基于容器技术的软件测试优化研究

刘钱超,董超群,张　垚
(江南计算技术研究所,江苏 无锡 214083)

摘　要:提高软件测试的缺陷检测能力,有效降低测试成本是软件测试优化研究中的关键问题。大数据、云计算时代的到来,伴随着软件功能的不断增强,软件规模和复杂度也呈爆发式增长,传统软件测试理论与方法暴露出许多亟待解决的问题。容器技术是一种基于 Linux 内核的虚拟化技术,能够达到接近物理主机的资源利用率,Docker 作为容器技术的典型代表,在软件封装和复杂系统搭建方面较传统虚拟机具有轻量、快速和高效等优势。文中首先分析了目前软件测试面临的问题,并基于 Docker 对软件测试过程中被测试软件的交付和安装部署方法、软件测试基础设施搭建方式、测试工具使用和软件可伸缩性测试方法等进行了优化。最后通过实验证明,基于 Docker 的软件测试优化方法可有效降低测试成本,提高软件测试的缺陷检测能力。

关键词:容器技术;Docker;虚拟化;软件测试;缺陷检测
中图分类号:TP31　　**文献标识码**:A　　**文章编号**:1673-629X(2019)04-0013-06
doi:10.3969/j.issn.1673-629X.2019.04.003

Research on Optimization of Software Testing Based on Container Technology

LIU Qian-chao, DONG Chao-qun, ZHANG Yao
(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

Abstract: Improving the defect detection of software testing and effectively reducing testing cost are the key issues in software testing optimization research. With the advent of the era of big data and cloud computing, with the continuous enhancement of software function, the scale and complexity of software also grows explosively. Traditional software testing theories and methods have exposed many problems to be solved urgently. The container technology is a virtualization technology based on Linux kernel, which can reach the resource utilization close to the physical host. As a typical representative of the container technology, Docker has advantages of lightweight, fast speed and high efficiency in software packaging and complex cluster system construction compared with traditional virtual machine. We analyze the problems in software testing at present, and optimize the method of delivery and installation of the software under test, the construction method of software testing infrastructure, the use of test tools and the test method of software extensibility based on Docker. Finally, it is proved by experiments that the software testing optimization method based on Docker can effectively reduce the testing cost and improve the defect detection of software testing.

Key words: container technology; Docker; virtualization; software testing; fault detecting

0　引　言

随着计算机科学技术的不断发展,软件在国民经济和社会生活等方面的应用越来越广泛和深入。应用的广泛性使得软件质量日益成为关注的焦点。软件测试作为保证软件质量和可靠性的关键手段,逐渐受到软件机构和开发人员的重视,软件测试阶段在整个软件开发周期中所占的比重也日益增大。大数据、云计算时代的到来,软件特征和表现形式都发生了深刻的变化,传统的软件测试理论与技术逐渐暴露出许多问题,制约了软件测试的发展,也使得软件测试成本不断增加。据统计,目前软件测试与维护成本占软件总开发成本的 40% ~50%[1],在关键领域的应用软件,软件测试成本甚至远远超过软件开发成本。在此情形下,研究如何利用新的理论、技术和方法来优化软件测试过程,从而提高测试效率和降低测试成本具有十分重要的意义。

收稿日期:2018-06-01　　**修回日期**:2018-10-10　　**网络出版时间**:2018-12-20
基金项目:国家自然科学基金(91430214);国家重点研发计划(2016YFB1000505)
作者简介:刘钱超(1994-),男,硕士研究生,研究方向为软件测试;董超群,博士,CCF 会员(30902M),研究方向为软件测试。

Docker 容器技术作为新一代的虚拟化技术,相比传统虚拟机技术,具有很多优势,如更高效的资源利用率、支持更轻量的资源分配、良好的软件封装性、更轻松的迁移和扩展等。文中主要讨论如何利用 Docker 容器技术具有的一些优势和特性对传统软件测试过程进行优化,以期解决目前软件测试中存在的问题并提高测试效率,降低测试成本。

1 问题分析及相关研究

1.1 软件测试面临的问题

随着大数据、云计算技术的不断发展,计算机软件的规模和复杂度不断增长,软件测试对象变得越来越复杂,导致传统软件测试理论与技术逐渐失去了适用性,极大地影响了软件测试工作。目前在软件测试工作中暴露出的问题主要表现在以下几个方面:

