Spatial Data Processing with MapReduce

Tilani Gunawardena *
etilani@gmail.com

Annamaria Vicari †
annamaria.vicari@ingv.it

Giansalvatore Mecca *
giansalvatore.mecca@gmail.com

* Department of Mathematics, Computer Science and Economics
University Of Basilicata, Potenza, Italy
† National Institute of Geophysics and Volcanology, Italy

***Abstract*—The current development of high performance parallel supercomputing infrastructures are pushing the boundaries of applications of science and are bringing new paradigms into engineering practices and simulations. Earthquake engineering is also one of the major fields, which benefits from above by looking for solutions in grid computing and cloud computing techniques. Generally, earthquake simulations involve analysis of petabytes of data. Analyzing these large amounts of data in parallel in thousands of nodes in computer clusters results in gaining high performances. Open source cloud solutions such as Hadoop MapReduce, which is highly scalable and capable of processing large amount of data rapidly in parallel on large clusters provide better solution compared to RDBDM. Both GPUs and MapReduce are designed to support vast data parallelism. For performance considerations, GPU computing could be adopted over low performing CPU systems. This paper discusses MapReduce system using Hadoop and Mars. Mars is a MapReduce framework on graphics processor. Hence, the proposition is to use GPU based systems for earthquake simulations in which Digital elevation model 3D data sets are fully materialized where scientist can make use of these data for various analysis and simulations.**

***Index Terms*—MapReduce, Spatial Data, GPU computing, CUDA, Hadoop, Big Data**

I. Introduction

Scientific Computing is applied to a variety of fields such as climate research, satellite feeds, genomics etc. Data sets are becoming more complexed and increasing in size by the day, often running into a petabyte range. Their analysis, archival and sharing have become difficult and challenging. The complexity of data is such that it led to the application of massively parallel supercomputing techniques during analysis. To meet the demands created by complex data sets, there is a need that algorithms use the parallel computer architectures effectively.

The fields similar to earthquake engineering tend to look for solutions in grid computing and cloud computing techniques to analyse large amount of data. In earthquake simulations it is not uncommon to see petabytes of data for analysis. Analysing these data in parallel in thousand of computer/node clusters enables high performance. Specially an open source cloud solution like MapReduce using cheap commodity hardware is a better solution in scalability and performance. Programming models similar to Hadoop MapReduce gives the capability to write applications which can process large amounts of data rapidly in parallel on large clusters.

Commodity computer graphics chips are probably todays most powerful computational hardware for the dollar. Many researchers and developers have become interested in harnessing the power of commodity graphics hardware for general-purpose computing. Recent years have seen an explosion in interest in such research efforts, known collectively as GPGPU computing. Further, the introduction of general purpose programming languages such as Nvidias CUDA, Stream, and OpenCL has promoted their use as general purpose processors.

An earthquake is the result of a sudden release of energy in the earth's crust that creates seismic waves. Earthquakes are caused mostly by rupture of geological faults, but also by other events such as volcanic activity, landslides, mine blasts, and nuclear tests. The processes that cause earthquakes mostly occur far below the earth's surface. Scientists have tried lot of different ways of predicting earthquakes, but none have been successful. They can get a good idea of where an earthquake is most likely to hit, but still can not tell exactly when it will happen. Vibrations can be detected just before an earthquake occurs, but this does not give enough time for people to escape. However, the probability of a future earthquake can be calculated, based on scientific data.

Scientist needs to generate ground characteristic data with low resolutions for earthquake simulations. Since all earthquake simulations need petabytes of ground characteristic data generated in an ad hoc fashion, generation of data is very expensive. For example ground model data for several large earthquake simulations can take several days or few weeks. Also most of these ground data are homogeneous. So instead of using all the data, if scientists could use materialised data(pre computed) generated in an ad-hoc manner without using super commuting centres, scientists and seismologist can get a huge advantage. In this paper we propose a MapReduce system using Hadoop and GPUs to get higher performance.

Figure 1 shows typical FE simulation, goal of this experiment is to mesh generating and partitioning, where scientists can use the final reduced(coalesced) results for further simulations. Also it is possible for scientists to generate these data in an ad hoc manner in less time without using super computing powers. Finally reduced results set is obtained to understand the fault regions in the ground without having to generate all

the mesh data. In seismologists and scientists' perspective, having a method to generate these ground characteristic data in few minutes/hours or querying materialized(pre computed) ground characteristic data is very useful.

