Portfolio-driven Resource Management for Transient Cloud Servers*

Prateek Sharma

University of Massachusetts Amherst

David Irwin

University of Massachusetts Amherst

Prashant Shenoy

University of Massachusetts Amherst

1 INTRODUCTION

Cloud computing has become popular in recent years for a wide range of applications, including latency-sensitive web services, computationally-intensive scientific workloads, and data-intensive parallel tasks. Recently, cloud platforms have introduced a new class of servers, called *transient servers*, which they may unilaterally revoke at any time. Transient servers (such as Amazon EC2 spot instances and Google preemptible VMs) typically incur a fraction of the cost of their regular (“on-demand”) server counterparts.

However, transient server revocations, which are akin to fail-stop failures, can lead to severe disruption in application performance and availability. Thus, despite the low cost of transient servers, making effective use of this new class of servers remains challenging. To minimize the effect of transient server revocations, applications must carefully select their fault-tolerance policy. Different applications, such as Spark, MapReduce, and MPI, have different tolerances to revocations, and require different application-specific mechanisms to handle revocations and their subsequent recovery. In addition, cloud providers provide a large number of transient servers with potentially different costs and failure-rates. For example, Amazon offers more than 250 spot instances in the US-East-1 region, and more than 2,500 globally. Thus in addition to fault tolerance policies, applications must also carefully select transient servers, if they seek to minimize their costs and running times.

2 SERVER PORTFOLIOS

To address the problem of transient server selection, we introduce a model-driven framework called *server portfolios*. Portfolios represent a virtual cloud cluster composed of a mix of transient server types with configurable cost and availability depending on the application’s tolerance to revocation risk and price sensitivity (Figure 1). Our portfolio model derives from Modern Portfolio Theory in financial economics, which enables investors to methodically construct a financial portfolio from a large number of underlying assets with various risks and rewards.

The flexibility and explicit risk-awareness that portfolios offer is not provided by prior work on transient server selection, which focuses largely on selecting *one* server type (among the hundreds that cloud providers offer). In contrast, portfolios are a *general* technique for constructing *heterogeneous* collections of servers for a wide

Figure 1: Given an application’s risk averseness, server portfolios provide lowest-cost heterogeneous virtual server clusters. Choosing a portfolio from a larger collection of servers (all US-east-1 vs. only r3) results in higher cost saving.

range of risk tolerances and application preferences. We use portfolio modeling as part of the design of an application-independent framework for supporting transiency, called ExoSphere. ExoSphere uses portfolio modeling to expose virtual clusters of transient servers of different types to different applications.

Along with portfolio modeling, ExoSphere also supports custom application-specific policies for handling transiency. In particular, ExoSphere adopts an Exokernel approach by exposing a set of basic mechanisms that are common to all transiency-enabled applications. These mechanisms can be used by applications to design custom policies for handling revocations, saving state, and performing recovery. We distill a general set of transiency-specific policies and mechanisms into a well-defined API that is exposed to applications. Applications then implement various policies in handler functions defined in the API, which are triggered based on changing market conditions, e.g., when prices increase or a revocation occurs.

3 EVALUATION

We have implemented ExoSphere by extending Mesos and integrating it with EC2’s spot market. ExoSphere’s transiency API allows us to develop transiency-aware variants of multiple applications such as Spark, MPI, and BOINC, with modest effort. ExoSphere simplifies the development of these applications, while also improving their performance and decreasing costs compared to prior approaches.

Compared to existing greedy approaches, ExoSphere’s diversified portfolios can mitigate the risk of simultaneous revocations by more than 10×, while achieving more than 80% cost savings compared to on-demand servers.

*Full version of the paper available at <http://lass.cs.umass.edu/publications/pdf/exosphere-sigmetrics17.pdf>

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGMETRICS '17, June 5–9, 2017, Urbana-Champaign, IL, USA

© 2017 Copyright held by the owner/author(s). ACM ISBN 978-1-4503-5032-7/17/06. DOI: <http://dx.doi.org/10.1145/3078505.3078511>