**Cloud Test Service: Reloading Tests to Cloud**

**Summary**

Cloud Test Service (<http://cloudtest.sh.intel.com>) is a user-friendly resource and application management framework, specially designed for diverse and complex software products’ quality assurance. Cloud Test Service provides XML descriptor to define interactive web UI layout to accept end users’ inputs, together with adaptive control flow to agilely execute tasks. It also abstracts a set of Java API to build cross-platform, manageable tasks. With built-in support for emerging cloud techniques, Cloud Test Service is able to quickly enable various application testing tasks to meet on-demand business requests.

**Description of Problem**

Software product testing facility is a special software, which is designed to test and validate another software or any other products. Although there’re many mature frameworks for teams of various scale, we think there’re still improvement areas for such frameworks:

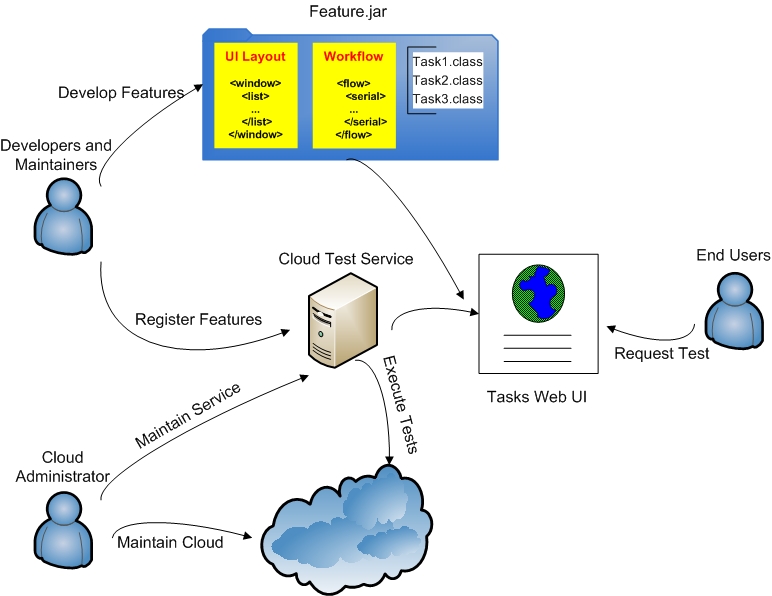
* Traditional testing frameworks usually manage the resources statically, which leads to inefficient resource sharing across teams. It’s also hard to meet rapid ever changing business requirements as more maintenance effort is required while the resource continuously growing. Considering an urgent customer workload which requires all team’s resource, traditional testing frameworks have to be terminated firstly, then engineers reconfigure or even reinstall all the hardware resource to be suitable for the requested test. After the test is finished, all the hardware resource have to be restored for original testings. This is even worse when the team works on multiple projects and each project has its own fixed resources.
* Traditional testing frameworks are usually based on scripts or unit tests which are not intuitively to implement rich interactive user interface and parallel testing. Just like other software products, testing frameworks are also expected to be a SaaS (Software as a Service) to improve productivity of testing and validation. In most cases, a product testing web site is supposed to provide various UI elements such as ListBox, TextInput, RadioButtons, etc to accept end users’ test configuration, then the framework uses the received configuration to lauch the corresponding testing tasks. This obviously relieves developers and testers from manually executing complicated testing tasks with a huge list of options. Unfortunately, traditional testing frameworks are not designed to easily get SaaS ready.

Let’s take famous CruiseControl as an example, it provides excellent continous integration, monitoring and reporting. However, CC (CruiseControl) doesn’t have native support for distributed testing and automatic resource management. It’s also not trivial to implement interactive web interface for testing tasks. CC reaches its design target well, however is there any possibility to enhance team’s productivity even more?

**Solution**

Cloud Test Service is such a sophiscated web application written in Java which tries to resolve above problems by various ways:

* The built-in support for various Cloud APIs helps Cloud Test Service to dynamically manage virtualized machine resources. After correctly configured, Cloud Test Service knows how to connect to all the virtual machines to monitor and dispatch the executions. Now, the maintenance effort for a large pool of physical servers is transformed to maintain a private cloud by the team, or even nothing if all tests are using the virtual machines in the public cloud. Creating different configurations for different testing tasks is simplified to create different virtual machine images, and then register them to the cloud. Now still considering an urgent customer workload testing request, engineers are just to create a set of virtual machine images which meet customer’s requirements, and configure them in the Cloud Test Service. There’s no hardware resource maintenance effort to support this task.
* Cloud Test Service provides a unified web interface to manage all resources, testings, and configurations which is the portal for the team. And the usage model could be as below:

Typically there’re 3 roles in a team: Cloud Administrator, Test Developers, and others.

1. Cloud Administrator maintains the private cloud and the whole infrastructure used by Cloud Test Service.
2. Test Developers are responsible to develop new testing tasks to meet team and customers’ requirements, then register into the Cloud Test Service.
3. Other team members usually navigate to Cloud Test Service web site to submit tasks, and wait for the result. He or she can also monitor the submitted tasks.

* Cloud Test Service is based on an extensible architecture which defines a set of interfaces for external plugins. The plugin’s XML description file defines the Web UI layout and task execution workflow. The Web UI layout is dynamically rendered to an interactive web page, so that user can easily configure all required testing parameters. When user clicks “Submit”, all the parameters are transmitted to the runtime engine of Cloud Test Service, which dynamically executes all tasks defined in the XML description. The actual execution workflows are the Java applications bundled in the same plugin, which strictly follow the defined application programming interface. Traditional testing framework can be easily migrated to Cloud Test Service by adopting the provided API so that enable the SaaS style in a graceful way.

**Results/Impacts/Relevance**

Although Cloud Test Service is not originated from CruiseControl, it aims to be a better continous integration framework so that help to in futher enhance Intel software products’ quality.

The project was delivered successfully and achieved all planned features in Q4, 2010. Current deployment uses Eucalyptus as the private cloud provider, and dramatically reduced resource management efforts. To support team’s projects, 10 physical servers are configured for Cloud Test Service framework to support at least 50 virtual machines with 15 types of virtual machine images covering CentOS/Redhat/SuSE Linux, Windows XP, Windows Server 2003 and Windows Server 2008. The Cloud Test Service really enhanced team’s testing efficiency. It saved at least 40 hours (4 hours for installing operating system and basic testing environment \* 10 supported platforms) for our typical product release test. We also developed 3 plugins to migrate legacy testing framework to Cloud Test Service. The testing experience is improved a lot due to the dynamic resource sharing, and on-demand self-help style web interface provided by Cloud Test Service.

**Lessons Learned**

There seems no silver bullet for software quality, so Cloud Test Service can not suit to all areas of software product testing. However, this leaves room for us to innovatively invent and integrate more and more techniques to improve Intel software quality.

References, if any

CruiseControl: <http://cruisecontrol.sourceforge.net/>

SaaS: <http://en.wikipedia.org/wiki/Software_as_a_service>