A simulation test-bed for the design of dependable e-services

PANAJOTIS KATSAROS CONSTANTINE LAZOS
Department of Informatics
Aristotle University of Thessaloniki
54006 Thessaloniki
GREECE
{katsaros, clazos}@csd.auth.gr

Abstract: - In this paper, we present the design and development of a simulation test-bed, for the performance analysis of dependable and mission critical e-services. Key characteristics of such systems, like for example, the object state transfer and recovery policies to be applied together with the chosen load distribution strategy, play the determinant role in the service's performance and customer response perception. Analytic models usually fail to realistically capture the effects of the available design alternatives. On the other hand, the published simulation studies are usually bound to algorithms, which are not covered by the recently published standards, for the development of object based services. Our work aims to provide a comparison framework that adheres to the published standards, allows the compositional development of the service configurations of interest and the estimation of meaningful performance measures, in an efficient way.

Key-Words: - simulation, quality of service, fault tolerance, load distribution, state transfer and recovery schemes, performance evaluation, distributed object systems, e-services

1 Introduction

The proliferation of matured standardized object middleware and the ever-decreasing Internet access costs lead to the development of innovative e-services, in commerce, education, finance, banking, medicine etc. Recently, the emerging need for the design and development of dependable and mission critical e-services resulted in the set up of additional standardization activities, for provision of the required quality of service and fault tolerance attributes ([1], [11] and [8]).

However, the usual build-and-test approach, adopted in most projects, implies high development costs and little chances for scalable and reusable designs. On the other hand, analytic models are bound to assumptions, which diverge from reality and do not make feasible the comparative evaluation of composite policy configurations.

Simulation is the only alternative that allows the representation of the resulted software and hardware resource contention, in an arbitrary level of detail, thus capturing the essence of the quality of service differences, in the studied policy configurations.

Our work aims in providing a simulation test-bed that allows to:

- mix and match different policies from those described in the already published standards,
- evaluate and compare their effectiveness based on meaningful, appropriately chosen estimates
- in an efficient way.

We assume that the observed failures conform to the fail-stop model ([13]), which means that objects fail by crashing, without emission of spurious messages. We do not make any assumptions about the network topology or the protocols making up the interprocess communication service, except that communication is accomplished through lossless FIFO channels.

In such an asynchronous heterogeneous object application, we are interested for modeling object failures that do not recur after recovery. Some of them may be hardware dependent (e.g. insufficient memory) and others may be attributed to sources of non-deterministic behavior, like:

- when a certain sequence of invocation requests results in an invalid object state,
- the use of local timers,
- the use of input-output to local devices,
- the use of multithreading, etc.

This paper describes the software's functional characteristics and the performance measures of interest, for selecting the most appropriate

- object state transfer (checkpointing) policies and intervals,
- failure monitoring schemes and intervals,
- failure recovery strategies and
- service load distribution policy,

for the development of either dependable or mission critical e-services. Dependable e-services are required to succeed the specified levels of availability, in normal or degraded mode of operation. On the other hand, mission critical e-services are required not to fail for the specified period of time. Steady-state simulation is employed for the first type of study, as opposed to the second one, where we use transient simulation, with different performance measures of interest.

Currently, object or service failures may occur only by taking advance of the software's primitive failure forcing facilities. Thus, we do not proceed to the description of advanced failure biasing and importance sampling techniques, which are under development and will result in accurate and efficient estimation of the measures of interest.

Related work has been reported in [12] and [4]. The first paper describes UltraSAN, a Stochastic Activity Net based modeling tool, with analytic and simulative solution facilities. The second one describes DEPEND, a simulation - based environment, with advanced fault injection features. Our test-bed differs, compared to them, in the following:

- it is designed exclusively for simulating the policies specified in related object middleware standards ([11] and [8]),
- it is not restricted to the estimation of dependability measures, like reliability and availability, but it also produces estimates for mean response times and utilizations, which guide the process of state transfer placement and load distribution,
- the systems' and components' failure behavior will be simulated by advanced failure forcing and biasing schemes, accompanied by appropriate well-founded importance sampling techniques (as in UltraSAN and [10], [9]), for efficient estimation (in contrast to DEPEND's efficiency improvements, which lie on the notion of variable aggregation and decomposability [2]),
- estimates will be produced by advanced output analysis methods ([10] and [3]), for increased credibility and finally,
- our software is open to the implementation and study of new workload dependent failure models.

