LARGE DATA & LARGE COMPUTING IN A HAZARD ANALYSIS WORKFLOW

Rohit Shivaswami, **Abani K. Patra**, Vipin Chaudhari

University at Buffalo, Buffalo, NY14260

abani@buffalo.edu

.

DISCS @ SC13

Denver, CO, November 18, 2013

November 18, 2013

The Problem

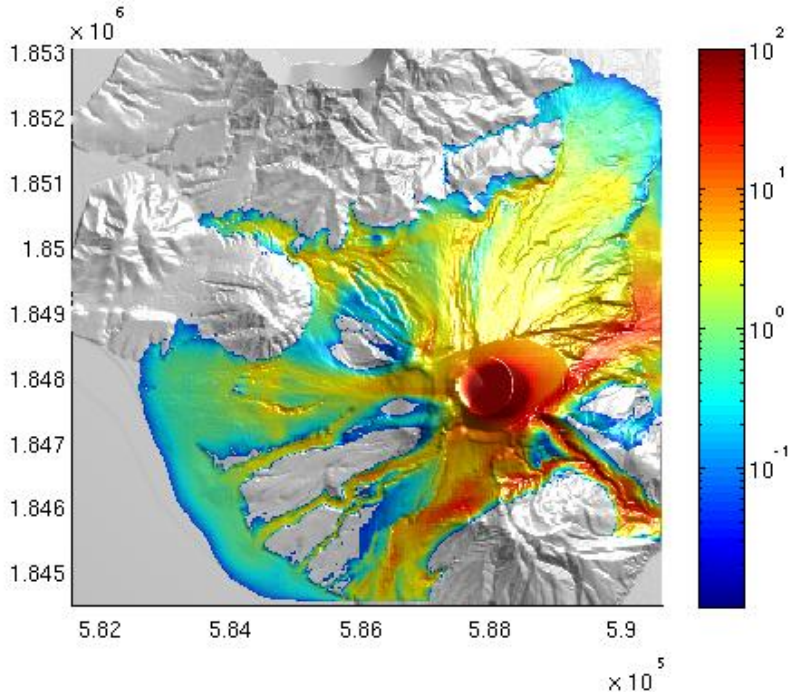
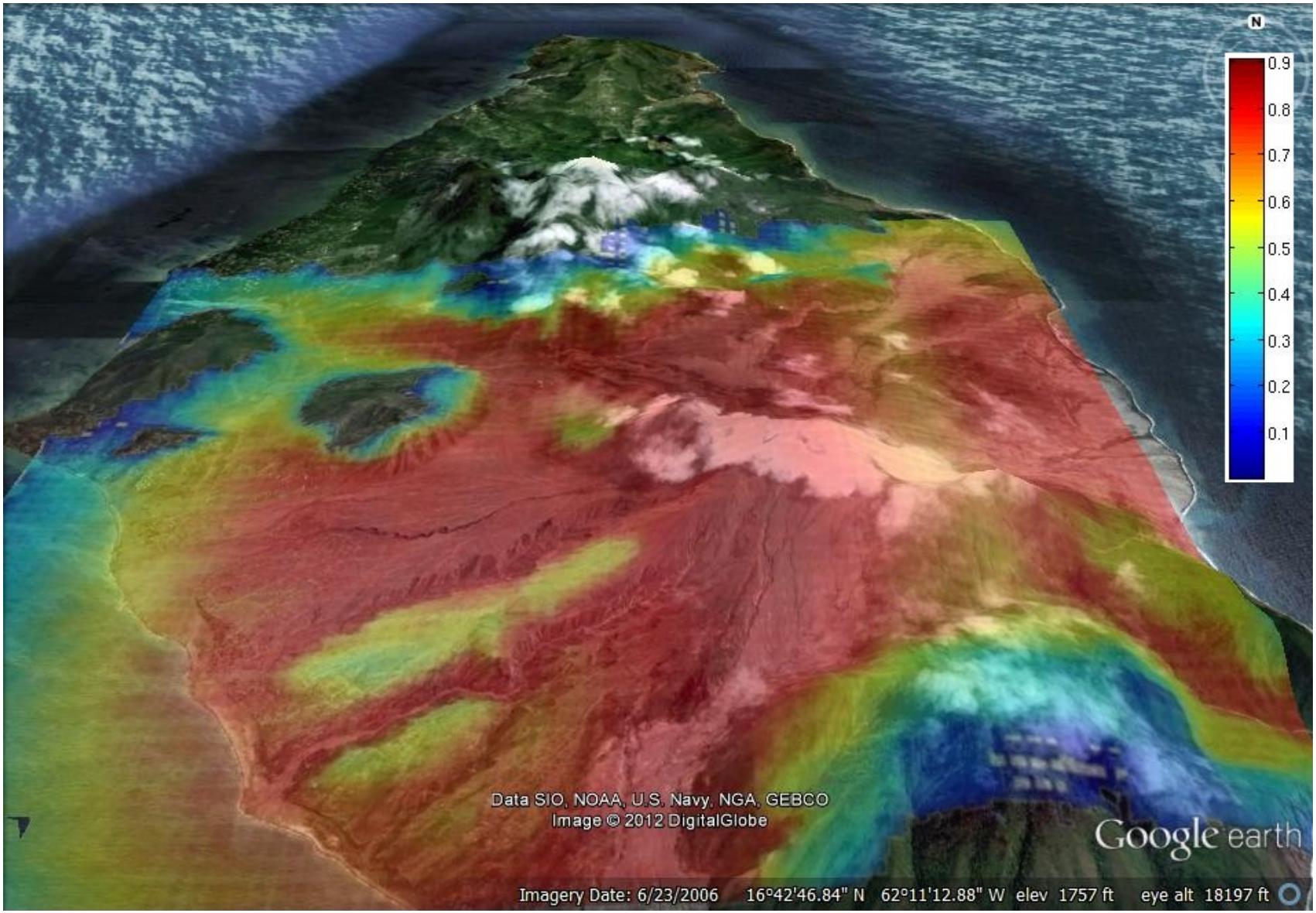


Figure : Example flow simulation of Montserrat volcano using the TITAN2D

- Good simulators of complex multi-scale hazard physics are available.
- They have uncertain data, large compute needs and massive data output.
- Decision making requires uncertainty quantification!



Hazard map with probability of inundation exceeding threshold.

OVERVIEW

- Uncertainty Quantification $\Rightarrow$ a) Ensembles; b) Reasoning with the outcomes (e.g. full flow fields)
- Twin computational challenges of managing large data and performing cpu intensive processing.
- A more accurate construction of emulator by replacing Tessellaton with Trees for neighbor searching.
- We present a new approach which uses a mix of hardware viz. IBM's Netezza database and cluster based supercomputer- in an integrated workflow.

OUTLINE

- Introduction
- Background - Application of Uncertainty Quantification to Hazard Map generation for Debris Flows
- Emulator and its construction
- Computational challenges in Hazard Map Generation
- Layout of Integrated Wokflow and its details
- Results

INTRODUCTION

- Hazard map[a] is a predictive map for a region which provides a probabilistic measure of a hazard (geophysical flows in this case). e.g. 100 year flood map
- Hazard map generation require large data movement and complex computations.
- Massive data leads to problem of storing data, data mining and I/O bottlenecks (I/O bandwidths are significantly lower than processor speeds.)
- High Performance Clusters, Parallel high level I/O libraries, Parallel file systems, Active disks[b], MapReduce[c], Databases[d] etc.

