Large Data and Computation in a Hazard Map Workflow Using Hadoop and Neteeza Architectures

Shivaswamy Rohit
Dept. of Mechanical and Aerospace Engineering
University at Buffalo
New York, USA
shivaswa@buffalo.edu

Abani K. Patra
Dept. of Mechanical and Aerospace Engineering
University at Buffalo
New York, USA
abani@buffalo.edu

Vipin Chaudhary
Dept of Computer Science and Engineering
University at Buffalo
New York, USA
vipin@buffalo.edu

ABSTRACT

Uncertainty Quantification(UQ) using simulation ensembles leads to twin challenges of managing large amount of data and performing cpu intensive computing. While algorithmic innovations using surrogates, localization and parallelization can make the problem feasible one still has very large data and compute tasks. Such integration of large data analytics and computationally expensive tasks is increasingly common. We present here an approach to solving this problem by using a mix of hardware and a workflow that maps tasks to appropriate hardware. We experiment with two computing environments – the first is an integration of a Netezza data warehouse appliance and a high performance cluster and the second a hadoop based environment. Our approach is based on segregating the data intensive and compute intensive tasks and assigning the right architecture to each. We present here the computing models and the new schemes in the context of generating probabilistic hazard maps using ensemble runs of the volcanic debris avalanche simulator TITAN2D and UQ methodology.

Keywords

Uncertainty Quantification, Data Warehouse Appliance, Hadoop

1. INTRODUCTION

Uncertainty Quantification work flows pose significant computational and data handling challenges in most HPC platforms since they are largely not designed for such workflows. We present here an approach designed to surmount issues arising from recent work [3, 14] that integrates the outcomes of large full field flow simulations into a probabilistic hazard map – often required in near real time. Early versions of related work involving only the data warehouse appliance will appear in [12] A hazard map, as stated here, is a predictive map for a region which provides a probabilistic measure of a hazard (e.g. geophysical flow reaching certain depths that can be considered hazardous/critical). Simple uncertainty quantification using a Monte Carlo approach for generating such hazard maps will require $O(10^6)$ such simulations – beyond current computational capabilities. Instead for the generation of hazard map we employ a statistical model termed emulator which acts as a fast surrogate for the simulator. Our work here is focussed on generating hazard maps for geophysical flows of volcanic origin using ensemble runs of the Titan2D simulator [3] (see fig. 1 for a sample simulation). Since each flow simulation generates upwards of 2GB of data, a full ensemble with 2048 simulations generates almost 600GB of data which we have to then use to construct emulators – Bayes Linear Models (BLM) [3] in our case –which involves the inversion of a covariance matrix of a size corresponding to the ordinality of the data. In previous work [3, 14], we proposed methodology based on localization and parallelization to make the problem tractable. Despite these simplifications, the remaining problem is still interspersed with both data and compute intensive tasks presenting a daunting task. The computational difficulties include managing and accessing select entities from the large data and of processing it using compute intensive operations(matrix inversion). Our approach to addressing these difficulties is to decouple the data mining and compututationally intensive tasks through carefully crafted workflows. While one workflow incorporates a massively parallel architecture, namely the Netezza database, an alternate workflow employs a the popular open source version of the Map-Reduce model, namely Hadoop, both in conjunction with a high performance cluster. Problems involved in data extraction, movement over the network, and replication are addressed. The proposed computational methodologies also allow us to greatly improve the resolution of the developed hazard maps by using tree based localization (rather than the simpler tessellation based localization used earlier [3]) allowing access to more data in the inference process and hence developing a more accurate localization of the covariance used in the hazard map construction.

For very large data there is no debate that computation must be moved close to where the data resides which is very different from traditional scientific computing which involves moving data to the computing. The standard practice in scientific computing requires the data to be moved from the storage area network (SAN) to the compute environment for analytics. Data extraction, movement over the network, and replication constitute one of the most time consuming phases of many large data analysis problems [4]. In-database

DISCS-2013 November 18, 2013, Denver, CO, USA

http://dx.doi.org/10.1145/2534645.2534648.

