关于新一代操作系统的思考

陈榕 2016.9.15

以史为鉴可知兴替

如果我们人为用十五年的时间窗口去划分计算机历史，虽然不是很准确，但也大致能发现有意义的规律：

- 1945-1959 计算机萌芽期，人作为操作系统时代；
- 1960-1974 大型机 OS/360 任务批处理机操作系统，人工辅助下的机器主导时代；
- 1975-1990 小型机 Unix 多任务分时处理，机人交互操作系统的时代；
- 1991-2004 PC 机 Windows 消息机制下的事件驱动编程，人机交互操作系统时代；
- 2005-2019 智能手机注册/回调机制下的事件驱动编程，利用特定个人硬件机器的人人交互操作系统时代。

2020-？沿着历史走来，用于通用计算的计算机越来越小，比手机更小的计算机作为常用人机交互终端已经没有多大意义。

我们遐想未来：人与人、人与机通过无所不在的周边 IoT 计算机交互。人不再与固定手机绑定，而是与容器虚拟机绑定。宿主操作系统（Host OS）与访客容器虚拟机操作系统（Guest OS）分离。

容器虚拟机操作系统具备弹性，动态适配于形形色色、大大小小的通用计算机终端硬件。容器虚拟机操作系统按需加载，独立演进，成为未来操作系统发展的主流。

时代造就操作系统，操作系统造就时代

通用计算机操作系统功能大致分为上下两块：

- 由内核（Kernel）+网络（TCP/IP）实现对计算机与网络硬件的抽象描述；
- 由中间件/框架+虚拟机/容器实现对形形色色应用的支撑。

从 1970 年代 IBM 大型机 OS/360，到 1980 年代 VAX 小型机操 UNIX，操作系统内核技术及 TCP/IP 互联网技术日臻完善。尽管 UNIX 开创性地设计了外设驱动模型及对结构化 C 语言程序运行的支撑，直到 1985 年之后，苹果 Macintosh、微软 Windows 视窗技术普及之前，操作系统作为应用运行平台的人机交互界面仍旧非常简陋，中间件框架及容器基本不存在。

今天的 Linux 操作系统诞生于 1990 年代初期，基本是用开源理念对 UNIX 的重复实现，在技术上谈不上多少创新，但是开源的理念迎合了操作系统作为生态环境的必然。由此酝 酿了操作系统不再是“产品”的观念，赋予操作系统作为各类软硬件产品赖以生存的载体的角色，大概可以类比商家与政府，创业公司与孵化器的关系。

随着 1946 年 ENIAC 作为第一台被世人关注的通用计算机投入使用，到 1950 年代中期，冯·诺依曼在论文中总结了集体研发的两项基本原则：程序作为数据加载到内存运行，计算机体系结构由寄存器、内存、磁盘等三层架构组成。遵循这两项原则设计的计算机也常被后人称为“冯·诺依曼机”。在之后的三十多年中，无数英雄人物、商业公司试图挑战冯·诺依曼机的原则，比如：向量机、堆栈机、数据流机、Lisp 机、Prolog 五代机等。那时与各种机器配合的操作系统也是层出不穷，每年小有名气的操作系统发布十几个。

回首往事，人们意识到一个非常不幸的事实：尽管那时的机器规模不大，操作系统编写核心团队也不过五到十人，但不论设计新机器还是开发新的操作系统，研发周期起码要五年。摩尔定律说：每两年左右 CPU 的速度就翻番。结果导致新型的机器的效率不如老机器换个新 CPU。另一个更不幸的事实是有人证明各种机器都与冯·诺依曼机等价，操作系统的消息机制与调用机制也等价。到 1990 年代中期，超大规模集成电路技术成熟，十几个人的团队，五年已经做不了任何新机器，也不可能完成任何一个有实质意义的通用操作系统了。

天不变，道亦不变。既然至今所有计算机都是冯·诺依曼机，Linux 内核作为最普遍采用的计算机内核也就越来越被接受了。不是说 Linux 内核无懈可击，但既然等价，在某些指标上好个百分之多少的内核，已经很难撼动 Linux 内核的地位了。微软最近也在努力推进其开源战略并积极贡献代码到 Linux 开源社区。

Linux 加上 UNIX 的历史已经将近半个世纪了，比大多数 IT 工程师的岁数还大，再因为开源又普及，大家毫无疑问地以为 Linux 是操作系统。如果我们仔细想一下，曾经 DOS 也被称为操作系统，后来 DOS 上面运行 Windows。由于一段时间内 Windows 的市场占有率奇高，大家跟着微软顺嘴叫 Windows 操作系统。那么到底 DOS 是操作系统还是 Windows 是操作系统？Android 运行于 Linux 内核之上，Android 是操作系统还是 Linux 是操作系统？这个问题我们留在后面章节来回答。

施乐（Xerox）与网景（Netscape）的故事

Window 视窗技术既不是苹果，也不是微软发明的，而是一间曾经赫赫有名的施乐公司（Xerox）发明的。为了方便人与机器交互，施乐发明了鼠标。为了方便机器与机器交互，施乐发明了以太网。为了便于程序员编写人机交互程序，施乐发明了近乎完美的面向对象语言 Smalltalk-80（1980 年）。就凭这四样技术，大家有没有想过施乐不发财都难。

