分数：	
评卷人：	

研究生(数据中心技术)课程论文(报告)

学号　　M202073595

姓名　　陈新宇

专业　　电子信息

课程指导教师　　施展、童薇

院（系、所）　计算机科学与技术

2020 年 12 月 25 日

更加高效、可靠的云端基础设施

硕 2011 班 M202073595 陈新宇

摘 要 随着数据增多，计算量增大，移动互联网的普及让人们需要随时接入服务，因而需要降低终端压力，云计算应运而生。云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池，这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。云计算依赖于数据中心中的存储、网络、虚拟化等等多种基础设施或技术。云计算是基于共享基础架构，软件、硬件、平台等 IT 资源将以服务的形式提供给使用者。数据中心最初的目标是对资源的管理，管理的主要是计算资源、网络资源、存储资源三个方面。数据中心是通过互联网提供服务，因而高效安全的网络是确保云计算服务进行的基础，除此之外，就云主机内部而言，也需要网络保证最基本的可迁移性和隔离性。本文重点围绕关于云端基础设施的 5 篇论文展开综述，分别包括了云存储、虚拟化与数据中心网络、云计算的一致性与正确性三大主题。

关键词 云计算；SoCC；云端基础设施；云存储；虚拟化与数据中心网络；云计算

Bigger cloud,faster calculation

ChenXinyu

Abstract With the increase of data and the amount of calculation, the popularization of mobile Internet makes people need to access services at any time, so the pressure on the terminal needs to be reduced, and cloud computing came into being. Cloud computing is a pay-as-you-use model. This model provides usable, convenient, and on-demand network access, and enters a configurable computing resource sharing pool. These resources can be provided quickly with little investment Management work, or very little interaction with service providers. Cloud computing relies on multiple infrastructure or technologies such as storage, network, virtualization, and so on in the data center. Cloud computing is based on a shared infrastructure, and IT resources such as software, hardware, and platforms will be provided to users in the form of services. The original goal of the data center is to manage resources, and the main management is computing resources, network resources, and storage resources. Data centers provide services through the Internet, so an efficient and secure network is the basis for ensuring cloud computing services. In addition, as far as the cloud host is concerned, the network also needs to ensure the most basic transferability and isolation. This article focuses on an overview of five papers on cloud infrastructure, including three themes of cloud storage, virtualization and data center networks, and cloud computing consistency and correctness.

Keywords Cloud Computing; SoCC; Cloud Infrastructure; Cloud Storage; Virtualization and Data Center Network; Cloud Computing

收稿日期：2020年12月28日

1 引言

ACM Symposium on Cloud Computing（以下简称SoCC）是由美国计算机协会主办、聚焦云计算技术的一项学术会议。SoCC由SIGMOD（数据管理特别兴趣组）和SIGOPS（操作系统特别兴趣组）共同赞助，旨在聚集数据库和计算机系统两大领域的学者，共同推进云计算技术的研究与发展。这个年轻的会议在近些年蓬勃发展，云计算领域著名的集群管理工具Apache Hadoop YARN、数据库研究中常用的YCSB基准测试等优秀工作均发布在此会议上。本届SoCC会议从160余篇投稿中录用了论文39篇，这些论文涉及与云计算技术相关的方方面面，包括云基础设施、云存储、计算引擎、资源管理与调度、分布式一致性等等。每年的主旨报告都会总结云计算尖端研究中积累的宝贵经验，分析和展望云计算技术当前和未来发展的趋势。来自谷歌的John Wilkes和来自加州大学欧文分校的MichaelJ.Carey是2018年的报告者，他们的报告都关于“大”：更大的集群，和更大的数据。

随着信息技术的不断升级，业务种类的增多，各领域需要处理的数据呈现出数据量大(Volume)、数据生产速度快(Velocity)、多样性(Variety)及真实性(Veracity)“4V”特点。一般地，具备“4V”特征的数据集都可认为是大数据。而随着大数据时代的到来，面对海量的数据信息时，传统的数据分析技术无法完全挖掘其中蕴含的信息。基于此，云计算概念应运而生，另外，研究人员一直在探索的人工智能技术也随着云计算相关技术的应用而快速发展。因此，这些技术或研究领域相辅相成，共同构建了当前蓬勃发展的新一代的信息技术。

关于云计算的概念纷繁多杂，本文首先区分云计算、云平台、云服务等概念，具体解释如下：

(1)云计算是指采用虚拟化等技术构建的数据中心，然后按使用量付费的模式，为用户提供各种服务形式的应用。

(2)云服务是指利用云计算技术，能够根据用户需求，提供多元化的服务，该服务是可扩展、可伸缩的。

(3)云平台将虚拟化的计算资源、存储资源、网络资源统一管理，并面向用户提供服务，形成了云服务。

因此，这几个概念类而不同，又相辅相成，而目前大部分的技术研发、研究机构或公司均是围绕这三个方面进行的。

对于云计算的概念，目前还存在多种定义，一般而言，云计算是一种按使用量付费的模式，该模式提供可用的、便捷的、按需的网络访问，进入可配置的共享资源池，例如计算、网络、存储等资源，而用户只关心云平台所提供的服务。因此，云计算既指一种可以根据需要动态地提供配置以及取得供应的计算和存储平台，又指一种可以通过互联网进行服务的应用类型。并且用户的系统规模变化时，云计算系统能够根据用户的需求自由伸缩。

