several hours (30 hr or more) [1]. Also, hybrid analysis techniques that combine both supervised and unsupervised mechanisms, boosting techniques that iteratively apply ensemble classifiers to produce more accurate results [9] further increases the overall analysis time. Near real time responses to support disaster occurrence detection, damage assessment, relief assistance etc. activities, requires near real time processing, which will speed up the overall analysis process and facilitates rapid responses.

Different HPC technologies such as GPU, FPGA, MPI are already being used for satellite image analysis [10]. Recently, Content Base Image Retrieval (CBIR) system for hyperspectral data is developed using a cluster of 44 Nvidia TESLA M2070Q GPUs; it provides around 7000X speedup which is a state of art performance [11]. Another recent work as a performance study of HPC technologies, uses OpenCL on two heterogeneous platforms, GPUs and field programmable gate arrays (FPGAs), for hyperspectral image analysis [12] . G. Cavallaro et al. [13] have employed smart data analytics techniques using Support Vector Machine (SVM) based on MPI parallel computation and observed around 6X speedup using 16 nodes over a serial execution environment.

This work focuses on parallel execution of big data analysis processes using heterogeneous parallel platforms such as state-of-art multi-core GPUs and MPI platform.
Core objectives of this work includes,

- Parallel implementation of analysis workflow using MPI [14] on CPU cluster with 64 CPU cores
- Parallel implementation of analysis workflow using Multi-core K40 GPU
- Hybrid Parallel implementation of the workflow using heterogeneous parallel platform (GPUs, MPI)
- Performance evaluation of these platforms

2. METHODOLOGY

A high level workflow for analyzing disaster data is shown in figure 1. Spatio-temporal high resolution satellite images (may be from different sensors) can be given as input.

978-1-5090-3332-4/16/$31.00 ©2016 IEEE

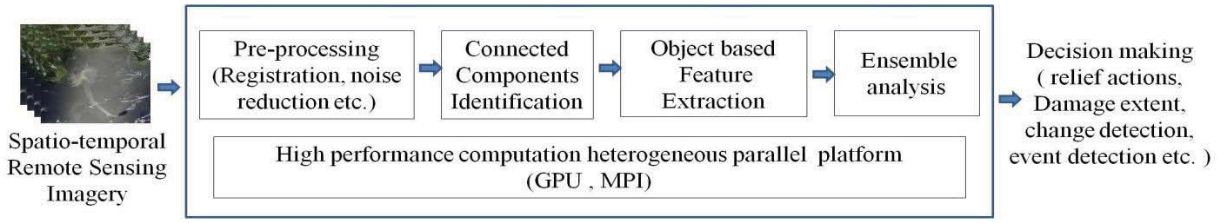

Spatio-temporal Remote Sensing Imagery
Pre-processing (Registration, noise reduction etc.)
Connected Components Identification
Object based Feature Extraction
Ensemble analysis
High performance computation heterogeneous parallel platform (GPU , MPI)
Decision making (relief actions, Damage extent, change detection, event detection etc.)

Figure1. Disaster related big data analysis framework for remote sensing data using parallel platforms

Preprocessing such as noise reduction, gap filling etc. can be performed. For each image, Segmentation / connected components are identified. Further, from each component, relevant and robust features can be extracted, suitable to the type of disasters. Finally, ensemble analysis is required, which will include ensemble classification approaches to produce most accurate results to support decision making in disaster management.

Each of these steps have to be implemented in parallel to provide near real time responses for disaster management activities. The below sections discusses one of the most important process of this workflow i.e. feature extraction and presents the parallel implementation details and performance evaluation of the same on MPI platform.

2.1. Computationally efficient feature extraction from Big data

LANDSAT 7 ETM+ (panchromatic) oil-spill images are considered for feature extraction. In literature, most of the oil spill detection work uses different types of analysis for accurate oil spill detection, mainly to avoid false alarm due to lookalike objects. Stathakis D. et al. [15] have performed neural network classification using 25 features, categorized as geometrical, physical and texture features, they have concluded that the method demands computational resources. Solberg et al. [6] have used various shape features for classifying oil regions such as *slick complexity*, *slick area*, *width* etc., and contrast-based features such as *slick local contrast*, *border gradient* etc. For oil spill detection from SAR images, Topouzelis K. [16] discusses various shape characteristics of the oil spill which helps to discriminate it from its lookalikes such as roundness, elongation etc.