电影、小说、报刊、杂志上都说 Steve Jobs 如何天才，去施乐 PARC 研究中心看到了视窗演示，发现了未来。等 Bill Gates 去 PARC 研究中心看到了视窗演示，就成了偷窃了未来。其实，施乐研究院既然发明了以太网和视窗技术，他们当然最先知道什么是互联网及计算机的未来，用不着 Steve Jobs 或者 Bill Gates 去指点江山。

当年施乐的看家产品是静电复印机，一台的售价上万美元。他们就是在高额利润下，不计成本投入研发，吸引大量优秀研发人员，顺带发明了许多计算机相关技术。这点非常像 AT&T 贝尔实验室，曾经利用电信通话的垄断性利润，发明了晶体管、超导等造福人类的技术。1984 年在美国电脑商店里，一台 Xerox Star Workstation 的售价是$14,000 美元。一台苹果或者 IBM PC 的售价大概是$2,000 美元。那时一辆汽车一万多美元，一栋别墅十几万美元。如果再考虑每两年电脑价钱掉一半，没人会去购买施乐电脑。华尔街股票市场是逐利的，不会允许施乐生产$2,000 美元的 PC，自然也就没有后来了。

唠唠叨叨，说这些跟操作系统战略有啥关系？近三十年来，操作系统内核争斗以 Linux 胜出告一段落，但伴随 Window 视窗技术的诞生，中间件/框架+虚拟机/容器技术却如火如荼地发展，还有愈演愈烈的趋势。

1995 年夏天，网景公司股票上市，当时提出的口号是“浏览器是未来的操作系统”，其影响至今不乏拥趸。前几年还红火一阵的 webOS、Firefox OS 的技术特点可圈可点。网景发明的 JavaScript 语言至今越来越普及了。浏览器的好处就不在这里赘述了，一句话，浏览器改变了世界。

浏览器引导的万维网（WWW）超越了公司的生死，网景、雅虎没了，成千上万新公司出现了，谷歌、亚马逊、Facebook、阿里、腾讯、百度是其中的佼佼者。政府搭台，公司唱戏，建议政府把资金投向开源、平台类技术，投到让所有创业公司受益的事业上。比如网景公司的技术（包括初始源代码）来自美国伊利诺大学（UIUC）超级计算机应用中心（NCSA）相关项目，而该中心的资金来自美国能源部和国防部的公共技术赞助基金，要求所有成果在互联网上公开。

前面提到冯·诺依曼计算机存储架构，不论 Linux 还是 Windows 操作系统的文件地址空间都是指向硬盘。硬盘上的资源无非两类：数据和程序。浏览器通过 URL 直接访问互联网门户网站上的数据文件。webOS、Firefox OS 等通过 URL 直接访问互联网门户网站上的 JavaScript 程序。如果读者看过谷歌 Google I/O 2016 大会的视频，谷歌正在积极推进的“Instant App”模式，点击链接就能立即运行应用，无需程序安装与卸载。

SUN 在 1992 年随着 Java 语言的推出，提出了“网络就是计算机”（The Network is the computer.）的口号。浏览器大概实现了一半，谷歌 Instant App 想实现另一半。二十年前点击 URL 网页链接就能得到无穷无尽的信息。网景当年的梦想今天也已经非常接近实现了，不久的将来，浏览器是未来的操作系统，操作系统就是未来的浏览器，不但能点击运行 JavaScript，能点击运行 Java，也能点击运行 C/C++编译的原生应用。

关于新一代操作系统的思考

名不正，则言不顺；言不顺，则事不成；事不成，则礼乐不兴。

如果不能用发展的眼光看操作系统，就不可能设计新的操作系统；没有新型操作系统设计，自然做不出来新型操作系统；没有新型操作系统，未来的操作系统生态就只能是海市蜃楼。

首先，Windows 是操作系统吗？假设我们从应用编程程序员的角度看操作系统 API（Application Programming Interface），能看到的最底层哪个系统的 API，那个就被称为操作系统。按照这个定义，应用程序员用 DOS API 编程，DOS 就是操作系统。应用程序员用 Windows API 编程，尽管 Windows 运行于 DOS 之上，Windows 就是操作系统。

这里我们强调一下，不考虑向后兼容的应用，操作系统必须能向前编写任意（图灵等价）应用，能在需要时段跑出 CPU 的速度。Win16/32 SDK 的应用编程 API 可以编写任意应用，而无需诉诸 DOS API，因此我们说 Windows 提供了完整的操作系统环境。

2000 年左右，美国 SUN 公司的 Java 语言大有占领世界的势头。他们认为 Java 与 C/C++相比的效率损失在四倍左右，也就是说 4-6 年之后，CPU 速度提升，利用 Java 可以完成 100%的应用程序实现。如此这般，Java 虚拟机提供的 API 就是操作系统 API。至于 Java VM 和系统底层抽象，TCP/IP 等网络协议栈仍旧采用 C/C++语言，这与应用无关，并不影响操作系统的定义。谷歌的 Android 采用 Java 作为母语，微软的 Windows Vista 采用 C#作为母语，两者都和 SUN 的布道脱不了干系。

1992 年 Java 语言诞生伊始，“Write once, run every where!”的理念声彻云霄，但至今未果。二十多年来 Java API 不能实现 100%的应用编程，这大概是不争的事实。Android 的松耦合编程框架已经深入人心，控制了近 80%的手机市场份额。但是 Android 上面的第三方游戏引擎、编解码器、浏览器、Office 等大型应用软件仍旧必须依赖底层 Linux 系统 API 的支撑。