(1)平台软件的复杂性和发布节奏之间的矛盾。如今,软件迭代速度不断加快,软件更新频率已由原来以月、年为单位发展到现在的以天、周为单位。据统计,大数据生态系统组件平均更新速度为 2 周,如此短的软件迭代周期给软件的测试效率提出了更高的要求[2]。

(2)被测试软件的交付和安装部署繁琐。随着计算机所控制的对象的复杂性程度不断提高和软件功能的不断增强,软件规模与复杂度也在不断增大,软件依赖和包含的组件变得十分复杂。目前被测试软件的交付大多是基于源代码或软件安装包方式,需要测试人员手工搭建测试环境并配置各种所需的软件依赖,十分繁琐,还可能因软件环境和配置的差异导致被测软件出错,无法保持软件“开发—测试—生产”环境的一致性。

(3)软件测试基础设施搭建速度慢、耗时长、不灵活。目前,软件测试环境的构建多是基于虚拟机方式,在物理机集群上新建大量虚拟机,在虚拟机内部布设被测软件所依赖的各种环境。这种方式由于虚拟机是比较重载的虚拟化技术,对宿主机系统资源消耗大,导致资源利用率不高。另一方面,基于虚拟机的资源管理较为笨重,难以实现资源的细粒度分配和应用的快速迁移与扩展。

(4)分布式软件的可伸缩性测试方法具有局限性。目前,对分布式软件进行可伸缩性测试时,采取的方法是在物理机集群上新建大量虚拟机,在虚拟机内安装配置被测试软件,然后测试其功性能指标。这种测试方法的主要问题在于,一是扩展速度慢,基础设施搭建和软件安装配置需要大量时间,延误测试进度。二是容易因物理资源有限而无法开展大规模可伸缩性测试,达不到可伸缩性测试要求,如当被测试软件需要在物理机上扩展 10 000 以上节点时,这在有限的服务器硬件资源条件下是无法进行测试的。

1.2 相关研究

如何利用现有理论、方法和技术来优化软件测试工作一直是国内外软件机构的研究热点,近年来有许多相关方面的研究工作。

文献[3]中,Scott Tilley 和 Tauhida Parveen 在 CMU/SEI 的 SMART[4]基础上,提出了 SMART-T 模型。SMART 模型是一系列指导方针,旨在帮助软件企业和机构对面向服务的体系结构(SOA)环境下重用遗留组件作为服务的可行性做出初步决策。与 SMART 模型不同的是,SMART-T 更专注于软件测试“Test”。它进一步明确了 SMART 模型在软件测试迁移至云环境的应用方式,主要由 3 部分组成:业务驱动、技术因素、运行结果。每一部分结束时都会做出将软件测试迁移至云环境的可取性、可行性和可接受性相关的决策,给软件测试机构提供了更具体明确的指导方针。Scott Tilley 和 Tauhida Parveen 也基于 Hadoop 平台构建了云环境下测试用例并行执行的分布式环境 HadoopUnit,在该分布式环境下,测试用例能够并行执行,可以有效地加快测试用例执行速度。但上述优化方法的不足之处在于,对于是否将测试环境迁移至云环境下的决策主要由人来评判,缺乏足够的数据支撑,可能会出现判断失误。

文献[5]中,Joshua Higgins 等基于 Docker 构建了一个 VCC(virtual container cluster)集群。VCC 不仅是一个分布式的容器集群环境,还是一个自动化的打包工具。将 VCC 应用到软件测试中,有效地提高了被测试软件的可移植性和可重现性[3]。但存在的问题在于,VCC 的应用场景具有局限性,主要针对分布式软件,对单机软件并不适用。

2 容器技术及容器云

2.1 Docker 容器技术

传统虚拟化技术如 VMware、Xen、KVM、Microsoft Hyper-v 等由于需要模拟一个完整的操作系统层,导致硬件资源利用率不高,给大数据云计算等大规模集群应用带来了不小的资源开销。为了寻求更高效的虚拟化技术替代方案,dotCloud 公司于 2013 年将一些现有容器技术进行包装整合后推出了开源的容器引擎项目 Docker,并将项目代码托管在 GitHub 上。得益于赶上一场 IT 开发行业从 SOA 架构向微服务架构的技术变革,Docker 迅速取得了成功,得到学术界和工业界的广泛应用[6]。

