AVAILABILITY METRICS: UNDER CONTROLLED ENVIRONMENTS FOR WEB SERVICES

Sandesh Tripathi[1], S Q Abbas[2], Rizwan Beg[3]

[1]Research Scholar, Department of Computer Science, Integral University, Lucknow
sandeshtripathi@rediff.comm
[2,3]Professor, Department of Computer Science, Integral University, Lucknow

ABSTRACT

Web Services technology has the potential to cater an enterprise's needs, providing the ability to integrate different systems and application types regardless of their platform, operating system, programming language, or location. Web Services could be deployed in traditional, centralized or brokered client server approach or they could be in peer to peer manner. Web Services could act as a server providing functionality to a requester or act as a client, receiving functionality from any service. From the performance perspective the availability of Web Services plays an important role among many parameters.

KEYWORDS

Web Services, Performance, Availability, Metrics

1. INTRODUCTION

Web Services have grown over the years as a key technology in distributed and dynamic environment for providing complex solutions using easier methods. Service oriented methodologies are being regarded as promising solutions for future applications [21]. The key challenges to keep web services up and running well, includes;

- Reliability- Due to the distributed nature of web service applications. It demands a stable and reliable network environment. Having different components distributed over geographically dispersed networks, availability of service, reliable communication and application performance becomes very important for successful deployment of application.
- Quality of Service- In addition to service reliability and availability, organizations need to prioritize requests. Requests need to be intercepted, analyzed, and directed to the proper resource to provide quality of service based on an organization's business policies.
- High Availability: As the demand of service increases, the availability of each component within the service and the applications that processes the requests will be critical. Key systems and devices that ensure web service availability and reliability will be required to direct requests to healthy resources.

  There is many more like Scalability, Performance, Application Security, Network Security. There is many more like Scalability, Performance, Application Security, Network Security. This paper does an investigation on web services availability, for architectures which involve some controller mechanism before final delivery of service.

DOI : 10.5121/ijwsc.2011.2301

In the author's previous work, a controller was created and implemented before final delivery of services for complying QoS, and choosing appropriate instance on the computing infrastructure.

2. WEB SERVICES AVAILABILITY

Web Services are loosely coupled self contained, units of code that are exchanged via messages to notify each other events, request information, or demand an action to be done on their behalf. When Web Services are combined with new code to be used in a web application, two areas are of concern, maintaining the availability of the web application and maintaining the availability of specific web service in that web application. With geographically dispersed services dependent on different data centres, if a single data centre goes down the entire application goes down.
[1] In the modelling of maintained or repairable systems with high availability requirements such as telephone switching systems, communication networks etc, the metric Reliability i.e. the probability that the system remains operational over a given time period, is an appropriate measure for evaluating the effectiveness of these type of systems.

Systems of the second class are usually operated continuously while at times for short duration their operation of unavailability can be tolerated. Therefore, redundancy is used to improve the performance under normal operation and to reduce the down time in case of a failure. In this class of systems, preventive and corrective maintenance can be performed to obtain the desired level of service. Availability, i.e. the fraction of time the system is operational is a more appropriate measure for evaluating the effectiveness of this class of systems. The main consideration is the service availability, which should be at least 99.99 %. The site is used by the users to get quotes on stocks and mutual funds, manage portfolios, conduct portfolio analysis, and to place orders to trade stocks and mutual funds. For example; www.xignite.com provides market data on demand through web services and in comparison market data feed the user can save money in building applications, save time (can be hosted in days in comparison to months), save hardware cost as well. In such cases availability and performance are the two key factors.
Comparison of using on demand market data web service and market data feed.