有人说 Android 是 Linux 操作系统的衍生品，尤其是至今这个论点还充斥媒体和专业舆论。毋庸置疑，Linux 架构沿用了 UNIX 架构，40 多年了，成熟稳定 ，并取得伟大的历史性胜利。

Android 出于商业考量选择 Linux 为基石，但并非 Linux 的衍生品，毕竟 Android 也可以选择其他操作系统内核为基石，比如最近 Google 也在尝试自己实现全新的 Fuchsia 系列内核（包括 Magenta 和 LittleKernel）。以发展的眼光看，Android 的缔造者们期望 Android 是操作系统，只是梦想还没真正实现。

1970 年代末期中国主流计算机是 DJS-130，该机源自美国 Data General 公司 1969 年发布的 Nova 机，其操作系统没有 BIOS，启动加载需要人工输入十三条指令（俗称手拨十三条）。后来的 PC、手机加载不同操作系统，如 Linux 或者 Windows，都要通过硬件内置的 BIOS。如今流行的 BIOS 标准是 Intel 主导的开源 UEFI 规范，其功能已经非常强大了。

今天的 Android、Blackberry、webOS、Chrome OS、 Firefox OS 等操作系统的底层都是 Linux 内核。iOS、Mac OS 底层是类似 Linux 的 Unix 系内核。以上多个操作系统的设计理念大相径庭。

Linux 的功能只是负责本地计算机硬件抽象，并不具备任何面向对象（Java 编程），面向服务（SaaS 编程），面向分布式（DCOM、.NET 编程），面向互联网（HTML/JS 编程），面向人工智能（逻辑编程）的理念。

我们认为 Linux 是现代高级 BIOS 的描述并不夸张。

本文中只是简单、非学术地讨论了 Windows 和 Android。并不是说各个时代没有其他操作系统，但相比之下，各个时代里操作系统第一名的市场占有率比第二名起码高很多。

其次，Windows 10 是 Windows 吗？Windows 是微软的注册商标。Windows 10 同时提供两套完全独立的应用编程 API 集合，一套是传统的、基于消息机制（Polling）的 Win32 编程 API，另一套是基于注册/回调的推送机制（Pushing）的 WinRT 编程 API。微软应用商店里的应用都是基于 WinRT 实现的应用，目前微软的应用商店中并不未涵盖著名的 Office 等大型软件应用。

撇开 Windows 商标，Windows 10 与 Windows XP 根本不是同一类（或者说同一代）技术。废弃 Win32 消息机制应用编程，启用全新的 WinRT API 应用编程的好处是什么？据微软说，未来采用统一的 Windows 10 操作系统可以支持手机、平板、电脑、电视、游戏机、服务器、工控嵌入式设备、眼镜、机器人等智能终端。微软还说，Windows 10 的未来不会有 Windows 11，这一款操作系统就是未来。

过去手机是 WinCE，桌面是 WinXP，服务器是 Win Server，游戏机是 Windows for XBox，工控机是 XP Embedded，微软一个公司同时做了五个操作系统发行版。今天的苹果的手机、平板是 iOS，便携电脑和台式机运行 MacOS。谷歌有 Android 和 Chrome OS。Linux 的主要发行版有 Ubuntu、Redhat、Debian 等。用过去的技术，大家都不能统一操作系统发行版，为什么现在 Windows 10 就能做到？

起码我们隐约能感觉到，新一代操作系统可能真的要来了，但关键肯定不在于 Windows 10 能在电脑上运行 Office。

以 HoloLens 眼镜为例，眼睛的 CPU 能力受到电源限制，眼睛的硬盘受到重量限制，眼镜的智能要对接云计算，眼镜还可能需要与周边的 IoT 设备进行动态交互。

Windows 10 的机器人项目代号是 Bamboo。机器人需要动态插拔“胳膊”、“大腿”；人工智能需要动态切换各种智能算法；VR/AR 需要有云（互联网）、雾（周边环境）、端（各种穿戴设备）的协同操作，传统操作系统（比如 Linux）的局限性越来越明显，新一代操作系统的痛点也就明确了。

我们认为统一应用平台的核心难点是 AR 眼镜和机器人，同时兼顾服务器、PC 机、手机、游戏机等传统计算机。

操作系统及互联网安全架构顶层设计

我们知道传统计算机是由 CPU 主板、硬盘、外设、I/O 终端（包括屏幕/键盘/鼠标）等四部分组成，如今的通用操作系统就是协调这四大件的应用软件运行平台。

我们特别指出常见的操作系统的安全机制只限于单机（本地），虽然操作系统提供网络接口，但并不负责网络安全。我们还注意到传统外设其实是自带嵌入式操作系统的不能独立上网的智能设备。随着 CPU 造价越来越低廉，目前计算机外设其实也是由 CPU、内存、永久存储组成的计算机，只是因为人

机交互及网络安全等原因被赋予了能力受限的角色，这个认识有助于理解 AR/VR 眼镜、机器人、智能家居安全问题及解决对策。

四种角色——任何 IoT 设备，手机、平板、眼镜等智能终端，机器人的胳膊、大腿、心脏、大脑的硬件，都无时无刻不在扮演这四种角色中的某一种或者二、三、四种角色。传统通用操作系统在一台计算机中协调四种角色，未来的操作系统要在去中心的分布式环境里，动态协调不同分立设备扮演好这四种角色。