Docker 作为软件技术层面可移植的轻量容器,是一种不受底层操作系统限制,即可在任何运行着

Docker 引擎的机器上运行的虚拟化解决方案[7]。Docker 容器技术相比于传统虚拟机技术,具有以下优势:

(1)Docker 可以将应用程序及其依赖环境打包成镜像文件,基于镜像文件可以在任何 Linux 系统中进行软件安装和部署,极大地提高了软件的可移植性和可重现性。

(2)Docker 是进程级的虚拟化技术,相比于传统虚拟机技术,由于没有了硬件模拟层开销,更轻量、快速和高效。在容器内运行应用程序时,与虚拟机技术不同,Docker 可以直接调用系统内核执行命令,不需要经过中间层翻译,因此资源的开销更小,启动速度更快。

(3)Docker 容器只需要数秒即可启动,比虚拟机需要数分钟启动更快,Docker 轻量和快速启动的特性使得较于重载的虚拟机而言,可以更轻松地进行迁移和扩展。

2.2　Docker 容器云

容器云是以容器为资源分割和调度的基本单位,封装整个软件运行时环境,为开发者和系统管理员提供用于构建、发布和运行分布式应用的平台[8]。容器云相较于虚拟机集群,凭借 Docker 轻量、快速、高效的特点,能够实现资源的细粒度分配和应用的快速迁移与扩展。

为了给容器化应用提供支撑环境和更好地优化软件测试基础设施搭建方式,文中基于 Docker、Zookeeper、Etcd 等技术搭建了面向软件测试的弹性容器的云平台(flexible container cloud for software testing, FCC-ST)。

FCC-ST 采用 B/S 模式,统一由浏览器认证并管理。它提供两种工作模式,分别是 Cluster Mode 和 Single Mode。其中,Cluster Mode 是由 Control Center 对集群进行监控和管理,Single Mode 是对单个物理节点进行监控和管理,包括对其上的容器进行新建、销毁和迁移操作,新加入的物理节点可以选择多种模式。FCC-ST 平台具有以下功能:

(1)物理节点和容器都可以快速灵活地进行弹性伸缩,只需在待加入的物理节点上运行相应的代理容器即可添加到 FCC-ST 弹性容器云中,容器新建、销毁和管理操作也都十分便捷。

(2)通过 Web 界面来新建、销毁和管理容器,十分方便,具有很强的易操作性。

(3)管理私有仓库,可以将自己部署好的私有仓库添加到 FCC-ST 平台中进行管理。

(4)集群容器调度,当需要部署一个镜像时,编写好配置文件,FCC-ST 会自动根据配置文件进行集群部署,并通过 WebHook 通知部署结果。

(5)对容器管理进行权限设定,分为普通用户和管理员用户,分别具有不同的权限。

3　基于 Docker 优化软件测试

凭借 Docker 具有的高效资源利用率、更轻量的资源分配、良好的软件封装性、更轻松的迁移和扩展等特性及 FCC-ST 平台,对图 1 所示的软件测试过程中被测试软件的交付和安装部署方法、软件测试基础设施搭建方式、测试工具使用和软件可伸缩性测试方法等进行优化,以解决目前软件测试面临的问题并提高测试效率。

(1)优化被测试软件的交付和安装部署。

被测试软件的交付和部署一直是困扰软件开发和测试人员的难题,一是软件安装配置本身比较复杂,耗时费力,二是难以保持软件开发环境和测试环境的一致性,经常会出现软件各项功能在开发环境中表现正常,在测试环境中却频繁报错的情况。针对这个问题,部分测试机构采取使用虚拟机打包被测试软件的做法,但因虚拟机体量过大,不利于软件的便捷交付。

Docker 正好可以比较好地解决这个问题。Docker 具有良好的软件封装性,能够快速、准确、标准化地封装应用程序并自动化部署整个运行环境。利用 Docker 将被测试软件及其依赖的各种运行时环境一起打包成容器镜像并交付给测试机构。测试机构只需要在任何支持容器运行环境下,就可以快速将被测软件安装运行起来,避免了复杂的软件安装和环境配置工作。另一方面,被测试软件容器化封装成容器镜像后,可以极大提高软件的可移植性和可重现性,在支持容器运行的环境下,基于容器镜像即可复现与软件开发环境一致的测试环境,有效地保持软件“开发-测试-生产”环境的一致性[9],避免因软件配置和环境差异引发的软件错误。图 2 为基于 Docker 的软件交付与部署示意图。