数据中心是云计算环境下分布存储的基础，云计算环境下的分布存储研究数据在数据中心上的组织和管理，为了提高可靠性和可用性，一般需要为一个数据对象创建若干个副本，或者以编码的形式提供一些冗余数据。数据对象及其副本或者冗余数据往往分布在数据中心不同的节点上，因此，数据的存取效率及可靠性与数据中心中节点的结构紧密相关。数据中心网络(data center network，简称DCN)主要研究构成数据中心中的各个节点之间的物理连接结构，即如何组织和连接数据中心中的各个服务器节点以及连接设备等，从而更加方便地为上层的各种应用和服务提供良好的接口。

依据数据中心中担任数据包的路由转发功能的节点类型，可以把数据中心网络分为 3 种：以交换机为中心的结构(switch-centric)、 以服务器为中心的结构(server-centric)以及混合结构(hybrid)。以交换机为中心的结构是指数据中心上的各个服务器通过交换机连接到一起，数据包的路由转发功能由交换机完成，服务器不担任数据路由转发的功能，只负责数据的存储和处理；以服务器为中心的结构是指通过为每个服务器安装多个网卡，然后通过网线把这些服务器直接连接到一起的结构。在这种结构中，没有交换机等数据转发设备，服务器不但负责数据的存储和处理，还要负责数据包的转发，担任交换机的角色；混合结构是以交换机为中心的结构和以服务器为中心的结构的一种混合，其中不但有交换机，也有部分服务器担任数据的路由转发功能。

2 研究现状

2006 年谷歌推出“Google 101 计划”，并提出了云计算的的概念，其提出的基础是随着数据资源的快速增加，需要相应的技术手段筛选或分析其中

有意义的信息。本文将云计算演化过程进一步细化为如下部分：

(1)资源汇集阶段：将分散的数据资源逐步实现集中化和标准化，并形成规模化的数据中心。 这些中心具有一定的基础设施，为后续数据的有效利用打下基础。

(2)资源应用阶段：为了降低运行成本、缩短业务执行周期、提高资源利用的灵活性，开始出现了物理资源虚拟化的应用；其本质是利用虚拟化技术屏蔽底层物理资源的异构性，进而实现资源利用率的提升和灵活快速的部署业务。

(3)资源共享协作阶段：由于数据中心的各种系统的初始投入较大，软硬件技术则将面临不断升级的压力，从而造成困境，即基于虚拟化的 IT 架构难以解决不断增加的业务对资源变化的需求。因此，云计算架构开始展现优势，其能够满足业务弹性拓展、按需服务等需求，最终形成一系列的云架构的 IT 服务。

产业界和学术界从 2011 年就开始了对后云计算时代网络计算模式的思考和探索，并且取得了初步的成果。雾计算(Fog / Mist Computing)[1]、移动边缘计算(Mobile Edge Computing，MEC)[2]、边缘计算(Edge Computing)[3]等多种新型网络计算模式被相继提出并开始得到初步研究和应用。当前的几种新型网络计算模式，虽然被提出时所考虑的技术和应用的出发点有些差异，但是其基本思想和核心理念是一致的：即都是试图将云计算中心的设备部署在物理或逻辑上距离终端和用户较近的基础设施上，从而利用这些较近的基础设施所拥有的计算和存储等资源来完成终端和用户想要完成的计算、存储和网络等各类任务。因为这些附近的基础设施一般具有更低的延迟，所以任务的响应时间可以很大程度上被减少，相应的也就可以提高用户的体验。

最初的单个处理机模式处理能力有限，并且请求需要等待，效率低下。后来，随着网络技术的不断发展，按照高负载配置的服务器集群，在遇到低负载的时候，会有资源的浪费和闲置，导致用户的运行维护成本提高。而云计算把网络上的服务资源虚拟化，整个服务资源的调度、管理、维护等工作由专门的人员负责，用户不必关心“云”内部的实现，因此云计算实质上是给用户提供像传统的电力、水、煤气一样的按需计算服务，它是一种新的有效的计算使用范式。并且，云计算是分布式计算、效用计算、虚拟化技术、web 服务、网格计算等技术的融合和发展，其目标是用户通过网络能够在任何时间、任何地点最大限度地使用虚拟资源池，处理大规模计算问题。目前，在学术界和工业界共同推动之下，云计算及其应用呈现迅速增长的趋势，各大云计算厂商如 Amazon，IBM，Google，Mi—crosoft，Sun 等公司都推出自己研发的云计算服务平台。而学术界也源于云计算的现实背景纷纷对模型、应用、成本、仿真、性能优化、测试等诸多问题进行了深入研究，提出了各自的理论方法和技术成果，极大地推动了云计算继续向前发展。文献[5]从几种典型的云计算实现方案角度，综述了云计算背后所采用的技术；文献[6]提出了云计算服务栈框架，综述归类了不同的云计算服务对应的服务层次。

无论是产业界还是学术界提出的近端云计算模式，它们能否得到广泛的接受和发展，取决于诸多因素的综合作用。虽然目前很难预测哪一种近端云计算模式将占据主导地位，但是可以肯定的一点是：近端云计算在不远的将来会得到大力的发展。然而，近端云计算的发展在面临许多机遇的同时，也面临着诸多技术挑战。

当前数据中心面临的挑战包括以下点：（1）传统网络结构是基于电分组交换的，一般是由接入层、汇聚层、核心层三层交换机组成的树型结构。随着网络业务更加多样以及网络规模的扩大，流量呈几何倍数增长且由南北向流量为主变为东西向流量为主，因此我们需要更高的可用带宽用以支持数据中心网络内部的大量数据及时、精准的传输。（2）高性能交换机十分昂贵，为了应对服务器规模的不断增加，数据中心往往需要不断升级替代已有硬件交换设备，然而这种扩展价格昂贵效果却不高。（3）传统树型结构存在单点失效的问题，核心层交换机一旦故障，很容易造成链路拥塞，为了避免单点失效往往需要准备冗余链路，这就进一步增加了数据中心的成本。（4）传统数据中心如果只进行数据存储分析业务，服务器仍需要处理 1015 字

[1]Bonomi F.Connected vehicles,the Internet of Things,and fogcomputing//Keynotes of the8th ACM International Workshop on Vehicular Inter-Networking.Las Vegas,USA,2011

[2]Bonomi F,Milito R,Zhu J, Addepallis.Fogcomputing and its role in the internet of things//Proceedings of the Ist Edition of the Mcc Workshop on Mobile Cloud Computing.

