International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 2, Issue 5, Nov 2013, 21-32

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

DESIGN OF ADAPTIVE STORAGE NETWORKS USING GENETIC ALGORITHMS

A. SHAOUT & S. SREEDHARAN

Department of Electrical and Computer Engineering, The University of Michigan Dearborn, Dearborn, Michigan, USA

ABSTRACT

The objective of this paper is to develop a fault-tolerant storage network capable of automatically managing and optimizing its reliability by monitoring and responding to failure indicators. This system is a fully automated self-healing network that is aware of its own behavior and failuzre profile, and is thus capable of proactively managing its own allocated storage units to minimize downtime and loss of information due to failure. We apply the latest research in disk failure model as a basis to implement new techniques for the construction of this fault-tolerant storage system.

KEYWORDS: Cloud Computing, Distributed Systems, Fault Tolerance, Operating Systems, Performance

INTRODUCTION

As software evolves into a service-based model offered entirely over the internet, the data managed by the software becomes increasingly centralized. As a result, software service providers require large networked datacenter infrastructures to support their customers. In such a large-scale operation, datacenter maintenance is the single largest operating cost. Furthermore, mechanical failure of components is a frequent occurrence in such large highly available networks. Failure results in information loss, service downtime and loss of customer goodwill. These losses are accentuated when business critical information is migrated to such cloud networks.

Most large datacenters are fairly homogeneous in their makeup. This is because using a limited set of hardware models allows the organization to leverage economies of scale by negotiating better prices with their hardware vendors and by developing specialized expertise in those specific hardware models. Managing downtime and failure in these networks simply involves managing and maintaining hundreds or thousands of these deployed units. Generally, datacenter maintenance involves backing up and moving data in response to hardware failure, network failure or network congestion. This activity involves manual intervention to initiate the maintenance task and monitor its progress.

There are several tools to simplify these tasks. But current tools and technologies used for backup and migration in these data center environment are generally designed for files systems and not for relational databases. Furthermore, such tools often require a considerable amount of human interaction to complete any maintenance task. The alternative to this solution is to use costly clustered databases that handle fault-tolerance implicitly. This alternative however does not scale to very large scale deployments because it presents other complications on the database management side, presenting issues like data replication, throughput & performance and failover. In this research paper we explore a new solution to all these issues.

The research conducted by this project lays the technical foundation for a new alternative – a fully automated self-healing network that is aware of its own behavior and failure profile, and is thus capable of proactively managing its own allocated storage units to minimize downtime and loss of information due to failure. The expected benefit of a system is a significant reduction on the cost of management of large storage systems while simultaneously improving the robustness and reliability of such systems.

Most contemporary database solutions use a centralized approach to data storage and management. Our approach differs from these alternatives in that it uses a decentralized storage mechanism. This mechanism works in conjunction with a centralized management orchestration service. This orchestration service, responsible for data integrity and management, uses artificial-intelligence techniques to optimize utilization of cluster components that follow a failure pattern. The failure pattern is mathematically modeled using a polynomial failure curve. The curve is used to predict the possibility of the failure of a storage device and thereby the possibility of data loss. By proactively migrating data based on the failure curve, we will demonstrate that the storage network as a whole is significantly more reliable than any contemporary deployment alternative. Furthermore, by taking this approach, we will be in a position to manage geographically distributed databases more easily than is currently possible.

To summarize, the technical objectives of this paper may be stated as:

- To develop an algorithm for self-healing networks capable of automatically managing and optimizing its performance by monitoring and responding to its state.
- To develop a model for a test environment that can be used to evaluate and simulate multi-year failure models of very large computational grids.

Background - Current Research: Our research began with a study of the state of the art in three relevant areas: hard disk failure models, cost estimation using AI techniques and cloud computing economics.

Moore [4], Schroeder [2] and Schroeder [5] conduct a thorough study of large disk populations to identify the significant factors that cause disk failure. Pinhero[6] takes the research further by defining failure in terms of disk usage profiles and I/O throughput in the Google cluster. Together the two pieces of research provide us the basis for a failure model that can be applied to a single disk based on usage profiles. Nisha [3] presents a study of the types of disk failures and their frequency on large storage networks. However the study does not provide a useful model to develop and test new algorithms for dynamic storage allocation.