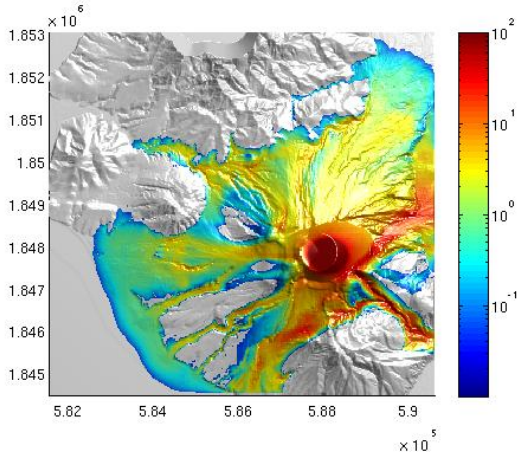

Figure 1: Example flow simulation using the TITAN2D toolkit. Maximum flow depth (pileheight) using Titan2D simulation. Simulation consumes hours of computer time and each simulation produces large volume of flow data.

computing pushes the processing into the database management system thus eliminating the need for data extraction/sampling and network data replication greatly improving scalability and performance of analytic computations. It provides the ability to efficiently analyze all available structured and unstructured data in their entirety and perform complex computations on the entire data as opposed to a limited/sampled set of data by allowing to execute analytics within the database management system, where the data resides, as opposed to a separate compute system. In addition to providing scalability and improved analytics performance, in-db approach (underlying db platform) ensures better data sharing/collaboration, security, availability, and governance [9]. Although in-database processing has been used widely in industry [5, 13, 8, 11, 6] for customer relationship management, its use in the scientific and engineering research has been limited.

The Map-Reduce model has been gaining lot of attention recently, and is being extensively used in applications requiring large volumes of data. Hadoop API based on Map-Reduce model provides a high level of parallelism and offers a level of abstraction that significantly simplifies programming with it. We present here two versions of integrated workflow mapping data and compute intensive operations on different architectures to optimize throughput. The Netezza server is essentially a large array of disks with FPGA in the disk architecture for in-slice filtering of data sets. In contrast Hadoop is far less expensive alterantive which not just offers an efficient job scheduler but also internally sorts the data. Significant changes to the underlying technique for selection of neighbors makes the process of the BLM emulator construction faster and more flexible. This flexibility makes it possible to adapt the data used for emulator construction without requiring any additional resources or time which would not have been easily possible with the previous computing model. In this paper we present details of execution and the challenges faced in implementing workflows along with a brief comparison of results.

2. HAZARD MAP GENERATION – DATA MANAGEMENT AND COMPUTING

To understand the complexity of the problem we present an outline of steps involved in a usual process of Hazard map generation.

Step 0: The first step is to run the simulator at well chosen inputs. The input parameters are sampled using a simple space filling design like Latin Hypercube to obtain 2048 sets of input. Multiprocessor Titan2D simulations of these inputs and post processing results in 2 gigabytes(GB) of flowdata per sample in the form of flow height records.

Step 1: The construction of the hazard map requires us to sample a tensor product of the input parameters and 2 space dimensions which results in as many as 10^8 data points. Emulator construction on this very large set is unaffordable so a simple decimation strategy is used to create a smaller set on which we construct the emulator. This downsampling is introduced to reduce their number to the order of 10^6. Furthermore, resamples from the generated emulator surface (for the final probability calculation) are also required to be generated and can be as many as 10^{10} in number.

Step 2: The size of the downsized data set makes it computationally impossible to fit a single emulator using all the data at once, which warrants the need for piecewise emulator obtained by localizing the covariance. The neighbor search used in identifying the regions for localization is thus an important pre-requisite for the functioning of the emulator and requires both sample and resamples to be searched from among the samples for neighbors. Both neighbor search and downsampling are highly data intensive tasks which require little computation but scanning of large datasets.

