The Truth Is Out There: Understanding System Behaviour By Exploiting System Logs

xxx

1 Introduction

As industry is moving towards *cloud computing*, many players in the industry are building large and complex data center applications on top of clusters of commodity PCs. Ensuring the correctness and performance of the applications are critical to providing sustained service to clients. However, bugs have been challenging service developers and maintainers continually.

However, the inherent complexity of the system hinders people from understanding the system easily. The system are usually built in a tiered fashion, with each tier providing certain abstraction to the upper layer. Within a logical tier, its function is implemented as a distributed system. It may consist of hundreds or thousands of distributed processes which works coorperately to fulfill the requests from upper tier. Within a particular process, asynchronous staged handling of messages is well adopted to fully utilize the computing resource on a node. By leveraing the above techniques, developers have built large scale of complex systems to serve requests worldwide. The handling of a request is splitted into pieces of tasks which are executed distrbutedly across tiers, processes and stages.

Understanding runtime behaviour of the complex system is key to verify system design, debug its correctness and performance problems. By tracking task pieces and their causal dependencies, we can construct task flow by linking together pieces of its execution throughout the system. In our previous work, we further developed techniques to automatically tracking tasks and infer hierarchical task models. Using the task models, developers can better understand the structures of components and their dependencies, and use debugging tools to instrument the system and verify the behavior of tasks at appropriate layers.

The production system contains abundant logging information on system status, but it is not fully explored yet. It is useful for two reason. First, the log is added by developers who are familiar with the system. The resulting log is a faithful records of system runtime behaviour. Second, log usually contains both state report and high level descriptions. The derived task models is of value for both automatic processing and human understanding.

The hierarchical structure of tasks is often consist with the hierarchy of data processed by tasks. For example. In cosmos, data are organized as streams and streams are consisted of extents. By inspecting cosmos logs, task that processing a stream is splitted into subtasks. The first subtask do some stream level processing (open, etc.), and the following tasks process extents in the stream. But the task boundaries are not marked in the log.

By mining and leveraging data hierarchy, we can use the information to infer the task hierarchy. A log item which starts to process a stream marks the begin of a task. The task lasts before the log item which process another stream. Within the task, it is splitted into subtasks at extent-processing boundary. In this paper, we present a method for automatically infer data hierarchy recorded in system log and use the information to construct task models. We also describe our experiences in using the inferred task models to understanding the system design and debug performance problems.

2 Design

In this section, we describe the detailed process of mining data hierarchy using system log. It works in two phases.

2.1 Extract key-value pairs

data format are limited, we can find data part easily, and track back in the text to find its key name.

2.2 Mining hierarchical relation of keys

- rule 1: keyS $>$ keyP, then keyS comes before keyP
- rule 2: keyS $>$ keyP, then there're 1-to-many mapping relation between keyS and keyP value set.

2.3 Construct hierarchical task models

this step is straightforward.

3 Discussion

Discussion. topic need decided

4 Experiences

4.1 Understanding system behaviour

understanding cosmos client, en and csm

4.2 Guide on debugging

using the model to guide on debugging cosmos network library.

5 Related Works

Related Works

6 Conclusion

Conclusion

7 References

References

References