***文献主题列表：***

虚拟机（Virtual Machine）和容器技术（Container Technology）

分布式系统可靠性技术（Reliability Techniques in Distributed Systems）

容错技术（Fault-Tolerance Techniques）

内存故障隔离方法（Memory Fault Isolation Methods）

系统性能调优技术（System Performance Tuning Techniques）

***A Comparative Study of Containers and Virtual Machines in Big Data Environment***

两种技术的目的：获得更高的硬件资源利用率，保持不同计算实例之间的隔离

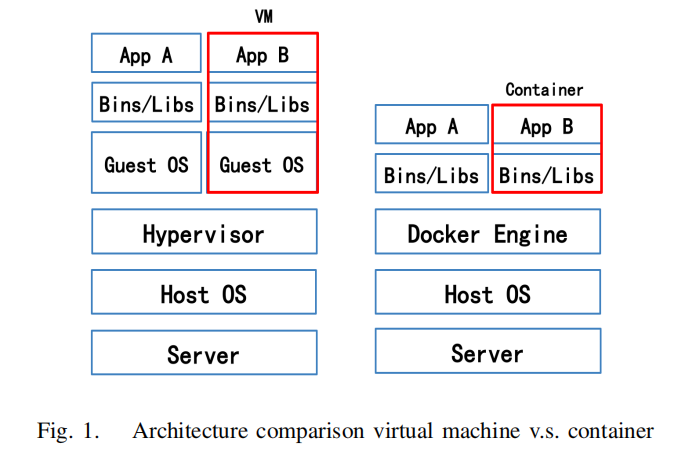
基于虚拟机的云面临的挑战：资源共享，资源转移，软件依赖

容器技术：便利性（部署，共享），多功能性

本文提出前人的分析不足，本文注重在云环境下的可拓展性与性能分析

研究目的：（1）管理人员的便利性 （2）性能与可拓展性 （3）分析性能不同的原因

体系结构的对比：



比较的差异：轻量级，hypervisor，容器可以使用同一个映像创建

Claus[26]比较了容器和虚拟机的虚拟化原理

结论是容部署更方便，性能更好

***Revisiting the History of Virtual Machines and Containers***

**虚拟机的技术发展路线：**

1960s M44/44X：虚拟内存的技术

CP-40/CMS and later CP-67/CMS ：可移植性方法（裸机编写程序可以在虚拟机中运行）

PC的出现：短暂地降低了人们对虚拟机技术的兴趣

**容器的技术发展路线：**

CAL-TSS：目标是一个不依赖于硬件的OS

Plessey System 250：实现了内存保护与进程隔离

现代虚拟机发展：

Disco研究项目：突出可拓展性

VMware：VMware将trapand-execute technique与动态二进制转换技术结合起来

Denali：半虚拟化技术

x86 Hardware virtualization extensions：QEMU+KVM的组合提供了一个完整的虚拟机实现

Hyper-V：Hyper-V granted special privileges to one guest, called the “parent partition”, which hosted the virtual devices and handled requests from the other guests.

现代容器技术：

POSIX capabilities === posix标准中的一个容器的想法

Namespaces and resource controls：

FreeBSD增加了Jails隔离了文件系统名称空间，隔离了进程和网络资源

Solaris添加了zone(有时也称为Solaris Containers)，将进程隔离到只能观察或通知同一 组中的其他进程的组中，将每个区域与隔离的文件系统名称空间相关联，并为共享资源 消耗设置限制(最初仅限CPU)。

SubDomain：它为与执行进程相关的有限系统调用增加了访问控制检查

Linux安全模块(LSM)框架：它提供了一组钩子，其他安全扩展(如SELinux或AppArmor)可 以在其中插入访问控制检查

Seccomp：在2012年，Will Drewr扩展了seccomp，允许使用Berkeley Packet Filter (BPF) 规 则动态定义系统调用的过滤器，这提供了足够的灵活性，使seccomp成为一种有用 的隔离技术。

