基金项目论文

基于 OpenStack 的云测试平台及其性能分析研究

丁小盼[1]，周浩[1]，贺珊[1]，陈朱管[2]，郭东辉[1,2]

(1. 厦门大学信息科学与技术学院 福建厦门 361005; 2. 厦门大学-新大陆 SOC 实验室 福建厦门 361008)

摘 要: 云测试是基于云计算的软件测试新模式。在传统的软件测试中，当测试任务高并发、原有的硬件资源无法满足测试时间要求时，需要更换硬件资源、重新配置测试环境。使用云测试可以实现按需分配硬件资源、无需重新配置测试环境,因而比传统软件测试更能节省测试成本。介绍了如何在 OpenStack 云平台上配置和运行软件代码覆盖率测试工具 SAT。实验表明，当测试任务高并发时，所搭建的云测试平台可以在不用重新配置环境的情况下满足用户对测试时间的要求，降低了用户测试成本。

关键词: 云计算；OpenStack；软件测试；云测试平台；SAT

中图分类号: TP393 **文献标识码:** A **DOI:** 10.3969/j.issn.1003-6970.2015.01.002

本文著录格式: 丁小盼，周浩，贺珊，等. 基于 OpenStack 的云测试平台及其性能分析研究[J]. 软件，2015，36(1)：6-10

Performance Study of Cloud Testing Platform Based on OpenStack

DING Xiao-pan[1], HOU Hao[1], HE Shan[1], CHEN Zhu-guan[2], GUO Dong-hui[1,2]

(1. *Department of Information Science and Technology, Xiamen University, Xiamen Fujian* 361005, *China*;
2. *Xiamen University-Newland Group SOC Lab, Xiamen Fujian* 361008, *China*)

【Abstract】: Cloud testing is a new method based on cloud computing to test software. In traditional methods, the original hardware resource may be inadequate when a large amount of concurrent test requests occur. In such case, the hardware resource should be redistributed to establish the needed test environment. However, the cloud testing can achieve distributing the hardware resource on-demand and need not to re-establish the test environment. Consequently, the test cost can be reduced. This paper presents how to combine OpenStack and SAT in order to build a cloud testing platform. The experiment validates that the established platform could both meet the demand on test time and reduce the test cost when massive concurrent test requests occur.

【Key words】: Cloud computing; OpenStack; Software Testing; Cloud Testing Platform; SAT

0 引言

云测试是基于云计算的软件测试新模式。软件测试工具 SAT[1]是一款常见的针对 C 语言的代码覆盖率测试工具。采用云计算平台来运行 SAT 工具不仅可以提高软件的测试效率，而且可以降低软件开发的测试成本。

目前主流的云测试平台中[2]，Soasta 主要用于测试 Web 应用和网站的性能；Sauce Labs 提供基于 Selenium 的测试服务，可跨多浏览器测试 Web 应用；BlazeMeter 主要用于负载与性能测试，能够创建负载测试脚本。这些平台能够实现的主要功能是性能测试、负载测试和 Web 应用测试服务，并没有涉及到软件代码覆盖率测试服务。常用的 C 代码覆盖率测试工具主要有 ATAC[3]、GCOV[4]和 SAT，其中 ATAC 和 GCOV 都是通过对代码的每个基本块进行插装来获取代码覆盖信息，而 SAT 是基于超级块支配图[1]插装的软件测试工具。它在不丢失代码覆盖信息的情况下能对插装探针个数进行有效优化，减少代码插装对被测程序性能的影响。但在 SAT 的常用运作模式中[1]，SAT 测试环境都是搭建在一台计算资源已经固定的单机环境上，而在系统使用高峰期，出现测

基金项目: 国家自然科学基金资助项目(No.61274133)；高等学校博士学科点专项科研基金资助项目(No.20090121110019)