(2)优化软件测试基础设施搭建方式。

容器化后的被测试软件在支持容器运行环境下即可部署运行,基于搭建的 FCC-ST 平台能够十分灵活迅速地搭建软件测试所需的基础设施环境。FCC-ST 支持物理节点的动态加入和删除操作,在 FCC-ST 的 Web 管理界面即可完成所有环境搭建过程,较于创建/销毁虚拟机方式搭建集群环境更高效,物理资源利用率更高。FCC-ST 同时支持容器化被测试软件的一键部署,在软件部署的配置文件中写好配置信息,FCC-ST 就可以根据设置好的配置信息将被测试软件自动部署到预设节点上,不需要过多的人工干预,十分的高效。

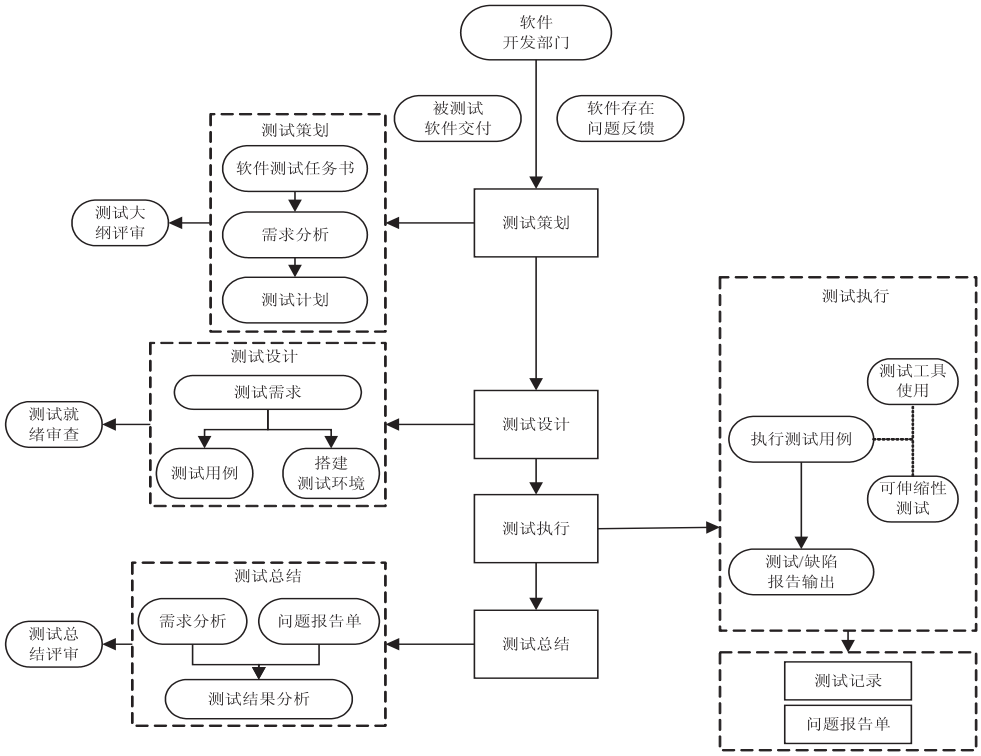

图 1 软件测试流程

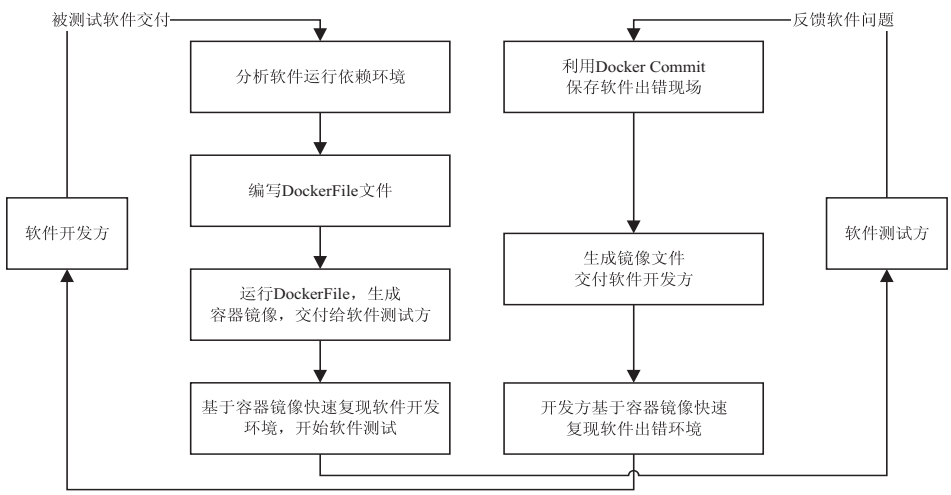

图 2 基于 Docker 的软件交付与部署示意

(3)优化测试工具使用方式。

软件测试工具大多都需要复杂的安装和配置才可使用,如 Junit、Jmeter 等[10],对刚接触的新手并不友好。利用 Docker 将测试工具及其依赖环境容器化封装成容器镜像,存储到 FCC-ST 中的 Private Registry 中,测试人员在需要使用测试工具时,直接在 Private Registry 中拉取相应容器镜像并新建容器即可使用,省去了复杂的安装和配置过程,“一劳永逸”。将测试工具容器化,能够帮助测试人员高效地使用软件测试工具,即用即取,有效提高测试效率。

