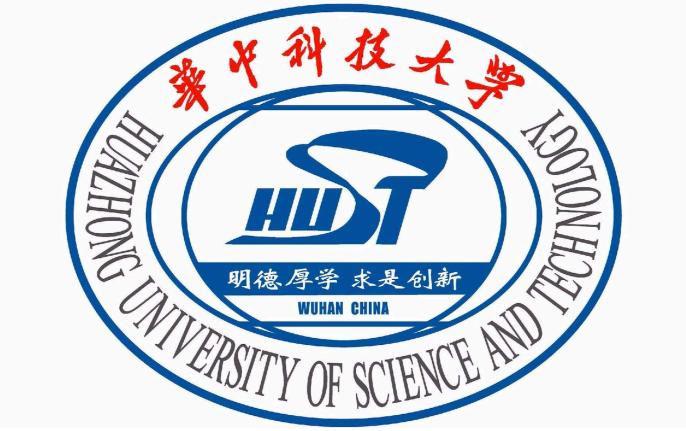
《Evolution of Cloud Operating System: From Technology to Ecosystem》



姓名：范唯 班级：CS1703

学号：U201714670

课程名: 云计算和虚拟化

指导教师: 吴松

前言

云计算与虚拟化这门课，可以称得上是大学三年以来专业通识教育课程里最为精彩与详尽的。在翻阅了吴老师的多篇文章之后，最终选定了《Evolution of Cloud Operating System: From Technology to Ecosystem》来进行品读。相较于其他在某一方面深刻钻研的论文，我更倾向于此篇更倾向于综合论述的文章，因为这会让我对云计算系统有一个更加宏观的概念与把握，来帮助我更好的去理解以及发掘自己的兴趣点。也希望更为详尽的陈述出文中的信息，以便日后时常翻阅。

阅读体会

Abstract The cloud operating system (cloud OS) is used for managing the cloud resources such that they can be used effectively and efficiently. And also it is the duty of cloud OS to provide convenient interface for users and applications. However, these two goals are often conflicting because convenient abstraction usually needs more computing resources. Thus, the cloud OS has its own characteristics of resource management and task scheduling for supporting various kinds of cloud applications. The evolution of cloud OS is in fact driven by these two often conflicting goals and finding the right tradeoff between them makes each phase of the evolution happen. In this paper, we have investigated the ways of cloud OS evolution from three different aspects: enabling technology evolution, OS architecture evolution and cloud ecosystem evolution. We show that finding the appropriate APIs (application programming interfaces) is critical for the next phase of cloud OS evolution. Convenient interfaces need to be provided without scarifying efficiency when APIs are chosen. We present an API-driven cloud OS practice, showing the great capability of APIs for developing a better cloud OS and helping build and run the cloud ecosystem healthily.

Keywords cloud computing, operating system, architecture evolution, virtualization, cloud ecosystem

有关于云操作系统的讨论，总是绕不开便捷与效率。然而这两个目标总是会发生相当大的冲突。便捷意味着更好的用户体验，尤其是工业级别的cloud OS，在许多应用场景下，可能开发效率的需求要远远大于运行效率。所以在观察云OS的发展历程时，总会有趣的发现Faster和Convenient两个命题总是相互驱动最后取折中的方案。

而在最近几年，国内的云服务商崛起迅猛，如老师课上所言。除去美国的几大巨头（Google,Amazon,Microsoft），阿里云已经基本上囊括了国内的市场的大多数份额。从底层架构出发，到API接口的设计，用户UI的考量，在早期都需要极其高昂的成本。正如下图中的体系结构，云计算系统的核心一般是基于unix系的系统。系统的扩展有纵向扩展（scale-up）和横向扩展（scale-out）两种方式，传统的单机操作系统关注前者，而云计算场景下需要关注后者才能满足计算需求。而基于OS Kernel，Core->LaaS->Paas->SaaS也是层层嵌套，API的集成等级也越来越高。