作者简介: 丁小盼(1991–)，男，硕士研究生，主要研究方向为云计算与软件测试

通信联系人: 郭东辉，教授，主要研究方向为集成电路设计

试任务高并发的情况下，由于计算资源的限制，会导致任务测试时间明显增加，无法满足用户对测试时间的要求。

传统的软件代码覆盖率测试在搭建好测试环境后，当业务变更或业务量增加时，由于原有硬件资源的限制，会导致用户的测试时间增加。为了降低用户的测试时间，需要更换硬件配置，重新配置测试环境(如系统重装，软件的安装、配置文件的修改等)，而这些工作无疑会加大测试成本的投入[5]。使用云计算平台则可以有效地解决这个问题。云计算平台具有“按需服务”的特点[6,7]，用户可以按需部署测试资源，不需要重新配置测试环境，因而测试成本可以明显降低。

除了 Soasta、Sauce Labs 和 BlazeMeter 等商业性的云测试软件之外，目前也有一些成熟的、开源的云测试软件可以利用，包括业内熟知的 Hadoop、Eucalyptus、Openstack、CloudStack 和 OpenNebula[8]。其中，OpenStack 虽然出现较晚，但是因为官方文档相对完备，安装配置非常容易，且兼容亚马逊公共云平台，因此很快获得广泛推广。OpenStack 已被成功地应用于数据仓库系统、混合存储系统、Web 应用测试等领域[9-11]，但在软件代码覆盖率测试方面还没有相关应用。

本文的第二部分将介绍 SAT 的基本原理和 SAT 在 OpenStack 云平台上运行的工作机制；第三部分阐述如何在 OpenStack 上配置 SAT，以形成一个完整的云测试平台；第四部分对这一平台的性能进行分析评估；最后，在第五部分给出本文结论。

1 相关原理

软件测试的一般流程为首先根据程序的内部结构设计测试用例集，然后将测试用例作为输入，对被测程序的运行结果进行分析。而在度量测试的完整程度，判断测试用例是否足够、测试何时可以结束时，就需要用到代码覆盖率测试。代码覆盖率测试工具 SAT 工作流程如图 1 所示[1]。

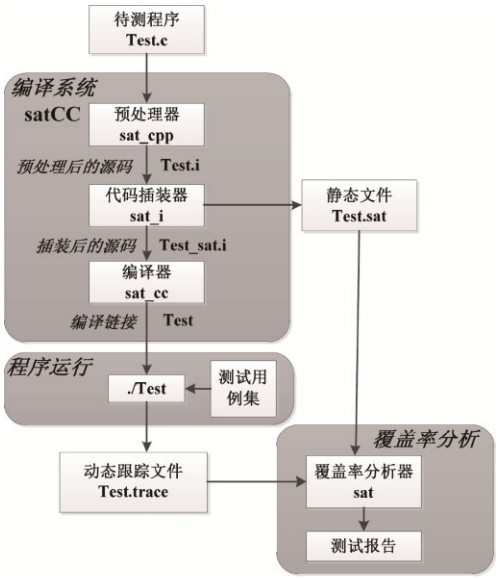

图 1　SAT 工作流程图

由图 1 可知，对任意一个待测程序，SAT 完成一个完整的测试流程都需要三个步骤：编译系统 satCC 将待测程序编译成可执行文件；输入测试用例，运行编译后的程序；覆盖率分析器 sat 根据前两个步骤产生的静态文件和动态跟踪文件，获得测试报告。在步骤一中，预处理器 sat_cpp 先将待测程序 Test.c 的源码展开，得到预处理后的源码 Test.i；接着代码插装器 sat_i 对 Test.i 进行分析，得到记录程序行列信息的静态文件 Test.sat，同时对 Test.i 进行插装，获得插装探针函数后的源码 Test_sat.i；最后编译器 sat_cc 对 Test_sat.i 进行编译，并链接相应库函数，生成可执行文件 Test。在步骤二中，将测试用例作为输入，运行程序 Test，生成记录程序运行情况的动态追踪文件 Test.trace。在步骤三中，覆盖率分析器通过对 Test.sat 和 Test.trace 文件进行分析，生成测试报告，获得 Test 程序代码的覆盖率情况。

