第33卷 第18期 计 算 机 工 程 2007年9月
Vol.33 No.18 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)18—0093—02 文献标识码：A 中图分类号：TP391

基于 FreeBSD 内核的 Chamber 技术

张 巍，卢 凯
(国防科技大学计算机学院，长沙 410073)

摘 要：服务质量支持技术和安全隔离技术是当前服务器操作系统研究领域的热点，也是服务器之间竞争的关键技术。该文分析了一些重要的资源控制技术，并结合这些技术提出了在 FreeBSD 操作系统中的 QoS 支持和安全隔离技术——FreeBSD Chamber(FC)机制。FC 机制包括了上层的虚拟执行环境和底层的资源管理框架，提供完全隔离的执行环境和系统服务质量支持。

关键词：操作系统；服务质量；资源容器；资源控制

Chamber Technology Based on FreeBSD Kernel

ZHANG Wei, LU Kai
(School of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】 Quality of service(QoS) supporting technology and security isolation technology are both hotspots in the research fields of server OS, which are also crucial for the servers' competition. This paper discusses some important resource control technologies, and proposes a FreeBSD chamber(FC) mechanism—the QoS supporting and security isolation solution for the FreeBSD OS. The FC mechanism, which includes virtual execution environment in the superstratum and resource management framework in the downstratum, can provide total isolating execution environment and QoS support.

【Key words】 operating system(OS); quality of service(QoS); resource container; resource control

1 与资源管理相关的技术

1.1 资源容器

资源容器[1]包含了一个进程或一组进程以处理一个服务类需要的所有系统资源。该服务类的所有用户态和内核态的进程都归入相应的资源容器，并按这个容器的优先级进行调度。而与之相关联的资源容器则作为资源代理，代表该服务类同其他代理竞争系统资源。资源容器机制只是让资源分配到正确的资源代理上，并不会改变资源本身。要实现区分的QoS，还必须和适当的资源管理策略结合使用。

文献[2,3]虽然分别对资源容器理论做了不同程度的尝试，但其实践还处在对少数几种资源的管理调度范围内。目前，资源容器理论尚未得到完整实现，还没有真正可以投入实用的资源容器框架。

1.2 Resource Control Lists

Resource Control Lists(RCLs)[4]的思想来源于对现代操作系统的思考：在现今大多数系统中，各种系统资源、特殊文件都处于一些访问控制的管理之下，那么CPU时间或网络带宽也应当能处于类似的控制之下。

RCLs 描述了“谁”(资源代理)可以得到“什么”以及“多少”资源，是资源控制和资源分配之间的一座桥梁，其结构如表 1 所示。

表 1 RCLs 资源管理结构

events	threshold	action taken / access allowed

RCLs 通常在操作系统的资源分离点如系统调用入口处指定，并且可随时间变化。因此，可以被用来捕获资源请求的变化。RCLs 与资源代理的绑定也可以随着进程计算行为的变化而动态变化。系统使用事件来捕获这种变化。当一个事件产生而且其值大于极限值时，可根据 RCL 的 action/access allowed 域作出响应。如果事件是由于对资源的访问产生的，那么该访问将被限制；如果事件是由于其他原因产生的，那么根据实际情况下一步可以：(1)使新的资源代理关联到资源预约；(2)改变现有资源预约的属性；(3)改变 RCL 的入口等。

1.3 其他相关技术

微软在其实验性实时操作系统Rialto中提出了一个与资源容器相似的Activity[6]概念。在这个实时系统中，微软使用了一个资源集(resource set)对象来表征资源本身及其数量；使用Activity对象作为资源分配的接收对象，并将资源的使用记在其账上。通常每一个程序或应用对应一个Activity对象。

另外，与操作系统资源管理相关的技术还有 IRIX 系统下的软件性能单元(software performance unit, SPU)，Scout 中的 Path，Eclipse 中的预留域(reservation domains)等。