---

[a] Dalbey et al '09, '11, Stefanescu et al '12
[b] Acharya et al'98, Riedel et al,'01
[c] Dean et al'08
[d] Kouzes,et al'09, Oguchi et al '01

Background

Hazard Map $\Leftarrow$ O$\left(10^6\right)$ ensemble of flow simulations – Months of supercomputing!!.

Limited O(10^3) simulator runs used to create an emulator which can then be sampled.

Simulator

A simulator for our purpose is Titan2D[a] , a computational fluid dynamics code which solves a system of hyperbolic equations equations (shallow water like equations) to simulate large geophysical flows using depth averaged model.

[a] Patra et al(2004)

Emulator

A fast surrogate for simulator - would otherwise take several days using a Titan2D.

Emulator : Mathematical Background

Bayesian Regression

$$s(x) = \beta G(x) + \varepsilon \tag{1}$$

$$\varepsilon \sim N(0, \sigma^2)$$

Bayes Linear Equations

$$E(s(y)|s(x)) = E[s(y)] + Cov[s(x), s(y)]\, Var[s(x)]^{-1} (s(x) - E[s(x)]) \tag{2a}$$

$$Var[s(y)|s(x)] = Var[s(y)] + Cov[s(y), s(x)]\, Var[s(x)]^{-1} Cov[s(x), s(y)] \tag{2b}$$

s(x) is the response function.

β is the vector of least square co-efficients.

ε is the the error model.

G(x) is the matrix of basis functions evaluated at the sample points.

E[s(y)|s(x)] is the Bayes Linear Mean and Var[s(y)|s(x)] is the Bayes Linear Variance.

Bayes Linear Emulator

$$E(s(y)|s(x)) = g(y)\beta + r(y)^T R^{-1}\varepsilon \tag{3a}$$

$$Var[s(y)|s(x)] = \sigma^2(1 - r(y)^T R^{-1} r(y)) \tag{3b}$$

$$r_i(y) = exp\left(-\sum_{n=1}^{N_{dim}} \theta_n (y_n - x_{i,n})^2\right) \tag{4}$$

g(y) is the matrix of basis functions evaluated at the resample points

R is the matrix of the correlation functions at x such that $R_{i,j} = r_i(x_j) = r_j(y_i)$.

$\varepsilon = s(x) - G(x)\beta$ is the true error evaluated at the sample points.

θ_n is the vector of hyper-parameters or roughness parameters.

Localization and Parallelization

- The computational effort in R^{-1} in (3) dominates the workflow and for the very large data sets becomes infeasible.
- To make it computationally feasible R^{-1} is approximated as

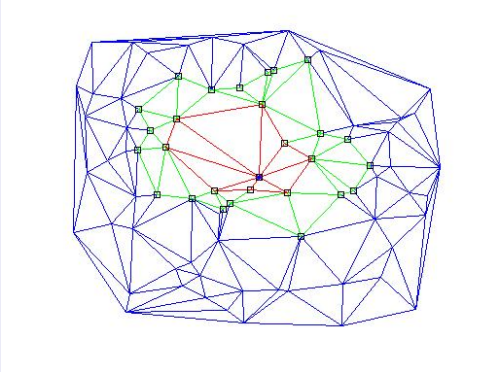
$$R^{-1} \approx \sum_{i=1}^{N} R_i^{-1} \tag{5}$$

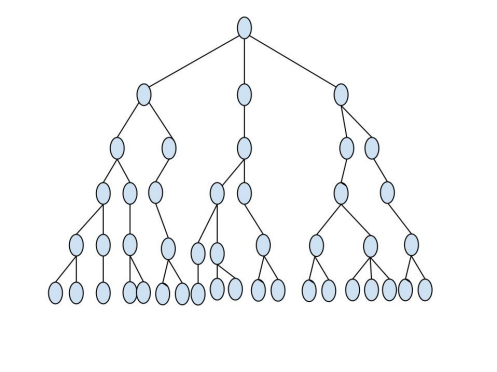
where each R_i is a restriction of R defined as below:

$$R : \Omega \to \mathbb{R}, \Omega = \cup_{i=1}^{N} \Omega_i, R_i : \Omega_i \to \mathbb{R}$$

- dia(Ω_i) is related to the local strength of the correlations. Schemes of this type widely used in weather forecasting – Gneiting '99, ...

Tessellation vs Trees





- For Ω_i points Dalbey'09 proposed a tessellation based approach wherein the input parameter space is tessellated and correlations further away than n (n=2,3 ...) hops in the tessellation are disregarded.
- We propose here an alternate methodology based on a simple scaled distance metric(D_s) and a tree based organization of the data.

$$D_s = \sqrt[2]{\sum_{n=1}^{C_{dim}} (x_n - y_n)^2} \tag{6}$$

where C_{dim} is the number of dimensions.

Tessellation vs Trees

Implemented R+ type tree data structure for storing and searching of multi-dimensional point objects. Test case 4 input parameters – 2 parameters for pile centre and iinternal friction) and 2 spatial parameters.

- Tessellation (using delaunay triagulation) results in more simplices than the sample points and requires more search time. Searching for 2^{17} points, of 4 dimensions, among the 55874 simplices generated using 2048 *samples* took 3049 seconds or close to 51 minutes.
- Using tree, the search was performed in under a minute.
- Trees allow more flexibility in selection of neighbours than tessellation.
- For higher dimensions tessellation based method takes longer and even breaks down in some cases.

Hazard Map Generation

- Titan2D simulations are performed for 2048 sets of input parameters sampled using LatinHypercubeDesign resulting in $\approx$ 2GB of flowdata.
- 10^8 samples from the tensor product of the input parametrs and 2 space dimensions have to be downsized/downsampled to $\approx 10^6$.
- Neighbour search involving $\approx 4x10^6$ samples and close to 10^{10} resamples.
- Construction Millions of emulators (one emulator about each sample).
- Emulator construction is dominated by matrix operations performed through an iterative non linear optimization technique.
- Scanning of massive dataset from emulator for computing barycentric weights and for aggregation.

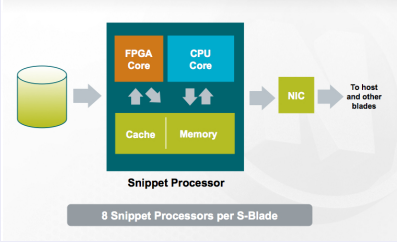
Challenges

- Data storage and data mining operations for downsampling and neighbour search - These require few cpu cycles but rapid and repeated passes through dataset.
- Construction of millions of emulators - Requires less data movement but highly CPU intensive.
- Aggregation of data - Scanning and grouping very large dataset.

Problem of large number of computations is intertwined with the problem of data mining and massive data movement. We introduce a *divide and conquer* model that separates data mining operation from complex computations.

