内存虚拟化是虚拟化技术中至关重要的环节，它提供了虚拟机间内存的隔离与保护。Xen 采用影子页表机制来实现硬件虚拟机的内存虚拟化。

Xen 内存虚拟化实现

——影子页表内存管理机制

辛晓慧

在 Xen 中，影子页表（Shadow Page Table）的设计源于页表与 TLB 的基本交互过程,，这是实现分页机制的内存访问的核心。事实上，整个影子页表被看作是一个相对于客户机页表（Guest Page Table）而言的巨大的 TLB（Translation Lookaside Buffers）。影子页表和客户机页表不一定需要一致，只要影子页表的内容能正确仿真客户机的 TLB 即可。本文简述影子页表，并介绍影子表机制在 Xen 中的实现。

影子页表简介

1．启用影子页表的原因

Xen 利用影子页表机制来实现硬件虚拟机的内存虚拟。宿主机就是真实的物理机器，Xen 的监控程序就运行在宿主机上。客户机是指在宿主机上执行的硬件虚拟机，也被称为虚拟域。客户机认为它所拥有的内存地址空间总是从 0 开始，但它在宿主机上执行时不可能总是拥有从 0 开始的地址所在的物理内存。也就是说客户机的物理地址并不等于宿主机上的机器物理地址。图 1 描述了客户机的物理地址与宿主机上机器物理地址的关系。
这样监控程序必须把客户机线性地址到客户机物理地址的转换修正为客户机线性地址到宿主机物理地址的转换。这样的转换显然不是客户机的页表所能支持的，客户机的页表只知道客户机的物理地址，而监控程序为了实现对各个客户机的隔离与保护，也不会让客户机了解宿主机的物理地址。对于完全虚拟化的客户机，监控程序甚至不能够修改客户机的页表。但是，客户机的线性地址到宿主机物理地址的转换是保证客户机在宿主机上访问内存运行正确的核心环节，这样，为了支持和保存这种转换或映射，并能根据客户机修改页表的需要及时更新，Xen 就启用了另外一张页表，这就是影子页表。

2.影子页表

影子页表是监控程序真正使用和维护的页表。客户机执行时，监控程序在宿主机的页表基地址寄存器（CR3）中放入的是影子页表中指向最高级的影子页表的指针（物理地址）。当物理机上没有监控程序运行时，在物理机的页表基地址寄存器中放入的是物理机上运行的惟一操作系统所指向最高级页表的指针。在 Xen 中客户机维护着客户机自己的页表，而监控程序维护着影子页表。客户机实际上是通过影子页表在访问真实的机器物理地址，影子页表以客户机页表为蓝本建立起来，并且会随着客户机页表的更新而更新，就像客户机页表的影子。这就是我们称它为影子页表的原因。
在 Xen 中，我们把客户机的物理地址称为客户机物理地址，宿主机的物理地址称为机器物理地址。对于每个客户机，在 Xen 中都有两张表来表示客户机物理地址与机器物理地址的转换或映射关系。一张表称为客户机物理地址到机器物理地址的转换表（P2M table），以客户机物理地址为索引，可以在这张表中找到该物理地址所对应的宿主机机器物理地址。另一张表称为机器物理地址到客户机物理地址的转换表（M2P table），以宿主机机器物理地址为索引，就可以在这张