Step 3: Using neighborhood data, emulator is constructed about the sample points through an iterative process. The functioning of emulator can be understood from the following equations:

$$E(s(y)|s(x)) = g(y)\beta + r(y)^T R^{-1}\epsilon \tag{1a}$$

$$Var[s(y)|s(x)] = \sigma^2(1 - r(y)^T R^{-1} r(y)) \tag{1b}$$

$$r_i(y) = exp\left(-\sum_{n=1}^{N_{dim}} \theta_n (y_n - x_{i,n})^2\right) \tag{2}$$

g(y) being the matrix of basis functions evaluated at the resample points and β being the vector of least square co-efficients. R is the matrix of the correlation functions at x such that $R_{i,j} = r_i(x_j) = r_j(y_i)$ and σ is the variance.

$s(x) = \beta G(x) + \hat{\epsilon}$ is the response function.

ϵ = s(x) - G(x)β the true error evaluated at the sample points. θ_n is the vector of hyper-parameters or roughness parameters and Ndim is the number of dimensions associated with the data set.

At each iteration β and R^{-1} are computed using updated values of hyper-parameters (θ). Mean and variance are then evaluated for the resamples and adjusted

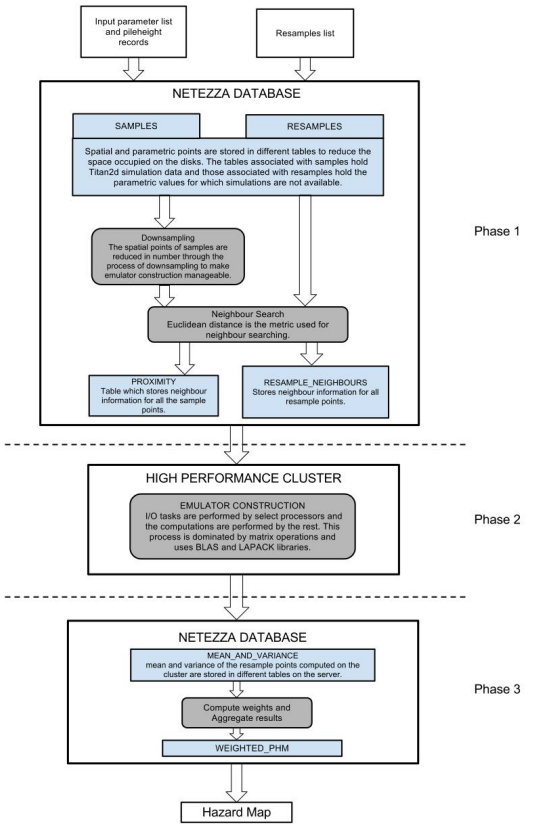

Figure 2: Illustration of the integrated workflow using Netezza architecture.

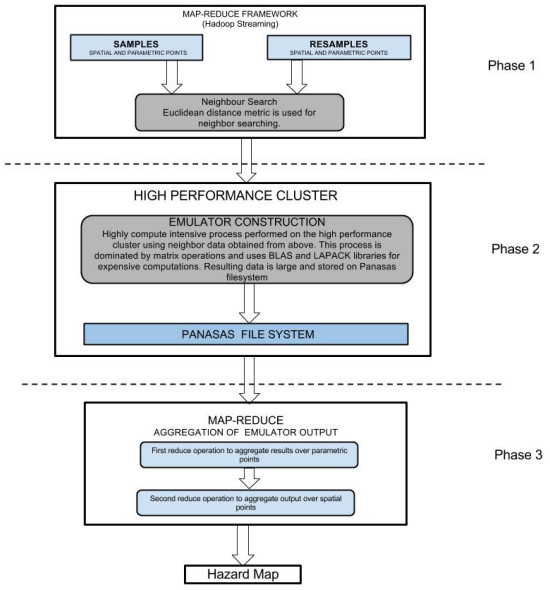

Figure 3: Illustration of the integrated workflow with Map-Reduce framework

using bayes linear equations. Typically, a hazard map requires constructing a few million emulators. The emulator construction dominated by $O(n^3)$ matrix operations is a highly compute intensive process but also embarrassingly parallel.

