ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

A Non-Linear Indoctrination Model to Reduce the Data Recovery and Effecting Cost in Cloud Atmosphere

R.Kavitha, Sundararajan.M, Arulselvi S

Assistant Professor, Dept. of CSE, Bharath University, Chennai, Tamil Nadu, India

Director, Research Center for Computing and Communication, Bharath University, Chennai, Tamil Nadu, India

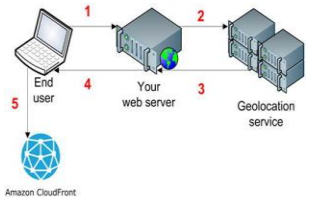
Co-Director, Research Center for Computing and Communication, Bharath University, Tamil Nadu, India

ABSTRACT: In this paper, we formulate a non-linear indoctrination model to minimize the data retrieval and execution cost of data-intensive workflow in Clouds. Our model retrieves data from Cloud storage resources such that the amount of data transferred is inversely proportional to the communication cost. We take an example of an 'intrusion detection' application workflow, where the data logs are made available from globally distributed Cloud storage servers. We construct the application as a workflow and experiment with Cloud based storage and compute resources. We compare the cost of multiple executions of the workflow given by a solution of our non-linear program against that given by Amazon Cloud Front's 'nearest' single data source selection. Our results show a savings of three-quarters of total cost using our model.

I. EXISTING SYSTEM

Amazon Cloud Front

Cloud Front is a web service for content delivery. It makes it easier for you to distribute content to end users quickly, with low latency and high data transfer speeds. Cloud Front delivers your content through a worldwide network of edge locations. End users are routed to the nearest edge location, so content is delivered with the best possible performance. Cloud Front works seamlessly with the Amazon Simple Storage Service, which durably stores the original, definitive versions of your files.



Amazon CloudFront Functionality

Amazon CloudFront has a simple, web services interface that lets you get started in minutes. In Amazon CloudFront, your objects are organized into distributions. A distribution specifies the location of the original version of your objects. A distribution has a unique CloudFront.net domain name (e.g. abc123.cloudfront.net) that you can use to reference your objects through the network of edge locations. If you wish, you can also map your own domain name (e.g. images.example.com) to your distribution. You can create distributions to either download your content using the HTTP or HTTPS protocols, or stream your content using the RTMP protocol.

To use Amazon CloudFront, you:

- Store the original versions of your files on an origin server. An origin server is the location of the definitive version of your object. This could be another Amazon Web Service – Amazon S3 bucket, Amazon EC2 instance – or this could be your own origin server.
- Create a distribution to register your origin server with Amazon CloudFront through a simple API call.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

- Use your distribution's domain name in your web pages, media player, or application. When end users request an object using this domain name, they are automatically routed to the nearest edge location for high performance delivery of your content.
- Pay only for the data transfer and requests that you actually use.
- Amazon Cloud Front's availability is backed with the Amazon CloudFront Service Level Agreement

Drawback:

- Cost of computing, storage and communication over cloud be very high.
- Intrusion Detection method not available.

II. PROPOSED SYSTEM

Scientific and commercial applications are leveraging the power of distributed computing and storage resources. These resources are available either as part of general purpose computing infrastructure such as Clusters and Grids, or through commercially hosted services such as Clouds. Clouds have been defined to be a type of parallel and distributed system consisting of inter-connected and virtualized computers. These computers can be dynamically provisioned as per users' requirements. Thus, to achieve better performance and scalability, applications could be managed using commercial services provided by Clouds, such as Amazon AWS, Google AppEngine, and Microsoft Azure. Some of these cloud service providers also have data distribution services, such as Amazon Cloud-Front.

However, the cost of computing, storage and communication over these resources could be very high for compute-intensive and data-intensive applications. Data mining is an example application domain that comprises of data-intensive applications often with large distributed data and compute-intensive tasks. The data to be mined may be widely distributed depending on the nature of the application. As the size of these data-sets increases over time, the analysis of distributed data-sets on computing resources by multiple users (repeated executions) has the following challenges:

- A well-designed application workflow: Large number of data-sets and mining tasks make the application complex.
- Minimization of communication and storage costs: Large size and number of distributed data-sets make the application data-intensive.
- Minimization of repeated data mining costs: Cost of computing (classification/knowledge discovery) and transferring of data increases as the number of iterations/data-sets increase.

Challenges listed above for data-intensive workflow by making the following three contributions:

1. We take Intrusion detection as a data mining application which will be referenced throughout the remainder of this paper. This application has all the features as listed in the previous paragraph when executing commercially. We design the application as a workflow that simplifies the basic steps of data mining into blocks.
2. We model the cost of execution of an intrusion detection workflow on Cloud resources using a Non-Linear Programming (NLP) model. The NLP-model retrieves data partially from multiple data sources based on the cost of transferring data from those sources to a compute resource, so that the total cost of data-transfer and computation cost on that compute resource is minimized.
3. We then apply the NLP-model on the intrusion detection application to minimize repeated execution costs when using commercial compute and storage resources. As an example, we compare the costs between our model and Amazon CloudFront.