[3]Patel M,Hu Y,Chanc,etal.Mobile-edgecomputing.France:European Telecommunications Standards Institute,Introductory Technical White Paper,2014

节的数据，如果考虑更加多样的业务，可想而知需要传输的流量十分庞大，这些多种多样的业务使得数据中心的流量模型更加复杂，各种调度也更加费时费力。（5）另一个制约数据中心规模的是带宽收敛，它限制了服务器之间的链路容量。在实际部署中，为了降低成本，树型拓扑在核心层由于交换机能力有限而出现链路带宽收敛所以存在带宽瓶颈，不能够满足服务器间大量的虚拟机迁移数据造成的东西向流量在核心层的汇集，其对分带宽、时延、吞吐量等指标均有降低，影响了服务质量（Quality of Service，QoS）和用户体验，造 成了很大的网络带宽浪费。（6）传统数据中心网络存在大量交换设备，由于缺乏有效的自动化运维及管理调控机制，所以难以控制进行实时操作调度，对全网拓扑的管理维护往往需要大量人工。下面简单介绍数据中心的现有技术：

（1）数据中心网络架构：数据中心除了电力温控等辅助设备外，最主要的就是由服务器机架和网络交换设备构成的内部网络[10]，传统的数据中心是树型的，机顶交换机（Top of the Rack，ToR）负责接入服务器主机，所有交换机按层次分类，主要分为核心层和边缘层，而大型网络还会在他们中间部署汇聚层交换机。在模块化机架设备普及的过程中，交换设备逐渐到达设备性能的瓶颈，同时由于数据中心的业务需求发生变化，对自身性能的要求也不断提高，传统数据中心网络面临着绿色节能、链路拥塞、可靠性、运维复杂等问题。

（2）数据中心流量监控技术：为了分析设计适合不同数据中心的流量调度策略掌握当前链路状态，确认业务部署是否合理，关键业务优先级是否保证，整体网络利用率是否健康，我们需要对数据中心流量进行抓取监控。传统数据中心流量的流量监控方法有 SNMP、NetStream、sFlow 和 Mirror。

SNMP 是使用最广泛的一种网络监控技术，采用轮询机制，管理设备通过公用节点访问从而获取网络设备各个端口上的流量信息实施监控。SNMP 可以掌握网络设备所有端口的流量大小和主要特征，但对报文内容无法获知。而且对于端口数量比较多的网络设备，SNMP 轮询一次设备上的所有端口会占用大量的时间，设备性能会受到很大影响。

NetStream 是基于对数据流的统计，部署在传统三层网络设备上进行数据采集，根据业务需求对链路带宽使用情况作出统计。局限在于只能统计流量的 IP、MAC、PORT。

sFlow 是基于 RFC3176 协议的，采用随机采样技术，提供流量完整的信息，可以实时的分析链路性能，且不使用镜像技术所以不占用大量网络资源。可以部署在三层网络中，用于实时分析流量传输状态。通过连续实时的方式监视各个端口由代理来转发被采样的数据流，相比其他方案，能够降低实施费用。

Mirror 镜像，十分简单原始的技术但能获得的信息却最为完整。部署在传统三层网络设备上进行数据采集。缺点在于 Mirror 需要占用太多的端口和带宽，还需要大量空间存储这些数据，而且考虑到大多数数据的拷贝是无用的，说明该种方法效率并不高。

（3）数据中心流量调度机制：数据中心流量路由调度机制是基于不同拓扑结构和流量模型的分析来对应设计的。分布式流量路由机制包括最短路径路由算法（Shortest Path Routing，SPR）、等价多路径路由算法（Equal Cost Multi Path，ECMP）、Valiant 平衡路由（Valiant Balance Routing，VBR）等。分布式路由针对各个数据流分别计算路由， 会出现流量碰撞，通常只能实现局部最优，不具有全局性，缺乏拥塞控制功能。

集中式流量路由机制包括 Hedera[9]、QoS[10]、Mahout[11]、MicroTE[12]、智能启发式[13]、TMS 等。集中式机制是由全局统一的控制器来获取网络中的流量，根据实时获取的动态数据再来指定对应的路由策略，理想情况下如果采用优异的算法能够实现链路的充分利用，但这 种路由机制通常是针对于特定的网络结构设计的，往往需要根据实际情况作出调整，部署设备及策略算法较为复杂。

（4）基于 SDN 的数据中心：以上数据中心网络的流量监控及路由调度技术都不能很好地满足

[9] Al-Fares M，Radhakrishnan S，Raghavan B，et al. Hedera: dynamic flow scheduling for data center networks[C]// Usenix Symposium on Networked Systems Design and Implementation，NSDI 2010，April 28-30，2010，San Jose，Ca，Usa. DBLP，2010:281-296.

[10] 林娜，邵志学. 基于蚁群算法的多路径 QoS 路由算法研究[J]. 沈阳航空航天大学学报，2011，28(1):67-71.

[11] Curtis A R，Kim W，Yalagandula P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection[C]// IEEE INFOCOM. IEEE，2011:1629-1637.

[12] Benson T，Anand A，Akella A，et al. MicroTE: fine grained traffic engineering for data centers[C]// Conference on Emerging NETWORKING Experiments and Technologies. ACM，2011:8.