举个例子，当我们介绍传统的浏览器，我们一定会强调互联网的另一端还有门户网站，中间要通过 HTTP、HTTPS 之类的网络协议连接。仔细一点的读者还会注意到浏览器可以离线访问本地硬盘上的 HTML 内容。四十年前传统电脑的数据文件和应用程序都存放在本地硬盘上。二十年前浏览器可以方便地访问互联网上的数据，程序仍然只能预装在本地。

如果我们把网站的内容全部存储在云盘里，浏览器是否稍加改良，仍旧可以继续工作呢？网站演变为云盘；浏览器演变为终端；HTTP/HTTPS 等网络协议演变为云盘计算机内部的 PCI 总线；CPU 主板的负载由云盘 CPU 与终端 CPU 分摊；外设终端周边 IoT 设备作为本地终端及云盘服务器上运行的 Web 服务。不但数据可以放在互联网云盘上，程序/软件服务也可以放在互联网云盘上。

以上云盘计算机顶层设计的好处是什么？务虚来说，这就是 25 年前人们说的“网络就是计算机——The Network is the Computer。” 务实来说，网络变为云盘计算机内部总线，所有网络数据包由操作系统代为收发。

想象移动运营商是一台网络计算机，世界上可以有大大小小许多虚拟运营商。每只手机或者虚拟手机里面只跑一个应用或者服务，并且不管应用还是服务都不允许直接访问运营商网络（不许直接读写 IP 包）。应用与应用，应用与服务，服务与服务之间的寻址都是通过某种 UUID （Universal Unique ID），比如手机号码。所有通讯数据包都由运营商代为传递。

容器方兴未艾，Android 作为容器操作系统，因为没有解决原生代码 JNI 的事情，容器底层漏了。作为轻量级容器内运行的操作系统 CoreOS 虽然轻巧，等仍旧采用 Sockets 通讯，容器侧面漏了。

假设完美的互联网网络没有漏洞，仍然不可避免第三方应用/服务发起 DDoS 攻击。假设理想的 Linux 操作系统没有漏洞，仍然不可避免后台服务/伪驱动监听鼠标、键盘并泄露用户隐私。这是由于互联网协议和操作系统顶层架构导致的问题。

面对当今泛滥的网络安全问题，大量开源代码良莠不齐，众多 IoT 设备真假难辨，旧酒囊装新酒的政策，补丁摞补丁，只是权宜之计，无法根本保障互联网安全。

新一代操作系统利用程序元数据（Metadata）实现的反射（Reflection）技术，可以将应用与服务（App/Service）包在容器里，所有通讯、安全等正交问题交给新型操作系统来运营。新一代操作系统不允许应用开发后台 Daemons，除非为了兼容 Linux，不建议使用 libc 库，也不建议使用 socket 及 TCP/IP，规避原生代码漏洞，不留病毒赖以生存的死角。

新一代操作系统底层也许仍旧采用 TCP/IP，不同点在于网络协议演变为网络操作系统内部总线。只许州官放火，不许百姓点灯——只许操作系统发包/收包，应用及服务都不许染指网络、安全等事宜。

单机硬盘是网络云盘的特例，可以通过 Localhost（127.0.0.1）这个互联网预留给本地计算机的网址来访问。网络云盘则不是单机硬盘的特例。新一代操作系统模型不但可以在云盘（网络）计算机上运行，也可以在单机里运行，统一了计算机本地应用与云计算应用的编程及运行。

曾经的一台计算机上运行一个操作系统，未来一台计算机上运行一个物理操作系统（Host OS）和一到多个虚拟容器操作系统（Guest OS）。其中有一个特殊的 Localhost 虚拟机容器，兼容运行目前 Android 应用。其它虚拟机容器按需启动，每个环境中运行一个云盘映射到本地终端的受限 Android 应用或者服务。应用与应用，应用与服务，服务与服务之间的通讯由新型去中心化的分布式网络操作系统平台自动生成。

移动终端上运行的通用计算虚拟机环境替换浏览器里运行的 JavaScript 计算环境，隐藏网络协议的云盘替换暴露网络协议的门户网站，这就是未来安全移动互联网的顶层设计。

新一代操作系统编程语言

1970 年代初期，随着计算机编程规模的增大，人们开始意识到软件危机来临——有时解决一个 BUG 会伴随产生更多 BUGs，导致软件工程无法按时完成。

人们开始认识到：滥用 GOTO 语句会导致程序可维护性变差；数据变量按照结构化（Structure）分组会使程序易懂；规范 if-then-else，while、for 等编程模型会程序逻辑流畅。经过几年的争论与沉淀，结构化程序设计的思想开始流行。1970 年代中期，Pascal 语言产生于欧洲，流行于世界各大高校教学。C 语言诞生于美国贝尔实验室，用于重新用高级语言实现 UNIX 操作系统，并取得巨大成

功。1980 年代初期，TCP/IP 协议也在 UNIX 之上得以用 C 语言实现。C 语言至今仍旧是系统软件编程的首选。

1980 年施乐（Xerox）发布了 Smalltalk-80 面向对象编程语言。1983 年兼容 C 语言的面向对象 C++语言在贝尔实验室面世，由于 UNIX 的影响力，C++语言在 1980 年代中期开始流行。面向对象的原则是数据结构进行封装并规范一组可以对该数据结构进行操作的函数集合。