Algorithm for Scheduling Heuristic

1: Compute ykj & dki,j
for all tasks by solving the NLP
2: **repeat**
3: Get all the 'ready' tasks in the work.ow
4: **for** each task tk . Tready **do**
5: Assign tk to the compute resource P for which ykj = 1
6: Fix partial data transfers dki,j from
Si to the compute resource Pj for which ykj = 1
7: **end for**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

8: Dispatch all the mapped tasks for execution
9: Wait for POLLING TIME
10: Update the ready task list
11: (Upload output .les of completed tasks to the storage central for distribution)
12: **until** there are unscheduled tasks in the ready list

Non-Linear Programming model design

We denote an application workflow using a Directed Acyclic Graph (DAG) by G=(V,E), where V ={T1, ..., Tn} is the set of tasks, and E represents the data dependencies between these tasks, that is, tdatak = (Tj, Tk) . E is the data produced by Tj and consumed by Tk. We have a set of storage sites S = {1, ..., i}, a set of compute sites P = {1, ..., j}, and a set of tasks T = {1, ..., k}. We assume the 'average' computation time of a task Tk on a compute resource Pj for a certain size of input is known. Then, the cost of computation of a task on a compute host is inversely proportional to the time it takes for computation on that resource. We also assume the cost of unit data access txcosti,j from a storage resource Si to a compute resource Pj is known. The transfer cost is .xed by the service provider (e.g. Amazon CloudFront) or can be calculated according to the bandwidth between the sites. We assume that these costs are non-negative, symmetric, and satisfy the triangle inequality: that is, these relations can be expressed as:

- txcosti, j = txcostj,i for all i, j . N, and txcosti, j + txcostj, k = txcosti,k for all i, j, k. N.
- ecost . 1/ {execution time or capability of resource} txcost. bandwidth OR = (tx cost/unit data)/site
- total cost of computation :

  C = ecost * etime + txcost * data + overheads

The cost-optimization problem is: Find a feasible set of 'partial' data-sets {dk i, j} that must be transferred from storage host Si to compute host Pj for each task (Tk. V) Such that the total retrieval cost and computation cost of the task on Pj is minimal, for all the tasks in the workflow (not violating dependencies) .

Non-linear model

Here, we try to get the minimum cost by formulating a non-linear program for the cost-optimization problem, as depicted in Figure 3. The formulation uses two variables y, d and pre-computed values txcost, ecost, txtime, etime as listed below:

- y characterizes where each task is processed. Yk j = 1 iff task Tk is processed on processor Pj .
- d characterizes the amount of data to be transferred to a site. e.g. dk i,j = 50.2 denotes 50.2 units of data are to be transferred from Si . Pj for task Tk.
- txcost characterizes the cost of data transfer for a link per data unit. e.g. txcosti,j = 10 denotes the cost of data transfer from Si . Pj . It is added to the overall cost iff dk i,j > 0 & yk j = 1.
- ecost characterizes the cost of computation (usage time) of a processor. e.g. ecostj = 1 denotes the cost of using a processor Pj . It is added to the overall cost iff ykj = 1.
- txtime characterizes the average time for transferring unit data between two sites. e.g. txtimei,j = 50 denotes the time for transferring unit data from Si . Pj . It is added to the Execution Time (ET) for every task iff dk i,j > 0 & yk j = 1.
- etime characterizes the computation time of a task averaged over a set of known and dedicated resources. e.g. etimek
  j = 20 denotes the time for executing a task Tk on a processor Pj . It is added to ET iff yk j = 1.

The constraints can be described as follows:

- (a) & (h) ensure that each task k . T is computed only once at processor j. P when the variable yk j > 0. For partial values of yk j, we round up/down to the nearest integer (0 or 1). Tasks are not partitioned or migrated.
- (b) & (c) ensure that partial data transferred and total data required by a task cannot be negative.
- (d), (e), (f) and (g) ensure that cost and time values are all positive.
- (i), (a) & (b) ensure that partial-data are transferred only to the resource where a task is executed. For all such transfers, the sum of data transferred should equal to the data required by the task, which is tdatak.
- (j) ensures that the total data transfer for all the tasks are bounded by the sum of data required by each task. This is important for the solvers to relate (h), (i) & (j),

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

- (i) & (j) combined ensure that whenever partial-data dk i,j is transferred to a compute host Pj , then a compute host must have been selected at j (ykj = 1), and that total data transfer never exceeds the bound tdatak for each task and in total.

To get an absolute minimum cost, we map the tasks in the workflows onto resources based only on cost optimization (not time). This eliminates the time dependencies between tasks. However, the task to compute-resource mappings and data-source to compute-resource mappings minimizes the cost of execution but not the makespan. The execution time of a task (ET k) is calculated based on the cost-minimized mappings given by the solver.