Step 4: In the last stage of hazard map construction, emulator output is aggregated using barycentric weights. This involves scanning the dataset, computing the euclidean distance of the samples that influence a resample point, evaluating their weights and assembling the results.

3. INTEGRATED WORKFLOW FOR DATA AND COMPUTE INTENSIVE TASKS

There are three dominant phases of Hazard Map generation namely: 1) Downsampling and neighbor searching, 2) Emulator construction, and, 3)Aggregation of resulting data. Our computational model based on a *divide and conquer* strategy, segregates these phases and performs them on either Netezza server or the high performance cluster depending on the computational requirements.

3.1 Downsampling and neighbor search

Both downsampling and neighbor search are ideally suited for distributed systems. For multidimensional dataset, such as ours, operations like neighbor search are afflicted with the *curse of dimensionality*. Several tree and clustering based methods have been proposed but most converge to sequential search for higher dimensions and/or are difficult to implement. Neighbor search operation on a distributed environment like Netezza server or on a cluster through Map-Reduce implementation like Hadoop allows for a much simpler algorithm and adapts to large datasets. Downsampling and Neighbor search on the Netezza system were easily accomplished because of its massively parallel architecture. Netezza's high performance stems from filtering data close to disk and from its MPP architecture. Since a significant amount of data movement is curtailed through the use of FPGA filtering, we abstained from using complex algorithms in favor of brute force techniques. All Netezza based implementations were in plain SQL which ensured parallelism and high performance.

We also tested same operations on the high performance cluster using python scripts, relying on Hadoop Streaming API for task distribution and scheduling. In a distributed environment the underlying algorithm for neighbor search remains essentially the same as the one used on Netezza server. The mapper computes the distances between two sets of data (X and Y) and prints out the result as key-value pair, key being the indices of dataset X and value being the indices of dataset Y and the euclidean distance. In the absence of a customized partitioner same keys are guaranteed to be dispatched to the same reduce task. The reduce operation merely involves printing out the indices of set Y against the keys obtained from set X. Hadoop based implementation as in the case with Netezza was simple and concise.

In both our implementations euclidean distance was the metric for neighbor search. The parametric neighbor search was performed independent of spatial neighbor search by separately treating the two sets of dimensions. Though the brute force method of neighbor search has a complexity of

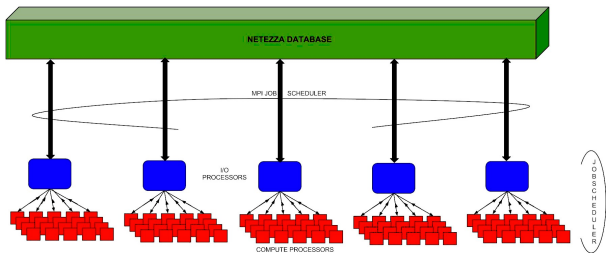

Figure 4: Schematic representation of integration of Netezza with high performance cluster

$O(n^2)$, it is well suited for distributed environment owing to good scalability.

3.2 Emulator Construction

Unlike the data mining operations up to this point, the emulator construction is a computationally expensive process dominated by matrix operations. Though emulator conctruction is an embarrassingly parallel process and Netezza server boasts of massively parallel architecture, it was, for variety of reasons as we describe below, appropriate to siphon data off the database to an architecture that could meet the emulator's computational demands. First, complex algorithms can not be easily translated into a declarative language like SQL. Second, the number of blades available in our systems put a limit on the available processors. Also the explicit use of high performance libraries like *blas* and *lapack* was not possible. Another subtle aspect about emulator which advocates against its construction on the database is that it requires small chunks of data, more precisely the neighborhood data. Netezza is more appropriate for scanning large datasets. We present here the steps taken and the challenges faced in integrating server to the cluster.

The high performance cluster and the Netezza system were connected using Netezza's ODBC driver. Data transfer across the two frameworks was possible largely through the use of named pipes. On linux systems a named pipe is a type of file - FIFO file (FIFO being the acronym for First In First Out). A named pipe does not store or buffer data and hence does not occupy storage space on the filesystem. Loading data from cluster to Netezza was achieved through the *nzload* feature. The *nzload* command is a SQL CLI client application which in conjunction with named pipes that makes it possible to load data to Netezza server from any remote client[1].