Cluster management：

Google部署Borg --->>> Kubernetes项目

文章结论：这两者从一开始就在一起发展。它还表明，这两个技术家族在为现代多租户基础设施提供安全隔离方面都面临着重大挑战。

***Containerisation for High Performance Computing Systems：***

***1 INTRODUCTION***

提出的观点：容器化的应用程序会变得复杂

文章贡献：

1. 调查先进工程在高性能计算系统的容器化
2. 介绍了HPC和Cloud环境中具有代表性的框架

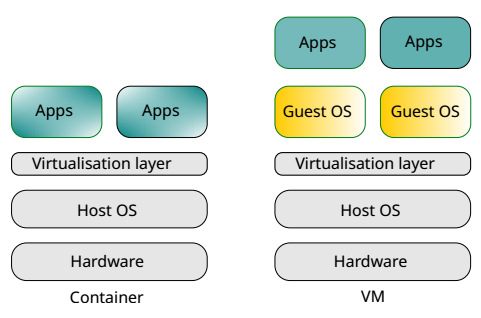
***2 CONCEPTS AND TECHNOLOGIES FOR CONTAINERISATION***

容器化技术在云和高性能计算系统上的主要区别在于安全性和工作负载类型

HPC应用程序往往需要更多的资源 ===>>> HPC社区开发了复杂的工作负载管理器

容器化技术和虚拟机技术：都是虚拟化技术差别在于------容器是映像的可运行实例，它将程序及其库、数据、配置文件等封装在一个孤立的环境中-------传统的VM将整个来宾内核(模拟操作系统)加载到内存中

技术结构示意图



容器的共享内核策略有一个明显的缺陷:Windows容器化的应用程序不能在Unix内核上执行

Linux下用户名称空间允许将用户ID (UID)和组ID (GID)从主机映射到容器，这意味着在容器内具有UID 0(根)的用户可以映射到容器外的非根ID(例如100000)。

一种流行的容器实现方案：Docker

Docker早期需要用户使用root权限，但是会导致安全问题

Docker发展之后演变为无root权限的守护进程，但是仍然面对安全问题

***3 CONTAINER ENGINES AND RUNTIMES FOR HPC SYSTEMS***

一些为高性能计算设计的架构：

Shifter ： NERSC开发的用于高性能计算的容器引擎的原型实现

支持MPICH + 不影响容器可移植性的情况下启用加速器支持

Charliecloud：

charlicloud运行没有特权操作或守护进程的容器。

Charlicloud可以将Docker 镜像转换为tar文件，并在HPC节点上解包。

解决了MPI的依赖项问题

Singularity：

优点说明（与docker相比）：

Running with user privileges

与高性能计算系统兼容性良好（支持需要的各种api）

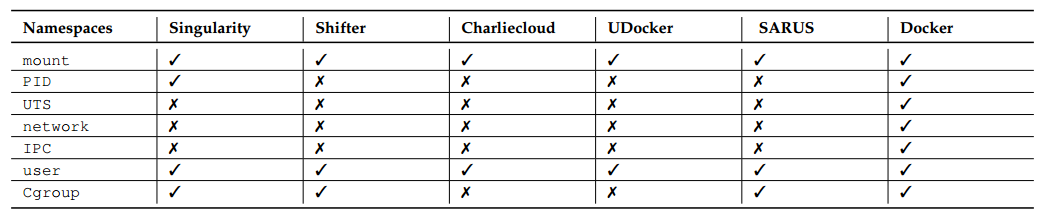
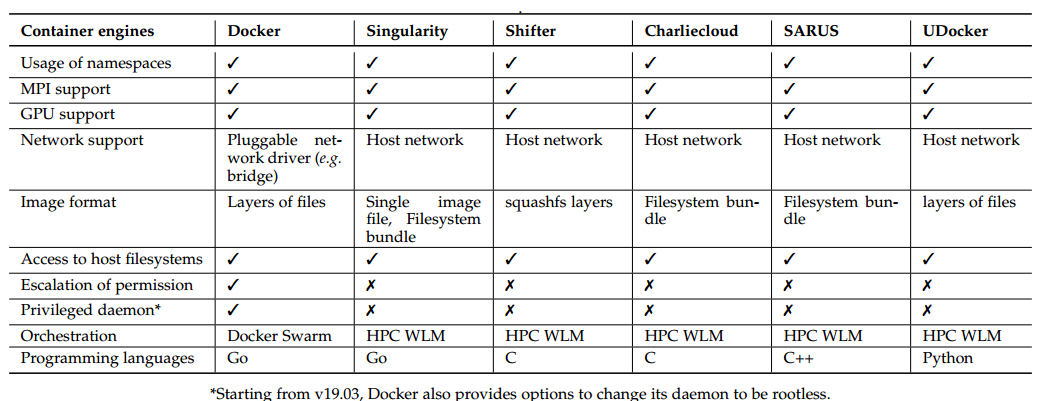
SARUS：