We looked at several applications of fuzzy logic in optimization applications, but none applied to the problems in cloud computing or large-scale storage systems. That said Andreou [1], Ismail [9] and Shaout [11] present some interesting application of fuzzy logic for software optimization and software-based control systems. Mengshoel [10] present refinements to the classic Goldberg variation of genetic algorithms, that we use extensively in this research.

Cloud computing economics research is a relatively new field. Assuncao [8] studies the cost benefit of using cloud computing technologies. Furthermore, the research explores the problem of throughput and performance in cloud architectures, specifically in the context of service handling and throughput using different allocation strategies. The research provides us with insights into the current best-practices for resource allocation in cloud architectures. In their research at IBM, Hu [7] also presents a study of the resource provisioning problems as it pertains to optimization of available resources given a specific service level constraint.

Based on our study of existing academic papers, it was concluded that although there are pockets of research in the three areas of interest described above, no single research integrated the ideas to solve the problem of reliability of cloud computing storage infrastructures. It was also recognized that there was very little current research on the use of AI techniques to dynamically optimize cloud-based resource allocation and compare its performance against the current best practices.

The past five years have seen a significant shift in the nature of information management. Three major trends define the nature of these shifts in this *cloud computing* era:

- As storage of information become increasingly centralized, large facilities hosting thousands of computers become commonplace. These facilities are commonly referred to as data centers.
- A marked shift from structured relational representation is giving way to unstructured representation of data and content.
- Cloud-based infrastructures leverage the storage and computational capabilities of data centers and provide them on the internet as a turnkey utility service.
- As the marketplace for cloud computing service providers consolidate, the economies of scale attained by the few large providers rapidly drives down the cost of deploying information on the internet thus making it more attractive than current alternatives.

Consequently, in most corporations, these infrastructures are increasingly viewed as a cost-effective alternative to maintaining a computing infrastructure and network in-house. Consequently, business critical data, from email, to databases, to document stores, are increasingly migrating from private corporate networks to secure cloud-based infrastructures.

These marked shifts in the nature of data management are driving us toward an increasingly fragmented and distributed representation of data and metadata. To make a viable business case for the storage of information in the internet, service providers of cloud infrastructures compete on price. To the provider, this translates to ensuring the lowest total cost of ownership (TCO) while maintaining exponentially growing server networks and delivering on a contractual obligation tied to a service level. These service level agreements (SLA) typically ensure 5-sigma or better uptime and protection against permanent data loss or corruption to ensure worry-free business continuity of their client's applications.

Providing this kind of a service cost effectively is not easy. Delivering service compliant with an SLA presents a unique technical challenge: How do you reduce cost of large scale infrastructures while proactively reducing downtime?

One possible solution is to accurately predict the failure of components on their network and consequently allocate resources to avoid loss due to the failure. But in order to do this, we must first understand how to model network failure accurately.

The ideal failure model balances failure costs with benefits of implementing a new storage management system. Thus this investigation began by defining the main problem: *What are the major cost variables in the running of an infrastructure?* Answering this question defines the components of failure costs. Once the failure costs are identified appropriately, a cost model can be developed around the cost variable. Consequently, as the research is focused on reliability, the cost model can be further simplified by eliminating the cost variables that are either not relevant, too miniscule to matter or are not directly related to failure and reliability of the network.

There are several components to infrastructure costs. The major costs of running a data center typically fall under the following categories:

- **Media/Server Costs:** This is fixed cost of acquiring the server, and storage media for the infrastructure
- **Maintenance Cost:** The cost of ensuring the upkeep of the servers- fixing failures, replacing components, upgrading components and retiring aged components.

- **Facility Costs:** This cost of operation includes costs for energy, telecommunication connectivity, cooling, administrative and so on.
- **Contractual Liabilities:** Hidden liability costs from contractual obligations like the SLA. This may be offset by insurance against failure and thus will have a significant impact on the TCO.
- **Licensing Costs:** Cost of licensing the operating software and other technology required to operate the infrastructure.

For the scope of this research, we narrow our focus to maintenance costs; more specifically, we look at the cost of maintenance of storage infrastructures.

Maintenance cost typically includes the cost of repairing and/or replacing failed or obsolete components and the cost of salary overhead for the human resources involved in the operation. This model however is unsuitable for our purpose. This is because it does not account the cost of failure and its impact on the SLA. We thus conclude that we cannot use the generally accepted models for cost accounting maintenance cost; we need something new. To define the cost of maintenance effectively the costing model must also fold into its calculation some element of system failure to factor in the cost of failure. Thus we turn our focus to the failure model of the most failure-prone hardware components: *Hard Disks*.

