Reproducibility as Side Effect

Shu Wang, Zhuo Zhen,
Jason Anderson
University of Chicago
Chicago, Illinois
{shuwang,zhenz,jasonanderson}@uchicago.edu

Kate Keahey
University of Chicago, Argonne National Laboratory
Chicago, Illinois
keahey@anl.gov

ABSTRACT

The ability to keep records and reproduce experiments is a critical element of the scientific method for any discipline. However, the recording and publishing of research artifacts that allow to reproduce and directly compare against existing research continue to be a challenge. In this paper, we propose an experiment précis framework that helps the experiment repeatability. Guided by the framework, we implement a prototype tool called **ReGen** which generates repeatable experiment scripts that can be used or shared along with a detailed experiment description automatically. Evaluation shows that **ReGen** is effective in reducing the researcher's efforts of creating a repeatable experiment in a real setting.

1 INTRODUCTION

The ability to keep records and reproduce experiments is a critical element of the scientific method for any discipline. However, the recording and publishing of research artifacts that allow to reproduce and directly compare against existing research continues to be a challenge. Computer science research is particularly difficult to reproduce when compared to other disciplines [1]. *Foremost*, this is partly due to **cultural factors**, e.g., the accepted medium of research sharing, the 8-page paper, is the primary consideration for paper acceptance and contribution evaluation. Yet, the paper itself is no longer suited to accommodate the level of detail necessary to communicate complex results, especially for applied computer science research, e.g. system, network, and database research. *Secondly*, researchers lack **incentives** for repeatable experiments, since there is a strong emphasis placed on publication of only novel and only positive results. *Finally*, it is extremely difficult to keep track of, communicate, and ultimately provide **mechanisms** to repeat and expand on existing research.

In recent years there has been an increasing recognition that being able to reproduce, conclusively compare, and directly expand the research of others is the best and fastest way to make progress in scientific and technological fields. This led to a cultural change: conferences, journal publishers, and standards organizations are beginning to encourage providing descriptions of how results can be reproduced. Yet, creating reproducible experiments today is still time-consuming: a scientist needs to take detailed notes not always knowing which specific detail will prove important and invest in streamlining their experiments, which often requires extra effort at a time when the amortization of this effort may be uncertain. Because making research repeatable is seen as a costly operation, many scientists see repeatability as a hard choice between investing the time in repeatability or advancing their scientific agenda.

Operating within a testbed creates a great opportunity to help resolve this dilemma as much of the information that is required is already recorded by the testbed in great detail: the Chameleon testbed records detailed description of hardware components, and is versioned whenever any of this information changes and allows users to create appliance versions. Furthermore, the specific resources allocated to the user, the appliance/image deployed, the monitoring of various qualities, are all recorded as part of logging activity on testbed services. In addition, most testbeds provide monitoring systems that the user can leverage to record information about experiment-specific metrics or even differentiation markers between experiments. Consolidating this already gathered information and filtering it for the user allows us thus to automatically generate a detailed and accurate description of all the actions taken to create an experimental environment and provide it to the user.

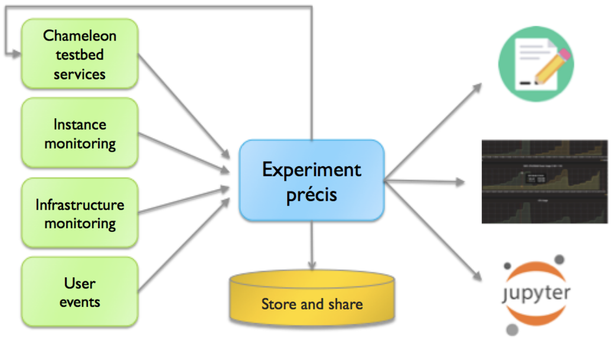

Figure 1: Experiment précis framework

In this paper, we propose the experiment précis framework that improves the experiment repeatability. We implement a prototype tool, which generates repeatable experiment scripts that can be used or shared along with a detailed experiment description automatically. We explore the possibility of experiment repeatability as a side-effect in the Chameleon testbed.

2 EXPERIMENT PRÉCIS

A Chameleon experiment précis represents exactly this information about user experiments in a form that can be consumed in multiple ways: from providing an experiment record, to its analysis, to repeating the experiment, potentially with variations. In a sense, an experiment précis is the equivalent of a Linux "history" command: it reflects the actions the user took when interacting with the system, it can be edited or processed to e.g., simplify the workflow it represents, and it can be streamed to a file and turned into a script repeating those actions that can be easily shared with others. Similarly, an experiment précis captures actions carried out in a significantly more complex environment that can be adapted in multiple ways (Fig . 1):

- **Experiment description**: an experiment précis can be used to simply as an informational tool for the user to recall or

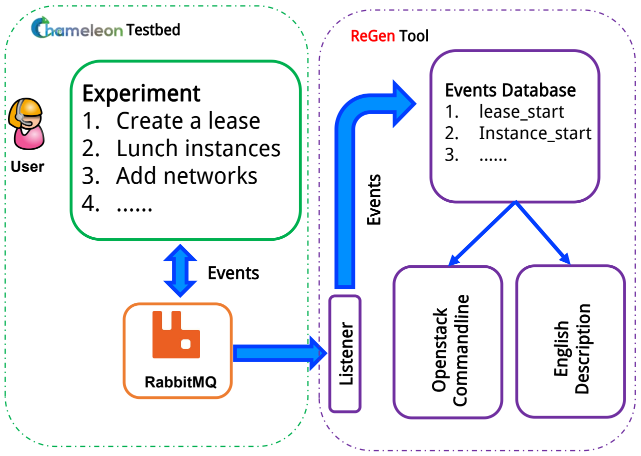

Figure 2: *ReGen* tool

share with others their experiment description; this can be done both via a machine readable format or by generating a description of the experiment in English such as can be pasted directly into the relevant sections of a scientific paper.