For each image, connected components (objects) are identified using 8-connectivity. Further, features are extracted for each component, 6 different features namely, Gray Mean (GM), standard deviation (SD), Slick Complexity (SC), Elongation (EL), Solidity (SL), Orientation (OR) are extracted as depicted in figure 2 which presents the MPI based feature extraction process. Attributes, *solidity* and *orientation* are used to identify the oil spill regions which distinguish between lookalike and oil spill regions. These features are described below in brief,

GM ensures spectral similarity i.e. the regions having similar spectral reflectance; Mean value of each connected component *C* is calculated as,

$$GM = \frac{\sum_{i=1}^{n} p_i}{n} \qquad (1)$$

Where, p_i is intensity value of each pixel that belong to *C,* and *n* is total number of pixels that belong to *C; n* is also referred to as the *area* of *C*.

Structural similarity such as homogeneity can be observed by obtaining Standard Deviation of the connected components. SD calculation is benefited because of earlier calculated GM of each connected component.

Slick complexity [6] is used to measure the complexity of the oil spill region. High Slick complexity value indicates complex shapes whereas low value indicates simpler shapes. Slick complexity is calculated as,

$$SC = \frac{\mathrm{P}^2}{\mathrm{n}} \qquad (2)$$

Where, *P* is the perimeter and *n* is area (i.e. total number of pixels) of the connected component.

Geometric similarity, which is based on the length of the region, can be observed using elongation attribute.

$$Elongation = \frac{Length\ of\ Major\ Axis\ of\ Convex\ hull\ enclosing\ C}{Length\ of\ Minor\ Axis\ of\ Convex\ hull\ enclosing\ C} \qquad (3)$$

Solidity is a ratio of the area occupied by the object to the area of its convex hull, basically it analyses the concavity or convexity of the region.

Orientation defines the angle between the *x* axis and the major axis of the convex hull drawn around the region i.e. connected components. Orientation helps to understand the direction of oil flow influenced by the water current and the wind. To compute orientation and major, minor axis of convex hull covering the region, the technique discussed in [17] is used.

Features from each connected components are extracted in parallel using MPI technology as shown in figure 2. Root process is responsible for distributing the input data and uniform number of components at each core from each node. All cores works in parallel, extracts the features from the connected components assigned to it; after finishing with all components, each core sends back the extracted features to root process. Root process collects the features after finishing with all components, and store it at appropriate

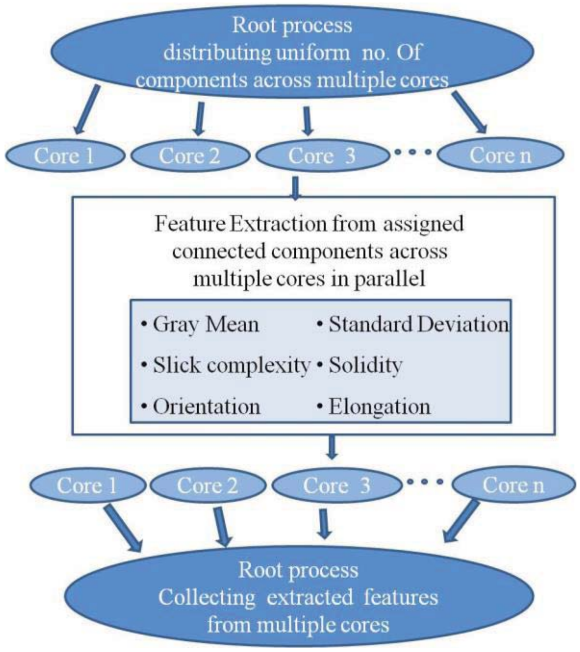


Figure 2. MPI based feature extraction process for oil spill detection

location of feature matrix, which holds all features of all connected components together. Next session discusses the MPI implementation details of parallel feature extraction process.

3. MPI IMPLEMENTATION DETAILS FOR FEATURE EXTRACTION PROCESS

Implementation is done by calling MPI routines in C code. MPI calls are invoked to initiate the MPI process after obtaining connected components from the image. The first process, i.e. rank 0 calculates the number of components that can be distributed to each core, such that all cores should get assigned equal amount of work, only the last process gets remaining components, may be lesser than number of components assigned to all other processes. The labeled components matrix and original image matrix (required for GM, SD computation) are broadcasted to all processes, following statement demonstrates the broadcast of the labeled components matrix,