[13] Ruff C C, Driver J, Bestmann S. Combining TMS and fMRI: from 'virtual lesions' to functional-network accounts of cognition.[J]. Cortex; a journal devoted to the study of the nervous system and behavior, 2009, 45(9):1043.

现有网络的需求。越来越多的业务被应用到数据中心内对我们的数据中心网络有了更多要求：服务器间的流量不仅具有突发性而且并不均匀，实时性较强，需要我们的数据中心网络具备负载均衡、绿色环保、虚拟租户隔离管理和 VM 的智能部署和迁移、自动化运维等功能，幸运的是，2008 年诞生于斯坦福的 SDN 技术在应对数据中心的这些需求中显示出巨大优势[14]。通过将 SDN 技术与数据中心相结合，部署统一的控制器来收集交换设备的状态信息，进行统一的调度管理、实时获取链路信息进行网络资源的调度分配以实现网络资源的充分利用。SDN 技术可以实现对数据中心网络资源的虚拟化，将网络抽象成类似硬盘资源和 cpu 计算资源一样的网络虚拟资源，实现资源的实时管理调度分配。同时由于数据中心是独立于广域互联网的半封闭网络，其结构规模可控，链路具有逻辑性，在其中部署 SDN 架构十分可行，因为本身 SDN 的诞生就是在半封闭的校园网之中。目前 Google 已经将其全球的数据中心升级成为 SDN 数据中心，将数据中心之间的链路接近 100%的利用起来。综上，数据中心网络是 SDN 架构最为热门的落地部署场景。

3 云端基础设施介绍

3.1 云存储

本次 SoCC 上就有多项工作致力于提升云端基础设施的性能和可靠性，会议有多篇论文关注云存储技术。如荣获本届会议最佳论文奖的 Netco，是由微软设计的一套针对分布式数据并行计算的数据预取和缓存系统。论文题目为《CacheandI/O Management for Analytics over Disaggregated Stores》，指出云计算中数据存储往往是与计算节点分离并分散的（Disaggregated storage）。作者考虑将存储分散设置为来自数据并行系统中的计算。并置缓存层使用计算机可以减少互连上的负载，但是这样做会带来新的资源管理挑战。系统 Netco 将数据预取到缓存中（基于工作负载的可预测性），并适当地划分缓存预取和服务之间的空间和网络带宽正在进行的工作。Netco 做出各种决定（要缓存的内容，何时缓存以及如何分配带宽）以支持端到端的优化目标，例如最大程度地增加达到他们的服务水平目标（例如期限）。这些思想的实现可在开源 Apache 中找到 HDFS 项目。在公共云上进行生产跟踪实验启发性的工作负载。因此 Netco 系统设计了一套数据预取策略，优化数据和计算的共同放置，提升计算任务的性能，实验表明 Netco 可以减少达 5 倍的远端 IO。

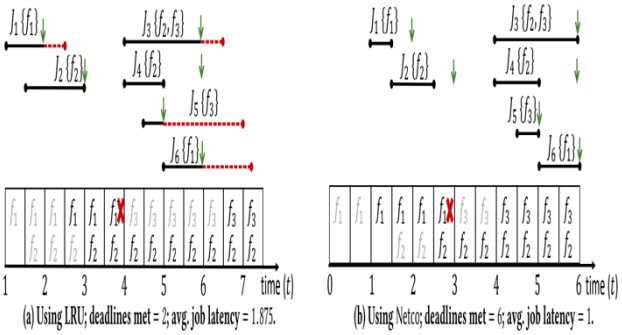

图 1 工作负载的执行延时

如图 1，正在运行的工作使用实线（黑线）显示，工作截止日期用绿色箭头。如果作业错过了截止日期，则截止时间之后的执行将以虚线（红色）显示。这些表指出了哪些文件是存在于缓存中；完整文件显示为黑色，部分文件显示为灰色。（红色）叉号表示正在从缓存中逐出文件。作业启动时不在缓存中的文件将从远程存储中读取，并且可以缓存。

3.2 Wharf

该论文题目为《Wharf: Sharing Docker Images in a Distributed File System》，Wharf 是一个用于共享 Docker 镜像的分布式文件系统。作者针对云计算越来越多地依赖于容器（如 Docker）技术的趋势，设计了这样一个 Docker 镜像共享系统，使得数据中心中大量使用相同镜像的任务高效地、合作地从存储系统中拉取和共享镜像文件。Wharf 可以提升镜像拉取速度达 12 倍，大大提升了基于容器的计算及对其管理的效率。

