Brief Announcement: The Inherent Difficulty of Timely Primary-Backup Replication

Pramod Koppol
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
pramod.koppol@alcatel-lucent.com

Kedar S. Namjoshi
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
kedar@research.bell-labs.com

Thanos Stathopoulos
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
thanos.stathopoulos@alcatel-lucent.com

Gordon T. Wilfong
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
gtw@research.bell-labs.com

ABSTRACT

We show that existing methods for primary-backup replication may disrupt the timing behavior of an underlying service to the extent of making it unusable. We prove that the problem is inherent to the primary-backup model.

Categories and Subject Descriptors

D.4.5 [**Operating Systems**]: Reliability—*Fault-tolerance*

General Terms

Theory, Reliability

Keywords

Primary-backup, replication, fault-tolerance, knowledge

1. INTRODUCTION

The advent of cloud computing, with its remote compute and data centers, is changing the way in which computing services are offered. We are interested in the question of whether *telecommunications* services can be offered in a cloud environment. These services differ from the ones hosted currently on cloud platforms in that they have rather strict requirements on delay and jitter. The requirements may be regulatory, or imposed by human perception; large values for delay, jitter, or packet loss can seriously degrade audio quality in a phone conversation. In addition, such services are expected to be highly reliable.

Traditionally, telecommunications services have been protected against faults using special hardware, which also allows the software to meet timing constraints [9]. The environment offered by compute centers is quite different: hardware is standardized, a shared broadcast medium is generally not available, and communication and computation failures may be frequent [4]. This model is akin to that of a distributed computing system.

PODC'11, June 6–8, 2011, San Jose, California, USA.
ACM 978-1-4503-0719-2/11/06.

A natural and important question, therefore, is whether time-sensitive services can be run reliably under a distributed computing model, while also preserving timing constraints. We examine methods for fault tolerance which are based on primary-backup replication. We show that known methods can give rise to large jitter and unbounded delays under high load, *even under failure-free operation*. We further prove that *any* primary-backup method operating under this model must have similar impediments; these result from an unavoidable synchronization between primary and backup. The proof is based on a theorem of Chandy and Misra [2], which connects increases in process knowledge to the existence of causal chains of messages.

The most closely related work which we are aware of is an analysis of the effect of primary-backup synchronization on real-time scheduling [10]. That paper accounts for delays caused by primary-backup synchronization by adjusting the admission policy for real-time tasks, allowing fewer tasks to be admitted, which effectively reduces throughput. The real-time task model is different from our setting, which is the processing of message streams.

A paper by Budhiraja et al. [1] gives upper and lower bounds on the blocking time of primary-backup protocols. The system models considered there, however, are different from ours in crucial respects. In the most closely related crash+link model, a transmission failure is counted against the failure budget—i.e., a protocol with a budget of $f = 1$ failures trivially meets its specification after failures on two links—whereas we consider transmission failure to be normal. Therefore, the non-blocking protocol given in [1] for crash+link failures does not apply, as message acknowledgments are necessary in our model.

2. LIMITATIONS OF KNOWN METHODS

A primary-backup protocol allows the state of a server, which can be thought of as a state machine, to be recovered after a failure of the machine on which the server is executed. We consider protection for a single crash failure. In *active replication* [6, 8], all replicas receive the same messages in the same order. This suffices for fault-tolerance if message processing is deterministic. In *passive replication*, a checkpoint of the server state is taken periodically; after

a primary failure, recovery proceeds with the backup server starting from the latest checkpoint state.

In both approaches, the root cause of timing disruptions is the synchronization between primary and backup. We examine more closely the source of the disruptions arising in Remus [3], which is a representative implementation of passive replication. An analysis of active replication results in similar conclusions. The essence of the Remus synchronization is as follows.

1. Primary and backup are synchronized at the start of an *epoch* (the period between checkpoints)
2. Primary receives and processes a sequence of input messages; generated output messages are buffered (Releasing output messages immediately can cause the environment and the backup to have inconsistent views of the primary state after a primary failure.)
3. Primary sends its current state to the backup and waits for an acknowledgment
4. Primary receives acknowledgment from backup; synchronization is complete
5. Primary sends the buffered output to the environment

Let T represent the round-trip time from the primary to the backup. Suppose that, without any synchronization, the primary is able to process input arriving at a constant rate $r > 1/T$. At rate r, at least one message arrives during synchronization (steps 3,4). As the primary is suspended at step 3, these messages must be processed in the next epoch (after step 4). If there is insufficient slack time during that epoch, some messages remain unprocessed. Repeating this sequence of events, unprocessed messages must accumulate beyond bound in input buffers, which results in an unbounded delay in processing input messages.