Background –Hard Disk Models: As the focus of our research is storage systems, our interest is primarily about the failure of hard disks. This is because hard disks are still the dominant storage media on most large-scale storage networks and they are highly prone to mechanical and electrical failure.

Most hardware vendors publish a variety of specifications about their products. These specifications also include a measure called MTBF (mean time before failure) that is a metric generally used as a measure of the reliability of the disk. Although a useful mechanical reliability metric, this value does not take into account - 1) the dynamic environmental characteristic of the network it is intended to operate in, 2) the failure profile of the disk over its entire lifetime, or 3) the operational characteristic of the network. Till recently, there were very few published research related to this form on disk failure. Fortunately, that has changed. Several recent publications shed light on the impact of various factors on disk failure. To implement a fairly accurate model of disk failure, we look at the components of disk failure documented in these papers.

Perhaps the research that has most influenced our research is Failure *Trends in a Large Disk Drive Population* by Pinheiro [6] and *Disk failures in the real world* by Schroeder [2]. The authors of these two papers conducted research on large populations of commercially available hard disks and discovered specific patterns in disk failures. They concluded that disk failure occurs due to the following factors:

- **Disk Deterioration:** The expected remaining time until the next disk was replaced grows with the time it has been since the last disk replacement. This does not seem to observe an exponential curve or a Poisson distribution. However, it was demonstrated that they can be modeled based on disk usage profile like the amount of disk I/O.
- **Infant Mortality & Wear out:** Disks failure is age dependant. In particular failure rates of hardware products typically follow a “bathtub-shaped curve” with high failure rates at the beginning (infant mortality) and the end (wear-out) of the lifecycle. Figure 1 shows the generalized failure model of a magnetic disk.
- **Environmental Characteristics:** Environmental factors like temperature and frequency of power failures although mentionable, were found to have a minimal influence on the disk failure profile over long test runs. This

is perhaps because the environmental variable are unlikely to change dramatically within a given operating environment

- **Background - Generic Failure Models:** We can now build a fairly simple mathematical model to describe the cost of downtime on a deployment environment of several computer nodes. For simplicity we will limit the costing function to model loss due to the likelihood that a disk failure, failure due to a usage overload, or downtime due to outage occurs. Although the algorithm provided here is flexible enough to handle other factors that influence the cost, we have chosen to keep the model simple. The following section describes the general principle used to describe a generic failure model.

Let the computer node be assigned a unique number ranging from 0 to $node_{max}$. Then, we may assume that the node allocation $\mathbf{N_D}$, of a deployment environment D is an integer given by:

$N_D = n; 0 < n < node_{max}$

and the 'Calamity' Cost of failure of a Deployment **D** (a measure proportional to the size of the storage units allocated on it when it fails), may be any arbitrary positive real number that is a measure of the weighted failure cost.

$C_D = r; r>0$ and $r \in R$

Now, if we assume that failure is limited to three independent events: disk failure, overload and downtime. Then the risk of failure/downtime on deployment, at time **t**, is defined as the probability of likelihood of failure due to the three independent events may be derived as follows:

If the probability of an arbitrary event occurring in the network is $\mathbf{P(N_D, t)}$, then the probability of it not occurring is calculated by $\mathbf{1- P(N_D, t)}$. Now, if we have the three aforementioned independent events to consider, then the probability of the system not failing $\mathbf{\tilde{R}(D,t)}$ due to either of the three events is given by

$\tilde{R}(D,t) = (1-P_{Disk}(N_D, t)) * (1–P_{Overload}(N_D,t))* (1–P_{Outage}(N_D ,t))$

Thus the probability of failure **R(D,t)**, due to either one of the three events is given by:

$R(D,t) = 1- \tilde{R} (D,t)$

Thus we can infer that, Cost of Deployment **E (D, t)** at time **t**,

$E (D, t) = C_D * R(D,t)$

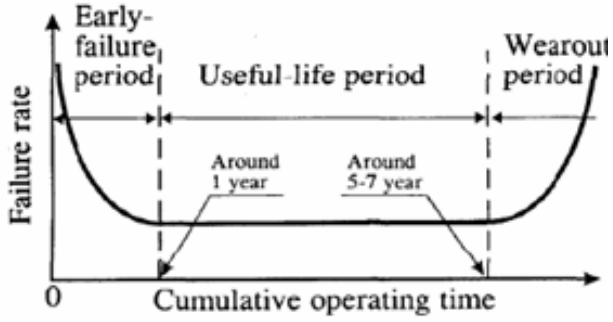

Figure 1: Generalized Failure Curve of Hard Disk Schroeder [2]