Netezza Data Warehouse[1]

- Massively Parallel Processing architecture enabled through a high speed network fabric.
- Netezza is equipped with snippet processing units (spu). Each spu is contains multi-core CPUs, multi-engine FPGAs and gigabytes of RAM
- FPGAs uncompress and filter out extraneous data early in the data stream to be processed by CPUs. – "in-database analytics"



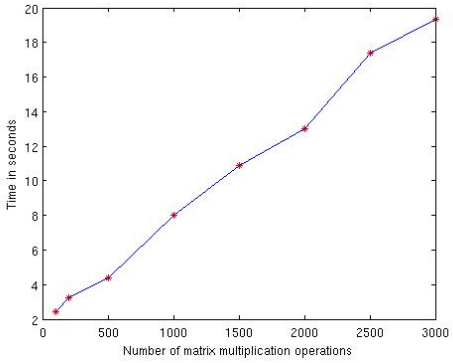
Downsampling and Neighbour search

- Downsampling was achieved through a single SQL query in much simpler and straightforward implementation compared to the previous work.
- Tesselation based method was replaced with a more accurate distance based search.
- Neighbor search based on euclidean distance. Searching for samples of the order of 10^{10} ($\approx$ 4 million) from among themselves and from a collection of resamples of the order of 10^{10} ($\approx$ 10 billion).
- All the operation performed using less than 60 SQL queries a significant improvement over thousands of lines of matlab implemntation.
- Netezza's datawarehousing ability allowed storage of all the data in one place as opposed to storing on several thousand ascii files on the cluster.

Netezza Database

Compute intensive Tasks - Not suited for Netezza

- Complex algorithms can't be easily translated into a declarative language (SQL).
- The number of SPUs puts a limit on the available processors.
- Explicit use of high performance libraries like BLAS and LAPACK is difficult .
- Emulator construction would have required days to finish.

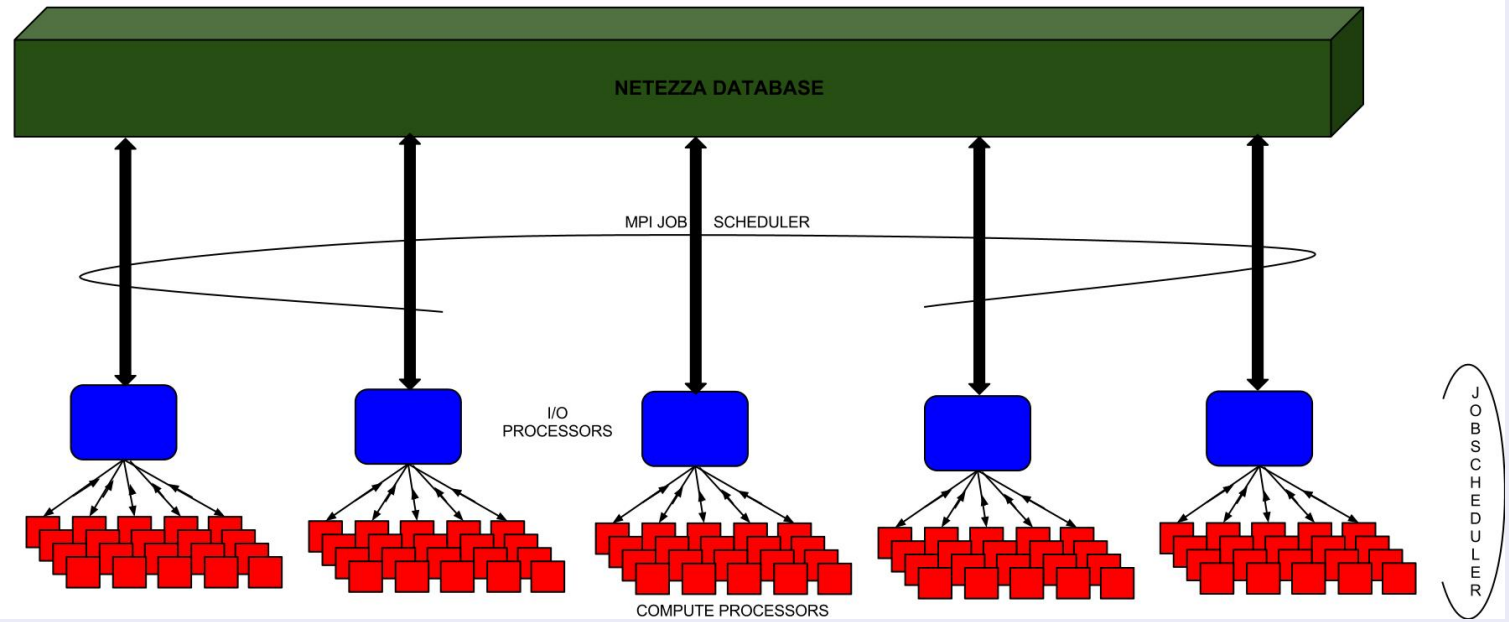


Plot of number of matrix multiplication operations of square matrices of size 200x200 against time.

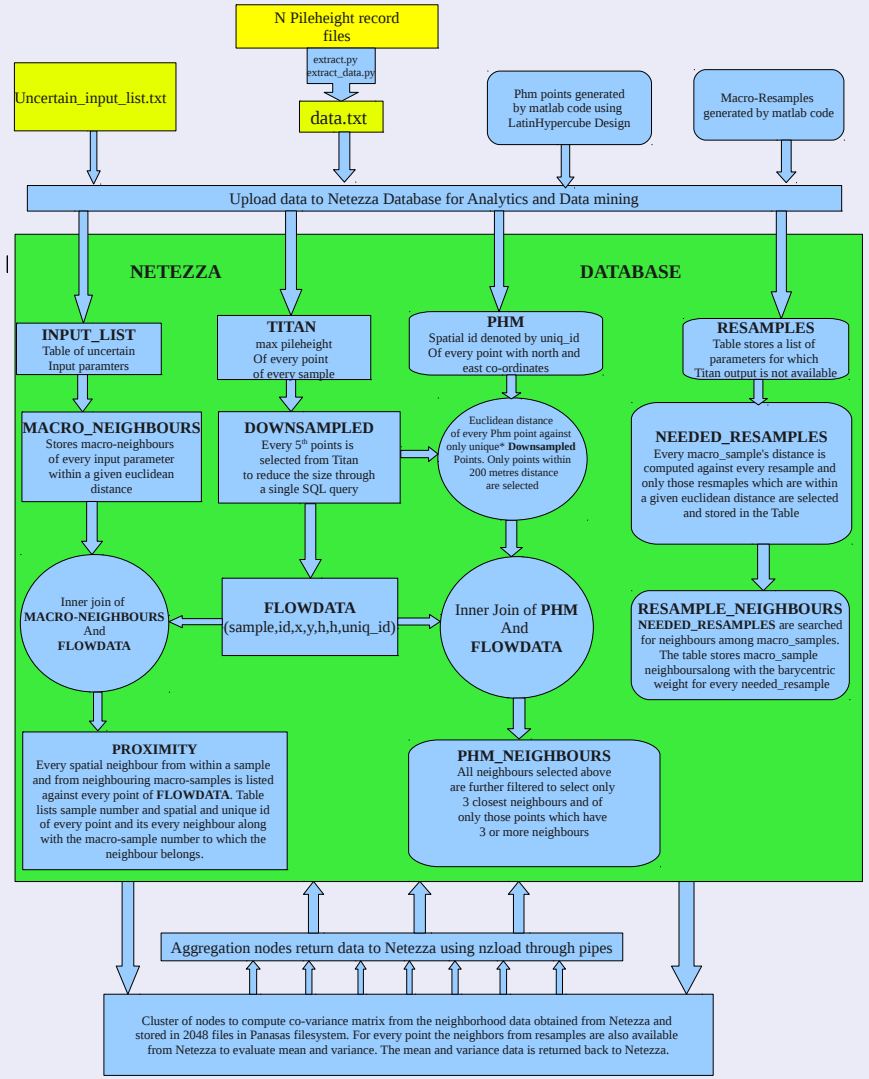
High Performance Cluster

- Large data requirements by the cluster leads to I/O bottlenecks.
- Resolved by creating a hierarchy at the cluster level.
- A few of the processors are identified as the I/O proessors perform the task of managing I/O - I/O forwarding
- I/O processors extract sufficient amount data from Netezza using pipes to feed the compute processors.
- An MPI job scheduler distributes the work to compute processors.
- The resulting data which requires aggregation is tranferred to Netezza via I/O processors using named pipes.