Emulator construction requires that each processor request the server for its share of data. Large number of concurrent I/O requests by nodes on the grid can result in I/O bottlenecks and processors starving for data. Such problems have been addressed in the past and techniques such as collective I/O[10, 7] and I/O forwarding[2] have been proposed. I/O forwarding mitigates this problem by passing the I/O requests of the all the nodes to only a select number of nodes called as I/O nodes. We adopted this method for our computing model with some modifications.

A few processors on the high performance cluster are identified as the I/O processors and the rest as compute processors. All the processors are assigned a group, with each group having one I/O processor and several compute processors. Each group operates independent of the other with I/O processors responsible for performing all I/O operations on behalf of the compute processors of their respective groups. I/O processors alone communicate with the Netezza server and any transfer of data occurs through named pipes. Multiple small requests are avoided and data is invariably transferred in bulk. I/O processors extract data from Netezza, disseminate it across the compute processors and deliver the data gathered from the compute processors back to the Netezza database. I/O processors of every group draw the neighborhood data of all the downsampled points corresponding to the pileheight simulation number assigned to them from sever, and store it in their data structures. Each compute processor based on the data received from the I/O processor of its group builds an emulator. The mean and variance data of the resample points, computed at the end is sent over the network to I/O processor.

The operations on the cluster are parallelized using MPI (Message Passing Interface). Data is transferred between the I/O and compute processors over the network using MPI protocols. An MPI job scheduler allows compute processors to notify the I/O processors about the status of their completion. When a compute processor finishes constructing emulator about a sample point, it prompts the I/O processor. The I/O processor responds by sending neighborhood data for the next point. The compute processors do not communicate among themselves and are self sufficient with the data received from the I/O processor. The mean and variance information received from the compute processors is allowed to accumulate with I/O processors and dispatched to the Netezza server at regular intervals. Another layer of MPI job-scheduler is responsible for co-ordination between the various groups of processors. A lone processor which holds the metadata maintains communication with the groups and assigns each group a simulation number to work on. Additionally it also prompts Netezza database to "generate statistics" after each load session.

Though we succeeded in integrating server with the cluster and in transferring data over the network, this implementation has apparent shortcomings. Firstly offloading compute intensive tasks from server to cluster defeats the objective of minimizing large data movement. Secondly only a small number of ports can be kept working between the two, to transfer data. Thus, while Netezza can easily house much more data, extracting it to more nodes on cluster is not feasible. It is thus clear that compute intensive tasks like emulator construction expose the vulnerabilities of even a specialized hardware like Netezza database. For the above mentioned reasons and also because of the complexity of implementation, we attempted to test the above model with Hadoop in place of Netezza. Hadoop provides a convenient alternative because it does not require data to be moved into or out of the cluster. The downsampled points stored on files on the Hadoop distributed file system were used as input to the mapper. The mapper itself was simply a python wrapper around the existing code to compute the emulator with output in terms of key value pairs. Hadoop is designed to take in large amount of input and distribute the tasks by spawning mappers on each of the slave nodes. In our application at less than 2GB the input data was very insignificant in size but the generated output was expected to be large. Also the emulator construction is highly cpu intensive. We observed that no more than 2 map tasks could be spawned

