Detecting Hidden Shared Dependencies via Covert Channels

Kaustubh R. Joshi
AT&T Labs Research
180 Park Ave, Florham Park, NJ, USA
kaustubh@research.att.com

Abstract

Designs for high availability using redundant subsystems rely on the independence of replicated subsystems to work. However, increasing prevalence of multiple administrative domains in distributed systems obscures dependency information. In this paper, we propose a new technique to detect hidden shared dependencies. We utilize the insight that shared resources often give rise to covert channels that can then be used to detect the presence of the shared resource itself. Through experiments, we show that the technique is feasible for an important subclass of dependencies - shared hidden bottlenecks.

1 Introduction

Distributed systems managed by multiple administrative entities are becoming increasingly common due to technological trends such as utility computing and service oriented architectures. In such scenarios, it is often difficult for any single entity to obtain information about the design of the whole system. For instance, information may be withheld to maintain competitive advantage, to preserve proprietary secrets, or to comply with legal requirements. Unfortunately, limited information about the sub-components of an overall system can have important design consequences. For example, when designing highly available systems, it is critical for redundant sub-systems to have independent failure modes. However, it is difficult to make such determinations without knowledge of the sub-systems' deployment configurations. For instance, if a backup web service is hosted by the same virtual web hosting provider as a seemingly independent primary service, the hosting provider resources become a hidden shared dependency and a common point of failure.

Detecting such hidden shared dependencies is an important but difficult problem. Trends such as virtual hosting and server consolidation make the problem even harder. Dependencies can be deduced through correlations in failure occurrences, but such data usually takes very long to collect. Some dependencies can be deduced through the external observables of a system. For example, do two web services use the same web server (i.e., Apache, IIS)? However, others are more difficult - e.g, Do the services have a shared network connection? Do they share the same backend? The deeper the shared dependencies occur in the system, the more difficult they are to detect.

We propose a novel approach to detect hidden shared dependencies, especially those due to resource sharing, without requiring the co-operation of the target systems. To do so, we use the concept of "covert channels" that was first introduced by Lampson in [2] as a term for processes leaking information by manipulation of their environment. It is well known that resource sharing between entities often induces covert channels that are very difficult, if not impossible to eliminate [3]. Our key insight is that conversely, if a covert channel is detected between two entities, it indicates with high likelihood the existence of a shared dependency between the two entities. Therefore, the basic idea is to attempt to communicate between the target entities over a predefined set of possible covert channels. Success indicates a dependency, and the type of the successful covert channel provides information about the type of dependency. To be effective, the communication attempts should be subtle, so as to not trigger adaptive behavior changes (e.g., resource reallocation) in the target entities.

2 Load Induced Signal Injection

To illustrate a concrete example of our approach, the covert channel we target is one in which an increased workload on one of the two target services (called the sender) causes an increased load on a shared resource, thus leading to an increase in the response time of the other service (called the receiver). Recently, [4] have shown that it is possible to stress-test a large fraction of even the largest Internet sites using bursts of relatively few requests (< 60) provided they are co-ordinated to arrive very close to each other. Our implementation uses a similar approach of send-

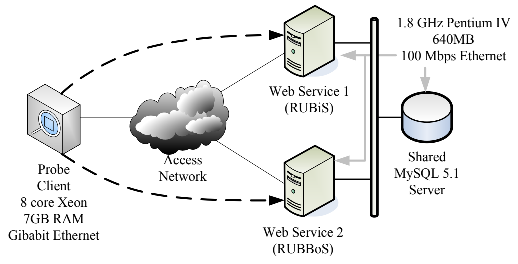

Figure 1. Experiment Setup

ing synchronized bursts, or *epochs*, of requests within several milliseconds of each other to temporarily increase the workload on the sender system.