(4)优化软件可伸缩性测试方法。

可伸缩性[11-12]是分布式软件的一个重要特性,可伸缩性测试的目的是测试软件在扩展到很多物理节点上运行时,其功能指标是否正常。目前的测试方法都是基于虚拟机,由于虚拟机是一种十分笨重的虚拟化技术[13],在有限的物理资源条件下无法开展大规模节点的可伸缩性测试,因而极大地降低了软件测试在应用大规模部署场景下的缺陷检测能力。

凭借 Docker 容器技术轻量、隔离和快速部署的特点,提出基于容器节点扩展的可伸缩性测试方法,将被测试软件及其依赖环境容器化封装为容器镜像,并部署在 FCC-ST 平台上,直接基于容器镜像批量新建大量容器来完成软件节点的扩展。这种方式凭借容器技术轻量、快速的特点,可以在有限物理资源条件下开展大规模的可伸缩性测试并节约大量测试时间,有效地降低软件测试成本,提高了软件测试的缺陷检测能力。

图 3 为软件可伸缩性测试优化示意图。

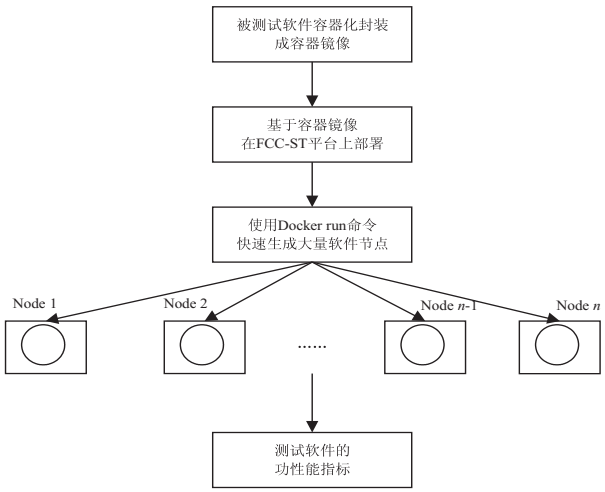

图 3　软件可伸缩性测试优化示意

4　实验分析

4.1　实验环境

实验基于以下实验环境:物理机型号为 Mac Pro,操作系统为 macOS Sierra 10.12.1,处理器为 3.5 GHz 6-Core Intel Xeon E5,显卡为 AMD FirePro D300 2048 MB,内存为 32 GB 1866 MHz DDR3,虚拟系统软件为 VirtualBox 5.1.30,虚拟机系统版本为 Ubuntu 16.04。