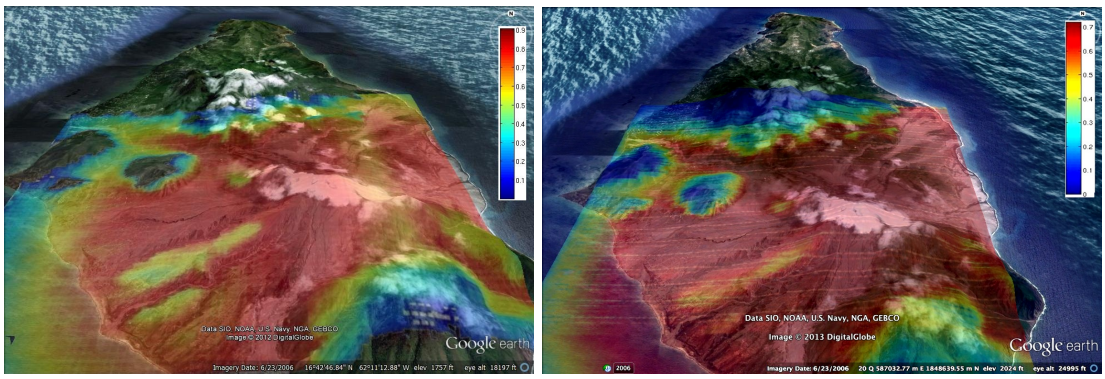

Figure 5: Figure shows the probabilistic hazard maps for the island of Montserrat in the Caribbean for flow exceeding 0.2m of depth. One on left was generated using Netezza database based model and the one on right using Hadoop API

on each node regardless of the number of cores on them. On certain nodes, with say 12 cores, having only 2 map tasks running was a severe under utilization of the resources. Also, previous experience dictates that at least 500 processors be used for emulator construction to finish the hazard map in reasonable time. Under these circumstances we found it to be more appropriate to perform emulator construction independent of Hadoop environment. The resulting output was stored as key-value pairs on ASCII files.

3.3 Aggregation Of Results

The first two moments are to be computed for every resample point, which as mentioned earlier, could be as many as 10^{10} in number. Furthermore most of these points occur in the neighborhood of more than one sample point. In the previous work [3], the weights were pre-computed owing to the fixed number of neighbors which made immediate aggregation of data possible. A distance based method of neighbor search is computationally more expensive. The

Netezza database offers aggregate functions and can easily house massive data. The "group by" feature of any database is primarily aimed at operations like reduction and aggregation. Besides, computing the weights required repeated scanning of large parts of the data sets. Netezza is well suited for such aggregation of data. The mean and variance computed by the compute processors is directed to the Netezza server through the I/O processors. The cluster and the Netezza server are connected by 10Gb network at the Center for Computational Research, Buffalo. Using Netezza's nzload command and with multiple named pipes, data is concurrently moved from I/O processors to the respective tables on Netezza. The mean and variance data is massive and runs into billions of rows on the server. It is important that the tables storing such data are already distributed on columns on which they are aggregated. This ensures prior partitioning of data about those columns which significantly reduces the computation time. During the implementation of certain aggregation queries we found, that by having the table (with approximately 2 billion rows) distributed on the columns to be aggregated on, the total time of operation was reduced from approximately 30 minutes to under 3 minutes.

A Hadoop based aggregation of the emulator output was performed by chaining of map-reduce operations. As the term suggests, aggregation, did not require any specific map operation. As the output was required to be aggregated over multiple set of keys, aggregation was split into two separate reduce operations. Both the tasks involved only stdin and stdout operation in the mapper. The reducer was responsible for assembling the weighted mean for records with same keys. The replication factor was fixed at 1 for Hadoop implementations.

4. RESULTS

We put both computing models was put to test with the task of creating the hazard map for the volcano on Montserrat island. 2048 sets of input parameters were generated using Latin Hypercube sampling and Titan2D simulations of these inputs were performed. The probabilistic hazard map shown on the left in figure 5 was generated, in 6 hours time using Netezza based workflow. Downsampling and neighbor search operations were performed in under 30 minutes of time. The computation time on the cluster was reduced to a little more than 2.5 hours using 512 processors on 43 nodes of 12 cores each and by keeping 16 connections open between Netezza and the cluster. The 512 processors were divided into 16 groups with each group having 1 I/O processor and at most 32 compute processors. A maximum limit of 120 was enforced on the size of the covariance matrices. Also the radius of search was reduced from 100 metres for spatial dimensions and from 0.2 for parametric dimensions(scaled). The final aggregation was completed in 2.5 hours, and was performed entirely on Netezza server.