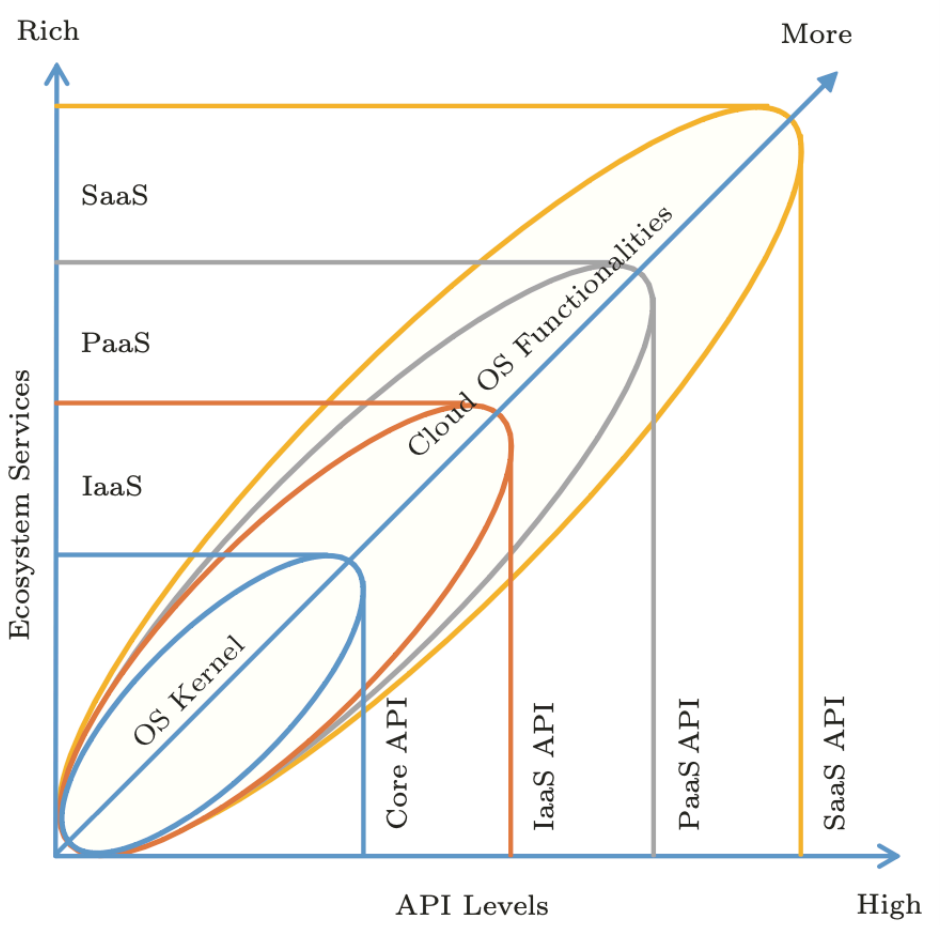


Fig.1. Cloud OS API levels and the ecosystem services.

从十分抽象的角度来说，计算这种行为可以被视为0和1的序列化互传。那么服务端在本地或者远端就无足轻重了，云计算系统也便应运而生。对于云服务商来说，只需要提供更为方便高效的API即可，而软件开发者们则需要根据这些API来增删软件的交互接口，OpenStack与Hadoop便是两个非常完美的例子。但是在云服务开发者仍然会面对一个挑战——“如何权衡隐私性与凯源性。”许多时候使用开源软件的大众版本无异于会增加被攻击的概率，但是纯粹的为了个人用户去开发特殊版本来进行兼容又会变得冗余而昂贵，因为在持续集成的驱动下，任何小的改动都是牵一发而动全身的。所以这个难题被甩到了云服务商的面前——“如何定义更便捷高效但是安全稳定的APIs”

similar but slightly different interfaces. Thus, the cloud application developers have to face the problem of vendor lock-in which means that applications built for one specific provider will not run on the other platforms. Even for building the private cloud environments using open source software, the developers have to rely on the specific open source implementation of specific version. This will impede the construction of cloud ecosystem or