2 Replication management and failure monitoring and recovery features

Replication may be used as a means, either to succeed the required fault tolerance attributes or for load balancing purposes. In this paper, our description is restricted to the features used, in providing fault tolerance.

According to [11] and [8], replication management may be based on the notion of the object group and includes creation, deletion and replication of existed application or infrastructure objects. In our test-bed, we have implemented the following styles of replication:

- Active replication
  Each object replica processes all the invoked object methods. The failure of a single replica is masked by the presence of the other replicas that also perform the same invocation requests and generate the desired results. Duplicate invocation requests as result of redundant processing are detected and suppressed. When in a degraded mode of operation, the number of live replicas falls bellow the specified minimum number of replicas, each failed replica is being recovered. This involves the completion of an object state transfer from a live object replica (if there is any). If, in the course of this state transfer, the interacting replicas receive additional invocations and responses, they are all enqueued locally and subsequently being applied to them.
- Warm passive replication
  Only one of the object group replicas processes the received invocations. This is the so-called primary replica. The remaining passive replicas, known as backups, have been activated and their state is continuously synchronized with the primary replica's state, according to the specified state transfer policy.
- Cold passive replication
  The backup replicas have not been activated, but the primary's state is periodically captured and stored into a log. If the current primary fails, a new one is elected and then activated. The log is then used to initialize the new primary's state.

State transfer durations depend on the object state size (granularity), the bandwidth and the processing speed of the slowest backup replica (for warm passive replication). A state transfer may be initiated only when it is not violating the object group's replica consistency. Thus, it is postponed, when the primary is in the middle of an invocation service or it happens to be blocked, waiting for a response. In the course of the state transfer, new invocations may be received, but they cannot be processed, before the end of it.

The object state transfer policies that are currently implemented include:

- the use of a fixed time interval between state transfers and
- the use of a fixed number of completed object invocations between state transfers (load-dependent policy)

Failure recovery depends on the chosen replication style and the applied policies and takes place right after the failure detection. In active replication, it includes just the time taken to transfer the current state of another live replica. On the other side, the contemporary function of multiple identical object replicas gives rise to heavy hardware resource contention, with possibly undesirable implications to the offered quality of service.

Object failure recovery, in warm passive replication, includes the election of a new primary (if there is any) and the replaying of the invocations logged, since the last object state transfer. A failed replica remains in the recovering state up to the first subsequent state transfer, from the new primary. Cold passive replication also requires the activation and the logged state transfer to the new primary.

Failure detection takes place as a result of periodic “i am alive” messages that have to be sent by each object or replica (for active replication) and to be received by a transparent monitoring service (we do not consider hardware resource contention for the failure monitoring function). When a failure is detected, the recovery process is then initiated, according to the specified policy.

3 Performance and dependability measures of interest

Our simulation test-bed provides a means for realistically modeling the individual objects interaction effects, as regarding:

- the simultaneous resource possession caused by the synchronous, often nested object invocations, which block the callers, until they get the replies and
- the hardware resource contention, as a result of the chosen replica placement to the available processing nodes.

The service dependability measures of interest, which may be the subject of a quality of service contract, are:

- the steady-state service availability, given as the fraction of time the service is operational (this happens if specified combinations of interacting objects are operational),
- the service response time in the worst case and its mean,
- the overall service throughput and
- the mean time to an unrecovered service failure (MTTF), for mission critical e-services (transient measure).

Different estimation methodologies have to be used in each case, depending on whether the target measure is a transient or a steady-state measure, on whether the service exhibits a regenerative simulation sample path or not, etc.

Other useful measures are:

- the ratio of mean requested service time to the estimated mean response time, for assessing the hardware resource contention caused by the chosen replica allocation, the selected state transfer intervals and the used load distribution policy and
- the individual mean object response times and utilizations, for determining undermost state transfer intervals.