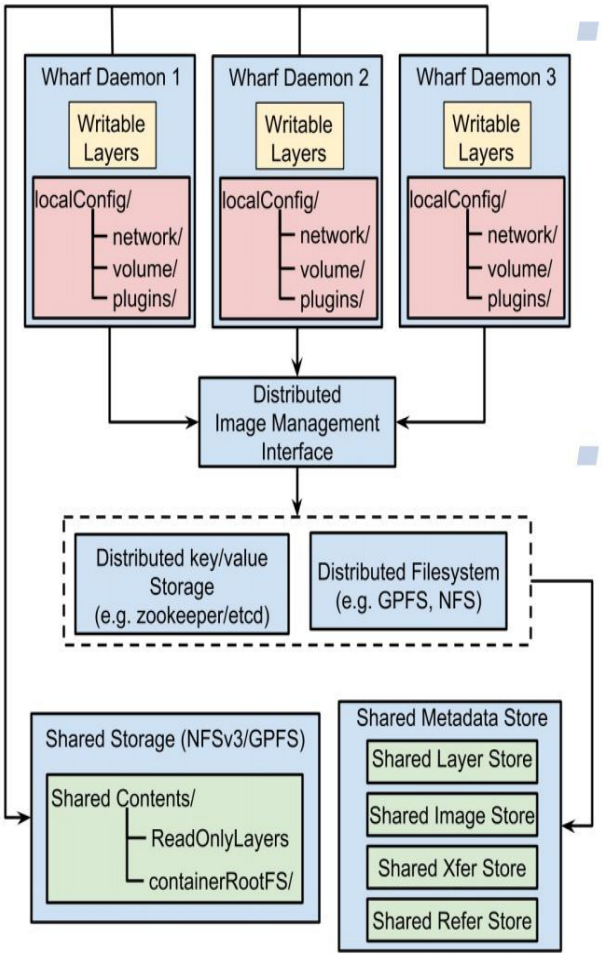

图 2 Wharf 通过一个分布式文件系统使得容器（Wharf Daemon）之间可以高效地共享 Docker 镜像

3.3 虚拟化与数据中心网络

虚拟化和数据中心网络是云计算构建的基石，而今年 SoCC 就有若干篇针对这两项技术的文章。数据中心往往将物理的计算资源虚拟化为虚拟机等单元，从而提供不同用户之间的性能隔离。系统虚拟化（例如虚拟机抽象）已被确定为多租户云中隔离的事实上的标准形式。其中论文《Unikernels as Processes》做的工作提出将虚拟机中常用的 Unikernel 作为进程运行在外部操作系统上，相对于直接运行在虚拟化的资源上，可以在不丢失性能隔离的前提下带来更高的灵活性。

从 2014 年以来，容器以一种不可逆转的态势席卷了全球，Unikernel 是很多人眼中的下一个容器。了解 Unikernel 前首先需要了解什么是 kernel，kernel 是操作系统中的一个概念。应用要运行起来，肯定要和硬件打交道，但是如果让应用都直接操作硬件，那一定是一场灾难。那内核就是在应用与硬件中间的一层抽象，内核提供了对底层硬件的抽象，比如把硬盘抽象成了文件，通过文件系统进行管理。传统的内核会将所有的硬件抽象都实现在其中，其中的代表就是 Linux，这样的内核被称为宏内核（Monolithic Kernel)。在宏内核中，所有的模块，诸如进程管理，内存管理，文件系统等等都是实现在内核中的。这样虽然不存在通信问题，但是任何一个模块的 bug 会使得整个内核崩溃。于是学者们提出了微内核（Micro Kernel）的概念，在内核中只保留必要的模块，比如 IPC，内存管理，CPU 调度等等。而其他，诸如文件系统，网络 IO 等等，都放在用户态来实现。这样会使得内核不那么容易崩溃，而且内核需要的内存小了。但是由于模块间的通信需要以 IPC 的方式进行，因此有一定的 overhead，效率不如宏内核。后来又有了混合内核（Hybrid Kernel)，把一部分不常使用的内核模块，或者是原本需要的时间就很长，因此 IPC 的 overhead 看起来就不是那么夸张的功能，移出内核，而其他的就会放在内核里。

Unikernel 是专用的，单地址空间的，使用 library OS 构建出来的镜像，Unikernel 主要目的是减小不必要的代码逻辑，让最简单的操作系统来运行应用。比如运行 web 应用，可能只需要运行网络处理逻辑、磁盘处理逻辑、语言解释逻辑、cpu 调度逻辑就可以了，其他的权限管理、进程切换、可视化、音视频播放等都可以删除掉。另外其最大的卖点就是在于没有用户空间与内核空间之分，只有一个连续的地址空间。这样使得 Unikernel 中只能运行一个应用，而且对于运行的应用而言，没有硬件抽象可言，所有的逻辑，包括应用逻辑和操作硬件的逻辑，都在一个地址空间中。Unikernel 镜像都很小，启动很快，而且安全性很高。Unikernel 在真正实践中，如何开发与测试是一个值得关注的问题。在开发过程中，开发者可以假定自己在传统的操作系统上进行开发，而所有内核相关的功能，暂且由开发机的操作系统提供。而在测试环境中，大部分 Unikernel 的实现会将应用代码与需要的内核模块构建成 Unikernel 后，再将其跑在一个传统的操作系统上，利用传统操作系统上的工具来测试 Unikernel。

在这篇论文中，Unikernel 实际上并不需要虚拟硬件抽象，但可以达到相似的水平通过利用现有资源作为进程运行时的隔离内核系统调用白名单机制。而且，显示运行进程的 unikernel 减少了硬件要求，可以使用标准过程调试和管理工具，并提高了

Unikernel 表现出的出色性能。

Unikernel，在我看来，是另一种形式上的容器。在一个 Unikernel 中，只能运行一个应用，这与容器的哲学不谋而合。但现在容器最吸引人的特性并不是它的便捷，而是在它的分发。Docker 让我们看到了，原来应用的分发可以这么无痛。而 Unikernel 与容器相比，虽然可以做的更小更安全，而且也不需要有 Docker Daemon 这样的后台程序存在，甚至不需要 Host OS，或者 Hypervisor，但是它一是与传统的软件过程有较大的出入，二是在分发等等方面不能做到像容器那样方便。所以它目前肯定不会成为主流的应用分发方式，还需要进一步探索。