2 基于 FreeBSD OS 的 Chamber 技术

针对服务器操作系统对 QoS 和安全的需求，本文设计了一个基于 FreeBSD 操作系统的 FreeBSD Chamber(FC)机制。该机制基于内核层的资源容器，提供完全隔离的执行环境和系统服务质量支持。

FC 机制将 FreeBSD 操作系统分成一个 3 层结构，如图 1 所示，从下到上依次为硬件资源控制层，硬件资源虚拟层和虚拟执行环境层。最底层实现资源控制机制，直接控制物理资源的分配、调整和约束，是整个 FC 机制的底层支持。中间层将所有或部分物理资源抽象为资源池，并负责向上层应用提供系统资源，以满足应用运行的需要。最上层为虚拟执行环境层 FVE(FreeBSD virtual environment)，它虚拟出实际

作者简介：张 巍(1982－)，男，硕士研究生，主研方向：系统软件；卢 凯，副研究员

收稿日期：2006-09-26 **E-mail：**darkchampionzw@yahoo.com.cn

运行所需的系统环境。

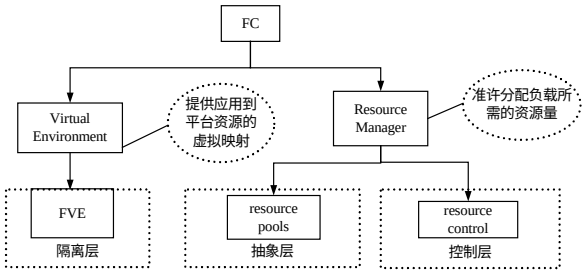

图 1　FreeBSD Chamber 组成结构

下两层都归入 FreeBSD 资源管理框架，主要为负载分配系统资源，并监视这些已分配资源的使用情况，如果需要还要调整这些分配以适应负载的变化。

上层的 FVE 和 FreeBSD 资源管理框架相结合，就可以实现提供隔离的执行环境和区分的 QoS 支持。如果需要还可以将 resource pools 置成 disabled，这样 FVE 使用的就是整个系统的资源，它也就变成了一个平凡的虚拟机。

2.1　隔离层

FreeBSD 虚拟执行环境 FVE 使用软件定义边界，模拟出一个应用执行所需的系统环境，包括文件系统、设备、网络、系统资源和安全等。系统仍然采用 FreeBSD 原有 ABI/API 以保证系统的兼容性。

每个 FVE 可有自己的 ID，并且可以独立重启或关闭；同一 FVE 内的进程通过 IPC 进行通信，不同 FVE 内的进程只可以通过虚拟的或者物理的网络进行通信；应用错误被隔离在一个 FVE 内；同时禁止 FVE 访问暴露系统数据的设备。

每个 FVE 内设置一个根资源容器，负责该执行环境中的资源调度。资源容器代替进程向系统请求资源，逻辑上可以包含一个负载所需的全部系统资源，而且可以准确地计算和调度一个负载在内核态和用户态所消耗的资源。资源容器可用描述符来标示，其语义同于文件描述符。子进程从父进程继承这些描述符，并且可以通过 Unix 文件描述符传递机制在同一 FVE 中不同的进程间传递这些描述符。

2.2　抽象层

所有的系统资源(CPU 资源除外)都可被抽象成资源池的形式，如图 2 所示。一个 FVE 绑定一个资源池，对 FVE 内的所有资源容器来说，该资源池内的资源是其唯一可见的。

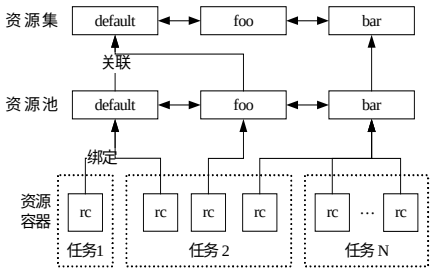

图 2　资源池示意图

图 2 给出了一个配有 3 个资源池和 3 个资源集的情形。资源集“ foo ”没有和任何资源池关联，因此，任何资源容器都不可以与其绑定。2 个资源池(default和foo)关联在同一个资源集上(default)，而任务 2 内的资源容器绑定在不同的资源池上。为了满足QoS的需要，可对资源容器的CPU使用百分比进行明确的限制(图 3)。为了保证该限制同时又不至于引起饿死和不公平，本文采用了lottery调度算法[5]。Lottery调度算法高效可控，它使用一个ticket代表资源权重，调度器随机产生一张lottery ticket，谁握有这张lottery ticket谁就可以获得资源分配，那么一个资源容器获得资源的期望值与其所持有的ticket数量成正比，并满足二项分布。