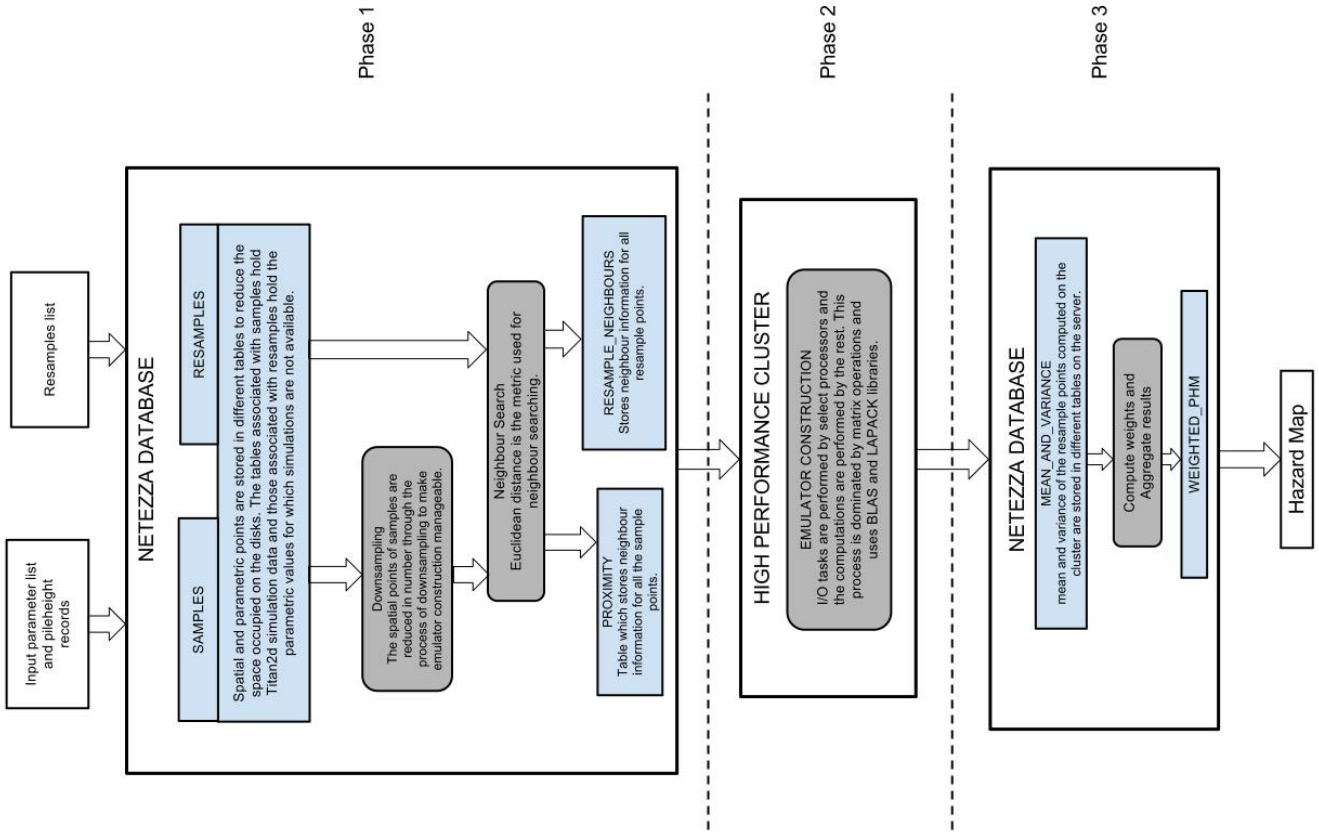
Computing Model



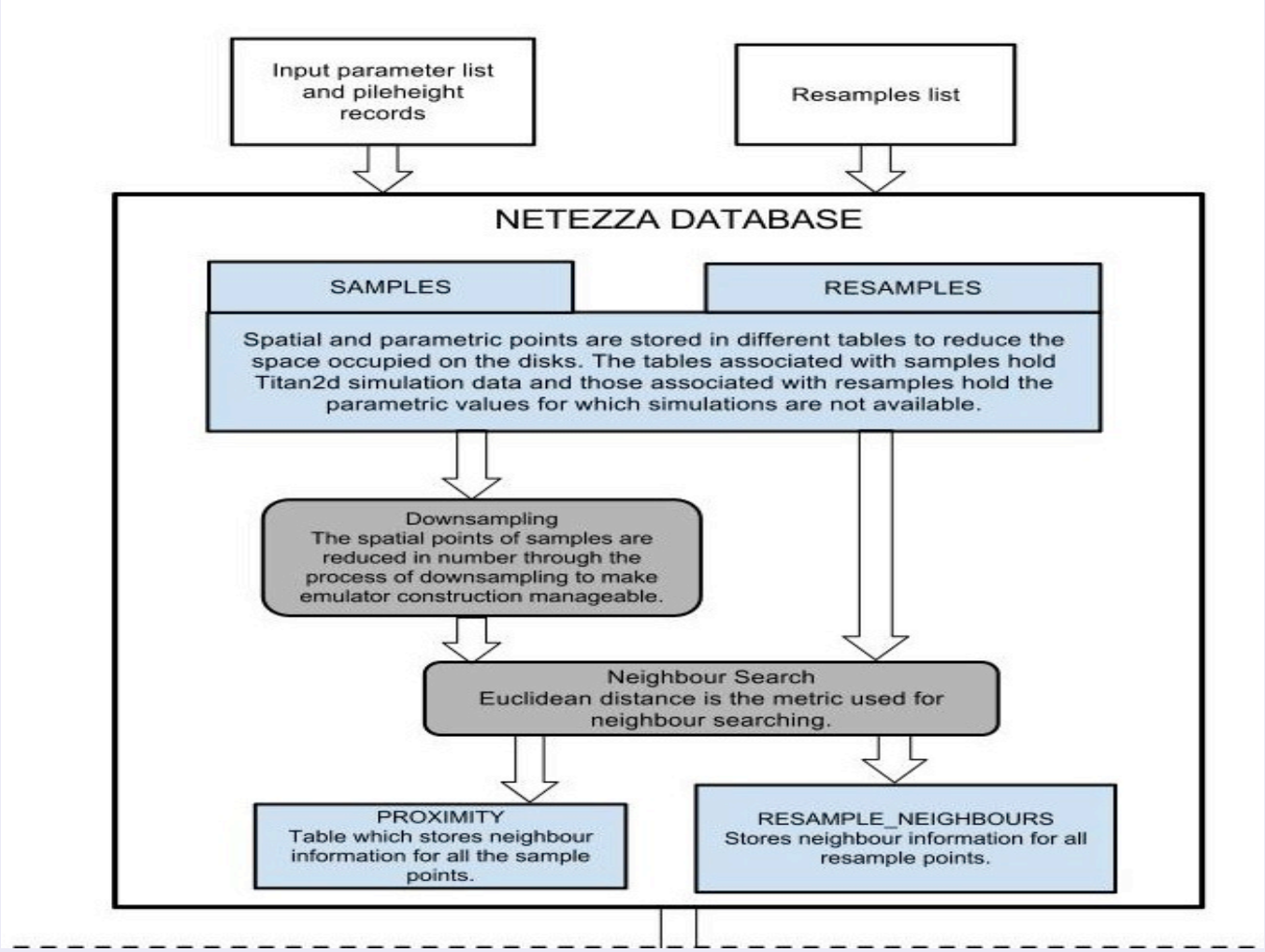
Integrated Workflow Diagram



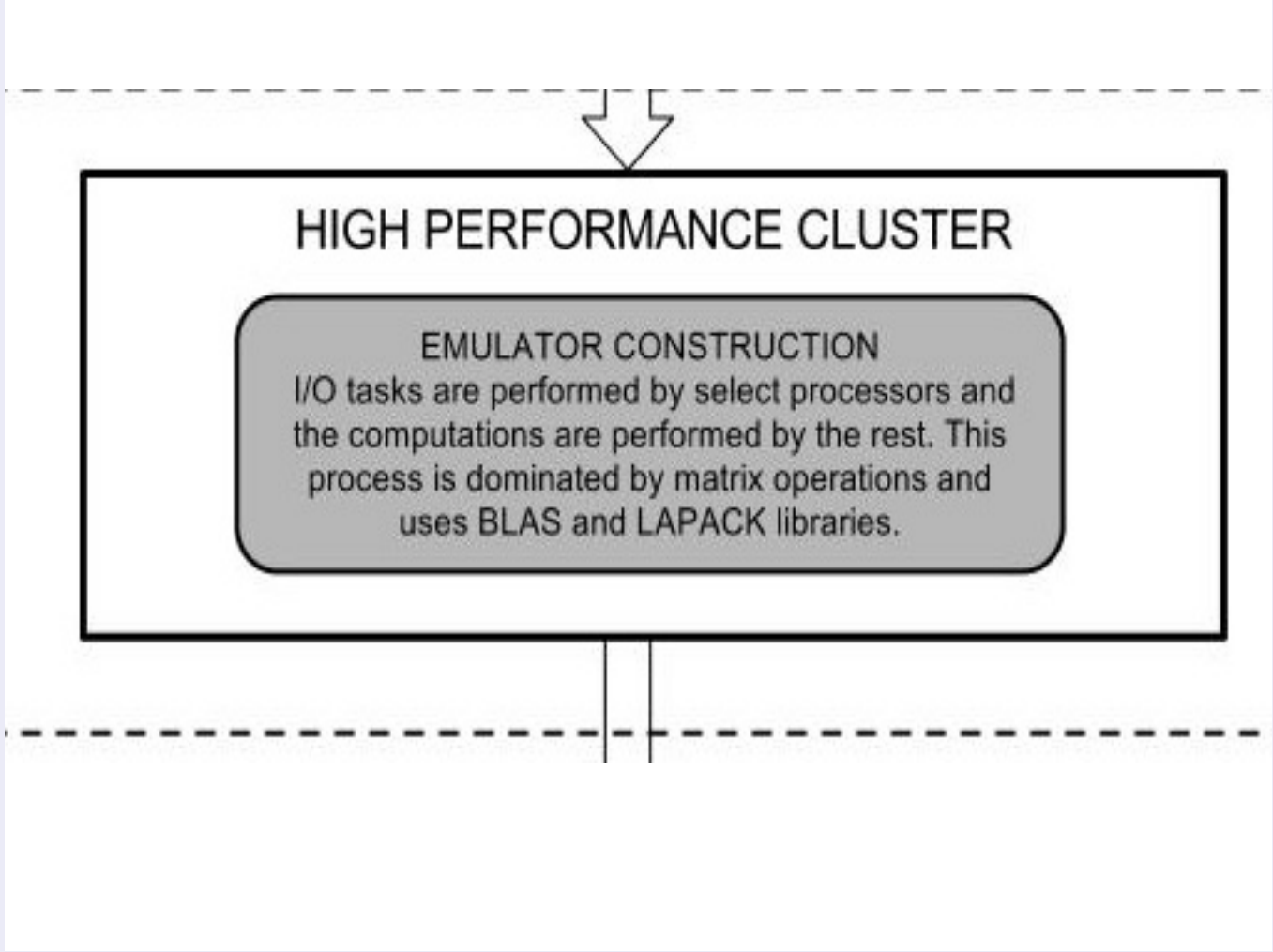
Integrated Workflow Diagram



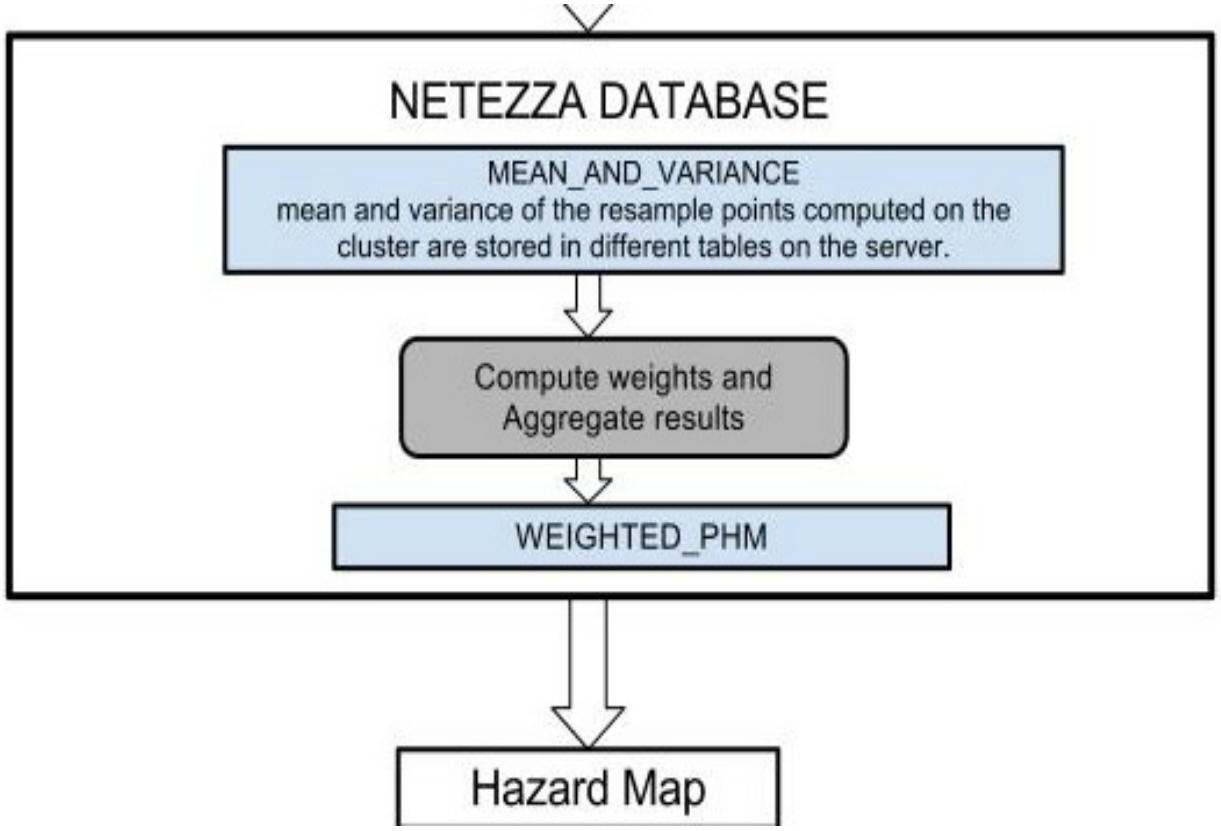
Integrated Workflow Diagram – Phase 1



Integrated Workflow Diagram – Phase 2



Integrated Workflow Diagram – Phase 3



Results

- Downsampling and neighbour search time 30 minutes.
- Computation time on High Performance Cluster 9 hours
- Number of neighbourhood records on table - $\sim$ 2 Billion records.
- 504 processors functioning as 18 independent groups were responsible for emulator construction and the evaluation of mean and variance.
- Each group had 1 I/O processor and atmost 27 compute processors.
- Approximately 4 million covariance matrices of size 300x300 were computed through an iterative process and the moments were computed for several billion resample points.