RoGUE（RDMA over Generic Unconverged Ethernet）是一种在以太网上的 RDMA 实现。作者提出传统的 RoCE 依赖于优先级流量控制（PFC）机制，但是 PFC 带来了诸多性能稳定性方面的副作用。RoGUE 通过一种新型的拥塞控制和恢复机制，解决了前述的 RoCE 中存在的问题。融合以太网上的 RDMA（RoCE）保证了低延迟和商品网络上的 CPU 使用率低，并且对云基础架构服务。当前的实现需要使用基于背压的拥塞的优先流量控制（PFC）控制为 RDMA 提供无损网络。不幸，PFC 损害了网络稳定性。结果，RoCE 的采用速度很慢，需要复杂的网络管理。最近诸如 DCQCN 之类的漏洞会降低网络风险，但不会彻底解决问题。该文描述了 RoGUE，一种新的拥塞控制和恢复且不依赖 PFC 的以太网 RDMA 机制。RoGUE 通过软件实现，以支持向后兼容性并适应网络发展，但仍允许使用 RDMA 实现高性能，同时支持 RC 和 UCRDMA 传输。我们的实验表明，RoGUE 的性能和 CPU 利用率均达到或优于原生 RDMA 协议，但可以宽容网络拥塞。

3.4 云计算的一致性与正确性

3.4.1 SDPaxos 介绍

数据副本是用来保证部署在云端服务的可用性的经典手段。在“一致性与正确性”论坛中，有三项工作都关注了副本协议一致性的问题。如 CRIC 是一种针对跨地域、跨云提供商情景提出的一种一致性协议。现在有许多服务会部署在不同的云提供商的数据中心中（Multi-cloud），但这样我们不能依靠云提供商来保证数据跨云存储的一致性。CRIC 利用不同云提供商提供的有限的接口，巧妙地实现了跨云的存储一致性。

由北京大学和微软亚洲研究院系统组合作完成的工作 SDPaxos，同样也关注跨地域副本问题，将副本协议中的复制和排序两种语义分离开来，并将复制操作分散化，而将排序操作中心化，通过这样的“半分散式”设计，SDPaxos 解决了传统协议的低性能或性能稳定性差的问题。SDPaxos 在 Strong Leader 和 Leaderless 中间做了一个优雅的折中，使得协议远比 EPaxos 清晰，但又支持了损失很小的多点并发复制。

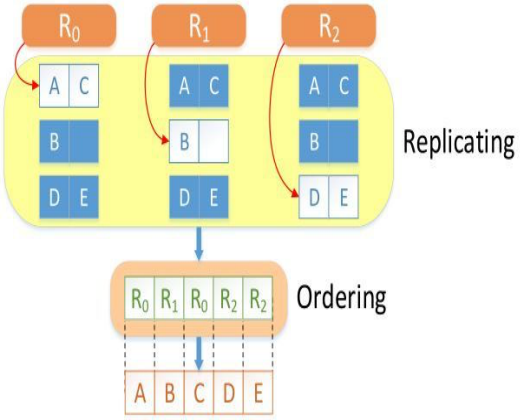

Figure 2: Separation of replicating and ordering.

图 3 SDPaxos 分离了副本协议中“复制”和“排序”两种语义并使用了不同的通信模式，
提升了协议在动态变化的环境中的性能

3.4.2 SDPaxos 基本设计

SDPaxos 的基本思路是复制和定序分离，复制使用的是类似 EPaxos 中的方法，每次客户端的请求都是指定一个 Command Leader 来处理，但是处理请求顺序的是一个中心化的 Sequencer 节点处理。这个看起来就是一个更折中的方法。由于定序带来的开销比复制小的多，这个定序的中心节点的负载也会小于 Multi-Paxos/Raft 中的 Leader/Master。在 SDPaxos 中，Instance 的定义和一般的没有什么不同。不过这里它分为了 C-instance 和 O-instance，分别对应复制和顺序。一个副本 R0 在收到了一个客户端的请求之后，它就成为这个客户端的这个请求的 Command Leader，选择一个 C-instance，然后使用 C-accept 操作将这个请求复制到其它的副本。同时，R2 作为 Sequencer 在收到了这个信息是的时候，R2 提出这个请求的 O-instance，然后发送 O-accept 给其它的副本(带有全局 slot 的信息)。其它的副本

在收到了 C-instance 和 O-instance 之正常情况下分别回复 C-ACK 和 O-ACK 给 R0，在收到了多数的副本的两个 ACK 回复之后，既可广播 C-commit 和 O-commit 信息。

```
# C-phase
Replica Rn on receiving a client request for command α :
sendC-accept(n,i,α,bali)to at least a majority
if the C-accept is not sent to the sequencer then
    send C-request(n, i) to the sequencer
increment the C-instance counter i
Any replica R on accepting C-accept(n,i,α,bali):
sendC-ACK(n,i,α,bali)toRn
Rn on receiving C-ACKs from a majority of replicas:
commit Cni and broadcast C-commit(n,i,α,bali)

# O-phase
Sequencer Rm on receiving C-accept(n,i,α,bali) or
C-request(n, i) from Rn:
if i ≥ number of O-instances for Rn in sequencer's
assignment log then
        send O-accept(j, n, balj ) to at least a majority
including Rn
        increment the O-instance counter j
Any replica R on accepting O-accept(j, n, balj ):
send O-ACK(j, n, balj ) to Rn
```

SDPaxos 定义一个 Command 为 ready 状态指的是可以安全地回复客户端。在三个或者超过五个副本的系统中，就是常用的等到多数的 O-ACKs 之后就可以执行提交 O-instance 的操作。当这个副本已经提交了这个 C-instance Cni(及这个副本的第 i 个 C-intance)且至少有 i 个 O-instance 的时候就可以返回客户端，在三个副本的时候，如果 Command Leader 和 Sequencer 不是同一个副本，那么它们两个就可以构成一个多数，正常情况下 SDPaxos 运行只要个 RTT 就可以 Commit，因为 Sequencer 返回 Command Leader 的时候就带有 O-instance 的信息。而如果是多个，则多于 1 RTT 才能 Commit。

```
# Ready condition for three or more than five replicas
if Rn receives O-ACKs from a majority of replicas then
    commit Oj and broadcast O-commit(j,n,balj)
if Rn has committed Cni and at least i O-instances for Rn
then
    respond to client
```