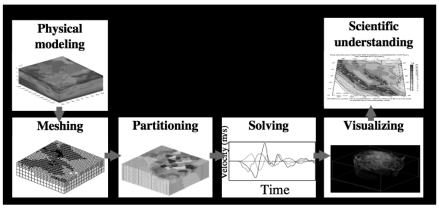

Fig. 1. Typical FE Simulation

II. Related Work

Schlosser et al [1] proposed a MapReduce system for materialized community ground models for large-scale earthquake simulations.

Euclid project [2] is developing the capability to perform physical simulations by computing directly on databases. All steps of the process, including mesh generation, solving, and visualization, work by directly querying and updating databases. With this database approach, the size of simulations is limited by disk capacity rather than by memory capacity, allowing scientists and engineers to run much larger simulations on their desktop systems than what is currently possible. The etree library has been widely used in the seismology community to store and represent models of the ground used in simulation. Southern California Earthquake Center (SCEC) has adopted etrees as the standard representation and API for ground models of Southern California.

Etree Library: Etree library [3] is part of a euclid Project, where research effort is aimed at developing the capability to perform physical simulations by operating on databases. For earthquake modeling [4] etree based method provide more performance. Etree library is a system for manipulating large octrees on disk from C programs. With this approach, the size of an octree is no longer limited by the size of the main memory, but instead, by the size of the disk space. The main memory serves as a disk cache to boost the performance. The etree library operates on etrees. To the external world, an etree is simply an ordinary UNIX file. Internally, an etree encapsulates the details of an octree and has a disk-resident index structure for fast octant data access.

III. Community Velocity Models

Community Velocity Models are program which take latitude, longitude and depth tuples as inputs and output set of ground characteristic at those locations. Latitude,longitude and depth values are coordinates of 3D space where we can get each point of the earth interior. These CVM are computer programs (Fortran code) and associated files that are downloaded, compiled, and run locally. The 3D CVM provides a unified reference model for the several areas of research that depend of the subsurface velocity structure in their analysis. [5], [6]

I SCEC Community Model: Community Velocity Models within SCEC (Southern California Earthquake Center)are computer programs and data files that provide information about earths material properties from the surface to depths below 100km: CVM-H [7] , CVM-S

II The Wasatch Front CVM

III New Madrid region CVM [8]

IV. Spatial Indexing

Scientific applications that query into very large multidimensional datasets are becoming more common. Efficiently querying that data, however, is a considerable challenge as the data is two-dimensional (or more). Extensive research has been carried out on multidimensional indexing structures, to enable efficient range queries and nearest neighbour searches. Common spatial index methods are, Spatial Index, Z-order(curve), Quad-tree, Octree, R^+ trees, R^* trees, Hilbert R-tree [9], [10].

A. Octree

An octree [3], [11]–[15] is a tree-based data structure for managing sparse 3-D data. Each internal node has exactly eight children. Octrees are fundamental in many computer graphics and simulation applications. In an octree 3D space is recursively subdivided until a reaching required voxel resolution is reached. As in Figure 2 Octrees can be drawn in different but equivalent ways,

I Domain representation: where the octree is drawn as an explicit decomposition of some rectangular domain.

II Tree representation

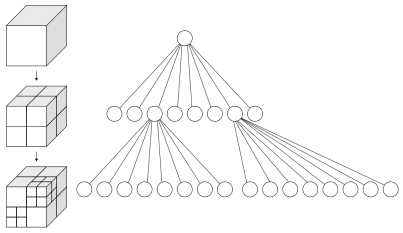

Fig. 2. Recursive subdivision of a cube into octants

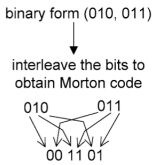

Fig. 3. Interleaving bits to obtain morton code in 2D

B. Z-order curve

In mathematical analysis and computer science, Z-order, Morton order or Morton code is a function which maps multidimensional data to one dimension while preserving locality of the data points [16]. Morton code is used for out-of-core algorithms that are designed to process data that is too large to fit into a computer's main memory at a time.