语言的目的是为了表达思想。1986 年美国伊利诺大学（UIUC）首次尝试用 C++设计一款名为 Choices 的操作系统，并得到世界范围学术界的瞩目。人们开始激烈辩论，用面向对象语言实现的操作系统是否可以更好地帮助应用程序员用面向对象的思想来分析并解决问题？用结构化的 C 语言实现的操作系统是否也可以帮助应用程序员用面向对象的思想来分析并解决问题？当然，面向对象的目的不外乎方便应用编程，减少 BUGs，易于维护代码。

1990 年左右，面向构件（Component）的 CORBA 及微软 Windows 支持的 COM（Component Object Model）开始进入市场。构件是二进制模块化封装的对象，而非基于源代码封装的对象。构件产生的原动力来自操作系统需要支持软件模块独立升级，而无需重新编译不相关的软件构件代码，比如不同版本的 Windows 打补丁时只需要升级修改过的动态链接库（DLL）。这点对于大规模、超大规模软件开发商来说尤为重要。在 1990 年代，所有微软内部用 C/C++语言开发的工程都必须采用 COM 的规范编程，IBM 内部采用 SOM 规范编程。与此同时，广泛流行于 Windows 社区的的面向对象快速应用编程 MFC 框架，在微软内部被明令禁止使用。这也印证了大型系统软件编程与小型应用软件编程的侧重点有本质不同。

2000 年新世纪伊始，随着 Java 语言的流行，Metadata（也称为 Class Information）在编程语言中的作用越来越被人理解。面向服务（Service Oriented）就是给构件配备 Metadata 的编程模型。简单的鉴别方法就是 C/C++程序编译后的目标代码模块叫 Object 文件（用.o 或者.obj 作为后缀），Java 或者 C#程序编译后的目标代码模块叫 Class 文件（用.cls 或者.class 作为后缀）。操作系统根据软件代码模块中的 Metadata 来自动实现跨语言、跨运行环境的执行代码生成，使各种语言之间自动适配，使应用与服务之间无远弗届。

2010 年以后，由于 Java 或者 C#并未能实现百分之百应用的目标，高效的原生（Native）语言也需要沿着软件即服务的理念演进。谷歌的 Go 语言，苹果的 Swift 语言，微软修改过的 C++（MSVC）都在原生语言的基础上引入了 Metadata 的支持。再重复一遍：语言的目的是为了表达思想。我们不但要关注这些新语言的语法特性，我们更要理解新语言的核心思想。

我们知道，操作系统的重要作用就是支撑应用程序运行。不同语言编译出来的程序需要相应的运行平台支撑。伴随 C/C++语言而来的支撑环境是 UNIX；Java 语言的支撑环境是 JVM；JavaScript 语言的最初支撑环境是浏览器。我们不妨认为 JVM 就是 Java 程序的操作系统，浏览器就是 JavaScript 的操作系统。

随着 2010 年面向服务的语言的目标代码回归原生属性，让不同语言自动互通互联，让应用自动适配远在天边近在眼前的服务，新一代原生操作系统必然应运而生。

新一代操作系统应用模型特点

一般人的大脑存储大约五万行代码。也就是说，一个职业的程序员，工作 2-3 年，大概能对五万行代码了如指掌。不是说他能把每行程序都背出来，而是说一旦问题发生，他基本能马上直觉地猜到问题定位。最好的程序员据说能记得十万行代码，这也就是个经验值。产品经理不能指望程序员更多了，毕竟大脑也是存储，装了新代码，老代码就删除了。

1970 年代人们记得五万行机器指令汇编，就能做相应那么多功能。1980 年代人们记得五万行 C 代码，能做的事情就多了许多。1990 年代记住五万行 JavaScript 的人比记住五万行 Java 的人干的事情粗不少，但规模大许多。到 2000 年就有 Python 了，蟒蛇大概是蛇类最大个头的啦。

由于软件产业近四十年来的飞速发展，创造了太多昼夜暴富的传奇。更多人把更多眼光投向市场，资本渐渐淡忘并忽略了前面提到的十年一个脚印，默默耕耘的人们。

1970 年代人们第一次意识到软件危机的恐慌，至今流传着“人月神话”（Men-Month）的迷思。同是那个十年，集成电路技术使计算机硬件日新月异，速度更快，效率更高，功能更强。不难相信，也就是那个时代，人们提出了”软件集成电路“（Software IC）的概念，开始了面向对象的追求。

构件就是二进制代码模块；服务就是能在运行时动态自动拼装的构件；能动态拼装的软件服务就是四十年前人们梦寐以求的软件集成电路。十个用脚本语言动态拼装的软件模块能完成多少功能？百个动态拼装的模块呢？

脚本语言拼装软件服务应该是新型操作系统运行平台的特色之一，只有原生语言作为母语的操作系统显然不能胜任了。

五万行代码编写的软件服务构件意味着什么？五万行 C/C++写的软件服务可以高效、精准，比如游戏引擎。五万行 Java 写的软件服务便于维护，功能强大。

原生语言的回归也应该是新型操作系统运行平台的特色之二，只有 Java 语言作为母语的操作系统显得有些英雄迟暮。

我们知道有的操作系统只能用 JavaScript 语言开发应用，有的只能用 Java 语言开发应用，对于特定场景，特定公司的产品，这并无可厚非，甚至简单就是美。但对于产业或者互联网生态来说，一种语言，一个框架，以静止的眼光看世界，其局限性显而易见。

新一代操作系统必须使用系统原生语言实现，比如用 C/C++语言实现。比如 Elastos 具备特有的 400 左右（有限集合）系统调用 API，提供类似安卓 Java 的 C++类库。通用操作系统应该支持多种语言，多种框架，尤其是未来的语言和未来的框架。Windows 10 的 WinRT 编程底层系统 API 的个数有待考证。