从上述 SAT 的工作流程可知，每一个待测程序都需要经过插装编译、程序运行、覆盖率分析这三个步骤。每个步骤所执行的程序分别为 satCC、Test、sat。这三个程序会产生对应的进程，假设为 P1、P2 和 P3，它们按顺序依次耗费 CPU 时间来执行。在 SAT 的常用运作模式中，假设测试环境有 N 个 CPU 内核，有 M 个待测程序。当 M=1 时，P1、P2、P3 都只有 1 个，它们将按顺序执行；当出现测试任务高并发的情况，即 M 大于 N 时，P1、P2、P3 进程都对应有 M 个。我们以 M 个 P1 进程的执行情况为例，P2、P3 进程的执行情况类似。多任务操作系统中，在某一个时间点，一个 CPU 只能执行一个进程，所有正在运行的进程会交替占用 CPU，N 个 CPU 也只能并行执行 N 个 P1 进程，剩下的 P1 进程会存放在队列中，等待其它进程的 CPU 时间片用完，才会去执行。因此，在 CPU 内核数固定的情况下，测试任务并发请求数越大，用户需要的测试时间就越长。

为了降低用户的测试时间，需要增加用户所使用的测试资源。传统增加测试资源的方法是通过购买配置更高的硬件，但更换硬件后需要重新配置测试环境，这又会引入新的测试成本。本文通过在 OpenStack 云平台上运行 SAT 工具来改进传统方法存在的缺陷。在 OpenStack 云平台上运行 SAT，就是通过 Web 或远程工具连接云平台提供的客户机(VM)，在客户机上配置和运行 SAT。OpenStack 云平台具有按需提供计算服务的特点：当

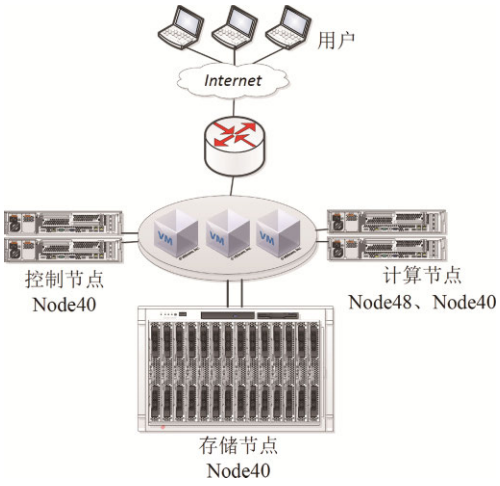

图 2　**OpenStack** 云平台工作原理

测试任务的并发请求量较大时，云平台可以根据用户需求来增加计算资源供 SAT 使用。这一功能是通过将客户机在两个物理节点之间进行迁移实现的。客户机在迁移过程中会重启，但软件配置不会发生改变，因而无需用户重新配置测试环境，云平台就可满足用户对测试时间的要求。OpenStack 云平台的原理如图 2 所示[12]，它通过高速互联网络将大量分布式的计算、存储节点互连，组成大型虚拟资源池，并由控制节点进行管理和配置。终端用户可以通过互联网访问平台提供的虚拟资源。

OpenStack 主要采用 KVM(kernel-based virtual machine)虚拟化技术[13,14]将处理器、内存、IO 设备等硬件设施虚拟化。KVM 包括内核空间的 kvm 模块以及用户空间的 qemu-kvm 程序。kvm 模块负责实现处理器和内存的虚拟化，剩下的大部分 IO 设备的虚拟化(比如网卡、显卡、存储控制器和硬盘等)则由 qemu-kvm 负责。OpenStack 将底层硬件设施虚拟化，组成虚拟资源池后，用户相当于在使用裸机和磁盘，可以按需使用这些虚拟资源创建客户机。

在云平台中创建客户机后，由于业务的变更或业务量的增加，需要对客户机进行扩展。目前，OpenStack 中提供 Resize 功能实现客户机扩展。Resize 是 OpenStack 提供的 API，它通过将客户机在计算节点间迁移，来实现客户机硬盘、内存和 CPU 大小的调整。因此，当测试用户需要增加或减少硬件资源时，只需向平台管理员进行申请，管理员利用 Resize 功能就可以按需分配该测试用户所使用客户机的资源。

2　云测试平台的实现

有两种方案可以部署 OpenStack 云平台：一是参考 OpenStack 官方文档，依次手动安装 Nova(计算服务)、Swift(存储服务)、Neutron(网络服务)、Glance(镜像服务)、Keystone(认证服务)以及 Horizon(UI 服务)组件；二是采用自动化部署工具完成上述组件的安装。由于手动搭建 OpenStack 需要安装的软件和修改的配置文件太多，容易出错，因此我们采用自动化部署工具来进行安装。