	Market Data Feed	On demand data web service
Time of deployment	1-6 months	< 1 day
Total cost for subscription	Generally bulk so fixed price	Sold on transactional basis so depend on the data consumed
communications	Require leased line	On internet
Software	Requires software to parse, clean and access data	Using web service operations directly send to developer tools for immediate use
Hardware	Requires on premise hardware	Since passed directly to the applications in response to service request, no local storage is needed
Latency	Overall latency is determined by the network and customer's IT infrastructure	Depends on the cloud service provider system and the response request time of the internet
Availability	Availability is a combination of vendor's feed and customers hardware and software infrastructure	Is driven by the SLA signed by the service provider and service requester

Table 1: Comparison of Market Data feed and on demand web Service

Table 1 illustrates how the web service technology provides the savings and easy management of services but it all depends upon majorly the availability and performance of the service otherwise the SLA will be breached and heavy penalty may be incurred.

2.1. System failures Category

There are many reasons why a computer system may fail [8]. Let us categories the different types of failures, on the dimension of duration, effect and scope.

The first dimension is related to the duration of the failure and comprises following cases.

Permanent Failure: A system stops working and there is no possibility of repairing or replacing it. Such may the case, when the system is employed in space ship.
Recoverable Failure: In this case the system is placed back in operation after a fault is recovered. An example could be, a web site becoming inaccessible because its connection to the internet goes down.
Transient Failures: These failures are categorised by having a very short duration and may not require major recovery actions. Such is the controller class, where in a controller may stop giving permission for accessing because the QoS might deteriorate.

The second dimension is related to effect of failure.
Functional Failures: This is the case in which a system does not operate according to its functional specifications.
Permanent Failures: This is the category where, even though the system may be executing the requested functions correctly, they are not executed in a timely fashion.

The third dimension is related to scope of the failure.
Partial: In this case some services provided by the computer system become unavailable, while others can still be used.
Total: These failures are categorised by a complete disruption of all services by the computer system.

2.2. Failure and recovery:

The term system, platform and infrastructure can be used interchangeably referring the underlying infrastructure.
Using [2], the platform is classified into three types: (1) 'near-user' (2) 'in-middle' and (3) 'near-host'. A failure in 'near-user' portion, which is typically the user's subnet, disallows the user to access the rest of the internet. Similarly, 'near-host' failures make the web unreachable from the outside world. The 'in-middle' failure usually refers to the Internet backbone connection malfunctions that separate the user and the specific host, but the user may still visit a non trivial part of the internet.

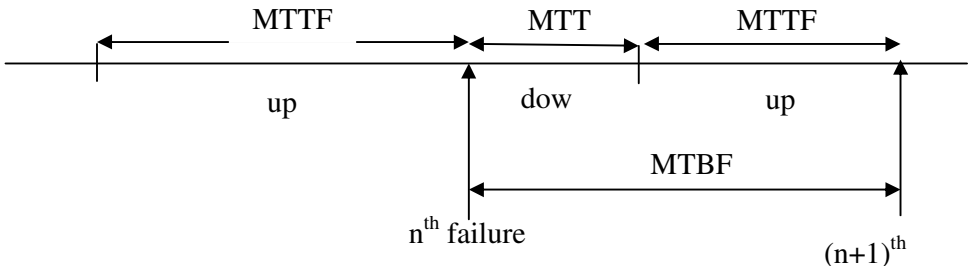

Figure 1.0 Classes of Systems according to Availability

Using a unified fail to recovery model that assumes time to failure (TTF) and time to recover (TTR) are exponentially distributed for all the three cases. Suppose once the system becomes operational, it takes certain time to fail again. The average time it takes the system to fail is called MTTF (mean time to failure). Once the system fails it takes certain time to recover from failure and return to operational state. The average time it takes for the system to recover is called MTTR (mean time to recover). The average time between failures is called MTBF (mean time between failures) and can be written as

MTBF = MTTF + MTTR (1), as shown in fig: 1

Availability is defined as the fraction of time that a component is operational.

3.0 Availability Classes:

Availability Class	Availability	Unavailable (min/year)	System type
1	90%	52560	Un Managed
2	99%	5256	Managed
3	99.9%	526	Well Managed
4	99.99%	52.6	Fault Tolerant
5	99.999%	5.3	Highly Available

Table 2: Classification of Availability Classes

According to these classes even if a web site has one hour of scheduled down time per week it is considered under a good class.

