Process-Oriented Recovery for Operations on Cloud Applications

Min Fu, Liming Zhu, Anna Liu, Xiwei Xu, Len Bass
Software Systems Research Group, NICTA, Sydney, Australia
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
{Min.Fu, Liming.Zhu, Anna.Liu, Xiwei.Xu, Len.Bass}@nicta.com.au

A large number of cloud application failures happen during sporadic operations on cloud applications, such as upgrade, deployment reconfiguration, migration and scaling-out/in. Most of them are caused by operator and process errors [1]. From a cloud consumer's perspective, recovery from these failures relies on the limited control and visibility provided by the cloud providers. In addition, a large-scale system often has multiple operation processes happening simultaneously, which exacerbates the problem during error diagnosis and recovery. Existing built-in or infrastructure-based recovery mechanisms often assume random component failures and use checkpoint-based rollback, compensation actions [2], redundancy and rejuvenation to handle recovery [3]. These recovery mechanisms do not consider the characteristics of a specific operation process that consists of a set of steps carried out by scripts and humans interacting with fragile cloud infrastructure APIs and uncertain resources [4]. Other approaches such as FATE/DESTINI [5] look at the process implied by a system's internal protocols and rely on the built-in recovery protocol to detect and recover from bugs. The problem we target is at a different level related to the external sporadic activities operating on a hosted cloud application.

Our overall approach is to explicitly model and analyze an operation as a process. We divide the process into sections consisting of steps. The division criteria can be different for different purposes, such as error diagnosis, conformance checking or recovery. For recovery, our initial division criteria include: 1) atomicity to achieve all-or-nothing for a group of actions making recovery easier; 2) idempotence to enable the same or parameterized actions to be re-executed for recovery; 3) fine-granularity to allow higher-level reuse of existing steps during recovery; 4) alternatives-friendly to allow alternative actions to be executed to reach the same expected result during recovery. We specify a set of assertions representing the expected outcomes produced by the execution of each section. We use our run-time assertion evaluation and monitoring system [7] to help detect errors at the end of each section and trigger recovery actions if necessary.

We applied our approach to a typical AMI-driven rolling upgrade process for AWS-hosted applications. Netflix has a well-known tool Asgard [6] supporting this process. We divided Asgard's internal recovery and error-handling mechanisms using our division criteria. We also enabled log-triggered assertion evaluation at the end of some sections. We were able to detect errors earlier and catch errors that would be missed by Asgard. However, we found that Asgard was not designed with the granularity and idempotence required for re-execution. So we could not conduct re-execution based recovery using selected pieces of Asgard's code. We did develop external alternative actions for recovering from some subtle errors. For example, upon detection of some unexpectedly stopped instances, our alternative actions either restarted these instances or killed them and relied on the auto-scaling group to restart them. Our recovery significantly shortens the time that Asgard would have taken otherwise.

At the moment, we are developing patterns and frameworks that can allow operators to easily design and instrument their scripts for marking atomic actions, re-execution blocks and alternative actions for recovery purposes. We are also integrating this with our assertion evaluation and monitoring framework [7].

SoCC'13, 1—3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1.
http://dx.doi.org/10.1145/2523616.2525958

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] D. Oppenheimer and D. A. Patterson, “Why do Internet services fail, and what can be done about it?”, Proc. 10th ACM SIGOPS European Workshop, September 2002.

[2] C. Colombo and G. J. Pace, “Recovery within Long Running Transactions”, ACM Transactions on Computational Logic, pp. 1-40, August 2011.

[3] L. DuBois, ”Disaster Recovery for Virtualized Environments: A DR Approach to Fit the New Datacentre”, IDC Presentation, March 2013.

[4] Q. Lu, L. Zhu, L. Bass, X. Xu, Z. Li and H. Wada, “Cloud API Issues: an Empirical Study and Impact”, Proc. 9th ACM SIGSOFT conference, 2013.

[5] H. S. Gunawi, T. Do, P. Joshi, P. Alvaro, J. M. Hellerstein, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and K.Sen, and D. Borthakur, “FATE and DESTINI: A Framework for Cloud Recovery Testing”, NSDI, 2011.

[6] Website: https://github.com/Netflix/asgard (last access time: 12 Aug 2013, 12:30).

[7] I. Weber, X. Xu, and et al., “Detecting Cloud Provisioning Errors Using an Annotated Process Model”, Submitted to Middleware 2013.