In the equation above, the probability of overload may be defined as an equation proportional to $\mathbf{U_{Usage}}$ (the total storage units on a node at a particular moment in time) and inversely proportional to $\mathbf{U_{Capacity}}$ (the total storage capacity of a particular node)

$P_{Overload}(N_D,t) \propto U_{Usage}(N_D, t) \propto 1/U_{Capacity}(N_D)$

and, the probability of outage

$P_{Outage}(N_D,t)$ = {0 if node is turned off, 1 if node is active}

and, the probability of disk failure,

$P_{Disk}(N_D,t)$ =

{ .10 if $t-t_{lastfailure} < 3$ month

.005 if $t- t_{lastfailure} < 1$ year

.02 if $t- t_{lastfailure} < 2$ year

.05 if $t- t_{lastfailure} < 3$ year

.04 if $t- t_{lastfailure} < 4$ year

.045 if $t- t_{lastfailure} < 5$ year

}

Note: P_{Disk} was derived using statistics based on research by Eduardo [6] at Google. The paper suggests three disk failure models and we run tests using each of these models to find the most suitable model for a given target environment.

Then the total cost of operation at a particular time is a cumulative sum of cost on all nodes:

$TCO(t) = \sum_{i=0:n} E(D, t)$

where **n** is the number of deployments.

Figure 2 represents a class diagram of a software implementation of the model.

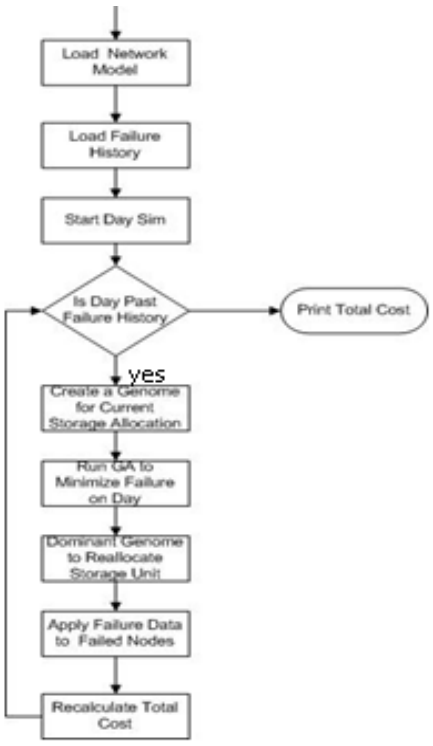

Figure 2: Algorithm of the Simulation Engine

Detailed Description - Overview: The model proposed in the previous is fairly simple. The simple mathematical model was used to create a simulation program that imports a database of network nodes (including its physical properties) and a database of historical failure incidents on that network. This program then uses a genetic algorithm to predict the failure of the each node based on the mathematical model. It then calculated the optimal allocation of storage units at a given point in time (for the purpose of this research, a storage unit is a single unit of storage – a customer account, a database or a document management system). Based on this optimal allocation model, we move storage units across the network to avoid loss due to failure. If a storage unit is on a failed node during the simulation, we calculate a weighted cost of failure proportional to the size of the storage unit. The simulation is run over a period of 1000 days and the accumulated total failure cost is used as a comparable measure of the maintenance cost. To account for the overhead of downtime due to migration, we also attach a weighted migration cost to the total cost function. This simulation is run over several variations to the cost function and the genetic algorithm itself and the behavior of the simulated environment are measured. The failure scenario of each simulation run uses actual failure data from a real world deployment of a cloud network. This ensures that the model closely resembles an actual deployment.

As a result, if the model is effective, the system will successfully avoid allocating a storage unit on a node when it fails. To evaluate the performance of this model, we compare the total failure costs of our new algorithm against the total failure cost of a standard round-robin allocation scheme for storage unit. The round-robin allocation scheme is the simplest and most common allocation scheme used in datacenters to allocate network storage resource. Such an allocation scheme requires only the set of available nodes as a input to the algorithm and is thus inherently simple. It has the added benefit that it spreads the risk of failure across the set of available machines thus minimizing risk through diversification. Figure 2 is a simplified flowchart of the execution of the algorithm.