当在非叶节点资源容器中绑定 Lottery 调度器时，叶节点资源容器仍然采用 FreeBSD 原有的 4BSD 调度器。

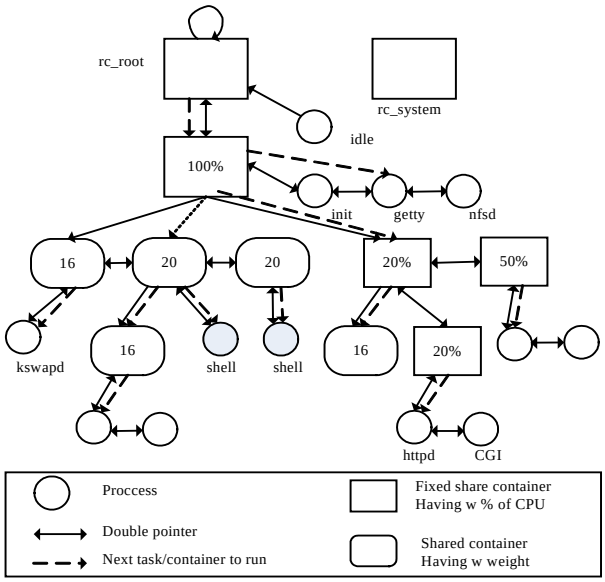

图 3　资源容器层级结构[3]

2.3　控制层

控制层实现对硬件资源的限制和控制。FreeBSD 资源控制层通过使用一个称之为资源控制(resource control, rctl)的数据结构来实现。同一资源可以有不同的资源控制，小容器级别的资源控制有更高的优先级。

资源控制的属性包含：

(1)特权级用于标示资源控制的控制级别，可以分为 3 级，第 1 级与资源软限制相关，可以被调用进程更改；第 2 级与资源硬限制相关，只可以被特权进程更改；第 3 级为固定特权级，不可更改。

(2)极限值用于对资源消耗限制作出规定，即资源代理被允许消耗的系统资源量。FreeBSD 资源限制极限值以 Unix 系统的 rlimit 作为基础，并提供与 rlimit 兼容的接口。rlimit 定义了进程消耗系统资源的约束边界。每个进程都从其祖先处继承这种约束。资源约束对应一对值，当前约束值(soft limit)和最大约束值(hard limit)。hard limit 必须大于 soft limit。只有拥有超级用户 ID 的进程才可以增大 hard limit。

(3)超过极限值后的动作包括本地动作和全局动作，用于说明当资源代理对资源的消耗超出限制时系统如何作出响应。动作定义由动作标志给出，例如 RCTL_LOCAL_DENY，RCTL_LOCAL_SIGNAL，RCTL_GLOBAL_NOACTION，RCTL_GLOBAL_
INFINITE 等。在操作系统的资源分离点如系统调用入口进行极限值检查时，一旦发现资源代理对资源的消耗达到资源控制的极限值时，就调用资源控制注册的动作函数，对该事件作出响应。

rctl 中保存了一个按升序排列的控制值链表，用游标 cursor 标示当前控制值。如果在动作标志中指定了 RCTL_GLOBAL_INFINITE 和 RCTL_LOCA L_MAXIMAL，即对资源消耗不作限制，则将 rctl 的 cursor 指向资源控制值链表中的下一个值，直到系统设定值为止。

3　总结

软分区技术可以有效提高系统资源利用率，并隔离错误；

(下转第 136 页)