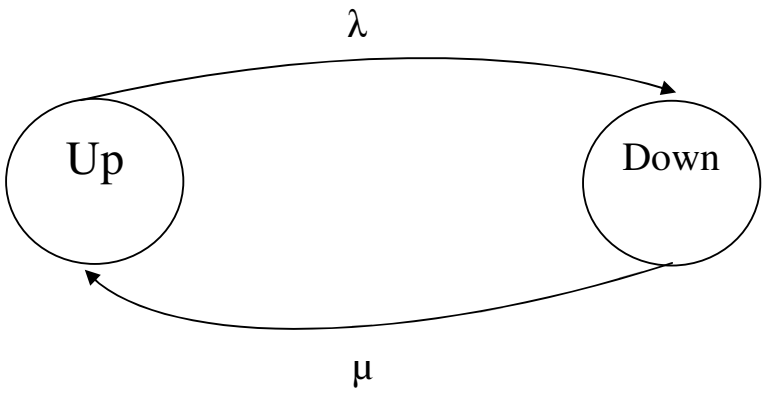

Figure 2.0 State Transition Diagram for availability computation

The system fails, i.e., goes from up to down with a rate λ and gets repaired, i.e. goes from down to up with a rate μ. These rates can be written in terms of the MTTF and MTTR as

$$\lambda = \frac{1}{MTTF} \text{ And } \mu = \frac{1}{MTTR} \quad (2)$$

Using the flow-in-flow-out principle, we can write that,

$$\lambda \times p_{up} = \mu \times p_{down} \quad (3)$$

Where p_{up} and p_{down} are the probability that the system is up and down, respectively, Note here that availability is simply p_{up} and that

$$p_{up} + p_{down} = 1 \quad (4)$$

Combining eq.3 and eq.4 we get that

$$A = p_{up} = \frac{\mu}{\mu+\lambda} = \frac{\frac{1}{MTTR}}{\frac{1}{MTTR}+\frac{1}{MTTF}} = \frac{MTTF}{MTTF+MTTR} \quad (5)$$

And $$U = p_{down} = \frac{\lambda}{\mu+\lambda} = \frac{MTTR}{MTTF+MTTR} = \frac{MTTR}{MTBF} \quad (6)$$

Where, U is known as the system *un-availability*. In most of the cases, MTTF >> MTTR, i.e. it takes significantly longer for the system to fail than to be repaired. Then, the unavailability can be approximated as

$$U \sim \frac{MTTR}{MTTF}. \quad (7)$$

There are two ways to improve availability [5]: reduce the frequency of failures or reduce the time to recover from them.

3.1. User's Behaviour:

Web user behaviour was proposed by Deng [4]. The ON period follows a Weibull distribution with the probability density function.

$$f(t) = \frac{k}{\theta}\left(\frac{t}{\theta}\right)^{k-1} e^{-\frac{t}{\theta}k}, \quad (8)$$

And the cumulative distribution function (cdf) is

$$f(t) = 1 - e^{-\frac{t}{\theta}k} \quad (9)$$

Constants k and θ are referred as the shape parameter and scale parameter of Weibull distribution. Typically k = .77 to .91 and $\theta = e^{4.4}$ *and* $e^{4.6}$ as in [4]. The duration of OFF period follows a general Pareto distribution with pdf

$$g(t) = \begin{cases} C\alpha\, m^{\alpha}/t^{\alpha+1}, & m \le t \le n, \\ 0 & otherwise \end{cases} \quad (10)$$

And the corresponding cdf is

$$G(t) = \begin{cases} C\left(1 - \left(\frac{m}{t}\right)^{\alpha}\right), & m \le t \le n, \\ 0 & otherwise \end{cases} \quad (11)$$

Where α, m, n are constants, with typical values α= 0.5 to 0.9, m=60 and n=6000. The constant *m* is called "ON-OFF" threshold, which means a series of requests with inter-arrival times within *m* constituting an ON period, and a request occurs more than time *m* after previous requests marks an OFF period.