表 1　构造云平台主要开源软件

名称	版本	说明
CentOS	6.4	开源 linux 操作系统
OpenStack	Havana	开源云平台管理软件
Packstack	Havana	OpenStack 部署工具

首先将 CentOS 6.4 作为平台的基础操作系统。CentOS 以最小化方式安装好后，采用 Packstack 作为自动化部署工具。Packstack 由 Puppet 语言编写，支持单节点 all in one 和多节点两种方式部署 OpenStack。本文采用 Packstack 的双节点 Gre 方式来部署 OpenStack 的“Havana”版本，平台使用的主要开源软件见表 1。另外，使用 Packstack 部署 OpenStack 要求硬件最少有 2G RAM、CPU 支持硬件虚拟化扩展以及至少一张网卡。

本文使用 2 台服务器来搭建 OpenStack 云平台，处理器都采用 Intel Xeon E5530，分别为节点 node40 和 node48。它们通过交换机连接，处于同一个局域网内，如图 2 所示。其中，node40 充当了控制节点、网络节点、存储节点和计算节点的角色。其 eth0 绑定 IP 为 192.168.0.40/32，连接公网；eth1 的 IP 为 10.1.1.40/32，连接内网。node48 是计算节点，只提供计算服务，其 eth0 对应 IP 为 192.168.0.48/32，连接公网；eth1 的 IP 为 10.1.1.48/32，连接内网。在运行 Packstack 时，最重要的配置文件是 Packstack-answers.txt，它记录了 OpenStack 需要安装的组件、主机的 IP 地址、数据库的账户和密码，以及用户访问 Dashboard 的账户密码等等。本文所使用的关键配置如下。

其中，配置 1-5 表示将存储、网络、镜像、认证及 UI 服务安装到 node40；配置 6-7 表示 node40 充当控制和计算节点，node48 充当计算节点；配置 8 表示通过同步时间服务器 0.uk.pool.ntp.org 的时间来确保计算节点和控制节点的时间同步；配置 9-11 表示控制节点与公网的接口为 eth0，控制、计算节点与内网的接口都为 eth1；配置 12-13 表示客户机之间采用 Gre 隧道协议进行通讯，且允许同时存在 1000 个 Gre 隧道数量；配置 14 表示设置管理员账号登录 Dashboard 控制面板的密码。

OpenStack 云平台搭建完成后，还需在云平台中创建客户机。在客户机上安装配置 SAT，并配置客户机的网络和防火墙规则，这样才能使测试用户使用云平台提供的软件测试服务。首先，我们使用 neutron 相关命令为用户创建虚拟网络，使用户的客户机能够正常连接互联网，关键代码如下：

```
neutron net-create net1
neutron subnet-create net1ip --name subnet1
neutron net-create net2 --provider:network-type local
```

其中 *net1* 是连接互联网的外网网络，*ip* 是外网网络的网段，*net2* 是客户机所处的内网网络，两者之间通过虚拟网关相连。为了保护客户机的安全，还需要设置客户机的防火墙规则，允许指定端口能被访问，关键代码如下：

```
nova secgroup-add-rule security_group tcp 22 22 0.0.0.0/0
nova secgroup-add-rule security_groupicmp -1 -1 0.0.0.0/0
```

表示该客户机开放了 22 端口和允许 icmp 协议的包进入，即允许 ping。最后，使用 nova 相关命令创建客户机，关键代码如下：

```
nova boot --flavor flavor_id--image image_id --security-groups sec_group    instance_name
```