In order to compute the Morton code for a given XYZ 3D grid coordinate, it is necessary to do bit interleaving as in Figure 3

V. MapReduce

MapReduce [17], [18] is a distributed programming framework originally proposed by Google for the ease of development of web search applications on a large number of commodity CPUs. For large-scale data-intensive computing, the MapReduce paradigm has come up as a highly successful programming model. MapReduce framework provides two primitive operations. (1) A map function to process input key/value pairs and to generate intermediate key/value pairs, and (2) A reduce function to merge all intermediate pairs associated with the same key.

A. Hadoop

Apache Hadoop [17] is an open-source software framework for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. The core of Apache Hadoop consists of a storage part (Hadoop Distributed File System (HDFS)) and a processing part (MapReduce). This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

B. MapReduce on GPU

Both GPUs and MapReduce are designed to support vast data parallelism. While the performance of CPUs has stagnated , both the programmability and performance of the graphics processor (GPU) have increased dramatically in recent years, with a broad variety of applications demonstrating order-of-magnitude gains in both performance and price-performance. Some MapReduce Frameworks on GPU are Mars, Grex [19], DisMaRC: A Distributed Map Reduce Framework on CUDA, MITHRA MapReduce Framework on GPU [20], GPMR MapReduce Frame work on GPU and Panda.

C. Mars

Mars [21] is a MapReduce framework, on graphics processors (GPUs). Mars hides the programming complexity of the GPU behind the simple and familiar MapReduce interface and It exploits the massive thread parallelism within the GPU. The programmer is required to define a small set of APIs that are similar to those in the CPU-based MapReduce framework. The rest of implementation details such as the GPU runtime are hidden by Mars framework. Mars hides the programming complexity of the GPU behind the simple and familiar MapReduce interface. Figure 4 illustrates the work flow of Mars. Similar to the CPU-based MapReduce framework, Mars has two stages, Map and Reduce. Before starting each stage, Mars initializes thread configuration including the number of thread groups and the number of threads per thread group on the GPU.

VI. System Model

A. CVM-H model

CVM-H model developed by the SCEC community to extract material properties in the region is used in this experiment. Also HR model which has 250m x 250m x 100m resolution is selected.

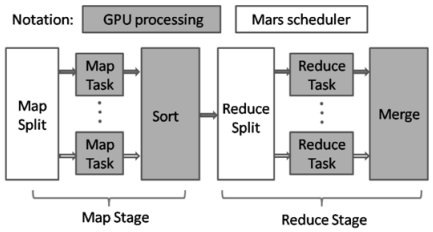

Fig. 4. The Work Flow of Mars on the GPU

B. How to find homogeneous regions

Morton code maps a point in a d-dimensional space to scalar integer. The mapping is computed by interleaving the bits of the binary representation of the fixed length coordinates of the point. Octants in Figure 5 can be represented as in Table I. The first 2 bits represent the level of octant and other bits represent the morton code. This is called locational code which represents any octant in octree structure in linear address space.

For example in an octree representation in level 3, it is possible to obtain maximum 8^3=512 voxels. So maximum coordinate will be (7,7,7) where 9 bits are needed to represent the coordinate. To represent the level 2 bits are needed.

Here neighbouring octants are generated by manipulating locational code. In the highest octree level last 3 bits are cleared. Then 6,9,12 .. least significant bits are cleared(put zero) by each level up. In this way sibling octants will have same intermediate keys.

For example for 8^4 records set has octree level 4 and to represent locational code totally 15 bits are needed. 12 bits(3*4) for morton code and 3 bits for level (level 4=100). To find sibling octants first the least significant 3 bits are cleared from locational code, then least 6 bits are cleared and so on for the each level up. In this case the sibling octants have same intermediate key(modified locational code) in each level .