A possible resolution is to drop some input messages, but that may result in a service failure (e.g., for voice packets) or in lowered throughput. Moreover, releasing output all at once (step 5) can cause bursty traffic. In an experiment with Remus on a single audio stream, we observed that burstiness (caused by step 5) and delay due to checkpointing can seriously degrade audio quality. The processor was lightly loaded, so we did not observe an input queue buildup.

3. FORMAL ANALYSIS

We show that timing disruptions are inevitable for any primary-backup mechanism. The central claim is stated in the following theorem. The computing model is that of asynchronous computation, crash failures, and message-passing communication over (normally) lossy channels.

THEOREM 1. *For any generic primary-backup mechanism there exists a service and an environment for which timing disruption is inevitable during fault-free operation.*

A proof of this theorem is in [5]; we give a sketch of the key argument. The main difficulty for the proof is that it must quantify over *all* correct primary-backup mechanisms. This is done through an analysis of the *states of knowledge* of the relevant parties: the environment (E), the primary (P), the backup (B) and the service (S). Knowledge is represented by an assertion, "M *knows* b", which holds for a process M and predicate b at a computation x if b holds at *all* computations y which agree with x on the history of events for M.

A key invariant is the following: for any computation x, and a predicate b on the state of S, if E *knows* b at x, then E *knows* (P *knows* (B *knows* b)) at x. From this, we show that any increase of knowledge by E (precisely, if $\neg b$ holds at x, and E *knows* b holds at an extension y of x) requires a causal chain of messages going through the processes in the order $P; B; P; E$ in the interval (x, y). This follows from a beautiful theorem of Chandy and Misra [2] which connects knowledge gain to the existence of causal chains. The chain $P; B; P$ is a round-trip synchronization between primary and backup, which must occur before a message to E increases its knowledge.

We choose a particular E and S with a computation which can be partitioned into infinitely many disjoint intervals, in each of which E gains new knowledge. By the previous result, a synchronizing process chain is required for each of those intervals. That induces the behavior analyzed in the previous section; hence, timing disruption is inevitable.

4. SUMMARY AND ONGOING WORK

We show that any primary-backup mechanism operating in a distributed environment must disrupt timing, even in the absence of faults. This holds for both active and passive replication strategies. We conjecture that some form of timing disruption is inevitable for all generic fault-tolerance mechanisms. If this conjecture holds, it may be necessary to create protection mechanisms which are specialized to each service and to the characteristics of its media traffic.

5. REFERENCES

[1] N. Budhiraja, K. Marzullo, F. B. Schneider, and S. Toueg. The Primary-Backup Approach. In Mullender [7].

[2] K. M. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.

[3] B. Cully, G. Lefebvre, D. T. Meyer, M. Feeley, N. C. Hutchinson, and A. Warfield. Remus: High availability via asynchronous virtual machine replication. In *NSDI*. USENIX, 2008.

[4] J. Dean. Software engineering advice from building large-scale distributed systems. `http://research.google.com/people/jeff`, 2007.

[5] P. Koppol, K. S. Namjoshi, T. Stathopoulos, and G. T. Wilfong. The Inherent Difficulty of Timely Primary-Backup Replication. Technical report, Bell Labs, 2011.

[6] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[7] S. Mullender, editor. *Distributed Systems (Second Edition)*. Addison-Wesley, 1993.

[8] F. B. Schneider. Replication Management using the State-Machine Approach. In Mullender [7].

[9] D. P. Siewiorek and R. S. Swarz. *Reliable Computer Systems (Second Edition)*, chapter High Availability Systems. Digital Press, 1992.

[10] H. Zou and F. Jahanian. A real-time primary-backup replication service. *IEEE Trans. Parallel Distrib. Syst.*, 10(6):533–548, 1999.