However, unlike [4], our goal is to detect changes in response time in a completely different system. Therefore, a workload fluctuation pattern (the covert signal) that is unlikely to occur naturally and that allows good detection sensitivity is required. The solution we propose draws upon our previous work [1] on using frequency domain analysis to boost distributed systems measurement sensitivity. The number of requests in each epoch sent to the sender system is periodically fluctuated according a square wave pattern at a user-specified frequency f. The response time of the receiver system during the epochs is measured using probe requests. The response times series at the receiver is then analyzed in the frequency domain by constructing its Fourier Transform. If a frequency spike is observed at the frequency f, we assume that a hidden dependency has been detected.

By choosing the type of transaction e.g., static, large, or CGI requests, it is possible to exercise different resources in the system i.e., the front-end, network, or back-end respectively and thus detect different types of shared dependencies. For example, if a covert channel is detected using CGI scripts but not large files, then it could be possible that the target services share an application server but not a network link.

3 Experiments

To test the feasibility of our approach, we used a testbed configured as shown in Figure 1. The setup includes PHP versions of two web services commonly used as benchmarks - the RUBiS auction site and RUBBoS bulletin board applications. Each service is deployed on its own physical host. However, they use a common database server hosted on a third host. Background traffic and the covert channel workloads are generated from a separate host as shown. RUBiS is used as the sender service and RUBBoS as the receiver service.

Two experiments were conducted - in the first (the database experiment), the database oriented SearchItems-ByCategory RUBiS transaction was used as the workload, while in the second (the base experiment), the short index.html transaction was used. In both cases, 100 epochs were issued and the number of concurrent requests per epoch was toggled between 1 and 10 at a frequency of 6 cycles per experiment. The StoriesOfTheDay database oriented transaction was used to probe RUBBoS response times.

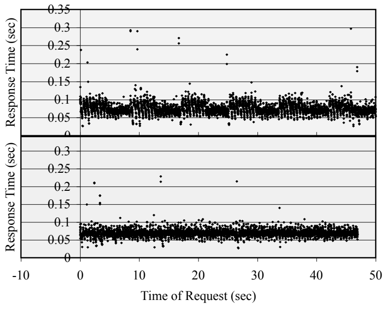

Figure 2. Original response time plot

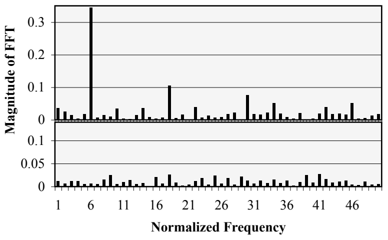

Figure 3. Frequency Domain plot

Figure 2 shows the raw response time series for RUBBoS with the database experiment on the top and the base experiment on the bottom. Periodic fluctuations can be seen in the database experiment, but not in the base experiment. That is consistent with the fact that the database experiment stresses a hidden shared dependency (the database), but the base experiment does not. The fluctuations are very small with a difference in means of 5.2msec compared to the service's standard deviation of 16msec. However, as seen in Figure 3, the plots of the magnitudes of the Fourier Transforms of the time series' paint a much clearer picture. The signal transmitted over the covert channel is clearly visible as a spike at a frequency of 6 in the database (top) results. The spike is more than 50 times the standard deviation of the remaining frequency components.

References

[1] S. Chen, K. Joshi, M. Hiltunen, W. Sanders, and R. Schlichting. Transaction dependency graph construction using signal injection. In *DSN 2007, Supplemental Volume*, 2007.
[2] B. W. Lampson. A note on the confinement problem. *Comm. of the ACM*, 16, 1973.
[3] I. S. Moskowitz and M. H. Kang. Covert channels - here to stay? In *Proc. of the Ann. Conf. on Comp. Assurance*, 1994.
[4] P. Ramamurthy, V. Sekar, A. Akella, B. Krishnamurthy, and A. Shaikh. Remote profiling of resource constraints of web servers using mini-flash crowds. In *Proc. of the USENIX Tech. Conf.*, 2008. To Appear.