Uptime-Optimized Cloud Architecture as a Brokered Service

Sreekrishnan Venkateswaran
IBM Corporation, India
Email: s_krishna@in.ibm.com

Santonu Sarkar
BITS Pilani, K K Birla Goa Campus, India
Email: santonus@goa.bits-pilani.ac.in

Abstract—**Enterprise workloads usually call for an uptime service level agreement (SLA) at the pain of contractual penalty in the event of slippage. Often, the strategy is to introduce ad-hoc HA (High Availability) mechanisms in response. Implemented solutions that we surveyed do not mathematically map their availability model to the required uptime SLA and to any expected penalty payout. In most client cases that we observed, this either resulted in an over-engineered solution that had more redundancies than was required, or in an inadequate solution that could potentially slip on the system uptime SLA stipulated in the contract. In this paper, we propose a framework backed by a model, to automatically determine the HA-enabled solution with the least TCO (total cost of ownership) for a given uptime SLA and slippage penalty. We attempt to establish that our work is best implemented as a brokered service that recommends an uptime-optimized cloud architecture.**

I. Introduction

Cloud Computing[1], specifically Infrastructure as a service (IaaS), is a software-driven, algorithmic paradigm that supplies on-demand, pay-per-use compute capability with an upfront uptime assurance.

There have been many efforts such as [3] from the academia and from the practice over several decades to design high availability on computer systems that propose new or improved methods for failover clustering. Papers such as [1] propose redundancy models for high availability of computer clusters. The work by [6] describes a clustered architecture that increases availability of middleware components. [5] and [7] pertain to improving cluster failover mechanisms. White papers such as [4] describe clustering strategies specific to certain operating systems.

However, to our knowledge, given a base cloud-hosted deployment architecture, there is no model-backed framework that has been proposed, which can automatically simulate and evaluate all HA (High Availability)-enabled variants of that base architecture, and determine the option that is optimal in terms of cost of clustering, given an uptime target and an associated financial burden to compensate non-adherence.

Cloud environments are increasingly turning hybrid, and cloud service brokers further simplify IT consumption while also bringing flexibility of choice. In this paper we propose a
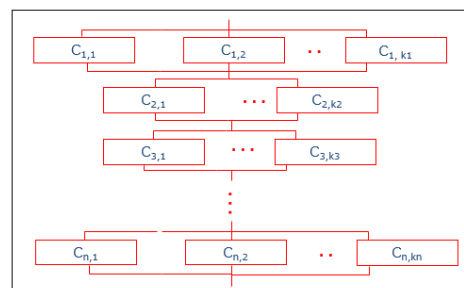
[1]The NIST definition of cloud computing. Special Publication 800-145, 2011

Fig. 1. Cloud-hosted Clustered IaaS Architecture of a System S

framework backed by a probabilistic model that can be implemented by a cloud service broker to automatically provide an uptime-optimized cloud solution architecture as a service. A broker sits at a cross-cloud cross-customer vantage point, that we show, helps model and evaluate permutations of cluster technologies that constitute cloud-hosted systems.

The paper has been organized as follows: In Section II, we first built a probabilistic model to construct expected uptimes for cloud-based systems that we consider as a serial combination of clustered compute, storage and network entities. We next simulate different levels of redundancies into these clusters and evaluate them based on the associated TCO (total cost of ownership). The TCO includes the cost of the infrastructure needed to embed redundancies, the labor needed to sustain the HA, and any penalty to be paid by the provider to compensate SLA slippage. Our proposed framework evaluates the TCO of all permutations of HA that can be engineered. The architecture that corresponds to the HA level that yields the least TCO is recommended as the optimal solution. We propose this framework to be realized as-a-service by a cloud broker since that is an entity at the best vantage point to gather values for the variables in our model. In Section III, we describe a client case study that establishes savings of up to 62% by leveraging our technique. Section IV examines any risks pertaining to the validity of our model evaluation. Finally, we conclude the paper and highlight future research directions.

II. Modeling Brokered Uptime-aware Cloud Architecture

Let us model a cloud-hosted system S as a serial combination of n clusters Let each cluster C_i be composed of K_i nodes as shown in Figure 1.