Detailed Description – Methodology: Our project focuses primarily on the reliability of distributed storage and the minimization of total cost of operation and reduction of failure cost. In this project we have invented a technology to maximize the utilization of our servers and to automate repetitive administrative tasks that today require human intervention. Our approach is to use a centralized orchestration service that uses AI techniques to optimize utilization of cluster components. By taking this approach we will be in a position to manage geographically distributed databases more easily than is currently possible.

Our first step in the project was to create a simulation environment of large-scale cluster environment based on 5-years of failure data published by the Los Alamos National Laboratory (LANL) super computing facility[1]. Using this simulation environment as our test bed, we experimented and implemented several calamity cost models to reduce the impact of failure on a simulated deployment of a 1000 node cluster managing up to 150TB of data. A typical simulation run took over 12 hours to execute. During the course of the project, three different models were attempted before we concluded on a final solution, each improving on the predecessor. At each stage, a limitation in our model resulted in a suboptimal algorithm. The final algorithm was able to reduce cost of deployment by as much as 15% over the round-robin allocation scheme.

Approach 1: Simple Grid Optimization Using Genetic Algorithm

Principle: A cost function based on disk failure models, outages and overload, was optimized by a genetic algorithm.

[1] Source: <http://institute.lanl.gov/data/lanldata.shtml>

Rationale: Genetic algorithms (GA) provide an efficient and proven method for optimizing real world cost models. GA based algorithms are simple and yet highly adaptive to sudden variations common in the real-world. Furthermore, if a generic mechanism to define cost functions can be implemented, the behavior of the GA can be modified simply by implementing a new cost function that the GA can then optimize.

Description: During this experiment, a generic mechanism to define cost functions was implemented in software. The genetic algorithm was then allowed to optimize the deployment during a simulated test run. Two deployment models were used – round-robin and first-fill (a machine is filled to max before another is used). Based on these experiments, we discovered that infant mortality (during the first year of disk deployment) significantly degraded the performance of the algorithm. The conclusion was that an accurate disk deterioration model was a perquisite for accurate network failure prediction.

Approach 2: Advanced Automatic Grid-Optimization Algorithm

Principle: Use cost functions based on disk utilization profiles to improve the algorithm.

Rationale: Based on the understanding of the limitations of our first approach, a modification to the costing function was made to incorporate three disk utilization models. The failures models appropriate for the disk utilization model, presented by Pinhero [6], were applied using the software.

Description: This experiment was inspired by insights published by Pinhero [6] that disk failure models are highly influenced by usage profiles. The published document described failure statistics for three (high, medium, low) models based on the amount disk I/O. This failure statistics was used to define a failure curve and thereby the failure function $\mathbf{P_{Disk}}$ using Newton's interpolation. The failure model was then used as the basis for optimized storage allocation. The three models were run against the simulation environment with considerable success.

It was discovered that the best model that suited the LANL simulation data, was a low disk I/O model. This agreed with the fact that the LANL cluster computing facility was indeed a relatively low disk I/O, high-CPU computing cluster and not a high I/O storage grid. However, the limitation of the algorithm was that random real-world 'black-swan' events derailed the genetic algorithm, when it crowded utilization on a few good machines that failed randomly. We concluded that a crowding avoidance mechanism had to be implemented.

Approach 3: Advanced Automatic Grid-Optimization Algorithm with Crowding Avoidance

Principle: Enhance the existing algorithm to avoid 'black swan' failures by spreading risk.

Rationale: By incorporating an exponentially rising cost proportional to utilization crowding, we spread the risk of failure over several machines.

Description: The final implementation ironed out the limitations of the existing algorithm by incorporating a crowding tax to the costing function. The algorithm was penalized when utilization was localized to select set of machines. The final algorithm provided us the results that we expected. We also experimented with variations in population and generations and its effect on the algorithm. This was necessary, because the choice of these GA variables have an exponential impact on the length it take the algorithm to compute an optimal solution.