新一代操作系统原生代码框架支持跨越互联网的分布式 WebService 调用，即新一代操作系统支持手机应用直接启动或寻找并使用大云、家庭云、周边（普适计算）设备上的服务，而无需了解底层网络协议，便于运营商进行网络优化及防止第三方应用实施网络攻击。

谷歌在 Google I/O 2016 大会上展示的“Instant App”。App 的链接图标（Icon）可以在社交网络中推送，安全无虞，用户不用事先安装，直接“点击运行”。这应该是新型操作系统运行平台的特色之三。

微软 Windows 10 鼓吹的“通用应用”（Universal App）及 Universal Windows Platform (UWP），用一个统一的软件平台支撑使多种屏幕、不同场景的终端，比如：手机、平板、电脑、电视、游戏机、眼镜、机器人等。不再刻意区分原生代码（如 ARM、X86）硬件，不再强调按照键盘、触摸、遥控器、手势等划分终端。应用点击运行应该是新型操作系统运行平台的特色之四。

既然点击运行可以实现，云盘映射到虚拟计算环境的图标点击运行只是特例。如果把个人云盘里建立过“快捷方式”（Short cut）的应用二进制执行代码事先缓存下来，就可以实现这些应用的脱网运行，即可以在飞机上、地铁里直接启动，无需每次诉诸网络。原生代码虚拟机容器里的应用运盘化应该是新型操作系统运行平台的特色之五。

由于移动终端与云盘的协同、统一，避免写死代码（Hard Code）绑定网络协议，操作系统可以在运行时根据应用程序特点，根据物联网 IoT 设备的配置，根据播放多媒体数据类型，动态选择最优化的

网络协议，比如从云盘下载文件与网络电影视频流采用不同的 P2P 网络协议，或者根据 IoT 设备的描述采用高通倡导的开源 AllJoyn 或者苹果把持 HomeKit 等物联网协议。因此，未来应用与服务运营化是新型操作系统运行平台的特色之六。

如上所述，新型操作系统的特色似乎非常科幻。以我们对现代编程技术的理解，以上目标可以划分阶段，逐步实施。假设以上架构能实现，不论应用还是服务都无法发起网络攻击，无法窃取他人隐私，无法传播病毒。计算环境与网络正交，计算环境与安全正交，编程更为简单。

所谓“大象无形，大乐稀声”，隐藏网络的去中心化计算环境～看不见网络就无法攻击网络，看不见的安全就最安全。

Elastos 云雾端统一应用平台

The best way to predict the future is to invent it. Alan Kay (1971) —— 预见未来的最好方法就是创造未来。Alan Kay 于 1970 年加入施乐（Xerox）公司的 PARC 研究中心，他是 Xerox 发明面向对象语言 Smalltalk 及视窗图形界面的灵魂人物。Alan Kay 也由此获得计算机领域最高荣誉——图灵奖。

Elastos 是中国团队正向研发的原创通用操作系统。Elastos 的名字隐含了弹性计算（或者云计算）的含义，其目的是创造移动终端上的虚拟容器内计算环境，并使该计算环境的永久存储与云盘对接，弹性云服务可以运行于互联网云端，也可以运行于移动终端。Elastos 的名字还隐含了终极操作系统的含义，希望其 400 多个系统 API 集合可以相对稳定，不会因为计算机终端用途不同，而分裂成不同的发行版。

最新的 Elastos 第三版从 2013 年 5 月开始产品化迭代周期，至今已经接近 Beta 版的水平，运行于 Moto X（XT1085）手机及 Lamobo-R1S 智能路由器之上。Elastos Framework 的 C++代码规模已达 300 万行以上，工程量巨大，开发团队辛苦了！Elastos 第三版从研发期伊始，即 2012 年 11 月 1 日，就决定采用 Apache 协议全部开源，参见 elastos.org 及 GitHub.com。

Elastos 兼容安卓应用，把两百多万行安卓 Java 框架（Framework）代码翻译成 C++，目的是让 C++、Java、JS 三类语言自动适配，无缝相互调用，无需人工编写 JNI 代码，实现一次编写各处运行。Elastos 是新一代所谓通用应用（Universal App）操作系统。

Elastos 的四个特色：

- 用 C++语言仿真了几乎全部 Android Java 编程 API，效率更高，可以更方便完成工控机软实时的任务；
- JavaScript、Java 还是 C/C++语言写的应用程序模块相互调用，无需手工编写 JNI，真正做到“Write once, run everywhere”，这也就是类似 Windows 10 推出的 Universal App 了。
- Elastos 建议应用避免应用直接调用 Linux 及 TCP/IP 的 API 来实现后台乃至远程服务，从而使第三方应用注入木马病毒或实施网络攻击难以得逞；
- 利用 Elastos 实现“云雾端”三位一体融合，把物联网 IoT 设备映射为移动终端直接访问的 Web Services，从而禁止 IoT 设备直接通过互联网泄露用户隐私或者实施 DDoS 等网络攻击。

Elastos 手机、平板、电视等的一个应用场景是作为 Elastos Hub（家庭云服务器）的远程 I/O 终端。但与经典远程终端不同的是它们可以脱网运行，比如在地铁里、飞机上等网络信号不好的情形下，仍然具有良好的用户体验。Elastos Hub 可以自带硬盘作为手机等移动终端的个人云盘，将来也可以采用 OneDrive 等网络云盘作为个人云盘。智能家居物联网设备作为 Elastos Hub 的禁止上网的外设。