*ierr = MPI_Bcast(band_1_conn, no_of_rows*no_of_cols, MPI_INT,0, MPI_COMM_WORLD);*

where *band_1_conn* is the labeled components matrix, *no_of_rows*no_of_cols* is the size of matrix, *MPI_INT* represents the data type of the matrix, *0* represents the process or rank broadcasting the data, *MPI_COMM_WORLD* is a MPI communicator. *MPI_Bcast* is observed as more time efficient as compared to *MPI_Send* when same data is to be transferred to all cores, also other processes need not have to explicitly collect the data using *MPI_Recv* as required by *MPI_Send* , data relevant to specific core is transferred using *MPI_Send.* Figure 3 shows the code snippet where rank 0 distributing the components across all cores. Root process computes *start_comp* , *end_comp* i.e. start and end component identifiers for all other processes so that all processes should gets uniform number of components; it then sends that data to all other processes. Each process then extracts the features only for the specific components assigned by root process.

```
if (rank == 0)
 {
  avg_comps_per_process = (comp / (numtasks-1)) +1; //
calculates no .of components to transfer to each processor
/* distribute equal no of components to each processor */
 for(id = 1; id < numtasks ; id++)
  {
     start_comp = (id-1)*avg_comps_per_process + 1;
     end_comp  = start_comp +avg_comps_per_process-1;
    if((comp - start_comp) < avg_comps_per_process) //for
     remaining last components
     end_comp = comp - 1;
     num_comps_to_send = end_comp - start_comp + 1;
      ierr = MPI_Send( &num_comps_to_send, 1 , MPI_INT, id,
send_data_tag, MPI_COMM_WORLD);
     ierr = MPI_Send(&start_comp ,1 , MPI_INT, id,
send_data_tag, MPI_COMM_WORLD);
                     :
  }
else // for all other process except root
 {    ierr = MPI_Recv( &num_comps_to_send, 1 , MPI_INT, 0,
send_data_tag, MPI_COMM_WORLD, &status);
  ierr = MPI_Recv( &start_comp, 1 , MPI_INT, 0, send_data_tag,
MPI_COMM_WORLD, &status);
                     :
for (k=start_comp;k<=start_comp+num_comps_to_send-1;k++)
{ // all the processors extracts the features only from components
assigned to it.
 }
```

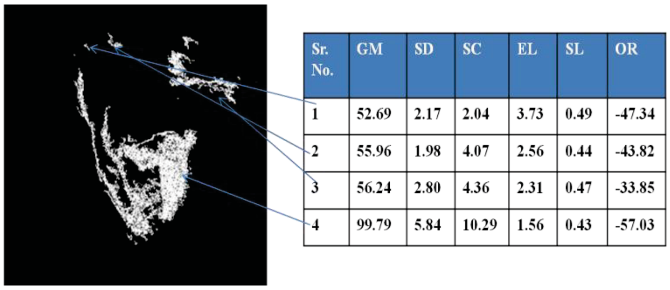
Figure 3. C MPI code snippet for feature extraction process from each connected components

4. PERFORMANCE EVALUATION

Oil-spill (panchromatic) images captured by LANDSAT 7 ETM+ are considered for analysis; Figure 4 shows the oil-spill objects obtained from one of the image and the table contains the features extracted from those objects, arrows from features table are pointing to corresponding objects. These extracted feature values are computed using C MPI code and verified against regionprops properties available in MATLAB Image Processing Toolbox [18].

4.1 MPI Performance Evaluation of Feature extraction

Execution platform is a cluster of 4 nodes, where each node has two Intel(R) Xeon(R) CPU E5-2640 processors, each processor has 8 cores. CUDA aware OpenMPI v1.8.1 is used. Execution time is measured by varying the number of cores as 8, 16, 32, and 64. Figure 5 presents the performance of features extraction process by varying number of cores against number of connected components.



Sr. No.	GM	SD	SC	EL	SL	OR
1	52.69	2.17	2.04	3.73	0.49	-47.34
2	55.96	1.98	4.07	2.56	0.44	-43.82
3	56.24	2.80	4.36	2.31	0.47	-33.85
4	99.79	5.84	10.29	1.56	0.43	-57.03

Figure 4. Sample features obtained from oil spill objects. (Arrows from Sr. No. pointing to corresponding objects)

Execution using all 64 cores performs 5X faster as compared to execution on 8 cores. Sequential execution of feature extraction process (without MPI) takes around 11 hours. GPU based CUDA implementation with CUDA aware MPI is under progress; but surely it will enhance the speedup over only MPI based implementation, as it will attempt multilevel parallelism.

5. CONCLUSION

Involvement of tremendous amount of remote sensing data to support disaster management activities in real time scenario clearly indicated the need of big data analytics framework based on HPC technologies. In this work, a disaster data analysis framework is proposed to suit the big data's high computational resources requirement. One of the most important process from this analysis i.e. feature extraction is implemented using MPI on 64 cores, it takes only 18 minutes whereas sequential execution takes around 11 hours for 33000 components.

Currently, hybrid parallelism using CUDA aware MPI that uses heterogonous platform i.e. GPU and MPI is being done to exploit multilevel parallelism.