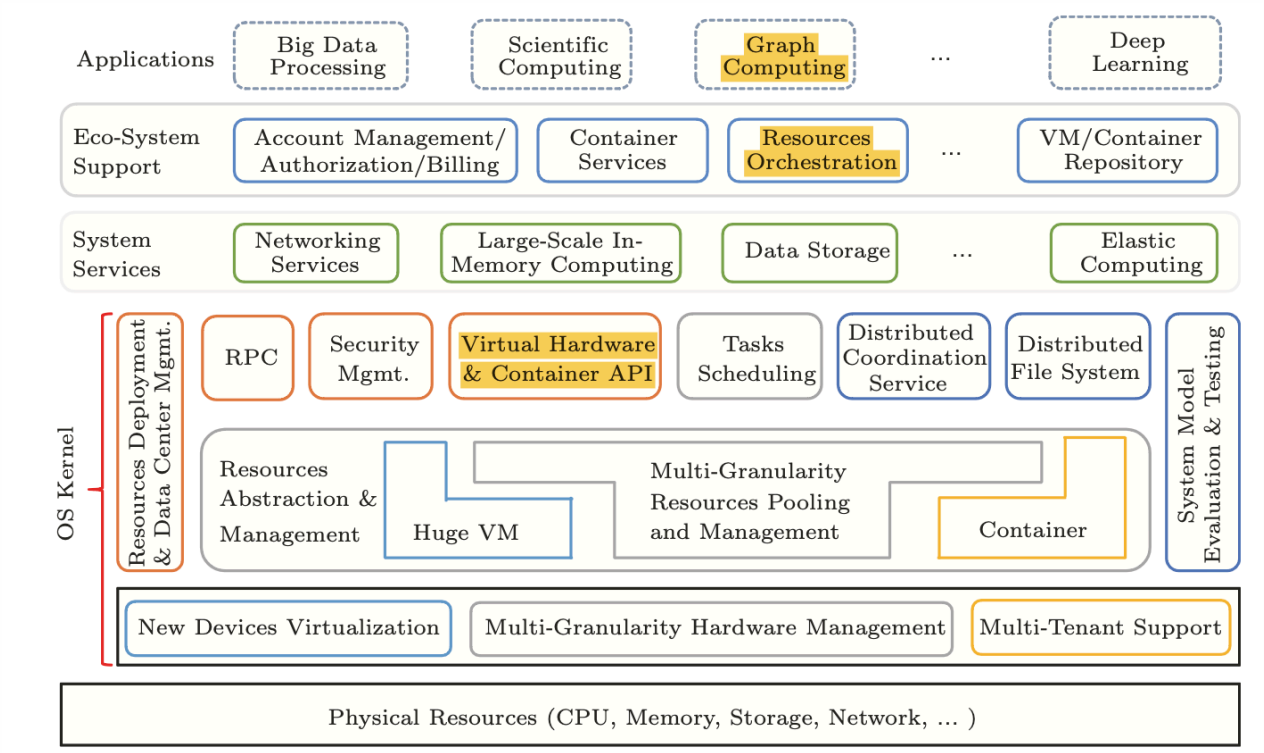
追溯到云计算的起步阶段，一般是围绕虚拟机来进行实现的。甚至在某一时期出现过“虚拟机==云计算”的命题。抛开这个话题，提升云计算性能的两大途径，一个是提升硬件性能，另一个就是优化软件。

- 硬件方面：硬件层的虚拟化、RDMA技术、网络拓扑结构的改进、单台机器架构的改进、存储系统的改进（固态硬盘、非易失性存储）。

- 软件方面：系统软件需要去适应新的硬件特征，比如多核计算机、SIMD指令集等。

为了应对大数据应用的高负荷，一些计算框架比如MapReduce、Spark便应运而生了。而为了应对众多的软件工程需要，APIs也需要进一步被明确分工，来降低版本维护的成本。正如手机系统中常常多APP的迭代成本要远低于单APP。而cloud OS的微内核也逐渐成为了一种趋势，需要插入新的应用服务时不至于让整个系统宕机。而现在许多管理软件如Docker也能实现类似的分离效果。

有关于云服务生态环境，许多大厂的APIs的定义不统一与混乱时常会引发一系列的问题。比如应用的迁徙时接口的不统一会导致许多重复无效的工作。不过在经过一段时间的混乱后，云开发商们开始着手于接口的抽象，Why？因为抽象意味着泛，兼容性会变得更强。但是核心的APIs必须要非常稳定且不能为了任何特殊的应用去做适配，底层的架构松散，那么上层结构便会经常崩溃。下图是一个文中非常经典的cloud OS体系结构图。从最底层的CPU、Memory...到顶层的Deep Learning等应用程序，基本囊括了全文的逻辑结构，可以供给以后进行反复翻阅来进行复习。



课程体会

在整个课程过程中，老师的讲述侧重点是更相关于云计算与虚拟化的入门，对于我这种纯粹的小白来说是十分友好的。从cloud OS的起源，以及各大云服务商的发展历程，`LaaS >> PaaS >> SaaS`。在每堂课上都会留有许多科普视频资料来供复习与参考。这点对于学生来说是比较有好的。但是也有如下几点建议，希望老师可以采纳。

- 可以布置一些有关基本概念的手写作业，来督促学生熟悉底层知识架构。
- 可以把结课的方式增加一些，比如可以让学生去自己在阿里云上搭一个云服务来跑一些服务，比如个人博客，爬虫，个人云盘等等。