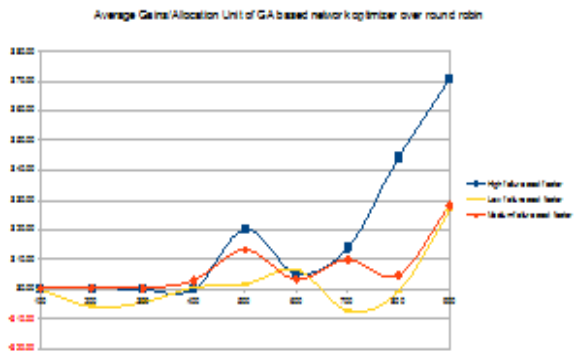

Figure 3: Average Cost vs. Failure Cost Factors

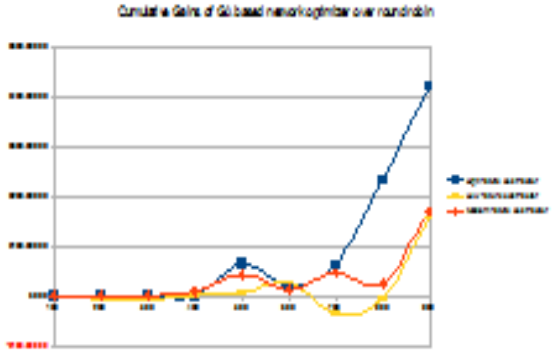

Figure 4: Cumulative Cost vs. Failure Cost Factors

Detailed Description – Algorithm: At the heart of the software is an optimization algorithm implemented using a genetic algorithm. Each node in the network is assigned a unique integer identifier. For each epoch in the simulation (a single day in the simulation run), the system generates a genotype consisting of a list of integers. The size of the genotype is the number of active storage units (also referred in the program as deployments). Thus a genotype is a mapping of a possible allocation scheme. Each integer in the genotype is set to the current node assignment of a storage unit in the network.

A genetic algorithm then evolves the allocation over several generations. The optimization function required by the genetic algorithm calculates the current calamity cost of a given allocation scheme. Each genome is an evaluation for its associated calamity cost and the algorithm is setup to minimize the calamity cost in a population. The most optimal genome generated by the algorithm defines the most optimal allocation of deployments on nodes at that epoch in time. The optimal genome is used to reallocate the storage units and migrated across nodes. Figure 2 is a simple flowchart of the algorithm described here.

The failure cost function is constructed by calculating the individual failure components and calculating the probability of a failure R(D,t). The of disk failure curve itself is calculated using a simple Newton-Raphson interpolator based on the disk failure profiles prescribed by the Google research in Pinhero [6].

Test Results: The final solution leveraged regression analysis, published by Pinhero [6], of disk failure models based on three I/O profiles of disks used in a large active data center. The failure curves were used to model a 'calamity' cost functions that predicted the cost of disk failure on a network at a specific period in time, given a network topology and database resource requirements on the network. A genetic algorithm was then used to optimize the costing function, which was then used to predict the optimal storage allocation for a given resource requirement on the network. The genetic algorithm was then allowed to run over a simulated period of 5 years. A typical test run takes about 10 hours on a standard dual core computer running WindowsXP.

Real-world failure data from the Los Alamos National Laboratory (LANL), supercomputing grid was used to simulate failure during the run. The test data provided disk failure statistics of a 1000 computer cluster node managing 150TB of data. The cumulative failure cost of the run over the 5 year period was compared with the cumulative cost of a baseline test run. The base line test run uses the current best practice of a round-robin resource allocation algorithm used by most data centers. Based on the results of each test run, an iterative approach was used to develop and improve the original genetic algorithm. The results from each test run was analyzed and based on findings in each run, new variables were introduced into the original costing function. The choice of variables was driven by known facts about the network like – I/O profile, resource crowding, and disk infant mortality. The algorithm was then further refined using weights in the costing function.

Based on several test runs, we tested:

- The impact of failure models P_{Disk} on the effectiveness of the model.
- The impact of a crowding avoidance on the effectiveness of the model.
- The impact of changing the properties of the genetic algorithm to improve the performance of the algorithm.