SARUS依赖runc1来实例化容器

UDocker：

一个简单的py打包器

两个表格指示不同的容器引擎的支持

注：容器化基准测试已经将所有模块包含在其映像中，映像作为单个文件挂载到每个节点上

本节描述的容器引擎的共同的特性：

•非根权限；

•通常可以将Docker映像转换为自己的映像格式；

•支持典型的高性能计算应用程序的MPI；

•使用主机网络而不是可插拔网络驱动程序

***4 CONTAINER ORCHESTRATION***

编排是指云计算或高性能计算系统的自动化配置、协调和管理。

HPC系统的工作负载管理器:

典型的HPC系统由工作负载管理器管理。

工作负载管理器包括 : 资源管理器和作业调度器。

PBS代表便携式批处理系统: OpenPBS，PBS Pro和TORQUE

4.1节就是说明了3个工作负载管理器的结构

容器编排器提供的功能：

•资源限制控制

•调度，确定优化节点上容器放置的策略

•负载均衡

•健康检查

•容错

•自动伸缩

下面介绍了几种容器编排器的结构：

Kubernetes：架构包括一个主节点和一组工作节点。Kubernetes在pod中运行容器， 这些pod计划在主节点或工作节点上运行。一个pod可以包含一个或多个容器。 Kubernetes通过部署提供服务，部署是通过提交yaml文件创建的。在yaml文件中，用 户可以指定要在集群上执行的服务和计算。

Docker Swarm:一个轻量级的编排器可，以应用于在HPC系统上

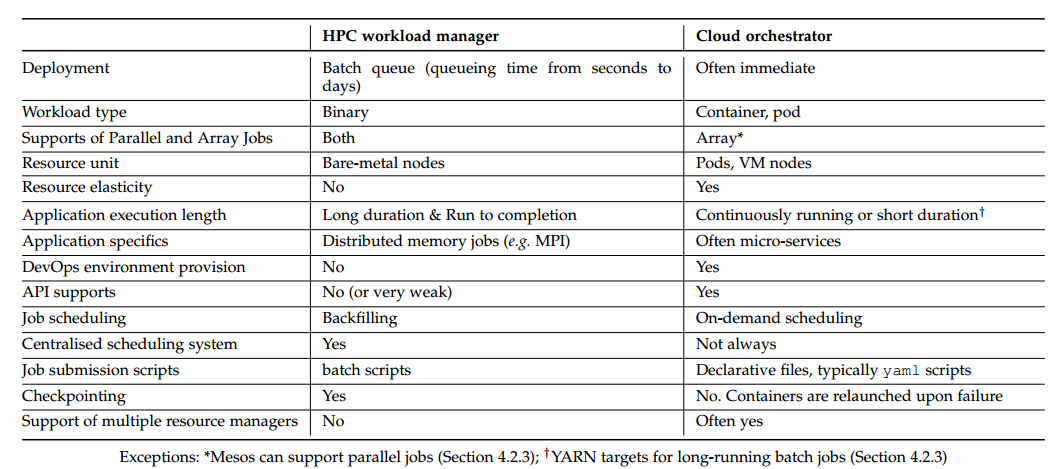
Mesos是一个非单片调度器，它充当一个仲裁者，在多个调度器之间分配资源，解决冲 突，并确保资源的公平分配。