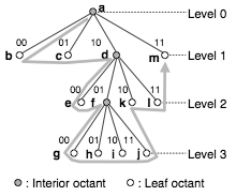

Fig. 5. octants tree representation

C. Thresholding

RST approach [22] (robust satellite technique) is a multi-temporal scheme of satellite data analysis that is already successfully used to monitor volcanoes at different geographic locations. Index of Change of the Environment (ALICE) to detect anomalies is defined as:

TABLE I
LOCATIONAL CODES

Octant	Locational Code (Level+Morton Code)
a	00_00000000
b	01_00000000
c	01_01000000
d	01_10000000
e	10_10000000
f	10_10010000
g	11_10010000
h	11_10010100
i	11_10011000
j	11_10011100
k	10_10100000
l	10_10110000
m	01_11000000

$$\otimes_v(x,y,t,T) = \frac{[V(x,y,t) - V_{REF}(x,y,T)}{\sigma_v(x,y,T)} \quad (1)$$

where x,y is location and t is time.The following equation was used to find threshold values in this experiment,

$$\otimes_v(x,y,z) = \frac{[V(x,y,z) - V_{mean}(x,y,z)}{\sigma_v(x,y,z)} \quad (2)$$

x,y,z : coordinates (longitude, latitude and depth values),
V(x,y,z) : velocity values in specific coordinate V_p ,V_s or ρ,
V_{mean} : mean value of V_p ,V_s or ρ,
σ_v :Standard deviation of V_p ,V_s or ρ ,
V_p : Primary Velocity,
V_s : Secondary Velocity, ρ : Dencity
ALICE value factor is set to 1 for generating materialised data.

D. Methodology

As Figure 6 indicates in the Map phase, the digital elevation model is divided in to small sub regions(voxels). So each node will query CVM model for required Vp,Vs,Rho value. Input for this is longitude, latitude and depth values. In the Reduce phase Vp,Vs,Rho values are compared to threshold value and homogeneous neighbours(siblings) are coalesced in each sub region. In the final output homogeneous regions are coalesced together. These pre computed values can be used instead of considering the whole region for further simulations, leading to higher performance. This system is implemented using both Hadoop cluster and Mars framework.

E. MapReduce Architecture

As in Figure 6, Vp,Vs and ρ values are generated for each X,Y,Z (longitude, latitude and depth) values invoking the CVM-H model. Each Map task considers a sub region and generate data sets for that sub region and reducer is a zero reducer. Zero reducer means reduce step will be skipped and mapper output will be the final output.
Figure 7 represents the system architecture for checking homogeneous regions. Here, each Map task takes 64MB or 128MB blocks and for each record it extracts locational code as the key and other data as values. Map task clear(put zero for least