其中 *flavor_id*，*image_id*，*instance_name* 分别对应客户机所使用的云主机类型，镜像文件以及客户机名称。上述步骤完成后，用户通过 Web 浏览器访问 http://192.168.0.40，就能连接新创建的客户机。

```
1. CONFIG_SWIFT_STORAGE_HOSTS=192.168.0.40
2. CONFIG_NEUTRON_SERVER_HOST=192.168.0.40
3. CONFIG_GLANCE_HOST=192.168.0.40
4. CONFIG_KEYSTONE_HOST=192.168.0.40
5. CONFIG_HORIZON_HOST=192.168.0.40
6. CONFIG_NOVA_NETWORK_HOSTS=192.168.0.40
7. CONFIG_NOVA_COMPUTE_HOSTS=192.168.0.40,192.168.0.48
8. CONFIG_NTP_SERVERS=0.uk.pool.ntp.org
9. CONFIG_NOVA_NETWORK_PUBIF=eth0
10. CONFIG_NOVA_NETWORK_PRIVIF=eth1
11. CONFIG_NOVA_COMPUTE_PRIVIF=eth1
12. CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=gre
13. CONFIG_NEUTRON_OVS_TUNNEL_RANGES=1:1000
14. CONFIG_KEYSTONE_ADMIN_PW=2aac98ac0cc64c9b
```

图 3　Packstack 配置

3　性能分析

为了评估 SAT 在 OpenStack 云平台的工作性能，本文设计了对比试验，即比较 SAT 在工作站和 OpenStack 云平台上完成相同数量测试任务所需的时间。

具体设计如下。首先在工作站 1、工作站 2 和 OpenStack 云平台的客户机上都最小化安装 64 位的 Ubuntu12.04，接着采用源码安装的方式将软件测试工具 SAT 部署到工作站和客户机上。为了防止其他服务对测试的影响，除了开启必要的如 sshd、network 等服务外，其他服务都不开启。工作站和 OpenStack 云平台客户机的基本初始配置对比如表 2。其中，客户机和工作站的内存、硬盘大小都保持一致；客户机初始 CPU 内核数也和工作站 1 的一致。

表 2　硬件配置对比

名称	云平台客户机	工作站 1	工作站 2
CPU 型号	Intel XeonE5530	Intel Xeon E3-1220L	IntelXeonX5450
CPU 内核	2 个	2 个	8
内存	4G	4G	4G
硬盘	500G	500G	500G

本文选择 Bubble[15]作为待测程序，它利用冒泡排序算法对一个文件中的数字进行排序，是一个简单但有代表性的样例。Bubble 中的排序算法主要消耗的是 CPU 资源，可以方便我们观察实验现象。我们用 SAT 对 Bubble 进行覆盖率测试。首先，SAT 利用 satCC 去插装并编译 Bubble 源码；接着把需要排序的数据文件作为测试用例，运行 Bubble；最后用 sat 去分析 Bubble 的静态文件和动态追踪文件，获得测试报告。SAT 对 Bubble 的测试时间主要由 satCC、Bubble 和 sat 这三个程序运行所耗费的时间组成。为了准确测量每个程序的运行时间，我们采用 Linux 系统提供的 time 命令。该命令利用计时器来记录进程占用 CPU 的累计时间，可以准确统计出一个程序从开始运行到结束所耗费的时间。

当有多个程序需要并发测试时，假设需要测试的程序都为 Bubble，且个数为 Num，随着 Num 的增加，即测试任务并发请求数增加，SAT 在工作站和云测试平台中分别完成 Num 个 Bubble 测试的时间如表 3 所示。其中，N1 表示工作站 CPU 内核数，N2 表示云平台客户机 CPU 内核数。

表 3　SAT 测试时间对比

Num	工作站 1/s (N_1=2)	云测试平台/s (N_2=2)	云测试平台/s (N_2=4)	云测试平台/s (N_2=6)	云测试平台/s (N_2=8)	工作站 2/s (N_1=8)
1	1.854	1.955	1.974	1.961	1.976	1.734
5	4.508	5.031	2.742	2.737	2.187	2.028
10	8.866	9.264	4.940	3.697	2.942	2.748
15	13.853	15.156	6.850	4.976	4.095	3.514
20	18.735	18.854	9.066	6.600	5.442	4.518
25	22.380	24.317	12.517	8.638	6.667	5.543
30	27.905	30.319	14.762	10.496	8.771	6.689