SDPaxos 针对 5 个副本的情况做了优化，使得 5 副本的情况下也可以在 1RTT 就可以 Commit。这里 C-instance 的操作依然是没有变的，必须多数确认(C-ACKs)。而对于 O-instance，从前面 3 副本的情况下可以看出，如果 Sequencer 和 Command Leader 不是同一个，则只能保证 O-instance 被 2 个副本确认的情况下才能满足 1RTT 的要求。SDPaxos 的优化就是值等待 Sequencer 的 O-accept 返回，加上自己就是两个。但是如果这两个故障的时候，就可能无法恢复全部的信息，导致不能满足容错 2 个的要求。所以这里必须使用额外的条件，及这里要保证在 O-commit 的时候前面的已经提交，这里的前提是 C-instance 这个时候必然是在多数节点上保存了的，故障了两个节点数据不会丢，主要是要恢复顺序信息。另外还配合了 SDPaxos 恢复的算法。

```
# Ready condition for five replicas
if Rn is the sequencer then
    commit Oj and broadcast O-commit(j, n, balj ) on
receiving O-ACKs from a majority of replicas
    if Rn has committed Cni ∧ any Ok with k ≤ j (Oj is the
ith O-instance for Rn) is accepted by a majority then
        respond to client
else
    commit Oj and broadcast O-commit(j, n, balj ) on
receiving the O-accept from the sequencer
    if Rn has committed Cni ∧ any Ok withk ≤ j (Oj is the
ith O-instance for Rn ) is accepted by itself and the
sequencer then
        respond to client
```

这里对 SDPaxos 对设计选择进行简单描述，从对 Leader 的依赖这个角度看，SDPaxos 在 Design Space/Spectrum 中选择了中间的一个折中位置。

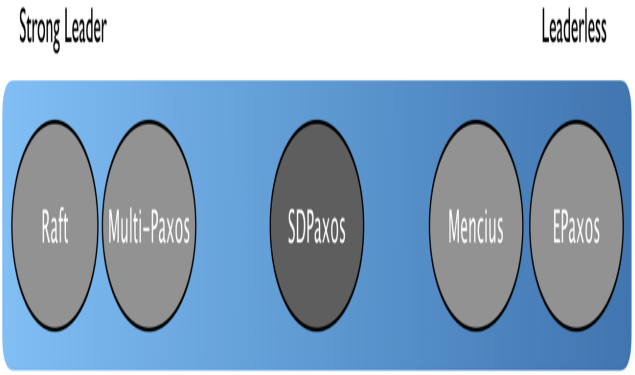

图4 SDPaxos地位

如图 4 所示，左一 Raft 协议强依赖 leader，例如在选举时必须选择在某个多数派中拥有最新日志的节点。左二 Multi-Paxos 协议的 liveness/progress 依赖 stable leader，但协议正确性不依赖。跳过 SDPaxos，右二 Mencius 让每个节点轮流 propose command，例如三副本 configuration 下，replica 0 propose slot 1,4,7...3x+1...的 command。最右 EPaxos 最极端，任意节点 propose command，冲突的 command 在 replication 过程中检测并确定顺序。SDPaxos 的折中是，command order 仍由单点确定，但各副本都可以 propose 和 replicate command。可以做个比喻，Raft 就像是苏联的社会主义计划经济，ordering/replicating 都由中心节点来管理；EPaxos 就像资本主义市场经济，各个 replica 充分竞争，自行解决 ordering 问题；SDPaxos 就像是社会主义市场经济，ordering 由单点的 sequencer 来确定，replicating 由各个副本来做。

3.4.3 SDPaxos 恢复机制

SDPaxos 为了处理 Sequencer 的故障，通用引入了 View Number 的概念，照样会在一个选举的过程。除了 5 个副本的恢复外，SDPaxos 的恢复过程没有特别的地方。由于 SDPaxos 对 5 个副本情况下的优化，Rm 代表在这个故障中非 Sequencer 的那一个。恢复的过程需要从存活的版本的 C-instacnes 中找到 Rm 编号最大的一个，Sequencer 找到合适 O-instance，然后“补全”对应 hole。一个基本的伪代码如下：

```
# Recovery of O-instances
Input: C[m]: number of C-instances of Rm accepted by the majority voters;
       O[m]: number of O-instances for Rm (Rm is the replica other than the old sequencer and the majority voters in five-replica groups)
if N==5 then repeat
  propose Rm's ID in the first empty O-instance
  O[m] ← O[m] + 1
until O[m] ≥ C[m] ∧ there is no empty O-instance preceding the C[m]th one for Rm

foreach empty O-instance between non-empty ones do
  propose no-op
```

3.4.4 SDPaxos 优化

SDPaxos 另外的一些优化的策略有：

1.消息合并，在一些情况下可以将 C-instance 和 O-instance 合并发送，减少传输包的数量；

2.掉队检测，可以 Sequencer 能否按序处理请求来检测 Sequencer 的状态，在判断 Sequencer 状态不是很好时可以启动 ViewChang；

3.Sequencer 分组，在一些具体的相同中，比如 KVS 相同可以根据一致性 Hash 来划分 Sequencer；

4. O-instance 批量处理。

3.4.4 SDPaxos 优势

1.协议可理解性：SDPaxos 清晰的多。这里举几个 EPaxos 中很不直观的地方：EPaxos fast quorum 采用的是 thrifty 模式，只能向特定几个 replica 发送 accept request；accept request 要携带 dependency 信息。