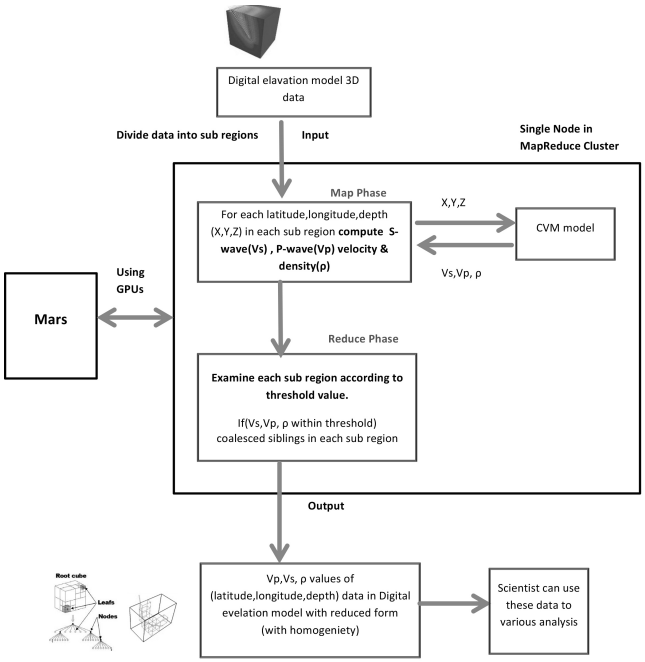

Fig. 6. System architecture

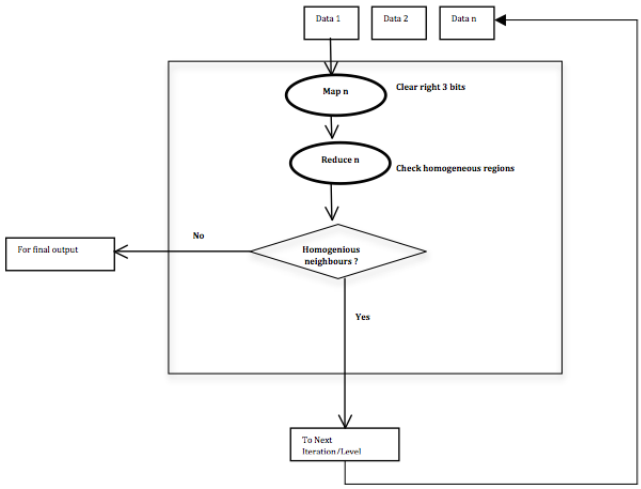

Fig. 7. MapReduce Phase in a Single Node in n^{th} Iteration/Level

significant bits) the last 3 bits in the locational code and send the cleared locational code as key and other data as values. So in Reducer phase, it takes modified key(modified locational code) as *key value* and the record set with same modified key as *value list* to check the homogeneity in the neighbouring octants. If Vp,Vs and ρ values of all records with same key are homogeneous, a single value will be returned instead of all record set for the next Iteration . If the values are not homogeneous, all the records will be returned as final output. This process is repeated till there are no more homogeneous regions in the data set. Here n level can have maximum n iterations or less.

VII. Results

Table VII represents the maximum voxels/records size that can be in each octree levels after invoking the CVM-H models. These data(records) represents longitude,latitude, depth values and Vp,Vs, and ρ values.

TABLE II
MAXIMUM VOXELS IN EACH OCTREE LEVEL

Octree Level	Maximum Voxels
Level 1	8^1=8
Level 2	8^2=64
Level 3	8^3=512
Level 4	8^4=4096
Level 5	8^5=32768
Level 6	8^6=262144
Level 7	8^7=2097152
Level 8	8^8=16777216
Level 9	8^9=134217728
Level 10	8^{10}=1073741824

The results from Octree level 9 is shown in Table III, initial data includes 134217728 records and after analysing for homogenous records in neighbouring octants, it returns final output as 99298212 records. So the initial data are reduced by 26.1% factor. These threshold parameters for Vp,Vs and rho values are originally specified by ground modelling experts. Results for different data sets are shown in Table IV

Figure 8 represents, initial data records (8^4 records) with X,Y,Z(longitude,latitude,depth) values against density in level 4 and Figure 9 represents reduced final output records.

A. Execution Time

1) Execution Time with Hadoop: Table V represents execution time for 17 million and 134 million records with 2 and 3 node cluster. Here, three computers are used with following configurations:

TABLE III
LEVEL 9(134 MILLION RECORDS): DATA IN EACH ITERATION AFTER CHECKING THRESHOLDING

		Records	Compression
Initial Data Records		134217728	
After Iteration 1	To Next Iteration	4783234	28.51%
	For Final Output	95951856	71.48%
After Iteration 2	To Next Iteration	319019	34.70%
	For Final Output	3123434	65.29%
After Iteration 3	To Next Iteration	23384	34.66%
	For Final Output	208437	65.34%
After Iteration 4	To Next Iteration	3469	48.83%
	For Final Output	11964	51.16%
After Iteration 5	To Next Iteration	473	37.53%
	For Final Output	2167	62.47%
After Iteration 6	To Next Iteration	62	34.25%
	For Final Output	311	65.75%
After Iteration 7	To Next Iteration	12	41.93%
	For Final Output	36	58.06%
After Iteration 8	To Next Iteration	3	66.67%
	For Final Output	4	33.33%
After Iteration 9	To Next Iteration	0	
	For Final Output	3	
Initial Records: 134217728 Total Records For analyzis: 99298212 Percentage : 73.9% Reduced data : 26.1%			

TABLE IV
COMPRESSION OF THE CVMH MODEL DATA(250X 250X250)M RESOLUTION OCTANT(CUBE) WITH ALICE=1

Octree Level	Sampled Octants (Initial Records)	Coalesced Octants (Final output)	Compression
Level 5	32768	3967	12.10%
Level 6	262144	150808	57.52%
Level 7	2097152	1308378	62.39%
Level 8	16777216	10467876	62.39%
Level 9	134217728	99298212	73.98%