Results

cont..

- 1.5 TB of data was transferred over the network to Netezza server over 18 different pipes.
- This data was stored on 18 different tables occupying a little more than a total of 50 billion rows.
- The final step which required scanning billions of rows of data to evaluate the barycentric weights and aggregate the results was also per formed on the Netezza server and took 2.5 hours of time.
- The hazard map was generated in 12 hours of time.

Hazard Map

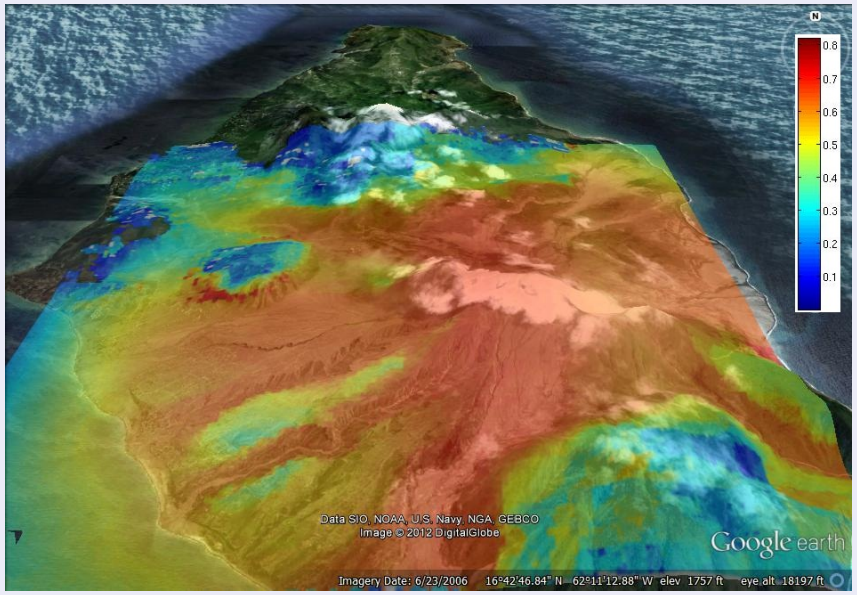


Image of the Montserrat Island with and without the Hazard Map. The Hazard Map shown here was generated using larger euclidean distance of serach.

Results

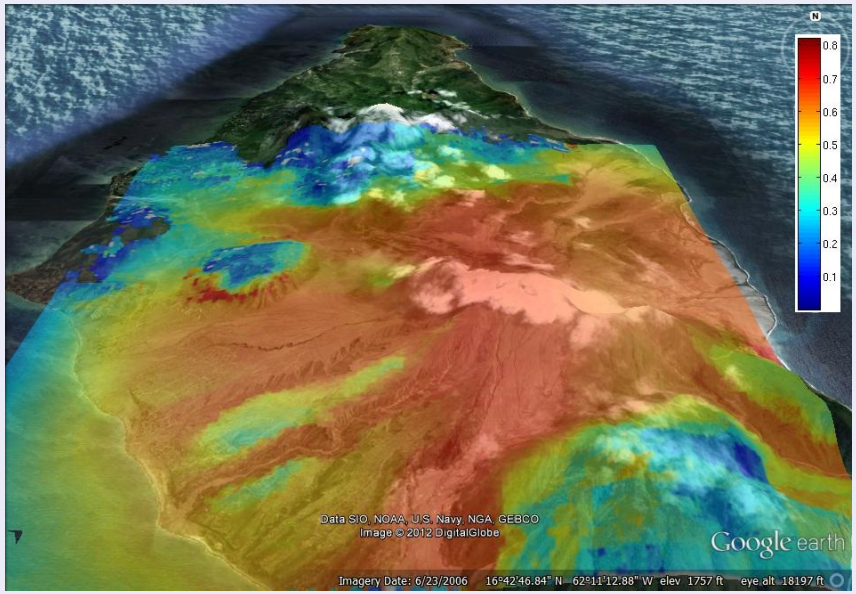
Improving the computation time

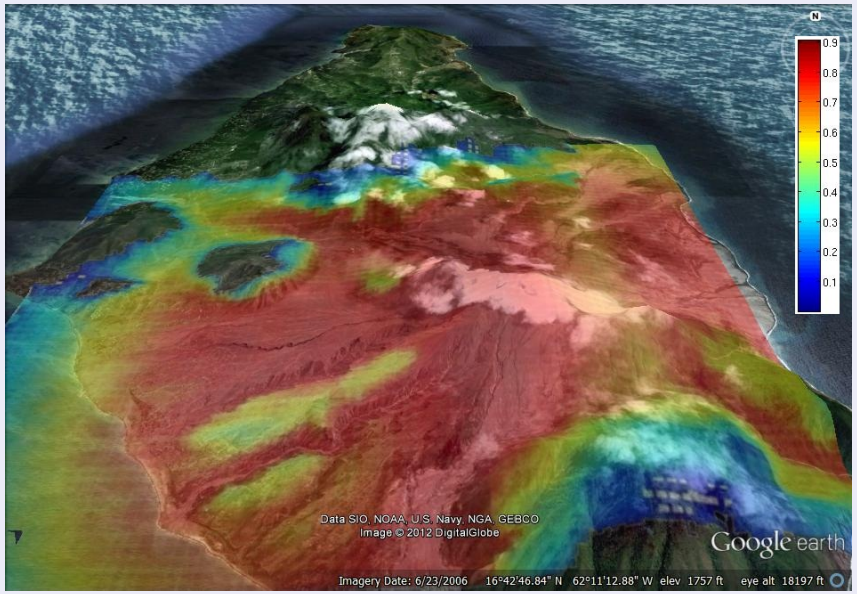
- Reduce the number of write sessions per load session. A write session involves writing data to the pipe. A load session is complete when pipe opened for writing is closed.
- Communications between I/O processors and server were made less frequent.
- I/O processors could now disseminate data to compute processors with fewer interruptions.
- Avoid simultaneous aggregation of data on server while loading the tables which put severe strain on the server.
- Block writing of data to server.

Results

- New Hazard map generated in less than 6 hours time with cluster time reduced to 2.5 hours.
- 512 processors on 43 nodes of 12 cores each and by keeping 16 connections open between Netezza and the cluster. Each group had 1 I/O and atmost 32 compute processors.
- A maximum limit of 120 was enforced on the size of the covariance matrices. dimensions(scaled).
- The operations prior to emulator construction took 10 minutes and the final aggregation was completed in 2.5 hours, both of which were performed entirely on Netezza server.

Hazard Map





Hazard Map generated using more localized emulators (through smaller euclidean distance of search).