A. High Availability (HA) Solutions in Practice

In a cloud based system, each cluster is built with the "k-redundancy" model. Let $\widehat{K_i} \le K_i$ be the maximum number of failed nodes that can be tolerated by the HA infrastructure of C_i. For instance, the case study shown later in this paper uses a VMWare ESX based HA solution with 3+1 cluster configuration, which means that the cluster has $K_i = 4$ nodes, of which three nodes must be *active* for the cluster to be operational, with one standby node, i.e. $\widehat{K_i} = 1$. If one of the operational nodes breaks down, the HA infrastructure, VMWare ESX in this case, brings the standby node to action. There is a small latency involved depending on the nature of the standby mode (hot, warm or cold) during which the cluster temporarily remains unavailable. This latency is commonly known as the *failover time*. The failover time t_i (in minutes) for C_i is a sum of (i) Time to detect that the currently active node in cluster C_i is down (ii) Time to bring up the failover node if it is on standby and (iii) Time for the failover node to take over from the primary node.

Now consider the scenario when the number of node outages in C_i exceeds $\widehat{K_i}$, which is 1 in this example. In that case, the cluster *breaks down* beyond immediate recovery.

Because the cloud based system we consider is a serial combination of n clusters, the entire system can *a)* Witness outage when one or more clusters break down completely or *b)* go down temporarily during a cluster failover.

B. System Failure Model

Since the above two events are mutually exclusive[2], the overall downtime probability of S can be expressed as:

$$D_s = B_s + F_s \tag{1}$$

where
B_s = System downtime probability due to breakdown of one or more clusters and
F_s = System downtime probability due to failover latencies while clusters recover from node outages

1) Calculating $\mathbf{B_s}$: Let the probability that a node in cluster C_i is down be P_i. Then the probability that cluster C_i is up is $\sum_{j=K_i-\widehat{K_i}}^{K_i} \binom{K_i}{j}(1-P_i)^j \cdot P_i^{K_i-j}$.

So system downtime probability

$$B_s = 1 - \prod_{i=1}^{n} \sum_{j=K_i-\widehat{K_i}}^{K_i} \binom{K_i}{j}(1-P_i)^j \cdot P_i^{K_i-j} \tag{2}$$

2) Calculating $\mathbf{F_s}$: Let f_i be the average number of failures experienced by a node in cluster C_i in a year. Total downtime due to failover transactions in cluster $C_i = f_i t_i (K_i - \widehat{K_i})$ since there are $(K_i - \widehat{K_i})$ active nodes in cluster C_i at any given point of time. Since P_i is the probability that a node in cluster C_i is down, it is also the probability that a *currently active node* in C_i is down. However, multiple clusters might simultaneously be experiencing failover transactions. The downtime
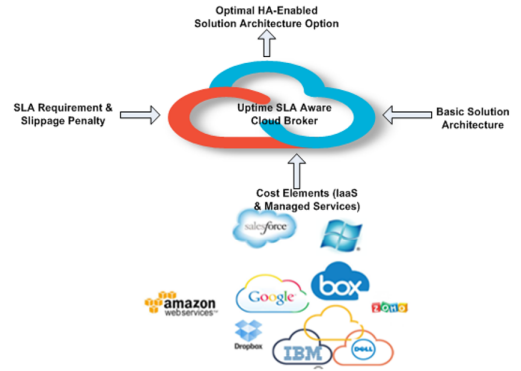

Fig. 2. Uptime Service Level Aware Cloud Brokerage

probability due to failover transactions in cluster C_i, when no other clusters are experiencing outages is $F_s(C_i) = \frac{f_i t_i (K_i - \widehat{K_i})}{\delta} P(X_i)$, where X_i is the event that none of the clusters other than C_i are experiencing failover events[3], $P(X_i)$ is the probability of this event, and δ is 525600, the number of minutes in a year. Since $P(X_i)$ is the probability that all the currently running $(K_j - \widehat{K_j})$ nodes in every cluster $C_{j,j \ne i}$ are up and running, $P(X_i) = \Pi_{j=1}^{n,j\ne i}[(1-P_j)^{K_j-\widehat{K_j}}]$. Substituting $P(X_i)$, we get $F_s(C_i) = \frac{f_i t_i (K_i - \widehat{K_i})}{\delta} \prod_{j=1}^{n,j\ne i}[(1-P_j)^{K_j-\widehat{K_j}}]$. Generalizing, the system downtime probability due to all failover transactions across clusters

$$F_s = \sum_{i=1}^{n} F_s(C_i) = \sum_{i=1}^{n} \frac{f_i t_i (K_i - \widehat{K_i})}{\delta} \prod_{j=1}^{n,j\ne i} [(1-P_j)^{K_j-\widehat{K_j}}] \tag{3}$$

We can apply Equation 2 and 3 to compute the total downtime probability D_s of system S in Equation 1.

