Singularity：重新思考软件堆栈

这篇文章是由微软研究院的研究员所著，介绍了他们的研究成果，一个全新的系统架构和操作系统：Singularity。

从一开始，Singularity 就是受到以下问题的驱动：如果一开始就以可靠性为主要目标而全新设计一个软件平台，那这个平台将会是什么样子的呢？微软研究院设立这个项目的目的就是为了设计一个以可靠性为主要目标的平台，该系统的目标是为了建立更为健壮和可靠的软件平台。Singularity 展示了使系统更健壮和可靠的新技术以及体系结构决策的实用性。

Singularity 的一个起始点就是：如果从底层开始就是一个安全的 managed 语言构建系统，我们能删除很多抽象层和迷之策略，managed 或者 GC 并不意味着又一层 overhead, 反而，你会得到更简单，更安全可靠，也更快的软件系统。首先，你不需要很复杂的虚存管理，所有应用可以跑在一个地址空间上，由类型系统和运行时提供隔离。它也意味着上下文切换的成本非常低，你可以跟 erlang 一样做超级多并发轻进程。单一地址空间和 GC 可以让你很放心地零拷贝在进程间或与内核/驱动传递数据 —— 传统系统里很多的 CPU 都耗在 copy one buffer to another 和为之服务的 encode / decode 上，在这里你都不需要了。就像 Joe Duffy 在 blog 里说的，这样做出来的系统很多时候比传统的所谓 native 系统要快。

大多数大型操作系统的思想都源自于 Multics，那时代的机器都在非常低速（100 kHz）和内存非常有限（10 KB）的情况下运行的，那时大多数的程序都通过汇编语言来编写，那时仅通过不同用户间的分时操作来达到文件的安全保护目的，那时网络还没有出现，那时的用户都是有明确的目的且训练有素的。而 Singularity 是基于当前世界的实际情况，具有快速运算的能力，大量内存的机器，高级的编程语言，久经考验的技术，及无知的用户等的目标而进行设计的系统。而 Singularity 就是为了解决这些问题，用来满足下一代用户以及开发者需求。

如论文里所提到的，“通道就是有两个端点间的双边信息管道。通道提供无损的，有序的信息队列”。 软件独立进程（SIP）之间使用这些通道进行通信，这与直接共享内存或其它普通的方式非常不同。一旦信息发出，SIP 就完全失去了控制权。所有使用这些通道的通信都必须遵循“通道契约”，该契约与 Web Services 非常相似，允

许系统更高层次的通信验证和使用并行计算（pi-calculus）的形式。并行计算和一致性检验提供了严密的基础，在此基础上更能去搭建实用的，有发展前景的系统。

Singularity 用到了许多“新”的编程工具——所谓“新”这个字需要琢磨一下，因为实际上他们已经存在好一段时间了，只是脱离了主流或者是操作系统的领域——其中很大部分与 Sing#语言有关。Sing#是另一个非常出名的 MSR 项目 Spec#的派生项目，它是一个基于规范的语言，该语言允许定义前后条件以及对象的不变量，允许通过法则交验器（Theorem Prover，boogie）来检验系统的状态。状态检验（Static verification）可以查找多种通常的程序逻辑错误，如不恰当的使用一个在编译时而不是运行时的方法。因外界不断的进化要求我们去维护日渐复杂的系统，许多人相信状态检验会变成未来几年的主流。