Singularity

这篇文章主要介绍了sigularity项目。每一个操作系统都是一系列的设计理念的体现。操作系统经历发展以后，系统中普通存在缺陷。Singularity采用了全新的编程语言，新的软件验证工具，采用了基于软件独立进程的软件体系结构。Singularity具有三个关键性的体系结构特征,1软件独立进程(SIP)，2基于Contract的信道，3基于Manifest的程序。

软件独立进程提供一个免于外界干扰的程序执行环境；基于Contract的信道使得进程间进行快速、可验证的消息通信成为可能;基于Manifest的程序则对软件独立进程所要运行的代码进行精确的定义并且限定了它的所有行为。

SlP是一个系统资源容器，提供程序执行的环境。但是与传统的进程不同的是，SIP充分地利用了现代编程语言的类型和内存安全性检查特征，极大地降低了隔离安全代码的开销。每个SIP的执行过程中都会有一个安全标识，该标识会和系统的安全特征相关联;同时，它还承担着错误隔离的功能。SlP之间不会共享数据，因此所有的SlP之间的通信都必须通过信道(channe!)的消息交换来完成。singularity使用了严格的、静态的、可验证的消息规范。SlP是一个密封的代码空间。作为一个密封的进程，SIP不能为自身动态加载或生成代码。因此必须为操作系统和应用程序采取一种扩展结构:扩展代码在一个新的进程中执行，与它的宿主SlP独立开来。密封进程提供了很多优势，可以提升用于静态检测缺陷的程序分析工具的能力，可以提供更强壮的安全结构(例如可以通过代码内容来识别一个进程)，也不需要在运行期执行环境中重复系统的访问控制。SIP之间的隔离依赖于编程语言的类型和内存安全检查，而不是用于内存管理的硬件。通过结合静态验证和运行期检查，Singularity确保某个SIP中的用户代码不会访问该SIP之外的内存区域。使用软件而不是硬件保护机制提供的进程间隔离，使得多个SlP可以驻留在同一个物理空间或虚拟地址空间。在一个最简单的Singularity实现里，内核和所有的SIP都共享一个单独的内核模式地址空间，这样做可以大大降低进程切换的开销。与硬件隔离机制下的进程通信开销相比，SIP之间的通信开销要小得多，另外，进程的创建与销毁开销也相应得到了减小。在软件隔离模式下，创建一个SIP并不需要创建相应的页表，不需要清空TLB。

信道是一个具有两个端点的双向信息通道。信道能够提供一个无损的、有序的消息队列。客户端可以为驱动提供多个缓冲区。

Singularity第三个基础性结构特征是其基于Manifest 的程序(MBP)。MBP是一种通过静态Manifest进行定义的程序，任何一个在Singularity上运行的程序都必须具备一个Manifest。用户在执行一个程序时，实际上是调用其相应的Manifest，而不是像其他系统一样去执行一个文件。