看到这里，读者可能要问是否 Elastos Phone 与 Elastos Hub 必须配对运行？答案是不需要。如果为了兼容安卓，并认为安卓的安全机制及预防网络攻击、保护隐私泄露已经足够好，读者只需把 Elastos OS 作为 C++版的安卓就可以。

Elastos 不是 Linux，Elastos 也不是 Android。对吗？曾经网页数据在互联网上流动，深刻改变了我们的生活。软件程序将在互联网上流动，我们可以期待什么呢？不论如何，为了运行互联网上流动的程序，我们需要新一代的操作系统，比如 Elastos。

Elastos 的商业机会本来是想为天下先，可我们太弱小了，团队执行力、技术底蕴都受到诸多挑战，贻误了不少机会。随着中国电子业基础工业的崛起，软件的重要性越来越明显。随着 Android 的成熟与开源，为我们提供了足够多的生态应用 App 及开发素材。微软的 Windows 10 后发先至，开始推动新一代 UWP 操作系统的宣传，这对 Elastos 有利，毕竟市场还需要教育。

开源社区和中国市场都不可能一家独大，就像 1990 年代初期的 Linux，弱小不可怕，活着就有机会。最核心的一点就是六分之一的成本。施乐当年的视窗工作站售价是 PC 的六倍，尽管先知先觉，但也无可奈何地失去了市场。

新一代操作系统为什么必须开源

天行健，君子以自强不息。地势坤，君子以厚德载物。（源自《易经》乾卦的卦辞）

操作系统作为产品内部的固有组成部分时，比如路由器里面的 Linux 作为其嵌入式操作系统，用户购买产品时一并包含了该操作系统。操作系统的产品级优化是产品核心竞争力的一部分，读者很容易理解，一个路由器公司不愿意将其开源，给竞争对手以把柄，以其人之道反制其人之身。产品是残酷竞争的、阳刚的、利己的、你死我活的——这就是商场上面的所谓：天行健，君子以自强不息。

操作系统作为孕育生态环境的通用操作系统支撑平台时，比如 Linux，比如 Android，其目的主要是孵化更多第三方产品茁壮成长，纵然第三方产品间可能相互竞争，但作为生态平台的操作系统需要采取中国古代建极绥猷的无为治国方略。（注：“建极绥猷”是太和殿匾额，意为：欲建盛世，则需顺势而为；“无为”是交泰殿匾额，皇上放玉玺的地方，提醒其切忌参与竞争。）一句大白话总结：公司卖产品，政府靠税收。生态系统是制定规则的、阴柔的、利他的，抚育合作伙伴成长——这就是商场上面的所谓：地势坤，君子以厚德载物。

1990 年代初期的 Linux，不论从代码规模还是技术优势，都不是当时的 Windows 的对手，但由于其开源策略迎合了生态环境规律，在 IBM、Intel、Google 等国际巨头的支持下，慢慢赢得了市场的认可。

读者可能会问，微软的 Windows，苹果的 Mac OS、iOS 并未开源，他们不也都给予第三方合作厂商非常巨大的生态发展空间吗？据微软自己鼓吹的资料，Windows 每挣一美元，其生态圈内的合作伙伴会挣到八美元，由此形成正反馈，带动了微软在 PC 领域的垄断。苹果也说过，App Store 每挣一美元，其合作伙伴会得到更多的利益。这都说明生态操作系统是“利他”的，但没有说明必须开源。

仔细想想，过去的通用操作系统是为单台计算机设计的，PC 机上的应用不能跑在苹果机上，反之亦然。PC 机上的两大阵营分别是 Windows 和 Mac OS。手机上的 Android 和 iOS 是两大阵营。各个生态之间还是竞争关系。

开源的 Linux 曾经想竞争 PC 桌面，但失败了。失之东隅，收之桑榆。Linux 在互联网云服务器领域及手机移动终端的底层物理机（Host OS）操作系统取得了巨大成功。究其原因，互联网云服务器及手机移动终端与云服务器需要互通互联，Linux 及 TCP/IP 开源便于其他公司参与互联网协议的互通互联，共同营造世界范围的互联网生态，而非狭义的某个操作系统生态。

前面提到，互联网（Internet）、万维网（WWW）超越了某个公司的生死，促进了人类共同的科技繁荣与进步。Linux 失去的是单机操作系统市场，赢得了互联网大经济圈的市场。Linux 的开源有意无意暗合了历时发展规律。

Android 的开源本来是为了后发制人，干的是搅局的勾当。但亚马逊把 Android 拿去做了许多和有创意的产品，比如电子书、Echo 家庭语音助理等。阿里云 OS、华为手机 OS、小米 OS、乐视 OS 大概都或多或少受益于 Android 的开源。目前 MTK 已经开始生产 Android 路由器芯片。也有人开始研究 Android 服务器（俗称 Headless Android）。由于 Android 的开源，我们可以预料其有机会慢慢走出单机操作系统的囹圄，开辟一片互通互联的新天地。

AT&T 公司的贝尔实验室发明 UNIX 和 C/C++语言；Xerox 公司的 PARC 实验室发明面视窗技术和 Ethernet 网络技术；UCLA 大学创造了早期的互联网技术；伊利诺（UIUC）大学创造了万维网（WWW）技术。这些公司与科研机构自己阴差阳错没有利用其技术发财，而是奉献给了人类文明。