Total uptime probability of S is

$$U_s = 1 - D_s \tag{4}$$

C. Service Level Aware Model with Hybrid Cloud Brokerage

In the framework that we propose, a cloud service broker is provided the following inputs as shown in Figure 2:

1) The base cloud solution architecture in terms of the compute/storage/network topology on a specified underlying hybrid environment, mapped to customer requirements
2) The uptime SLA requirement U_{SLA}, expressed as a percentage, along with the contractual slippage penalty SP per unit time (say per hour) of system unavailability
3) Being a hybrid cloud services broker, it is aware of the rate-carded price C_{HA}, which is the monthly infrastructure cost to engineer redundancy for the proposed level of HA plus the labor cost needed to deploy and sustain that HA for a period of one month

By virtue of its vantage point above clouds and being the provisioning pane into these clouds across customers spanning a long timeline, the broker determines and maintains a database of

---

[2]Disregarding the possibility of an unrecoverable failure during cluster failover

[3]We ignore the error of counting intra-cluster node failover times when more than $\widehat{K_i}$ nodes in C_i fail simultaneously

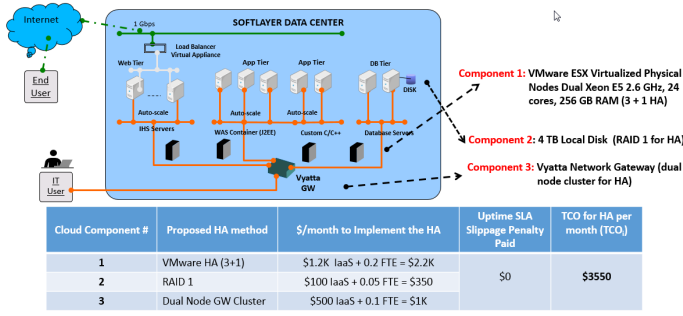Fig. 3. HA with As-Is Strategy (Also Option #8)

Cloud Component #	Proposed HA method	$/month to implement the HA	Uptime SLA Slippage Penalty Paid	TCO for HA per month (TCO_i)
1	VMware HA (3+1)	$1.2K IaaS + 0.2 FTE = $2.2K		
2	RAID 1	$100 IaaS + 0.05 FTE = $350	$0	$3550
3	Dual Node GW Cluster	$500 IaaS + 0.1 FTE = $1K		

1) The P_i and f_i across IaaS components across clouds
2) The t_i for various components in the underpinning hybrid environment

Using the above, the brokerage service models all permutations of HA-enabled system constructs (k^n possibilities, where n is the number of clusters and k is the number of HA choices per cluster) that conform to the input base solution architecture. Further, for each candidate system deployment i, it estimates the monthly TCO_i (total cost of ownership) as the sum of (i) The cost to implement and sustain the proposed HA-enabled system and (ii) Any expected slippage penalty, which is calculated by determining expected system uptime using Equation 4 and applying penalty clauses to projected downtime durations beyond the contractual SLA:

$$TCO_i = C_{HA} + (\frac{U_{SLA}}{100} - U_s)\frac{\delta}{12 \times 60}SP$$
$$= C_{HA} \text{ when } (\frac{U_{SLA}}{100} \le U_s) \quad (5)$$
$$OptCh = \min_{i=1, k^n} TCO_i \quad (6)$$

Note that $(\frac{U_{SLA}}{100} - U_s)\frac{\delta}{12\times60}$ converts downtime to slippage hours a month. The brokerage service then chooses the HA-enabled topology with the minimum TCO (i.e. $OptCh$) as the recommended HA strategy.

III. EVALUATING THE MODEL WITH A CLIENT CASE STUDY

We considered a customer cloud solution designed as a three tier architecture comprising of application, database and web service tiers. The system, deployed on the IBM SoftLayer Cloud [2], was a serial combination of 3 clusters at the compute, storage and network layers, respectively. The system uptime availability stipulated in the contract was 98%; we assume a penalty of $100 per hour of outage for illustration. In response, the provider added an adhoc level of HA in each of the three layers. The deployed solution and the cost to implement the HA is shown in Figure 3. The cost includes the monthly infrastructure cost of clustering on the SoftLayer cloud and the monthly labor (at $30/hour in this case) to deploy and sustain the HA layers.