4.2　实验过程及结果分析

实验 1 对文中搭建的 FCC-ST 平台的性能进行测试,对比物理机裸机和虚拟机,使用 LinpackMPI 测试工具测试其性能损耗。实验 2 为基于大数据基础软件 Hadoop 节点扩展测试实验,对基于容器节点扩展的软件可伸缩性测试方法和基于虚拟机的可伸缩性测试方法进行对比测试,验证优化方法的有效性。

实验 1:FCC-ST、Native 和 VM 计算性能对比测试。

FCC-ST 平台和 VM 都是搭建在裸机(Native)之上的,使用 LinpackMPI 测试工具对 FCC-ST 平台、Native 和 VM 环境下的 CPU 浮点计算能力进行测试,并在实验过程中将 VM 和容器的资源使用限制解除。该实验参考了文献[14]中的测试方法和步骤。

测试结果如图 4 所示。

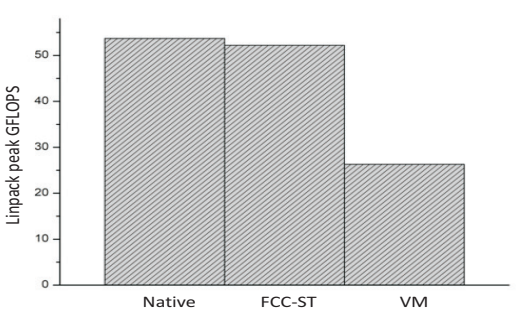

图 4　性能测试对比

由实验结果可知,FCC-ST 容器云平台的计算性能几乎与物理裸机相当,只有很少的计算性能损耗,而虚拟机计算性能损耗甚至超过 50%。分析原因是虚拟机本身需要模拟整个操作系统层,开销很大,而 FCC-ST 是基于容器构建,是操作系统级的虚拟化,没有硬件模拟层消耗,因此几乎没有计算性能损耗,与裸机计算性能相差无几。

实验 2:大数据系统基础软件 Hadoop 的软件节点扩展测试。

对大数据系统基础软件 Hadoop 进行伸缩性测试,基于下述两种方式完成大数据系统基础软件 Hadoop 的软件节点扩展。

1. 基于虚拟机方式。

(1)在虚拟机内部安装和配置 Hadoop 软件;

(2)在物理机上克隆已经安装配置好 Hadoop 软件的虚拟机;

(3)修改虚拟机内部的配置文件,包括 IP 地址和 Host 文件等;

(4)完成 Hadoop 节点扩展。

2. 基于容器节点扩展方式。

(1)利用 Docker 的镜像构建命令,将 Hadoop 及其依赖的运行时环境封装成容器镜像;

(2)基于容器镜像在 FCC-ST 平台上部署 Hadoop;

(3)新建 Hadoop Master 容器节点和 Slave 容器节点,shell 程序代码如下所示;

```
# N is the node number of hadoop cluster
#start hadoop mater container
sudo docker run -itd \
--net=hadoop \
-p 40058:40058 \
-p 8088:8088 \
--name hadoop-master \
--hostname hadoop-master \
scaleTest/hadoop:1.0 &> /dev/null
#start hadoop slave container
i=1
while[ $i -lt $N ]
do
echo "start hadoop-slave $i container"
sudo docker run -itd \
--net=hadoop \
-p 40058:40058 \
-p 8088:8088 \
--name hadoop-slave $i \
--hostname hadoop-slave $i \
scaleTest/hadoop:1.0 &> /dev/null
i=$((  $i+1))
```

done

(4)完成 Hadoop 节点扩展。

表 1 大数据系统基础软件 Hadoop 节点扩展测试结果

扩展节点个数	耗时/s	
	虚拟机方式	容器节点扩展方式
3	51	1
5	153	2
10	408	4
20	918	9
40	—	23
60	—	36
80	—	51
100	—	71
200	—	152
500	—	392
1 000	—	806

注:“—”表示在文中实验环境下无法进行测试

表 1 为实验测试结果,虚拟机由于物理资源消耗大,是比较重载的虚拟化技术,在文中实验环境下,只能支持扩展 20 个节点。而基于 FCC-ST 平台,凭借 Docker 容器技术的轻量、高效特点,Hadoop 可以扩展数千甚至上万节点[15]。图 5 为两种扩展方式耗时对比。可以看出,基于 FCC-ST 平台的扩展方式优势十分明显,不仅能在有限物理资源条件下开展比虚拟机方式更大规模的可伸缩性测试,而且在时间消耗上最少只需要虚拟机方式的百分之一,十分高效。