未来的操作系统一定是云-雾-端三位一体运营，凌驾于传统的物理单机节点与物理互联网组成的传统互联网之上，应用一次编写到处运行。如果我们没有开放心态与合作精神，我们不可能营造自己的新一代操作系统的生态。

我们相信，开源云-雾-端互联互通操作系统生态圈大概为数不会太多。Android 兼容很重要，否则很难找到应用开发商愿意为新的操作系统环境开发应用，鸡生蛋、蛋生鸡的悖论就破不了局。仅有 Android 兼容也不够，没有云-雾-端互联互通的大格局，就不能摆脱 Android 的阴影。

新一代操作系统的商业模式

传统的开源操作系统通常只有以下三种模式。

开源操作系统的第一个生存模式是“傍大款”。比如 Android 傍上 Google，作为其公司战略的一个有机组成部分，Android 不挣钱，但为其主营搜索业务带来巨大流量。这些年来，多次濒临生死抉择的教训告诉我们，开发 Elastos 操作系统的难点不在技术，尽管其背后的技术还是有非常多让我们值得骄傲的地方。没有类似谷歌、阿里之流的大型生态环境背景可以依托，Elastos 时刻面临旦夕祸福。同是神一样的 Andy Rubin，当年 Android 的前身 Danger 没有谷歌时也给他做死了。

开源操作系统的第二个生存模式是“非盈利基金会”。世界上比较著名操作系统基金会有 Linux Foundation；Firefox OS 背后的 Mozilla Foundation；ARM CPU 及 Android 背后的 Linaro 非盈

利技术服务公司等。这些非盈利组织背后由许多大公司以会费及赞助费的形势提供资金支持。中国半官方的 TD 产业联盟（TDIA）支持了中国自主产权的 4G 无线通讯技术 TD-SCDMA 的发展。虽然国家中长期规划也对一些操作系统厂家予以数年的支持，但相比之下，新一代操作系统发展更需要像非盈利组织那样专注及持之以恒。

开源操作系统的第三个生存模式是“技术咨询及外包”。伴随着开源 Linux 的成长，Redhat 作为商业公司为其提供技术及咨询外包服务。Ubuntu 背后有 Canonical 作为服务公司。Android 开源社区在中国有中科创达作为商业技术及咨询外包服务公司。中国最近二十年的发展，尤其是互联网社交生态圈的蓬勃发展，移动终端设备及物联网 IoT 设备制造的工业基础建立，对于类似 Elastos 等以网络安全为宗旨的操作系统来说，生存空间总体来看越来越好。

由于未来的新一代操作系统面向虚拟机及互联网，我们大概可以预见以下两种新的盈利模式。

新一代开源操作系统的第四个生存模式是“网络优化及代运营”。新一代操作系统会是多种终端（手机、电脑、游戏机、眼镜、机器人等）的统一应用（Universal Apps）平台，并且基于终端到云端协同管理，就像把传统的局域网需要专职管理员，未来操作系统需要根据云端部署情况不断优化网络，一定会需要 24x7 的运营管理人员。虽然终端及云端部分的操作系统代码都开放了，但外包部分运营给专业操作系统开发工程师，可以让合作伙伴在确保隐私不被泄露的前提下，提高工作效率。

以 Elastos 为例，我们不但开放了全部 Elastos 操作系统代码，我们还实现并开放了 Elastos 端到端的 P2P 智能家居网络代码（就我们所知，这也是目前唯一的完整开源 P2P 网络代码）。两者加起来总有超过一千万行代码。毕竟不是每个公司都有谷歌、微软、亚马逊那样的运营实力，如果一些小点规模的也想根据开源代码运营自己的特殊社交人群，部分外包网络运营更为现实。

新一代开源操作系统的第五个生存模式是“发行生态环境的虚拟货币”。趁着区块链大潮，做一个去中心化的数字资产物流公司。今天的区块链技术只讲金流如何可信，但几乎没人讲得清楚如何保障与金流呼应的虚拟数字物流安全。我们认为只有以新一代操作系统为技术保障，把握金流可信、物流安全的机遇，才能打造新一代互联网生态系统。

去中心化的区块链技术要求代码必须开源，因为区块链技术的本质特征就是它不把信任和信用的基础建立在对人和对机构的信任上，大家只信数学，只信算法，所以数学和算法就是信任的基石，如果这部分不公开，那就失去了跟全社会最有能力挑战它的高人过招并且成功过关的机会。如果这样，区块链的安全性就是值得怀疑的，至少是打了折扣的。

互联网是自然的去中心化网络。保障未来互联网网络安全，个人数据隐私的安全的新型操作系统必须开源。没有收益保障的正反馈奖励机制，生态圈也很难形成气候。比特币自 2008 年开始运转，以太坊社区的以太币自 2015 年开始运营，这两个完全开源的生态环境都成功地在世界范围发行了十亿美元以上的虚拟代币。

数字虚拟代币交易市场兴起，使我们看到了一种全新的开源操作系统生态环境的商务模式。

结束语

网络坍缩为分布式虚拟计算机系统的内部总线，由新型操作系统接管，意味着本地和异地的程序性资源（应用及服务）也和万维网（WWW）统一数据性资源（多媒体文件）一样“无差别化”了。

二十多年前浏览器带给人类的变革远远超出技术的范畴。未来操作系统与浏览器融合，只会更广泛、更深刻地影响人类生活的方方面面。