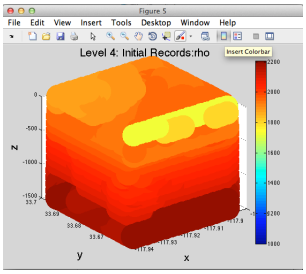

Fig. 8. Initial Input data for Rho(ρ) values: Level 4 (8^4 records)

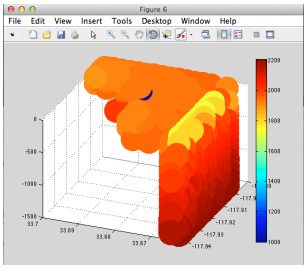

Fig. 9. Final output data for Rho(ρ) values: Level 4 (8^4 records)

TABLE V
OCTREE LEVEL 9 : EXECUTION TIME IN CLUSTER NODES WITH HADOOP [(CPU MHZ: 2999.622 ,CPU(S): 2,RAM: 8GB ,INTEL(R) XEON(R) CPU E3110 @ 3.00GHZ - 2NODES) & (CPU MHZ: 1200.000 ,CPU(S): 1,RAM: 1GB -1 NODE)]

	With 1 node	With 2 Nodes cluster	With 3 Nodes cluster
Level 9 (134million records)	2hrs,12min,25sec (7945 seconds)	1hrs,35mins,42sec (5742 seconds)	1hrs,15mins,48sec (3948 seconds)
Level 8 (17million records)	15 mins 34 sec (934 seconds)	10 mins 10 sec (610 seconds)	5 mins, 20 sec (320 seconds)

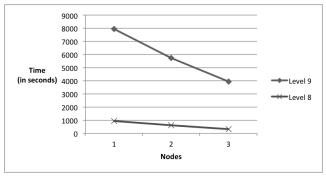

Fig. 10. Time taken to compute homogeneous regions in octree level 8 & 9 with 3 nodes using Hadoop

2) Execution Time with Mars : Table VI represents time taken for each octree level with Mars. With above system configurations Tesla C2075 GPU, only upto 8 levels can be tested (total data processing without dividing into sub regions). This is because it is not possible to upload map input data onto device memory, since device memory size is

TABLE VI
TIME TAKEN FOR MAPREDUCE PROCESS USING MARS WITH CUDA (TESLA C2075 , GPU RAM= 5GB,CPU RAM=4GB,2.93GHZ INTEL CORE I3,CUDA RUNTIME VERSION :6.5),UBUNTU 14.04

Octree Level	Sampled Octants		Mars (ms)
4	4096	File reading time	67.369
		Total Time	97.162
		Process time without file reading	29.793
5	32768	File reading time	151.223
		Total Time	231.406
		Process time without file reading	80.183
6	262144	File reading time	728.256
		Total Time	2011.891
		Process time without file reading	1283.635
7	2097152	File reading time	6301.407
		Total Time	13656.491
		Process time without file reading	7355.084
8	2097152	File reading time	153545.967
		Total Time	219692.911
		Process time without file reading	66146.944

insufficient to allocate data .

VIII. CONCLUSIONS

When the data set is large, it have to be distributed across hundreds or thousands of machines in order to finish processing in a reasonable amount of time. Over the past few years large clusters comprising 1000s of commodity CPUs, running Hadoop MapReduce, have powered the analytical processing of Big data involving hundreds of terabytes of information. Now with CUDA GPUs have created the potential for a new disruptive technology for Big data analytics based on the use of much smaller hybrid CPU-GPU clusters.

Two aspects are focused in this experiment. Those are, reduction of input data for the final analysis and to do this reduction effectively in terms of time. Hadoop cluster and GPU techniques are used as different techniques in this experiment to improve performance. As in Table IV data are reduced considering homogeneous regions.

In this experiment hadoop release version 0.22.0 is used. These results depend on various factors including nodes configurations, number of Maps, Number of Reducers, Input format (TextInputFormat, SequenceFileInputFormat etc..), Number of input files / paths, File types.

Comparing run time in Figure 10, 1.5X speedup can be seen when each node is increased in the cluster(scale out) .