6. REFERENCES

[1] S. Voigt, T. Kemper, T. Riedlinger, R. Kiefl, K. Scholte, and H. Mehl, "Satellite Image Analysis for Disaster and Crisis-Management Support," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 6, pp. 1520–1528, 2007.

[2] D. Espinoza-Molina and M. Datcu, "Earth-Observation Image Retrieval Based on Content, Semantics, and Metadata," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, pp. 5145–5159, 2013.

[3] R. Buaba, A. Homaifar, M. Gebril, E. Kihn, and N. Ngdc, "Satellite Image Retrieval Application using Locality Sensitive Hashing in L 2 -Space," In *Aerospace Conference, 2011 IEEE* pp. 1–7, 2011.

[4] C. A. Shah, Y. Sheng, and L. C. Smith, "Automated Image Registration Based on Pseudoinvariant Metrics of Dynamic Land-Surface Features," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3908–3916, 2008.

[5] C. Z. C. Zhu, H. Z. H. Zhou, R. W. R. Wang, and J. G. J. Guo, "A Novel Hierarchical Method of Ship Detection from Spaceborne Optical Image Based on Shape and Texture Features," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3446–3456, 2010.

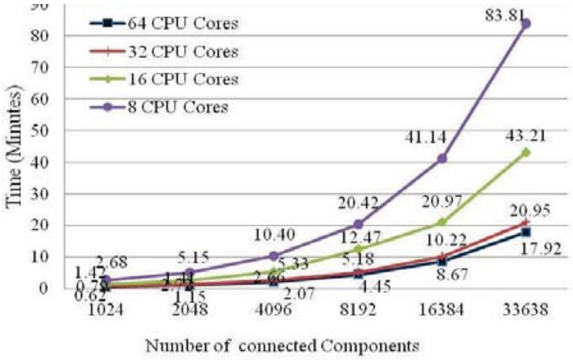


Figure 5. MPI Performance by varying number of cores

[6] A. H. S. Solberg, C. Brekke, and P. O. Husøy, "Oil spill detection in Radarsat and Envisat SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 3, pp. 746–754, 2007.

[7] D. Faur, "A Rapid Mapping Approach to Quantify Damages Caused by the 2003 Bam Earthquake Using High Resolution Multitemporal Optical Images," In *8th Inter. Workshop on the Anal. of Multitemp. Remote Sensing Images (Multi-Temp), 2015* pp. 4–7, 2015.

[8] L. Gueguen, M. Pesaresi, A. Gerhardinger, and P. Soille, "Characterizing and counting roofless buildings in very high resolution optical images," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 1, pp. 114–118, 2012.

[9] S. Xu, T. Fang, D. Li, and S. Wang, "Object classification of aerial images with bag-of-visual words," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 2, pp. 366–370, 2010.

[10] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High Performance Computing for Hyperspectral Remote Sensing," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 4, no. 3, pp. 528–544, 2011.

[11] J. Sevilla, L. I. Jiménez, and A. Plaza, "Sparse Unmixing-Based Content Retrieval of Hyperspectral Images on Graphics Processing Units," *Geoscience and Remote Sensing Letters, IEEE* 1 vol. 12, no. 12, pp. 2443–2447, 2015.

[12] S. Bernabe, F. D. Igual, G. Botella, C. Garcia, M. Prieto-Matias, and A. Plaza, "Performance portability study of an automatic target detection and classification algorithm for hyperspectral image analysis using OpenCL," In *SPIE Remote Sensing*, vol. 9646, pp. 96460M-96460M, 2015.

[13] G. Cavallaro, M. Riedel, J. Atli, M. Goetz, T. Runarsson, K. Jonasson, and T. Lippert, "Smart data analytics methods for remote sensing applications," In *Geoscience and Remote Sensing Symposium (IGARSS), 2014* pp. 1405–1408, 2014.

[14] Open MPI [Online]. Available: http://www.open-mpi.org/ [Accessed : 01 -Jan-2015].

[15] D. Stathakis, K. Topouzelis, and V. Karathanassi, "Large-scale feature selection using evolved neural networks," In *Remote Sensing* vol. 6365, pp. 636513–636513–9, 2006.

[16] K. N. Topouzelis, "Oil spill detection by SAR images: Dark formation detection, feature extraction and classification algorithms," *Sensors*, vol. 8, no. 10, pp. 6642–6659, 2008.

[17] D. Chaudhuri and a. Samal, "A simple method for fitting of bounding rectangle to closed regions," *Pattern Recognit.*, vol. 40, no. 7, pp. 1981–1989, 2007.

[18] The Mathworks Inc. [Online] Available: http://in.mathworks.com/help/images/ref/regionprops.html [Accessed: 16-Feb-2015].