The probabilistic map on the right in figure 5 was generated with Hadoop based model. The neighbor search operations were completed in 20 minutes of wall time. Emulator construction was performed individually by separately dispatching tasks on the cluster. This was completed in approximately 5 hours of time and resulted in 800GB of output data. Hadoop was extensively used for aggregation of results through two separate reduce processes. 200 reduce tasks were spawned over 30 nodes (1 master and 29 slave nodes) for the first reduce operation and the computations ended successfully in approximately 8 hours of wall time reducing

800GB of data to 190GB of output. The second reduce operation was completed with 100 reduce tasks spread over on 20 nodes (1 master and 19 slave) in 1.5 hours of time.

5. CONCLUSION

Through this paper we aim to address the simultaneous computational and data challenges in a Uncertainty quantification process through two different approaches - one hardware based and other using a more popular software tool on the traditional cluster. We successfully architected and implemented two functional workflows for the application of generating hazard maps for geophysical flows. Netezza based workflow offered a faster implementation for a moderate sized data such as ours in comparison to Hadoop based workflow. Data mining tasks required minimal work and its ability to perform fast analytics provided quick insight about the data. On the other hand, Netezza is an expensive hardware, with its SPUs (Snippet Processing Units) prone to wearing out. Integrating Netezza with the cluster presented numerous hurdles and the resulting implementation was not fault tolerant. Furthermore, restriction on the number of 'working' ports on Netezza makes the workflow less scalable. Hadoop in contrast is a much cheaper alternative, offering reliable and robust job scheduler and fault tolerance. It made our implementation of the workflow programatically easy and obviated the need to move data from the cluster. It is also more easily scalable. However, modeling, a compute intensive task, such as emulator construction, as a mapper posed a severe problem. Using Hadoop API for tasks deficient in input data but requiring more cpu cycles clearly suggested that it can't be used as an alternative job scheduler for compute intensive tasks.

6. REFERENCES

[1] Netezza data loading guide, October 2010.

[2] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. Scalable i/o forwarding framework for high-performance computing systems. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE, 2009.

[3] K.R. Dalbey. *Predictive simulation and model based hazard maps of geophysical mass flows*. State University of New York at Buffalo, 2009.

[4] D.Stodder. TWDI monograph series : Seven keys to high performance data management for advanced analytics, 2011. TWDI Research (1105 Media Inc.), 20.

[5] The Economist-Economist intelligent unit. Big data - harnessing a game-changing asset (white paper), 2011. The Economist Intelligent Unit. 1-32.

[6] J. Kobielus, B. Evelson, R. Karel, and C. Coit. In-database analyics : The heart of the predictive enterprise, 2009. Forrester Research Inc., 1-22.

[7] D. Kotz. Disk-directed i/o for mimd multiprocessors. *ACM Transactions on Computer Systems (TOCS)*, 15(1):41–74, 1997.

[8] J. Lovett. Beyond surface-level social media - using analytical assets for generation-next marketing (white paper)., 2011. Web Analytics Demystified. 1-19.

[9] N.Raden. Advanced in-database analytics done right hired brains research, 2010. (Hired Brains, Inc.) 1-17.

[10] R.A. Oldfield, L. Ward, R. Riesen, A.B. Maccabe, P. Widener, and T. Kordenbrock. Lightweight i/o for scientific applications. In *Cluster Computing, 2006 IEEE International Conference on*, pages 1–11. IEEE, 2006.

[11] SaS and Teradata Inc. Unlock the business value of enterprise data with in-database analytics (white paper)., 2009. SAS Institute Inc. and Teradata Corporation. 1-12.

[12] R. Shivaswamy, A. K. Patra, and V. Chaudhary. *Int. J. Computer Mathematics*, 2013. in press.

[13] Tableau software. Big data: Powering the next industrial revolution. tableau software, 2012.

[14] E. R. Stefanescu, M. Bursik, G. Cordoba, K. Dalbey, M. D. Jones, A. K. Patra, D. C. Pieri, E. B. Pitman, and M. F. Sheridan. Digital elevation model uncertainty and hazard analysis using a geophysical flow model. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 468(2142):1543–1563, 2012.