Map Reduce Framework

Hadoop

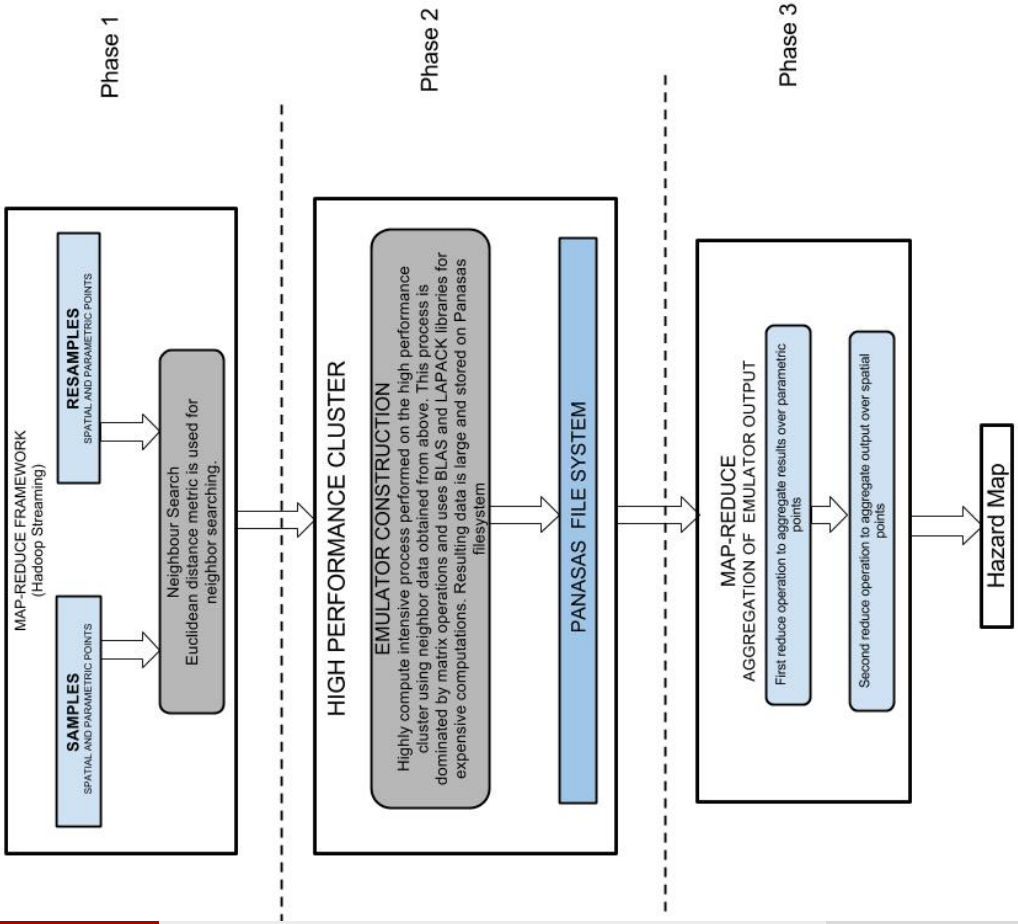
- Hadoop is a popular open source implementation of the Map-Reduce concept.
- Cheap alternative to Netezza database.
- Offers a fault tolerant scheduler.
- Hadoop streaming - can be used in conjunction with scripts in a variety of languages.

Hadoop

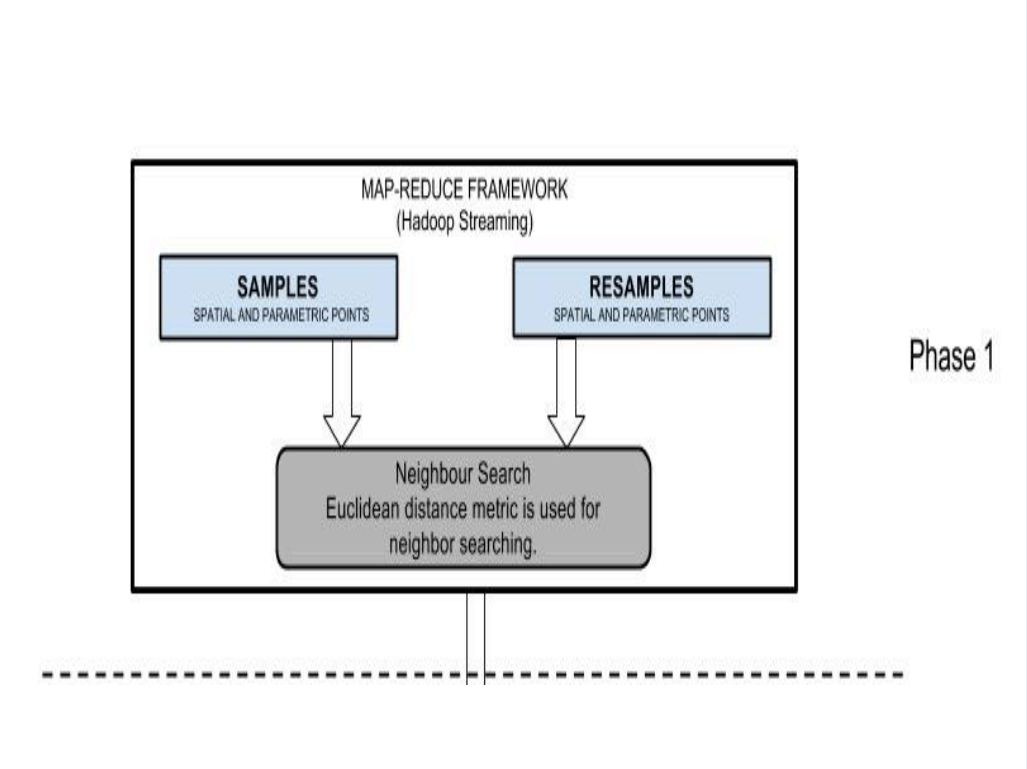
Motivation for using Hadoop

- Could be set up on the same cluster that performs emulator construction.
- This obviates the need to move data as data mining and computation can be performed on the same nodes.
- Good scalability. Netezza database's scalability is constrained because of a limit on the number of pipes and the number of processors on its rack.
- Easy to develop scripts in python and use them with Hadoop streaming.