从表 3 可以看出，当云测试平台和工作站有相同内核数的情况下(N_1=N_2=2或者N_1=N_2=8)，在处理不同规模的测试任务时，两者所用的时间相差不多。这说明，云测试平台可以实现和配置相同的物理机相似的性能。当测试任务规模很小，比如 Num 小于 5 时，传统的方法中，使用工作站 1 就可以满足用户对测试时间的要求；这时在云测试平台中，相应地，只需要调用两个 CPU 内核(N_2=2)的资源。当测试任务高并发时(Num 由 1 增加到 30)，为了保证测试时间要求，传统的解决方式是需要启用规模更大的工作站，即要将任务从工作站 1 过渡到工作站 2，这意味着要更换硬件资源，配置新的测试环境(如系统重装，软件的安装、配置文件的修改等)，而这些工作势必会花费一定的时间和人力物力；而使用云测试平台，利用 OpenStack 的 Resize 功能可将调用的 CPU 内核数直接调节为 8，重启客户机后新的测试环境就能投入使用，省去了重新配置测试环境的步骤。从表中可以看出，根据测试任务规模，云测试可以按需分配，任意调用 CPU 内核资源(N_2=2,4,6…直到所有的 CPU 被调用)，以满足不同的任务需要，因而同传统的解决方式相比，具有明显优势，能够降低测试成本。

4　结论

本文设计了一个基于 OpenStack 的云测试平台。通过将软件测试工具 SAT 部署在 OpenStack 上，用户可以通过 Web 获得平台提供的软件代码覆盖率测试服务。所搭建的云测试平台，因为可以实现对硬件资源的按需分配，无须重新配置测试环境，因而同传统增加硬件资源的方式相比，在测试环境的部署上提高了效率，节省了测试成本。通过对比 SAT 在工作站和 OpenStack 云平台上完成相同数量测试任务所需的时间，证明云平台可以实现和配置相同的物理机类似的性能，能满足用户对测试时间的要求。

参考文献

[1] 徐晓峰，陈艳，李伊飏，等. 基于超级块支配图插装的软件测试工具设计与实现[J]. 计算机应用研究, 2010, 27(3): 923–927.

[2] LEAH RK, OSSI T, KARI S. Testing in the Cloud: Exploring the Practice. IEEE Software[J], 2012, 29(2): 46–51.

[3] Dickey Thomas E. ATAC[OL]. [2014-12-07]. http://invisible-island.net/atac/atac.html.

[4] Free Software Foundation. Gcov—a Test Coverage Program[OL]. [2014-12-07]. http://gcc.gnu.org/onlinedocs/gcc/Gcov.html.

[5] 李乔，柯栋梁，王小林. 云测试研究现状综述. 计算机应用研究[J], 2012(12): 4401–4406.

[6] 刘鹏. 云计算[M]. 北京：电子工业出版社, 2011.

[7] 王象刚. 基于云计算的软件测试[J]. 软件, 2013, 34(12): 246.

[8] LI A, YANG X W, KANDULA S, et al. Comparing Public-Cloud Providers. IEEE Internet Computing[J], 2011, 15(2): 50–53.

[9] 姜毅，王伟军，曹丽，等. 基于开源软件的私有云计算平台构建. 电信科学[J], 2013(01): 68–75.

[10] FAN FL, YAO CL. The system implementation of complex product data warehouse based on OpenStack cloud computing platform technology[A]. 2013 International Conference on Information and Communication Technology for education[C]. Singapore: IEEE, 2013. 121–128.

[11] WU D W, WU Q, FU X C. A new hybrid storage system base on OpenStack[A]. 2014 International Conference on Mechatronics Engineering and Computing Technology[C]. Shanghai, China: IEEE, 2014. 5371–5376.

[12] WU L M, WANG H B, CHENG S D.Research on Virtual Machine Launching Mechanism of OpenStack[J].The Journal of New Industrialization, 2012, 2(8): 28–32.

[13] 任永杰，单海涛. KVM虚拟化技术：实战与原理解析[M]. 北京：机械工业出版社, 2013.

[14] 丁养志. 浅析虚拟化技术在云计算中的运用[J]. 软件, 2014, 35(3): 176–177

[15] ELINDA KM, MEGI T. An evaluation of java code coverage testing tools: OpenStack and OpenNebula[A]. 2012 5th Balkan Conference in Informatics[C]. Novi Sad, Serbia: IEEE, 2012. 72–75.