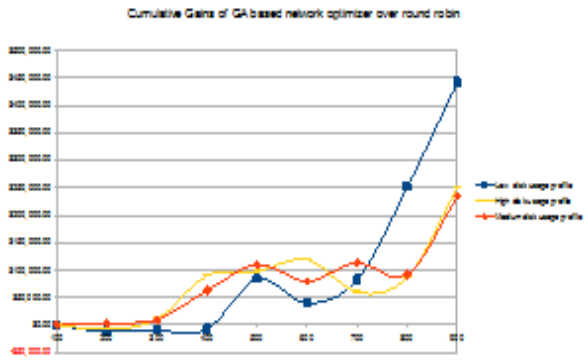

Figure 5: Cumulative Cost vs. Disk Usage Profiles

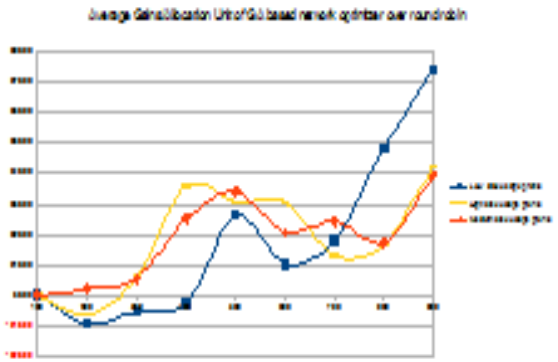

Figure 6: Average Cost vs. Disk Usage Profiles

Test Results- Impact of Crowding Avoidance: Early in our test runs, we discovered that the algorithm tended to crowd storage units in an attempt to optimize the failure costs. However, a random failure, or a 'black swan' incident, would cause the crowded node to fail, causing the performance of the algorithm to deteriorate significantly. Consequently, we implemented a crowding avoidance mechanism by introducing a higher cost on nodes with high utilization.

The results of the tests are presented in Figure 3 and Figure 4. The graph showing the average cost measures the average calamity cost per day of the test run. On the other hand, the cumulative cost graph display the total cost accumulated from the beginning of the run till that specific day. Based on the graph we can infer that the GA took around 400 simulation days before it could effectively improve on the round-robin algorithm. Furthermore, the usage of the low-disk I/O profile was the best failure model for the LANL storage network. However, it took over 700 days before it could leave behind the effectiveness of the other failure models.

Test Results- Impact of Failure Cost Factor: Based on our test runs, we find that the failure model has the most impact on the effectiveness of the algorithm. The failure model, for its part, defines the usage profile of the network. We used the three models proposed in the research by Pinheiro [6] and tested the effectiveness of each of the model against the LANL failure database.

We find that the algorithm accurately predicted that the low disk I/O failure model was most appropriate model for the LANL network. This is true because the LANL database is a computational cluster that is more CPU intensive when compared to its disk I/O.

The results of the tests are presented in Figure 5 and Figure 6. Based on the graph we observe that although the general trend new tests demonstrate the general characteristics of the original tests, the use of the cost function on an average improved over the original model by over 10%. The performance was compared with a baseline run of the round-robin allocation scheme over a 3-year simulation period.

Test Results - Impact of Population Size: As with all genetic algorithms, the population size has a significant impact on the speed of the algorithm. We attempted to reduce the population size from 100 to 33 genotypes. This significantly sped up the algorithm but the results were inaccurate. Based on our tests, for a network of a 1000 machines, a genotype of 100 or 1/10th of the total population was necessary to produce an effective result. During each test run, the algorithm also ran

100 generations before selecting the most optimal solution in the algorithm. The results of the tests are presented in Figure 7 and Figure 8. In comparison with the earlier tests, the new tests demonstrated an oscillating cost graph with no clear trend. This implies that the GA was unsuccessful in optimizing the cost function over the duration of the test run.