A. Leveraging our Model

We then applied our proposed framework that was implemented and embedded inside a cloud service broker. Our algorithm modeled all permutations of HA-enabled system constructs (k^n possibilities, where, in our case $k = 2$ and $n = 3$) that conform to the input base 3-tier solution architecture. Figure 4, for instance, depicts a permutation where there is no HA or clustering in any component in the system; Figure 5 is an embodiment of the same base architecture where only the network layer is clustered via dual gateways; the data in Figure 6 is associated with an HA architecture where only the storage layer has redundancy via RAID-1 (Redundant Array of Independent Disks), and so on. For each candidate system instance, our algorithm, implemented as a brokered service, calculated the TCO. The service finally chose the HA-enabled topology with the minimum TCO as the recommended hosting strategy for the workload in question. The 2^3 solution options and their associated TCOs are show from Figure 3 to Figure 9 (Option #7 that depicts HA for compute and storage is not shown other than in the tabular summary of Figure 10).

B. Summary of Results & Resulting Cost Efficiency

In Figure 10, we compare the TCOs between the as-is strategy (Figure 3) and our framework that simulated clustered scenarios, modeled associated TCOs, and determined the architecture with the minimum TCO. Our service thus evaluated all possible HA-enabled variants of the base configuration and applied them to Equation 6 to yield the recommended deployment, which is solution option #3 (Figure 6). If the possibility of slippage penalty is to be minimized, the recommended option is #5 (Figure 8). The savings that accrued as

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	None	0	$0		92.17% (43 hours slippage per month)	$100 per hour	$0 (HA) + $4300 (penalty) = **$4300**
2	5%	2	None	0	$0	98%			
3	2%	1	None	0	$0				

Fig. 4. Solution Option #1: No HA in the System

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	None	0	$0		94.01% (30 hours slippage per month)	$100 per hour	$1000 (HA) + $3000 (penalty) = **$4000**
2	5%	2	None	0	$0	98%			
3	2%	1	Dual Node GW Cluster	1 minute	$500 IaaS + 0.1 FTE = $1K				

Fig. 5. Solution Option #2: HA For Network Only

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	None	0	$0		96.78% (9 hours slippage per month)	$100 per hour	$350 (HA) + $900 (penalty) = **$1250**
2	5%	2	RAID 1	30 seconds	$100 IaaS + 0.05 FTE = $350	98%			
3	2%	1	None	0	$0				

Fig. 6. Solution Option #3: HA For Storage Only

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	VMware HA (3+1)	6 minutes	$1.2K IaaS + 0.2 FTE = $2.2K		93.04% (37 hours slippage per month)	$100 per hour	$2200 (HA) + $3700 (penalty) = **$5900**
2	5%	2	None	0	$0	98%			
3	2%	1	None	0	$0				

Fig. 7. Solution Option #4: HA For Compute Only

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	None	0	$0				
2	5%	2	RAID 1	30 seconds	$100 IaaS + 0.05 FTE = $350	98%	98.71% (0 hours slippage)	$100 per hour	$1350 (HA) + $0 (penalty) = $1350
3	2%	1	Dual Node GW Cluster	1 minute	$500 IaaS + 0.1 FTE = $1K				

Fig. 8.   Solution Option #5: HA For Storage & Network

Cloud Component #	Downtime without HA (P_i) (from Broker)	Average yearly failures (f_i) (from Broker)	Proposed HA method	Failover latency in HA mode (t_i) (from Broker)	$/month to Implement the HA (C_{HA})	Target Uptime SLA (U_{SLA}) (from Contract)	System Uptime with this architecture (U_i) (from Eq. #5)	Uptime SLA Slippage Penalty (SP)	Expected TCO for HA per month (Eq. #6)
1	1%	1	VMware HA (3+1)	6 minutes	$1.2K IaaS + 0.2 FTE = $2.2K	98%	94.91% (23 hours slippage per month)	$100 per hour	$3200 (HA) + $2300 (penalty) = $5500
2	5%	2	None	0	$0				
3	2%	1	Dual Node GW Cluster	1 minute	$500 IaaS + 0.1 FTE = $1K				

Fig. 9.   Solution Option #6: HA For Compute & Network

a result of applying our method was thus close to 62%. Note that the actual savings realized will depend on how adhoc the original redundancy engineering has been.

C. Improving the Time Complexity of the Algorithm

The technique we propose has exponential complexity. So if there are n components in a cloud-hosted solution system that can potentially be clustered, the algorithm has a time complexity of $O(k^n)$. However, the value of n in practice is usually low, most often under 10 and the number of HA choices do not turn out to be significantly high.