Apache YAR：是一个单片调度器，最初用于调度Hadoop作业。

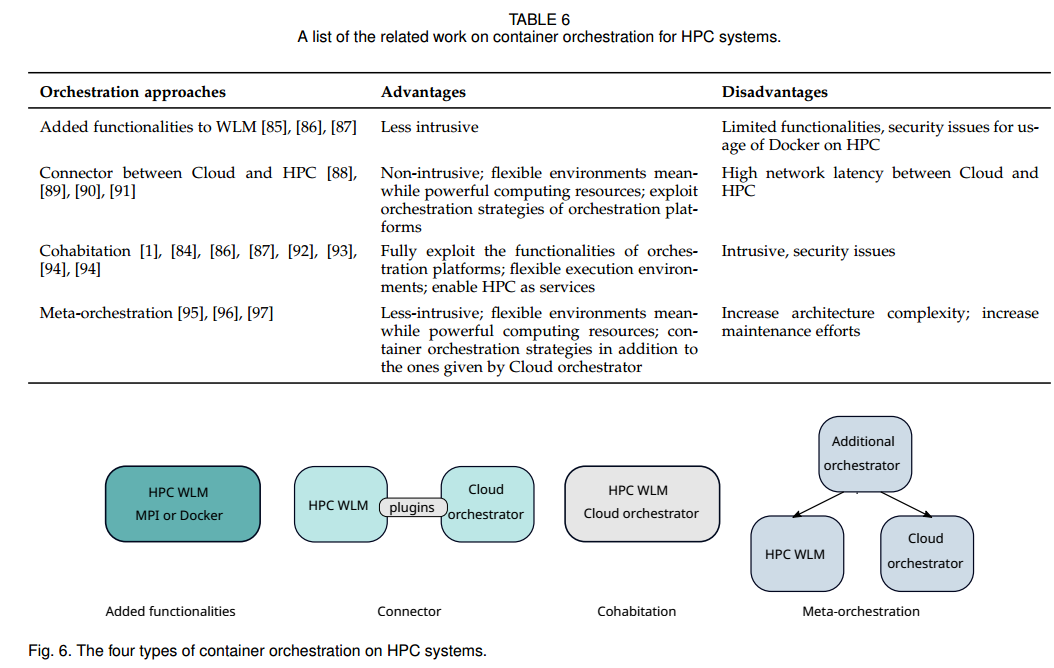
Ansible：它处理配置管理、应用程序部署、云供应、临时任务执行、网络自动化和多 节点编排。它不需要在节点上运行特殊的服务器或守护进程

OpenStack：以IaaS的形式部署在Cloud上。它可以用来部署和管理基于云的基础设施， 它为云解决方案提供了一个可扩展和高度自适应的开源架构，并有助于利用硬件资源。

这幅图展示的是传统HPC的负载管理器与云编排器的差别



下面这幅图说明了高性能计算系统容器编排的最新四类工作



1）为HPC工作负载管理器添加功能

2) 云端与高性能计算的连接器

3）共存

4）Meta-orchestration。在二者之上实现了一个额外的编排器

作者认为：容器编排器本身并不能满足HPC系统的所有需求，因此不能取代HPC中心中现有的工作负载管理器。

***5 RESEARCH CHALLENGES AND VISION***

容器化在高性能计算系统上应用面临着三方面的挑战：兼容性、安全性和性能

下面从具体议题说明：

1. 库兼容性问题

将容器库及其依赖关系映射到主机库可能导致不兼容（处理容器镜像和主机之间库不匹配的一个研究方向是在底层实现容器运行时库。）

（2）容器引擎和镜像的兼容性问题

不是所有的Docker镜像都可以被HPC容器引擎转换成自己的格式（这个问题需要容器标准化）

（3）内核优化

出于应用隔离的目的，主机禁止容器安装自己的内核模块。对于需要内核定制的应用程序来说，这是一个限制，因为它们的HPC主机的内核无法调优和优化

（4）安全问题

容器面临三大威胁：

•特权升级

•拒绝服务（DoS）

•信息泄露

1. 性能下降

作者提出的有趣的研究方向：

（1）HPC中AI的容器化

（2）HPC容器注册表构建

（3）DevOps

（4）Linux命名空间指南

（5）中间件系统

（6）资源弹性相关的工作

（7）向最小操作系统发展