We suggest undermost state transfer intervals to be detected based on a trade-off analysis, under the same failure model. Thus, the benefits derived from the state transfers are traded against the overhead they impose. An appropriate composite measure is the following ratio,

$$\frac{R_f \text{ with less state transfers} - R_f \text{ with more state transfers}}{R_o \text{ with more state transfers} - R_o \text{ with less state transfers}}$$

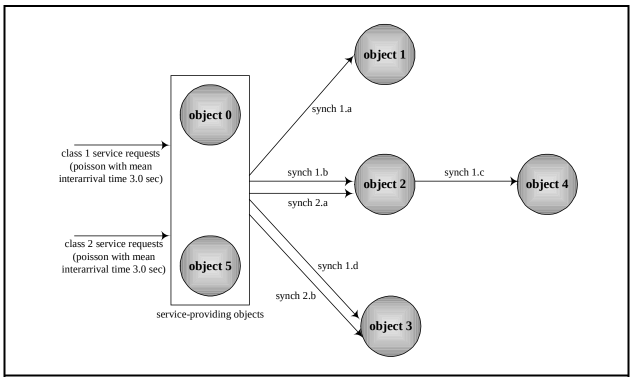
where R_o is the mean response time for serviced invocations not experiencing failure and R_f is the mean response time for invocations experiencing failures. Negative ratio values indicate that the chosen state transfer interval causes additional overhead that does not result in improvements to the service times of invocations experiencing one or more failures.

The state transfer intervals to be used will be longer than those given by the described trade-off analysis and their values will be influenced by the required quality of service attributes and the hardware resource contention imposed by the chosen replica allocation and the used load distribution policy.

Advanced failure biasing and importance sampling techniques will be used, for accurate estimation of the mean service response time and service throughput and availability. The overall estimation will be based on the regenerative method for producing confidence intervals ([5], [6] and [7]). A viable alternative for non-regenerative service configurations is a method based on splitting and batch means ([3]).

4 A case study

In this section, we proceed to the description of a sample service configuration and the results given by the current version of our simulation test-bed. The system under study disposes two service-providing objects (object 0 and object 5), which accept the invoked service requests, in a round-robin fashion (load distribution strategy). The service objects interactions are summarized in Figure 1.


object 1
object 0
synch 1.a
class 1 service requests (poisson with mean interarrival time 3.0 sec)
synch 1.b
object 2
synch 1.c
object 4
synch 2.a
class 2 service requests (poisson with mean interarrival time 3.0 sec)
object 5
synch 1.d
synch 2.b
object 3
service-providing objects
Figure 1 Object interactions for the sample service configuration

Each object group makes use of warm passive replication, with a minimum number of two replicas, allocated in processing nodes of different speeds. We did not consider replicas collocated in the same node.

A load-dependent state transfer strategy was chosen for all object groups. More precisely, a new state transfer was initiated after the completion of a fixed number of serviced object invocations. The state transfer speed was influenced by the allocated object state size (object granularity), the participating replica processing speeds and their inter-node allocated bandwidth. Finally, failure monitoring was simulated by periodic "i am alive" messages, sent by the corresponding object groups, in the specified frequency.

simulated time:	18000 sec	
number of invocations between state transfers:	55	
object 2 failure exponentially scheduled with parameter:	2341	
	mean respnse time (sec)	worst response time (sec)
class 1 service requests	7,60756	44,0935
class 2 service requests	7,12024	46,7736
	availability	utilization
object 0	99,6101%	82,1848%
object 1	99,5263%	17,5811%
object 2	98,9955%	42,4268%
object 3	99,4354%	56,5779%
object 4	99,7719%	11,0633%
object 5	99,4938%	82,4637%

system availability:	98,4348%

Table 1 Sample simulation results

invocations between state transfers	mean response class 1 no failure	worst response class 1 no failure	mean response class 2 no failure	worst response class 2 no failure	service availability no failure
165	6,52808	38,62900	5,88361	40,9284	99,501%
	8,09187	90,08840	7,64729	88,1666	99,071%
55	6,85649	39,68420	6,17036	39,7672	98,645%
	7,60756	44,09350	7,12024	46,77360	98,434%
25	7,02426	46,35750	6,47951	43,9989	97,244%
	7,45605	42,23310	6,88256	41,87440	97,018%

Figure 2 Response times and service availability for the simulated numbers of invocations between object state transfers