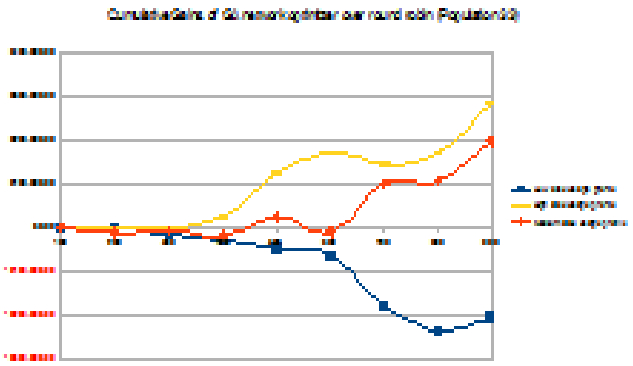

Figure 7: Cumulative Cost vs. Disk Usage Profiles with Reduced Population

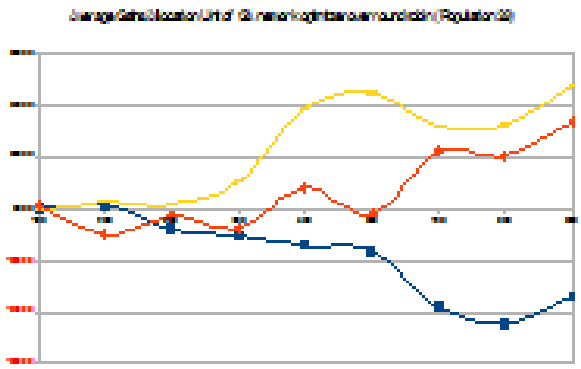

Figure 8: Average Cost vs. Disk Usage Profiles with Reduced Population

CONCLUSIONS

Our primary achievement at the conclusion of the project was the development of a new method to optimize the storage utilization of a distributed computing grid for cloud based internet solutions. Although focused on grid-computing problems, the solution has broad application on any network-based failure models where failure of individual components has a significant impact on operating cost or risk to individuals. This application includes failure optimization of - transportation networks, power grids, water distribution networks and wireless networks.

The final solution also provides a generic model to describe risk in a network and a methodology to optimize data center operational cost based on its unique risk model. Within the domain of storage computing the significant learning is that failure on a network can be effectively managed based on a few simple criteria – usage profile, risk distribution and failure models. Finally, our proposed solution also opens up several possibilities for future research on new failure indicators and its impact on operating cost.

In conclusion, the following results were identified as results of the project:

- The genetic algorithm developed during the project was proven to reduce the failure related downtime by over 55% during the test run.
- The performance of the algorithm was found to be greatly influenced by the disk usage profile. Therefore it was imperative to develop an accurate failure model for a specific data center based on its usage. However, the three models used in our research provide an adequate starting point for the development of the ideal failure model.
- It was discovered that on occasion, a random 'black swan' failure event could throw the genetic algorithm (GA) off track. This is to be expected in any real-world operational environment because the GA does not account for all possible variables in the target environment. The impact of such an event can be minimized by preventing the resource crowding nature of the GA. When compensated for this behavior the GA was able to adapt to these events effectively as well.

Conclusion: In conclusion to this research, the following areas of study may be planned as future avenues of exploration:

- The algorithm proposed by this research must be used to test its validity against other real world cluster failure data. Such failure data must cover an epoch of more than 700 days for the algorithm to be effective.
- The technique proposed in this research will need to be implemented into a cluster management technology and tested in a real cloud storage network.
- The techniques used for failure modeling has applicability in network-based engineering problems including transportation, distribution and communication. If failure data in such networks are available, the technique may be evaluated for its applicability to improve the performance and reliability of such networks.

REFERENCES

1. Andreas S. Andreou, and Efi Papatheocharous *Software Cost Estimation using Fuzzy Decision Trees* IEEE Computer Press, Washington D.C., 2008, ISBN: 978-1-4244-2187-9.
2. Bianca Schroeder, and Garth A. Gibson *Disk failures in the real world:What does an MTTF of 1,000,000 hours mean to you?* FAST'07: 5th USENIX Conference on File and Storage Technologies, 2007, Pp. 1-16
3. Nisha Talagala, and David Patterson *An Analysis of Error Behavior in a Large Storage System* Computer Science Division, University of California at Berkeley, 1999
4. Richard L. Moore, Jim D'Aoust, Robert H. McDonald and David Minor; *Disk and Tape Storage Cost Models* San Diego Supercomputer Center, University of California San Diego
5. B. Schroeder and G. Gibson; *The Computer Failure Data Repository (CDFS)*, Proc. of the Symp. on Reliability Analysis of System Failure Data, 2007
6. Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz Andr´e Barroso *Failure Trends in a Large Disk Drive Population* Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07), February 2007, Pp. 17-28
7. Ye Hu1, Johnny Wong1, Gabriel Iszlai2 and Marin Litoiu3 *Resource Provisioning for Cloud Computing* IBM Canada Ltd. 2009
8. Marcos Dias de Assunção, Alexandre di Costanzo, and Rajkumar Buyya *Evaluating the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters* HPDC'09, June 11–13, 2009, Munich, Germany. 2009
9. Ismail Saritas, et.al, *Fuzzy Expert System Design for Operating Room AirCondition Control Systems*, 2007
10. Ole J. Mengshoel and Davie E. Goldberg *Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement* University of Illinois
11. Adnan Shaout, Taisir Eldos and Najamuz Zaman *Scalable Autonic Processing Systems* International Journal of Computer Applications in Technology, Volume 16, No. 1, 2003