The algorithm can be rendered more time efficient by pruning solution paths that are already optimized. The algorithm starts by evaluating all HA permutations where only one component is clustered, then proceeds to permutations where two components are clustered, and so on. If a particular permutation yields an uptime greater than what the contractual SLA stipulates, super-sets of that permutation can be pruned since those will increase uptime (beyond the SLA) while also increasing cost. For example, after evaluating option #5, option #8 can be clipped from the search tree without evaluation.

IV. Threats to Validity

The following risks exist pertaining to *construct validity*: We rely on the cloud broker's long-term visibility across diverse deployments of multiple customers to gather and maintain values of P_i, t_i, and f_i. There could be skews in these values influenced by the demand and supply commerce pertaining to the broker's marketplace, but over the long-term, any

Solution Option #	Proposed HA-Enabled Solution	SLA Violation/Penalty Expected?	Expected TCO for HA per month
1	No HA for Compute, Storage and Network Elements	Yes	$4300
2	No HA for compute, No HA for storage, 2-node cluster for NW GW	Yes	$4000
3	No HA for compute, RAID-1 for disks, No HA for NW GW	Yes	$1250
4	HA for compute, No HA for storage, no HA for NW GW	Yes	$5900
5	No HA for compute, RAID-1 for disks, 2-node cluster for NW GW	No	$1350
6	HA for compute, No HA for storage, HA for NW GW	Yes	$5500
7	HA for compute, HA for storage, no HA for NW GW	Yes	$2850
8	HA for Compute, RAID-1 for disks, 2-node cluster for NW GE	No	$3550

Savings that accrued as a result of leveraging our proposed strategy:

Monthly TCO with As-is Strategy (Option #8)	Observed Monthly TCO with Proposed Framework (Model & Evaluate all options as a Brokered Services)	% Savings
$3550	$1350	62%

Fig. 10.   Summary of Results & Resulting Cost Efficiency

contortions should smooth out. Also, the reliability of the broker has not been considered in our model.

On risks pertaining to *internal validity*, SLA slippage penalty is a techno-commercial aspect, so there could be non-technical aspects at play when a provider fashions a strategy to address an SLA. This paper does not venture into territory outside of mathematics and technology.

We have attempted to establish *external validity* by applying our model to one family of solutions that apply hypervisor-level clustering for compute, RAID for data stores, and dual clustered gateways for network. In future work, we plan to expand our research and further generalize our conclusions by applying to more customer case studies and broader HA strategies such as described in the next section.

V. Conclusion and Future Work

In this paper, we proposed a framework and a supporting model to automatically simulate and evaluate all HA-enabled variants of a given base cloud architecture and determine the solution option that is minimalist in terms of total cost of ownership, given an uptime target. We also showed that cloud brokers are at good vantage point to leverage our framework and provide an uptime-optimized cloud solution architecture as a service. We applied our framework to a client case study and determined that our technique, had it been applied, would have resulted in a 62% savings on the cost that was expended to implement high availability.

Our model and framework needs to be tested on complex scenarios that include OS clustering for compute, Software Defined Storage and clustered file systems for data stores, multi-pathing for storage I/O, and BGP (Border Gateway Protocol) over dual circuits for network redundancies. Moreover, though our technique is agnostic of the underpinning cloud and managed service environments, our testing has been on a single cloud (IBM SoftLayer) and not on a hybrid environment. We propose to address these in future work. The larger goal of our research is to design what we envisage as next-generation cloud brokerage that constructs a commercial meta-cloud whose ownership is scattered across cloud providers.

References

[1] C. Bassek, S. Pierre, and A. Quintero. Redundancy schemes for high availability computer clusters. *Journal of Computer Science*, 2006.

[2] IBM Softlayer. http://www.softlayer.com/.

[3] M. R. Lyu. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill Book Company, 1995.

[4] Redhat whitepaper on low cost high availability clustering for the enterprise. https://static.redhat.com/legacy/whitepapers/rhel/Red_Hat_-_Winchester_Systems_White_Paper.pdf.

[5] Y. Shao, X. Zhu, W. Bao, W. Zhou, and W. Xiao. Chime: A checkpoint-based approach to improving the performance of shared clusters. In *IEEE International Conference on Parallel and Distributed Systems*.

[6] A. Stanik and M. H. amd Odej Kao. Failover pattern with a self-healing mechanism for high availability cloud solutions. In *International Conference on Cloud Computing and Big Data (CloudCom-Asia)*, 2013.

[7] V. R. Yerravalli and A. Tharigonda. High availability cluster failover mechanism using artificial neural networks. In *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2015.