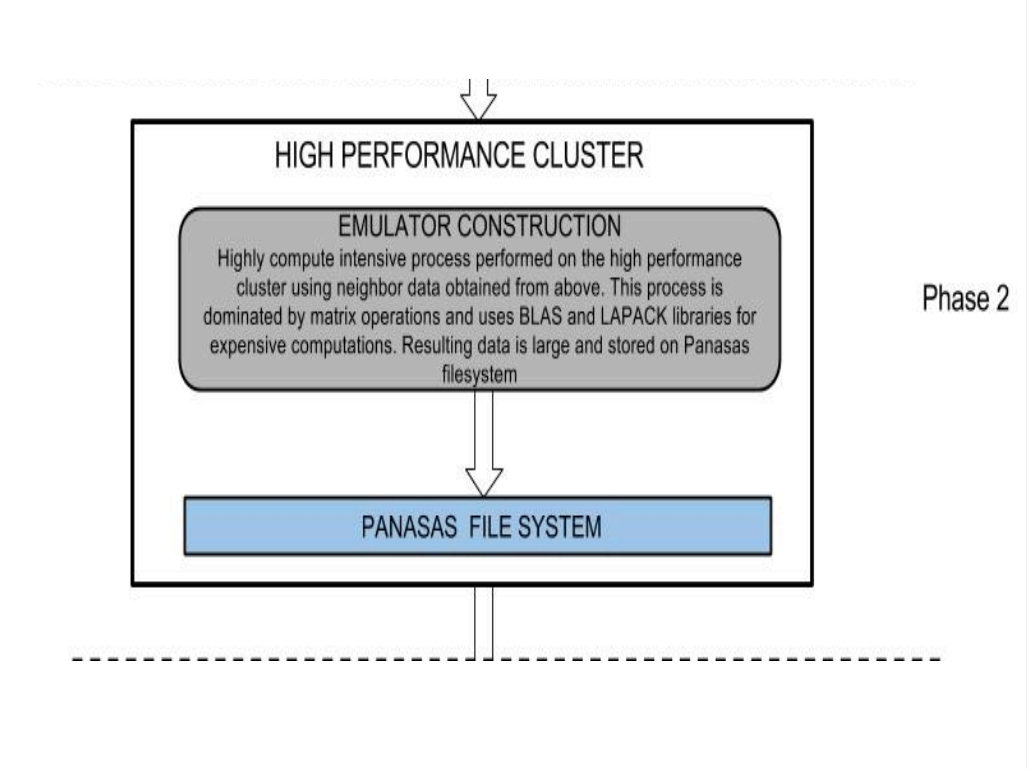
Integrated Workflow Diagram



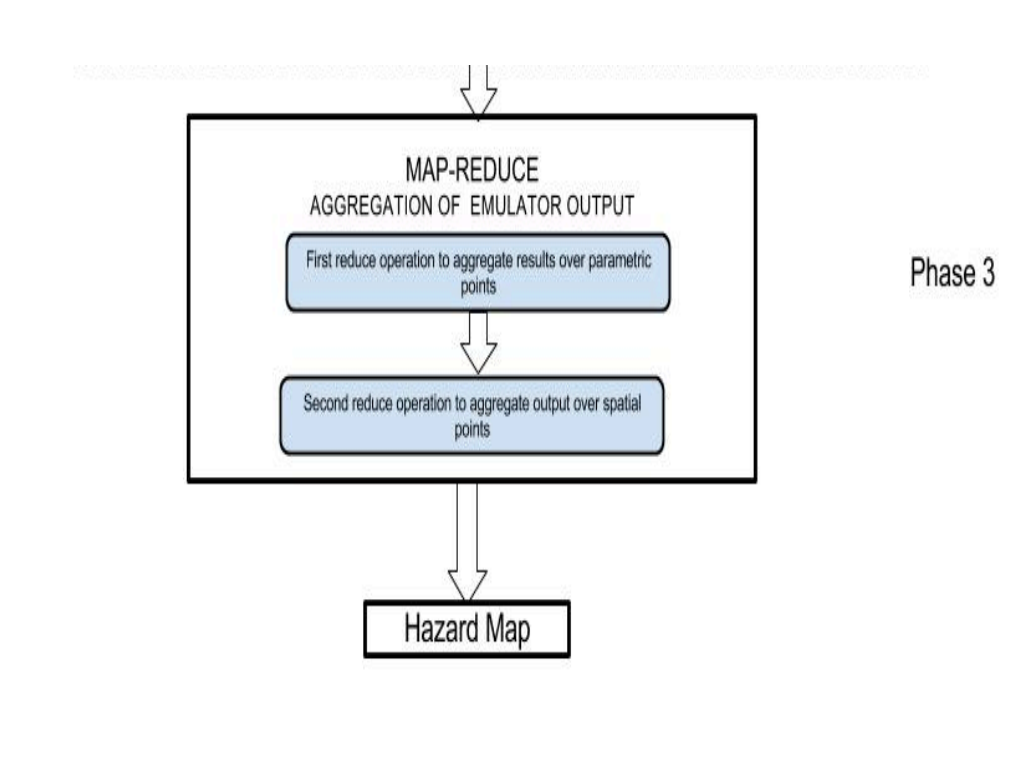
Integrated Workflow Diagram – Phase 1



Integrated Workflow Diagram – Phase 2



Integrated Workflow Diagram – Phase 3



Hadoop

Implementation Details

- Downsampling and Neighbor search were performed with ease on Hadoop.
- Mapper computes the distances between two sets of data (X and Y) and prints out the result as key- value pair.
- Reduce operation only involved printing out the output from mapper.
- Emulator construction on cluster - Wrap Phase 2 in mapper and Phase 3 in reducer.

Hadoop

Challenges

- Emulator Construction being compute intensive necessitates utilizing all processors of a multi-core machine.
- Only 2 mappers could be spawned on each multi-core node which was a severe limitation.
- Possible reasons for the above beahviour could be small input data and complexity of mapper operation.
- Emulator construction as such had to be performed without using hadoop and the output was stored on panasas filesystem.
- Phase 3 was performed using Hadoop using two reduce operations. Data had to be copied from panasas to Hadoop's distributed filesystem which consumed considerable time.

Hadoop

Results

Completion Times:

Phase 1 : 20 minutes

Phase 2 : 5 hours

Phase 3 : 9.5 hours

The above number do not include the time taken to copy data from panasas to HDFS. Phase 3 was the most dominant phase utilizing Hadoop and reduced 800GB of data to 190GB through 100 reduce tasks spread over 20 nodes.

Hadoop

Results

- The aggregation of data is a typical reduce operation.
- Phase 3 that was performed on Netezza was now performed using two reduce operations.
- Data had to once again copied from panasas to Hadoop's distributed filesystem.

THANK YOU

References

Netezza performance server® data warehouse appliance:an architectural comparison, November 2006.

A. Acharya, M. Uysal, and J. Saltz.

Active disks: Programming model, algorithms and evaluation.

In *ACM SIGPLAN Notices*, volume 33, pages 81–91. ACM, 1998.

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.

K.R. Dalbey.

Predictive simulation and model based hazard maps of geophysical mass flows.

State University of New York at Buffalo, 2009.

J. Dean and S. Ghemawat.

Mapreduce: simplified data processing on large clusters.

Communications of the ACM, 51(1):107–113, 2008.

S.C. Helmreich and J.R. Cowie.

Data-centric computing with the netezza architecture.

D. Kotz.

Disk-directed i/o for mimd multiprocessors.

ACM Transactions on Computer Systems (TOCS), 15(1):41–74, 1997.

R.T. Kouzes, G.A. Anderson, S.T. Elbert, I. Gorton, and D.K. Gracio.

The changing paradigm of data-intensive computing.

Computer, 42(1):26 –34, jan. 2009.

E. Krevat, T. Shiran, E. Anderson, J. Tucek, J.J. Wylie, and G.R. Ganger.

Applying performance models to understand data-intensive computing efficiency.

Technical report, DTIC Document, 2010.

M. Oguchi and M. Kitsuregawa.

Data mining on pc cluster connected with storage area network: its preliminary experimental results.

In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 7, pages 2145 –2149 vol.7, 2001.

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.

E. Riedel, C. Faloutsos, G.A. Gibson, and D. Nagle.

Active disks for large-scale data processing.

Computer, 34(6):68–74, 2001.

E. Riedel, G. Gibson, and C. Faloutsos.

Active storage for large-scale data mining and multimedia.

M. Stonebraker, D. Abadi, D.J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin.

Mapreduce and parallel dbmss: friends or foes.

Communications of the ACM, 53(1):64–71, 2010.

A.S. Szalay, G. Bell, J. Vandenberg, A. Wonders, R. Burns, Dan Fay, J. Heasley, T. Hey, M. Nieto-SantiSteban, A. Thakar, C. van Ingen, and R. Wilton.

Graywulf: Scalable clustered architecture for data intensive computing.

In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1 –10, jan. 2009.