3.2. Availability Metric description:

In this paper a availability metric calculation technique is proposed based on Service Status divided into four classes.

Service Available (SA), Service May Recover (SMR), Service May Not Recover (SMNR), Service Not Available (SNA).

In our previous work, where a controller is developed for choosing an appropriate instance, the controller aborts those service requests for which QoS does not comply. In all such cases the service requests may encounter unavailable web service. But it may happen that in next interval some of the services may be available after QoS satisfaction. Hence two more status is introduced known as Service may recover and Service may not recover.

Service Available (SA): This status indicates that the service is running stable and no invocation failure has happened, for these requests.

Service May Recover (SMR): This status indicates that the service is not currently available, but chances are there to recover it, because this unavailability is not due to failure but it is due to incompliance of QoS metrics by the controller.

Service May Not Recover (SMNR): This status indicates that the service is not currently available, but chances are less for recovery.

Service Not Available (SNA): This status indicates that service is down due to a specified reason.

In this approach the metrics computation is based on invocation of records, the model is simple, and in this model the short term down is further divided in two sub categories SMR and SMNR.

The metric estimation:

It's a three step approach :-(1) Calculate the success percentage for each sequence (2) Calculate the weighted average of success rates for status SMR and SMNR (3) Calculate the time percentage for each status.

Success rate: For Service Available: 1, For Service Unavailable: 0, For SMR and SMNR, several service invocation records are recorded and availability is computed as (5). For all unavailability cases a weighted approach is considered, where the weighting value is the time elapsed for each sequence interval. For each sequence $R_{k,j,}$ the time elapsed in this sequence is calculated by $\lceil t_j - t_i \rceil$.Suppose that the sequence is R^v=$\{R_{k_1,k_1'}, R_{k_2,k_2'}, \ldots\ldots\ldots, \}$, the calculation for overall success rate is based on following equation;

$$A(SR) = \frac{\mathrm{SM}\left\lceil t_{k_i'} - t_{k_i} \right\rceil * ASM_{SM}^{i,i'}}{\mathrm{SM}\left\lceil t_{k_i'} - t_{k_i} \right\rceil} \quad (12)$$

$$\text{And time percentage} = \frac{\mathrm{S_i}\left\lceil t_{k_i'} - t_{k_i} \right\rceil}{\sum_{i=1,\ldots,n}\left\lceil t_{k_i'} - t_{k_i} \right\rceil} \quad (13)$$

Where $\mathrm{S_i}$ is a status such as SA,SNA,SMR,SMNR.

3.3. Evaluation data Set:

The data is collected for six rounds for evaluating availability metrics, a simulated environment was created. Since EC2's servers are Linux based virtual machines running on top of the Zen Virtualization Engine [14]. A Linux machine having 1.7 GHz x 86 processor,1.75 GB of RAM,

160 GB of local disk is used for experiments. The virtualization is implemented on this machine as Amazon web Services are implementing, using Xen based Virtualization Environment. This virtual environment is also used by Amazon. This environment uses Xen hypervisor, the Domain 0, and 9 VM guests. These nine VM guests implemented nine instances under consideration. The system consists of one CPU and one disk. The workload is being driven by another machine using proxy-sniffer (a workload generator), which can also be used for Amazon EC2. The service demands at the CPU and disks are 0.03 sec and 0.05 sec, respectively. The SLA and the respective weights are:

- $R \leq 1.3\ seconds\ and\ W_R$=0.25,
- $X \geq 7\ requests/\sec and\ w_X$= 0.30, and
- $P_{rej} \leq 0.05\ and\ w_p$

During experiments, the arrival rate of requests started from a low of 7 service requests per second and the load was increased up to a maximum of 23 service requests per second, during a period of 1 hr and 40 min. The controller interval is of 300 seconds. During any interval with peak average loads of 23 service requests per second, 6900 requests arrive. At the maximum load of 23 service request per second, the resource bottleneck reaches close to 100 %, after this load was not increased further otherwise the probability of rejection would be turning up too high.

