1 Definition

1.1 Offline and Online Reconfiguration

Before giving the definition of *Reconfigurability*, we need to first discuss how system administrators reconfigure systems in the real world. The systems that we focus in this paper are the distributed systems that are widely deployed in the cloud such as Hadoop and MySQL.

Essentially, there are two methods to reconfigure such systems: offline and online.

In regard to offline reconfiguration, system administrators shutdown the whole system, make whatever reconfiguration needed, and then restart the whole system. Although this method provides a relatively safer way to reconfigure the system, it hurts system availability during the reconfiguration period. We call reconfiguration methods of this type as offline reconfiguration.

As opposed to offline reconfiguration, system administrators might also be able to make reconfiguration in an online fashion which can keep the system running normally during reconfiguration. Some systems (e.g., HDFS) provide native support for online reconfiguration for limited parameters. In addition, for many distributed systems, because components are usually replicated for fault tolerance, they natually own the opportunity to achieve online reconfiguration by incrementally performing offline reconfiguration for only a portion of the replications. We call the reconfiguration methods that keep the system running during reconfiguration period as online reconfiguration.

1.2 Definition of Online Reconfigurability

In this section, we will derive the definition of online reconfigurability gradually.

A reconfiguration operation can be defined as a 5-tuple (*m*, *p*, *v1*, *v2*, *t*), where *m* is the given reconfiguration metho, *p* is the parameter to be reconfigured, *v1* and *v2* are the values of *p* before and after reconfiguration, and *t* is the reconfiguration timing relative to global states. To simplify the discussion, we assume there are only two reconfiguration methods (i.e., *offline_m* and *online_m*) and they are implemented correctly.

To facilitate the definition, we assume the existence of a perfect **test oracle** which has the power to verify system correctness completely and accurately, involving functionality, availability, consistency. The oracle will verify the reconfigurability of parameter *p* by executing the given reconfiguration method *m* with different *v1*-*v2* value pairs at different timing points *t* until the oracle believes the verification is complete. If the oracle returns *OK*, it indicates that the system is correct and no errors will occur in the future caused by the reconfiguration; otherwise, the oracle returns *ERROR*.

Now, we will give the definitions of different levels of parameter reconfigurability. Parameter *p* is

- Online reconfigurable if: $\forall$ v1, v2, t, oracle(*offline_m*, *p*, *v1*, *v2*, *t*) == *OK* $\Longrightarrow$ oracle(*online_m*, *p*, *v1*, *v2*, *t*) == *OK*.
- MAYBE offline reconfigurable but NOT online reconfigurable if: $\exists$ v1, v2, t such that oracle(*offline_m*, *p*, *v1*, *v2*, *t*) == *OK* && oracle(*online_m*, *p*, *v1*, *v2*, *t*) == *ERROR*.
- Offline reconfigurable if: $\forall$ v1, v2, t, oracle(*boot_m*, *p*, *v1*, *t*) == *OK* && oracle(*boot_m*, *p*, *v2*, *t*) == *OK* $\Longrightarrow$ oracle(*offline_m*, *p*, *v1*, *v2*, *t*) == *OK*. *boot_m* indicates that no reconfiguration method needs to be used.
- Configurable(bootable) but NOT offline reconfigurable if: $\exists$ v1, v2, t such that oracle(*boot_m*, *p*, *v1*, *t*) == *OK* && oracle(*boot_m*, *p*, *v2*, *t*) == *OK* && oracle(*offline_m*, *p*, *v1*, *v2*, *t*) == *ERROR*.
- Every parameter should be configurable(bootable).

People may argue why we have to define parameter reconfigurability in this way. For example, why not simply define online reconfigurable as $\forall$ v1, v2, t, oracle(*online_m*, *p*, *v1*, *v2*, *t*) == *OK*, and offline reconfigurable as $\forall$ v1, v2, t, oracle(*offline_m*, *p*, *v1*, *v2*, *t*) == *OK*?

It is because we are unable to verify whether the given parameter value is valid or not. With the definition above, as long as we could find one invalid value, then parameter *p* will not be regarded as online/offline reconfigurable. Parameter values are invalid not only because the values are wrong but also may because the parameter value has conflict with other parameter configuration. Essentially, it is a configuration validation problem and, as far as we know, there is no tools that can validate configuration with high accuracy.

We did not make another assumption about the existence of an oracle that can always validate configuration correctly. Because configuration correctness is not necessary for us to define parameter reconfigurability.

Instead, our definition of parameter reconfigurability is based on the following hidden assumption: online reconfigurable $\Longrightarrow$ offline reconfigurable $\Longrightarrow$ configurable(bootable).

1.3 Defition without Test Oracle

In this section, we will remove the assumption used in the former definition.