- **Experiment analysis**: an experiment précis record, especially one containing the monitoring information, can be mined for correlation between various factors that may influence the experiment in non-intuitive ways.
- **Real-time experiment monitoring**: the experiment information can be imported to tools such as Jupyter to facilitate both analysis and management or the experiment.
- **Repeating an experiment**: a précis, in conjunction with testbed services, can be re-enacted in either the directly recorded or modified form, e.g., by substituting the appliance that was used or making a change to the type of resources.
- **Sharing with others**: just like a script derived from the history, a précis can be easily shared with others - in particular in a form standardized to work between two testbeds.

3 REGEN TOOL

ReGen is a prototype tool for experiment précis framework. As showing in Figure 2, *ReGen* aims at *(1)* collecting user events by attaching a listener to RabbitMQ, *(2)* consolidating them into databases, and *(3)* reconstructing an OpenStack command-line script along with detailed experiment description.

For the listener, we modify the configuration of OpenStack services used by Chameleon, so that it can emit detailed events information. All the event notification messages are bound to our ReGen listener. These events are imported into the database, and analyzed by ReGen . ReGen generates an OpenStack command-line script so that the user can easily repeat their experiment, keeping detailed records, and sharing the experiment with others.

Yet, ReGen is still facing at least the following two challenges:

- **Event Mapping**: It is important to filter out some unrelated/trivial events, and map the remaining into a machine-readable script.
- **Parameter Filling**: The script not only contains the general command-line operation, but also requires huge amount of configuration parameters, e.g. a specific instance id need to know before assigning the IP address to it. Some of them are static determined by default or experiment requirement, some of them are dynamic generated. How to automatically identify different type and fill with corresponding value are still challenging.

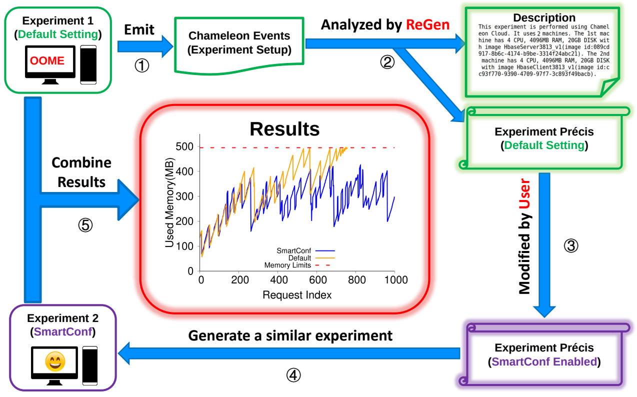
Description
This experiment is performed using Chameleon Cloud. It uses 2 machines. The 1st machine has 4 CPU, 4096MB RAM, 20GB DISK with image HbaseServer3813_v1(image id:089cd917-8b6c-4174-b9be-3314f24abc21). The 2nd machine has 4 CPU, 4096MB RAM, 20GB DISK with image HbaseClient3813_v1(image id:cc93f770-9390-4709-97f7-3c893f49bacb).
Experiment 1 (Default Setting)
OOME
Emit
①
Chameleon Events (Experiment Setup)
Analyzed by ReGen
②
Experiment Précis (Default Setting)
Modified by User
③
Combine Results
⑤
Results
Used Memory(MB)
Request Index
SmartConf
Default
Memory Limits
Experiment 2 (SmartConf)
Generate a similar experiment
④
Experiment Précis (SmartConf Enabled)

Figure 3: Evaluation of *ReGen*

4 EVALUATION

We evaluate *ReGen* in the DevStack environment (emulated actual Chameleon Testbed). A benchmark from Wang et al. [2] is used (This benchmark is about the default RPC queue size in HBase being too large, potentially causing an out of memory error). The goal is to show how effective our tool is for reproducing the experiment.

As shown in Figure. 3, the experimenter starts with an experiment using the default queue size. Under this default setting, the HBase Server will go out of memory very soon. At the same time, the experiment setup events are collected as a side effect. This requires no intervention from the experimenter. These events will be analyzed by ReGen .

ReGen summarizes the experiment environment, formalizes a standardized description, including both hardware (e.g. cpu, memory) and software information (e.g. image id), which could be directly incorporated into a scientific paper.

ReGen also generates an experiment précis for original experiment. The experimenter could easily modify it to invoke the same or similar experiment, like enabling SmartConf framework so that queue size could be adjusted automatically and avoiding out of memory error. Finally, the experimenter could combine results from these two experiments, and generate overall comparison between them.

5 CONCLUSIONS

In this paper, we propose an experiment précis framework for repeatable experiment, which eases the researcher's burden when preparing a repeatable experiment. We demonstrated that it is possible to capture a major part of experiment information automatically and faithfully as a side effect on the Chameleon testbed. We are able to use the captured information to repeat the experiment with controlled modifications. This also allows us to easily share our experiment with others.

REFERENCES

[1] Christian Collberg and Todd A Proebsting. 2016. Repeatability in computer systems research. *Commun. ACM* 59, 3 (2016), 62–69.
[2] Shu Wang, Chi Li, Henry Hoffmann, Shan Lu, William Sentosa, and Achmad Imam Kistijantoro. 2018. Understanding and Auto-Adjusting Performance-Sensitive Configurations. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 154–168.