ABSTRACT
Choosing the best-performing cloud for one’s application has become a critical problem for potential cloud customers. In this paper, we propose CloudProphet, a trace-and-replay tool to predict a legacy application’s performance if migrated to a cloud infrastructure. CloudProphet traces the workload of the application when running locally, and replays the same workload in the cloud for prediction. We discuss two key technical challenges in designing CloudProphet, and some preliminary results using a prototype implementation.

Categories and Subject Descriptors
C.4 [Performance of Systems]: General—measurement techniques, performance attributes; C.2.4 [Computer-Communication Networks]: Distributed Systems—distributed applications, distributed databases

General Terms
Design, Performance, Measurement.

Keywords
Cloud computing, performance, prediction.

1. INTRODUCTION
The public cloud computing market has grown dramatically in the recent years. Many companies, including Amazon, Microsoft, and Rackspace, have all released their own public cloud computing infrastructures, such as Amazon AWS, Microsoft Azure, and Rackspace CloudServers. These infrastructures, albeit offering similar services, can diverge significantly in terms of performance [3]. Hence, if a potential cloud customer is planning to migrate her legacy application into cloud, it is critical to know, prior to the migration: which cloud does my application perform best on?

The most straightforward way is to actually deploy the application on different clouds. This allows the customer to accurately test the performance of her application inside the target cloud infrastructures, but can incur significant deployment overhead and raise privacy concerns. For instance, the customer needs to install all required softwares on the cloud VMs. Some of the softwares might be proprietary and subject to licensing fees. Further, if the application is rich in data, uploading the data to each cloud can be a time-consuming process. A customer may also be unwilling to upload her sensitive data to many clouds just to test their performance.

In this work, we focus on predicting an application’s performance running on a target cloud infrastructure, without the need for deployment. There are several existing approaches for performance prediction. Standard benchmarks [1, 3] with similar workload characteristics as the application can provide a baseline for performance prediction. However, as the benchmarks are simple by design, it is challenging to map the complex and multi-tiered cloud applications to the limited set of benchmarks.

Another approach is performance modeling [5], which requires the construction of a mathematical model that can calculate the predicted performance of an application based on its execution profile. Performance models are abstract by definition and have many details omitted. For a complex system like cloud with multiple different types of resources, it is unclear how to construct a general yet precise performance model.

Our proposed system, CloudProphet, takes a different trace-and-replay approach [4] to enable accurate and application-specific performance prediction. During tracing, CloudProphet records the detailed workload information and the internal dependency of a representative application run. During replaying, CloudProphet runs an agent in the target cloud platform, which emulates the workload of the traced application run. Specifically, the agent actually replays the recorded workload while following the dependency. Finally, the performance of the agent is used to predict the actual performance of the application if deployed in the cloud.

The approach has several advantages. First, it enables application-specific performance prediction. Second, it does not require any a priori knowledge of the performance characteristics of the target cloud platform, because the agent in cloud replays real workload to test how efficient the platform is. Third, besides predicting the overall performance of the application, the approach can also predict the detailed performance of each cloud resource. We may further answer performance-related questions such as “is disk I/O likely to become the bottleneck of my application if migrated to cloud X?” Answers to such questions are critical to migration decision as well as resource planning.

In the following, we describe several key design challenges of CloudProphet.

2. CHALLENGES
2.1 Non-deterministic Application Workload
For many multi-threaded applications, such as databases and scientific computation tools [6], the workload on each thread depends on the order of the synchronization events (e.g., locks), which in turn depends on how the threads interleave with each other. Due to
the performance (e.g., CPU and I/O speed) differences in cloud, the application threads may interleave differently if migrated, which then leads to different workload. In this case, simply replaying the workload events collected locally can result in poor prediction accuracy.

We first illustrate the problem using a real example. Figure 1(a) shows part of the request handling code of a file hosting application UDDropBox, and (b) shows the lock and I/O events triggered by two application threads during tracing. Note that thread T2 fails to acquire the lock and has to sleep for a while before retrying.

A naive replay mechanism simply replays all events in each thread’s trace one after another. This may cause the replayed events to diverge from the real events of the application if migrated. For instance, if the cloud VM has faster disk I/O, T1’s unlock event may happen earlier than the first trylock event of T2 (Figure 1(c)). In this case, the first trylock of T2 would succeed, and the application would directly trigger open instead of usleep. However, if we replay strictly according to the event trace, T2 still needs to replay usleep and another trylock before open. This adds unnecessary overhead to T2.

CloudProphet introduces a novel mechanism to address this. During replay, CloudProphet detects any synchronization event that occurs out-of-order compared to the order in the recorded trace. An out-of-order synchronization event may cause the future workload events to diverge from the events in the current trace. If such an event is detected, CloudProphet pauses the replay process, and starts a new application run on the local machine to update the workload after the diverged point. Specifically, the new run is steered by enforcing the same order of the synchronization events as the one occurred in cloud [2]. After the application run has reached the diverged point, CloudProphet then collects the updated workload events, resumes the replay process, and uses those events for future replaying. The process is repeated if new out-of-order events are detected.

Consider again the previous example. With the new mechanism the cloud replayer will detect that the first trylock of T2 occurs out-of-order, because the event happens before T1’s unlock in the trace, while during replay it occurs after. CloudProphet then pauses the replay right after the trylock, and tries to update the future events through a new run of the application. During the new run, CloudProphet enforces the order between the trylock and the unlock, so that the trylock always happens after the unlock. In this way, the trylock is guaranteed to succeed, and therefore the application will immediately trigger open and other I/O functions afterwards. Finally, CloudProphet updates the event trace using the new local events that do not contain the extra usleep and trylock, and resumes the replay.

The mechanism does not make any assumption on the application model, and therefore works for arbitrary application theoretically. On the other hand, one major practical limitation is that it requires multiple runs of the application to obtain the right workload to replay, and this overhead can be prohibitively high if the application has many synchronization events.

2.2 Replay Computation Workload

To faithfully replay the computation (CPU and memory) workload of the original application, we need to trace the exact CPU instructions executed and the memory footprint. However, this can incur significant tracing overhead and increase the replayer complexity. CloudProphet instead adopts a simple linear model to map the local computation workload to the one in cloud. The model scales the CPU time measured locally by a constant factor calibrated by standard CPU benchmarks. The cloud replayer then uses a busy-loop to emulate the scaled workload. The model works reasonably well for most applications we have tested (the error rate is smaller than 30% for most cases), including memory-intensive applications in the SPLASH-2 benchmark [6].

3. PRELIMINARY RESULTS

Figure 2 shows the prediction results of UDDropBox with 1, 2, and 3 concurrent clients. Each client uploads ten 1MB files back-to-back. We predict the total processing time on an Amazon AWS m1.large instance. In comparison, we also show the real processing time when actually running Dropbox on the cloud VM, and the prediction results using the naive replay mechanism. The results of CloudProphet closely matches the real processing time. Moreover, with multiple threads the naive mechanism predicts almost ten-fold of the real processing time, suggesting that the workload collected on our local machine diverges significantly from the real workload in cloud. The result shows our approach is promising in predicting the performance of applications with non-deterministic workload.

4. REFERENCES