2.协议 CPU 开销：SDPaxos CPU 开销低，基本跟 Raft/Multi-Paxos 所需的开销接近。EPaxos 有大量处理 dependency 的 key 检测，会消耗大量的 CPU。

3.落地实现难度：同样是因为复杂性，EPaxos 协议在工程上很难落地，就 EPaxos 的 recovery 逻辑，几乎很难实现一个生产环境可用的正确版本，但 SDPaxos 就容易的多。

4．正确性直觉：虽然 EPaxos 也经过了论证和 TLA+证明，但考虑到 TLA+证明过程本身的可靠性，要说 EPaxos 协议没有正确性 bug，并不能容易结接受。相反，受益于 SDPaxos 协议的清晰，接受 SDPaxos 正确性在直觉上容易的多。

4 结语

云计算是基于多种技术的新兴计算模式，随着现代软件应用和商务处理的全球化、信息化和自动

化，必将为云计算的研究发展提供广泛的市场和应用背景。云计算不仅是虚拟化资源的集合，也不仅是在此之上的平台和应用实体的集合，而且是一种集虚拟化技术、网络技术、信息安全、效用计算、逻辑推理、软件工程、商务智能等技术为一体的新兴计算应用模式。无论是工业界还是学术界都提出了一系列实施技术和改进策略，并从理论和实际应用的角度进行了阐述。

本文首先介绍了云计算的相关背景，分析了云计算现有的优势和亟待解决的问题，然后由此给出了云计算的定义，通过 SoCC 2018 会议，重点围绕关于云端基础设施的 5 篇论文展开综述，分别包括了云存储、虚拟化与数据中心网络、云计算的一致性与正确性三大主题，概述了现阶段数　据中心中的存储、网络、虚拟化等等多种基础设施或技术的最新的研究方向，指出了各方向的研究目的。从理论和实际应用的角度分析了存在的问题，指出了未来云计算研发中应解决的关键问题和研究方向。由上面的讨论分析可知，应用向云计算模式的转变引发了一系列开放的问题，有待解决。

(1)用户在选择使用众多云计算服务时，如何选择需要的服务应用，通过什么标准度量云计算服务特征，避免选择的主观性。

(2)以往 web 服务定义的 WSDL 接口和 XML 数据类型方便用户的调用和信息的传输，需要考虑云计算的接口、数据类型怎样制定，采取何种具体的标准加强云计算供应商和用户间的互操作尚不明确。

(3)随着云计算模式的大量应用，是否所有的软件应用和开发都适合转向云计算的平台，这就需要考虑建立软件应用属性到云计算服务属性的映射，以判定云计算的属性是否适合软件应用的关键属性。

(4)如何划分 Saas 层次上云计算基本服务粒度，以便应用能够进行类似 Web 服务编排的服务组合，提高软件的重用性。

(5)云计算服务本身的质量关系到用户能否大量使用服务。目前，研究主要集中在底层基础设施服务的性能分析、优化以及测试研究，但是上层的云计算应用服务的测试模型和测试标准同样需要研究和关注。

参 考 文 献

[1] Bonomi F.Connected vehicles,the Internet of Things,and fogcomputing//Keynotes of the8th ACM International Workshop on Vehicular Inter-Networking.Las Vegas,USA,2011

[2] Bonomi F,Milito R,Zhu J, Addepallis.Fogcomputing and its role in the internet of things//Proceedings of the Ist Edition of the Mcc Workshop on Mobile Cloud Computing.

[3] Patel M,Hu Y,Chanc,etal.Mobile-edgecomputing.France:European Telecommunications Standards Institute,Introductory Technical White Paper,2014

[4] Satyanarayanan M.The emergence of edgecomputing.EEE Computer,2017,50(1):30-39

[5] 陈康,郑纬民．云计算：系统实例与研究现状[J].软件学报,2009,20(5)：1337 一l348 [博士学位论文/硕士学位论文].

[6] LenkA，KlemsM，NimisJ，eta1．What’s inside the Cloud?An Architectural Map of the Cloud Landscape[C]fProceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing.2009:2331

[7] VaqueroL,Roderc~MarinoL,CaceresJ,eta1.A break in the clouds:towards a cloud definition [J].SIGCOMM Computer Communication Review,2009,39(1):50 55

[8] GeelanJ.Twenty one experts define cloud computing.Virtualization[EB/OI].http://virtualization.sys—con.corn/node/612375. 2008.08

[9] Al-Fares M，Radhakrishnan S，Raghavan B，et al. Hedera: dynamic flow scheduling for data center networks[C]// Usenix Symposium on Networked Systems Design and Implementation，NSDI 2010，April 28-30，2010，San Jose，Ca，Usa. DBLP，2010:281-296.

[10] 林娜，邵志学. 基于蚁群算法的多路径 QoS 路由算法研究[J]. 沈阳航空航天大学学报，2011，28(1):67-71.

[11] Curtis A R，Kim W，Yalagandula P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection[C]// IEEE INFOCOM. IEEE，2011:1629-1637.

[12] Benson T，Anand A，Akella A，et al. MicroTE: fine grained traffic engineering for data centers[C]// Conference on Emerging NETWORKING Experiments and Technologies. ACM，2011:8.

[13] Ruff C C, Driver J, Bestmann S. Combining TMS and fMRI: from 'virtual lesions' to functional-network accounts of cognition.[J]. Cortex; a journal devoted to the study of the nervous system and behavior, 2009, 45(9):1043.

[14] Ghemawat S，Gobioff H，Leung S. File and sToRage systems: The Google File System[J]. Acm Symposium on Operating Systems Principles Bolton Landing，2003，37:29-43.