The total: k.T (ETk+waiting time) is the makespan of the work-flow with the minimum cost, where the waiting time denotes the minimum time a task has to wait before its parents .nish execution.

Cost Minimization for the Intrusion Detection Application

In this section, we describe the method we used to solve the non-linear program. We then describe how we applied the solution for minimizing the total cost of execution to the intrusion detection application workflow.

NLP-solver:

We wrote a program using the Modeling Language for Mathematical Programming (AMPL) for solving our NLP-model. We used DONLP2, a nonlinear program solver, to solve the model. The computation time of the solver to reach a solution (for a maximum of 2000 iterations) was less than 2 seconds, which is insignificant as compared to the data-transfer time in our experiments.

Partial-data retrieval and task-to-resource mapping:

Based on the integer values of yk j given by DONLP2, we statically mapped the tasks in the intrusion detection application workflow to each compute resource Pj . Data retrievals are also fixed for each ready task from each S based on the value of dk i,j and yk j = 1. The steps of mapping and data retrievals are given in Algorithm 1.

The heuristic computes the values for task mapping y k j and dk i,j for all the tasks in the beginning according to the solution given by a NLP-solver. As all the tasks in the workflow are mapped initially, the for loop preserves the dependencies of the tasks by dispatching only the ready tasks to the resources. For dispatched tasks, partial data retrievals to the assigned compute resource occur from chosen resources.

All child tasks wait for their parents to complete, after which they appear in the ready list for dispatching. The scheduling cycle completes after all the tasks are dispatched successfully. The output data of each completed task is staged back to the Cloud storage as part of the task's execution. The Cloud storage should ensure that the files are distributed to the edge-servers within certain time bound such that child tasks do not have to wait for availability of data longer than downloading directly from the Cloud's central server.

III. CONCLUSION

In this work, we presented the execution of an intrusion detection application workflow using Cloud resources, with an objective of minimizing the total execution cost. We modeled the cost minimization problem and solved it using a non-linear program solver. Based on the solution, we retrieved data from multiple data sources was mapped, unlike previous approaches, where data was retrieved from the 'best' data source.

REFERENCES

1. "Ultrasonic texture motion analysis: Theory and simulation" by J.Meunier and M.Bertand.
2. Kanniga E., Selvaramarathnam K., Sundararajan M., "Embedded control using mems sensor with voice command and CCTV camera", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) pp.4794-4796.
3. "Statistical and structural approaches to texture " by R.M.Haralick.
4. Das M.P., Jeyanthi Rebecca L., Sharmila S., "Evaluation of antibacterial and antifungal efficacy of Wedelia chinensis leaf extracts", Journal of Chemical and Pharmaceutical Research, ISSN : 0975 – 7384, 5(2) (2013) pp.265-269.
5. "A Theory of multiresolution signal decompression: The wavelet representation".
6. Subhashini V., Ponnusamy S., Muthamizhchelvan C., "Growth and characterization of novel organic optical crystal: Anilinium d-tartrate (ADT)", Spectrochimica Acta - Part A: Molecular and Biomolecular Spectroscopy, ISSN : 1386-1425, 87() (2012) pp.265-272.
7. "An Algorithm for image clustering and compression" by Metin KAYA.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

8. Thomas J., Ragavi B.S., Raneesha P.K., Ahmed N.A., Cynthia S., Manoharan D., Manoharan R., "Hallermann-Streiff syndrome", Indian Journal of Dermatology, ISSN : 0019-5154, 58(5) (2013) pp.383-384.
9. The complete reference JAVA by Herbert Schildt.
10. Hariharan V.S., Nandlal B., Srilatha K.T., "Efficacy of various root canal irrigants on removal of smear layer in the primary root canals after hand instrumentation: A scanning electron microscopy study", Journal of Indian Society of Pedodontics and Preventive Dentistry, ISSN : 0970-4388, 28(4) (2010) pp.271-277.
11. Bharthvajan R, Change Management - Models and Process, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 315-320, Vol. 2, Issue 1, January 2013
12. Bharthvajan R, Performance Management & Appraisal System in an Organization, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 7192-7195, Vol. 2, Issue 12, December 2013
13. Bharthvajan R, Marketing Research and Strategy-Down turn of Economy, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 5321-5324, Vol. 2, Issue 10, October 2013
14. Bharthvajan R, Exit Interviews, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 12073-12076, Vol. 3, Issue 5, May 2014
15. Bharthvajan R, ENTREPRENEURSHIP IN GLOBALISING ECONOMY IN ACCORDANCE WITH CAR INDUSTRY, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 14359-14361 Vol. 3, Issue 7, July 2014
16. Bharthvajan R, Evaluation of Training and Development Programme, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 13374-13377, Vol. 3, Issue 6, June 2014