Round 1	Service Available (SA)	Service May Recover (SMR)	Service May Not Recover (SMR)	Service Not Available (SNA)
Time percentage	0.840	0.158	.000	0.004
Availability (success rate)	1	0.64	0.31	0
Round 2				
Time percentage	0.616	0.176	0.161	0.034
Availability (success rate)	1	0.48	0.50	0
Round 3				
Time percentage	0.247	0.013	0.024	0.716
Availability (success rate)	1	0.32	0.64	0
Round 4				
Time percentage	0.731	0.134	0.056	0.071
Availability (success rate)	1	0.78	0.21	0
Round 5				
Time percentage	0.98	0.0	0.0	.000
Availability (success rate)	1	0	0	0
Round 6				
Time percentage	0.397	0.182	0.287	0.145
Availability (success rate)	1	0.63	0.37	0

Table 3: Availability computation for simulated services

At higher rates, since QoS is not meeting, the controller is aborting services, and the resulting probability of rejection is high. Table 3 shows the computed availability metrics for all six rounds

of data invocation. This approach gives a new dimension to see services availability, considering methods, where due to QoS considerations the availability may be at minor stake.

Conclusion:

The limitation of adopting time percentage in web services domain is that the higher frequency of accessing web services is not valid. Web Services spend considerably more time for XML serialization and deserialization. Web Service Availability is considered as one of the key properties for service oriented computing. The paper describes a new metric for web service availability. This metric convey more information for availability considerations.

REFERENCES

[1] A Goyal, S. S. Lavenberg and K.S. Trivedi, "Probabilistic modelling of computer system availability", Scientific Publishing Company .

[2] Bharat Chandra, Mike Dahlin, Lei Gao and Amol Nayate, " End to end Wan service availability", in third Usenix Symposium on internet technologies and systems, Jan 2001 .

[3] www.xignite.com

[4] S.Deng, "Empirical model of www documents arrival at access link," in ICC'96.

[5] E.A. Brewer, "Lessons from Giant-Scale Services,"IEE Internet Computing," July/August. 2001.

[6] M. Moser and W. Zhao, "Building Dependable And Secure Web Services", Journal of Software.

[7] K. S. Trivedi, Probability and statistics with reliability, Queing and computer Science Applications, John Wiley & Sons, Second edition, 2001.

[8] S. Lam and K. Chan, Computer Capacity Planning: Theory and Practice, Academic Press, London, England, 1987.

[9] R. S. Varga, "Matrix Iterative Analysis", Prentice Hall, 1962.

[10] S.S.Lavenberg, "Computer Performance Modelling Handbook", Academic Press.

[11] T. Nakagawa and A.L. Goel, "A note on Availability for finite interval", IEEE Transactions on computers.

[12] J.F. Meyer, "On evaluating the performability of degrading computing systems", IEEE transactions on Computers.

[13] OASIS. Specification: Business Process Execution Language for Web Services, 2004.

[14] *XenSource Inc. Xen. http://www.xensource.com*

[15] P. Tobias and D. Trindade, " Applied reliability", second edition, Kluwer,1995.

[16] W. Whitt. " The queing network analyzer", Bell System Technical Journal,1983

[17] M. Merzbacher and D. Patterson, "Measuring end user availability on web: Practical Experience", ICDSS 2002

[18] S . Gokhle, “Software reliability analysis incorporating second-order architectural statistics”, International Journal of Reliability, 2005

[19] Transaction Processing Performance Council. www.tpc.org

[20] V.G. Kulkarni, “Modelling and analysis of stochastic systems”, Chapman Hall,1995.

[21] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea. “Web Services enginerring: promises and challenges. In Proc. of the 24th International Conferance on Software Engineering.