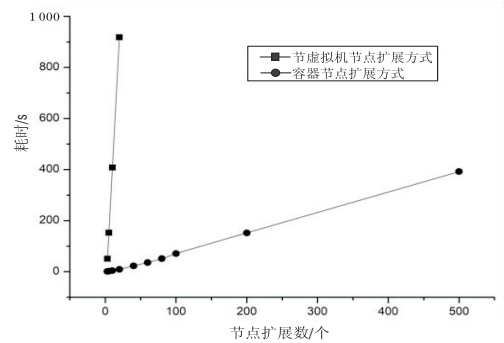

图 5 两种扩展方式耗时对比

4.3 实验结论

由实验 1 可知,基于 Docker 搭建的 FCC-ST 平台相比于虚拟机,更轻量、灵活,开销更小,资源利用率更高。FCC-ST 容器云平台兼具了虚拟机的隔离性优势和几乎物理机裸机的计算性能。实验 2 通过两种扩展方式的对比测试,基于容器节点扩展的软件可伸缩性测试方法克服了基于虚拟机扩展方式的笨重和资源开销大等缺点,在时间和资源利用率方面优化效果十分显著,验证了文中提出的基于容器节点扩展的可伸缩性测试方法的有效性和优越性。

5 结束语

分析了软件测试工作目前面临的问题,给出了相关研究,并基于 Docker 容器技术和 FCC-ST 平台对软件测试过程中的被测试软件的交付和安装部署方法、软件测试基础设施搭建方式、测试工具使用和软件可伸缩性测试方法进行了优化,并通过实验验证了优化方法的有效性。该研究工作具有一定的实用价值,能够在一定程度上提高软件测试效率和软件测试的缺陷检测能力,同时也可以给软件测试优化工作提供一定的启发和参考。基于容器技术设计一套完善的容器化软件测评工作流是下一步的研究方向。

参考文献:

[1] PATTON R. Software testing[M]. [s. l.]:Sams Publishing, 2006.

[2] 顾 荣. 大数据处理技术与系统研究[D]. 南京:南京大学,2016.

[3] TILLEY S,PARVEEN T. 云环境下的软件测试:迁移与执行[M]. 北京:科学技术文献出版社,2015.

[4] LEWIS G,MORRIS E,SMITH D. Analyzing the reuse potential of migrating legacy components to a service-oriented architecture[C]//European conference on software maintenance and reengineering. Bari,Italy:IEEE,2006:15-23.

[5] HIGGINS J,HOLMES V,VENTERS C. Autonomous discovery and management in virtual container clusters[J]. The Computer Journal,2016,60(2):240-252.

[6] 鲁 涛,陈 杰,史 军. Docker 安全性研究[J]. 计算机技术与发展,2018,28(6):115-120.

[7] 华为 Docker 实践小组. Docker 进阶与实战[M]. 北京:机械工业出版社,2016:313-354.

[8] 张 涛. Docker 全攻略[M]. 北京:电子工业出版社,2016.

[9] 王亚玲,李春阳,崔 蔚,等. 基于 Docker 的 PaaS 平台建设[J]. 计算机系统应用,2016,25(3):72-77.

[10] 王雅文,宫云战,杨朝红. 软件测试工具综述[J]. 北京化工大学学报:自然科学版,2007,34:1-4.

[11] 李 乔,柯栋梁,王小林. 云测试研究现状综述[J]. 计算机应用研究,2012,29(12):4401-4406.

[12] 杨靖琦. 云化业务平台可伸缩性研究[D]. 北京:北京邮电大学,2014.

[13] 汪 恺,张功萱,周秀敏. 基于容器虚拟化技术研究[J]. 计算机技术与发展,2015,25(8):138-141.

[14] FELTER W,FERREIRA A,RAJAMONY R,et al. An updated performance comparison of virtual machines and Linux containers[C]//IEEE international symposium on performance analysis of systems and software. Philadelphia, PA, USA:IEEE,2015:171-172.

[15] 杨 鹏,马志程,彭 博,等. 集成 Docker 容器的 OpenStack 云平台性能研究[J]. 计算机工程,2017,43(8):26-31.