Comparing with Hadoop results for 17 million records(level 8), it took 934 seconds with 1 Node & 610 seconds with 2 Nodes & 320 seconds with 3 Nodes in Hadoop Cluster. With Mars processing time is 66 seconds. It is possible to see 5X-10X speedup with 3 Node and 2 Node clusters. For the data that does not fit GPU memory, data is divided into parts so as to fit GPU memory.

IX. FUTURE WORKS

As future works Hadoop map reduce framework can be improved using Spark. Spark is a fast and general compute engine for Hadoop data and it provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation. Spark run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Also an algorithm can be implemented using thrust. Thrust is a powerful library of parallel algorithms and data structures. Thrust provides a flexible, high-level interface for GPU programming that greatly enhances developer productivity. Using Thrust, C++ developers can write just a few lines of code to perform GPU-accelerated sort, scan, transform, and reduction operations orders of magnitude faster than the latest multi-core CPUs.

REFERENCES

[1] S. W. Schlosser, M. P. Ryan, R. Taborda-Rios, J. López, D. R. O'Hallaron, and J. Bielak, "Materialized community ground models for large-scale earthquake simulation." in *SC*. IEEE/ACM, November 2008.

[2] "The euclid project," http://www.cs.cmu.edu/ euclid/, Aprial 2006.

[3] T. Tu, D. R. O'Hallaron, and J. Lopez, "The etree library: A system for manipulating large octrees on disk," *Engineering with Computers*, vol. 20, no. 2, pp. 117–128, 2004.

[4] V. Akcelik, J. Bielak, G. Biros, I. Epanomeritakis, A. Fernandez, O. Ghattas, E. J. Kim, J. Lopez, D. O'Hallaron, T. Tu, and J. Urbanic, "High resolution forward and inverse earthquake modeling on terasacale computers," in *Proceedings of SC2003*, Phoenix, AZ, nov 2003.

[5] H. Magistrale, S. Day, R. W. Clayton, and R. Graves, "The scec southern california reference three-dimensional seismic velocity model version 2," *Bulletin of the Seismological Society of America*, vol. 90, pp. S65–S76, dec 2000.

[6] "3d velocity model for southern california," http://www.scecdc.scec.org/research-tools/3d-velocity.html, 2012.

[7] "Cvm-h," http://scec.usc.edu/scecpedia/CVM-H, 2015.

[8] "Three-dimensional community velocity models," http://www.virtualuppermantle.info/3DCommunityVelocityModels.htm.

[9] "Hilbert r-tree," http://en.wikipedia.org/wiki/Hilbert_R-tree, 2015.

[10] H. J. Bierens, "Introduction to hilbert spaces," 2007.

[11] I. Gargantini, "An effective way to represent quadtrees," *Commun. ACM*, vol. 25, no. 0001-0782, pp. 905–910, dec 1982.

[12] S. Laine and T. Karras, "Efficient sparse voxel octrees – analysis, extensions, and implementation," NVIDIA Corporation, NVIDIA Technical Report NVR-2010-001, feb 2010.

[13] A. Knoll, Ed., *A Short Survey of Octree Volume Rendering Techniques*. Proceedings of 1st IRTG Workshop, 2006.

[14] J. Bielak, O. Ghattas, and E.-J. Kim, "Parallel octree-based finite element method for large-scale earthquake ground motion simulation," *Tech Science Press*, 2005.

[15] J. Veenstra and N. Ahuja, "Line drawings of octree-represented objects," *ACM Trans. Graph.*, vol. 7, no. 1, pp. 61–75, jan 1988.

[16] "Z-order curve," http://en.wikipedia.org/wiki/Z-order_curve, 2014.

[17] "Apache hadoop," http://hadoop.apache.org/, 2015.

[18] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, january 2008.

[19] C. Basaran and K. don Kang, "Grex: An efficient mapreduce framework for graphics processing units."

[20] R. Farivar, A. Verma, E. Chan, and R. Campbell, "Mithra: Multiple data independent tasks on a heterogeneous resource architecture," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, 2009.

[21] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a mapreduce framework on graphics processors," in *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. New York, NY, USA: ACM, 2008, pp. 260–269.

[22] C. F. F. Marchese, G. M. N. Genzano, N. Pergola, and V. Tramutoli, "Assessment and improvement of a robust satellite technique (rst) for thermal monitoring of volcanoes," Remote Sensing of Environment, 2011.