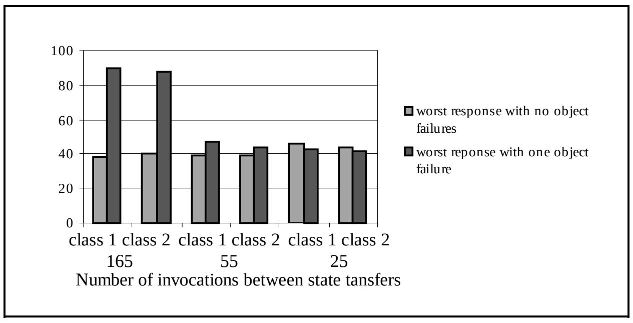


Figure 3 Worst response time improvements for the simulated numbers of invocations between object state transfers

The sample results given in Figures 2 and 3 indicate:

- when the selected number of invocations, between state transfers, is 165, this results in higher service availability and probably unacceptably high response times, for the service requests affected by an object 2 failure
- if end-user response is guaranteed to be varied up to 50 sec, then an appropriate number of invocations between state transfers is 55
- further decrease of the above number results in negligible improvements to the response times affected by an object 2 failure (rare event), at the cost of a significant deterioration to the overall mean response times.

5 Future work - Conclusion

In this paper, we described the functional characteristics of a new simulation test-bed, for the performance analysis of dependable and mission critical e-services. We were restricted to the currently implemented policies for providing fault tolerance and we discussed the use of valuable performance measures, for the development of either dependable or mission critical e-services.

The test-bed is to be equipped with advanced failure forcing and biasing techniques, accompanied by appropriate importance sampling and simulation output analysis methods, for accurate estimation of the performance measures of interest. Similar techniques have been also implemented in UltraSAN ([12]), but our environment is specifically designed for simulating the policies suggested in the recently published standards, without having to specify them in a specialized formalism, like the one used in UltraSAN.

Such a tool will provide a basis for studying new, load dependent failure models and customized load distribution strategies and for deriving realistic quality of service requirements.

References:

[1] C. Becker and J. Zincky, Quality of Service in Distributed Object Systems, In J. Malenfant, S. Moisan, A. Moreira (Eds.): ECOOP 2000 Workshops, LNCS 1964, Springer-Verlag, 2000, pp. 178-190

[2] P. J. Courtois, Decomposability – queueing and computer system applications, Academic Press, 1977

[3] P. W. Glynn, P. Heidelberger, V. F. Nicola and P. Shahabuddin, Efficient estimation of the mean time between failures in non-regenerative dependability models, In Proceedings of the 1993 Winter Simulation Conference, 1993, pp. 311-316

[4] K. K. Goswami, R. K. Iyer and L. Young, DEPEND: A simulation-based environment for system level dependability analysis, 46, 1, 1997, pp. 60-74

[5] P. Katsaros and C. Lazos, A technique for determining queuing network simulation length based on desired accuracy, International Journal of Computer Systems Science and Engineering, CRL Publishing, 15, 6, 2000, pp. 399-404

[6] P. Katsaros and C. Lazos, Regenerative estimation variants of response times in closed networks of queues, In Proceedings of the 2nd WSEAS International Conference on Simulation, Modeling and Optimization (ICOSMO), 2002

[7] P. Katsaros and C. Lazos, Return state selection for improved effectiveness in sequentially controlled regenerative simulation, (submitted), 2003

[8] P. Narasimhan, L. E. Moser and P. M. Melliar-Smith, Strong replica consistency for fault-tolerant CORBA applications, Journal of Computer Systems Science and Engineering, 2002

[9] V. F. Nicola, M. K. Nakayama, P. Heidelberger and A. Goyal, Fast simulation of highly dependable systems with general failure and repair processes, IEEE Transactions on Computers, 42, 12, 1993, pp. 1440-1452

[10] V. F. Nicola, P. Shahabuddin and M. Nakayama, Techniques for the fast simulation of models of highly dependable systems, IEEE Transactions on Reliability, 50, 3, 2001, pp. 246-264

[11] Object Management Group, Fault tolerant CORBA, OMG Technical Committee Document, 2001-09-29, September 2001

[12] W. H. Sanders, W. D. Obal II, M. A. Qureshi and F. K. Widjanarko, The UltraSAN Modeling Environment, Performance Evaluation, 24, 1, 1995, pp. 89-115

[13] R. D. Schlichting and F. B. Schneider, Fail-Stop processors: An approach to designing fault-tolerant computing systems, ACM Transactions on Computer Systems, 1, 3, 1983