表中找到该机器物理地址所对应的客户机物理地址。

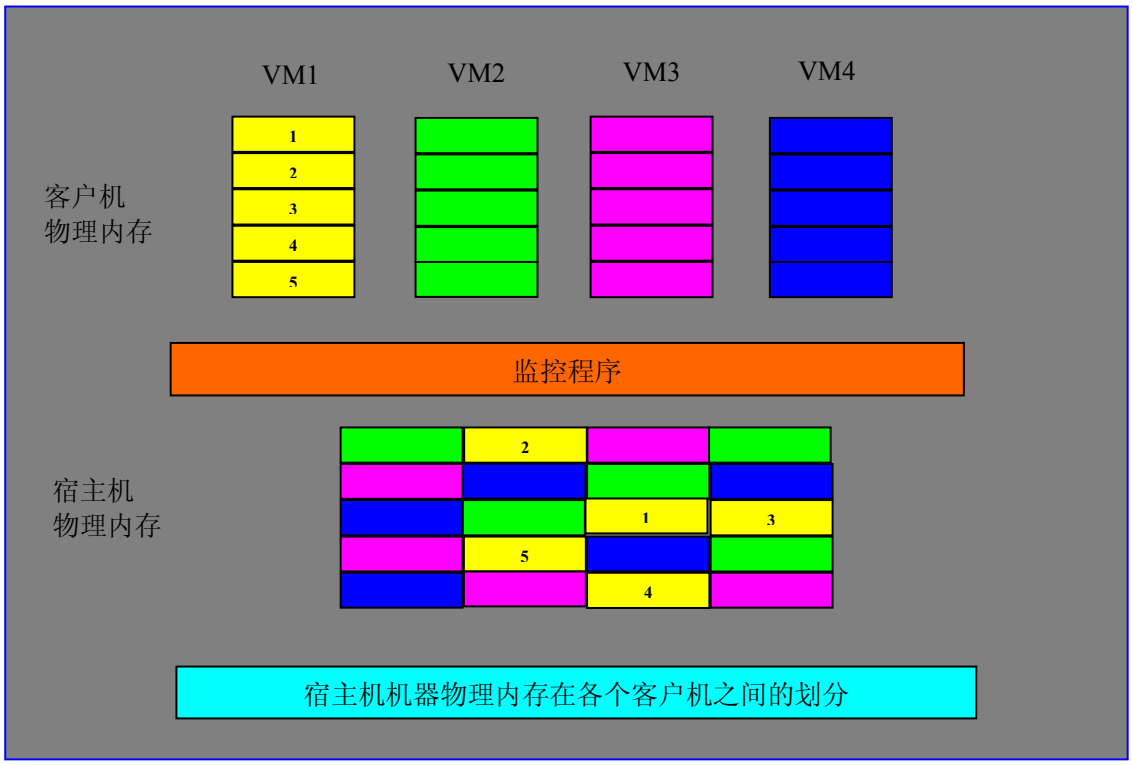

图 1 客户机物理地址与宿主机机器物理地址之间的映射

每一级客户机页表都有相应级别的影子页表与之对应，每一页客户机页表中有规定数目的页表表项，每一个页表表项中都包含一个客户机物理地址，而在该页客户机页表所对应的影子页表中，也有相应数目的页表表项，每个页表表项都包含了一个机器物理地址。

这些机器物理地址可以分为两种。一种是前文提到的在 P2M table 中对应了客户机物理地址的机器物理地址，另一种是指向了影子页表的机器物理地址。每当新生成一张影子页表时，监控程序会分给这张影子页表 4KB 大小的空间，有一个真实的机器物理地址指向它。如果该影子页表对应了最高级的客户机页表，则该机器物理地址就可能是宿主机 CR3 中的指针，否则该机器物理地址就会出现在该影子页表的上级页表的表项中。在最低级的影子页表中，其页表表项包含的一定是在 P2M table 中对应了客户机物理地址的机器物理地址。这样，最低级的客户机页表表项与最低级的影子页表表项才能指向相同的机器物理页。这也是客户机能通过影子页表访问真实的物理内存的关键所在。

为了生成新的影子页表，并能追踪与这影子页表对应的客户机页表，监控程序提供了一张哈希表和一个哈希函数。

在对应的客户机页表和影子页表中，相应位置上的客户机物理地址与机器物理地址之间，有两种对应关系，一种是通过哈希表来得到，一种则通过 P2M table 来转换。图 2 描述了这两种页表的对应关系。

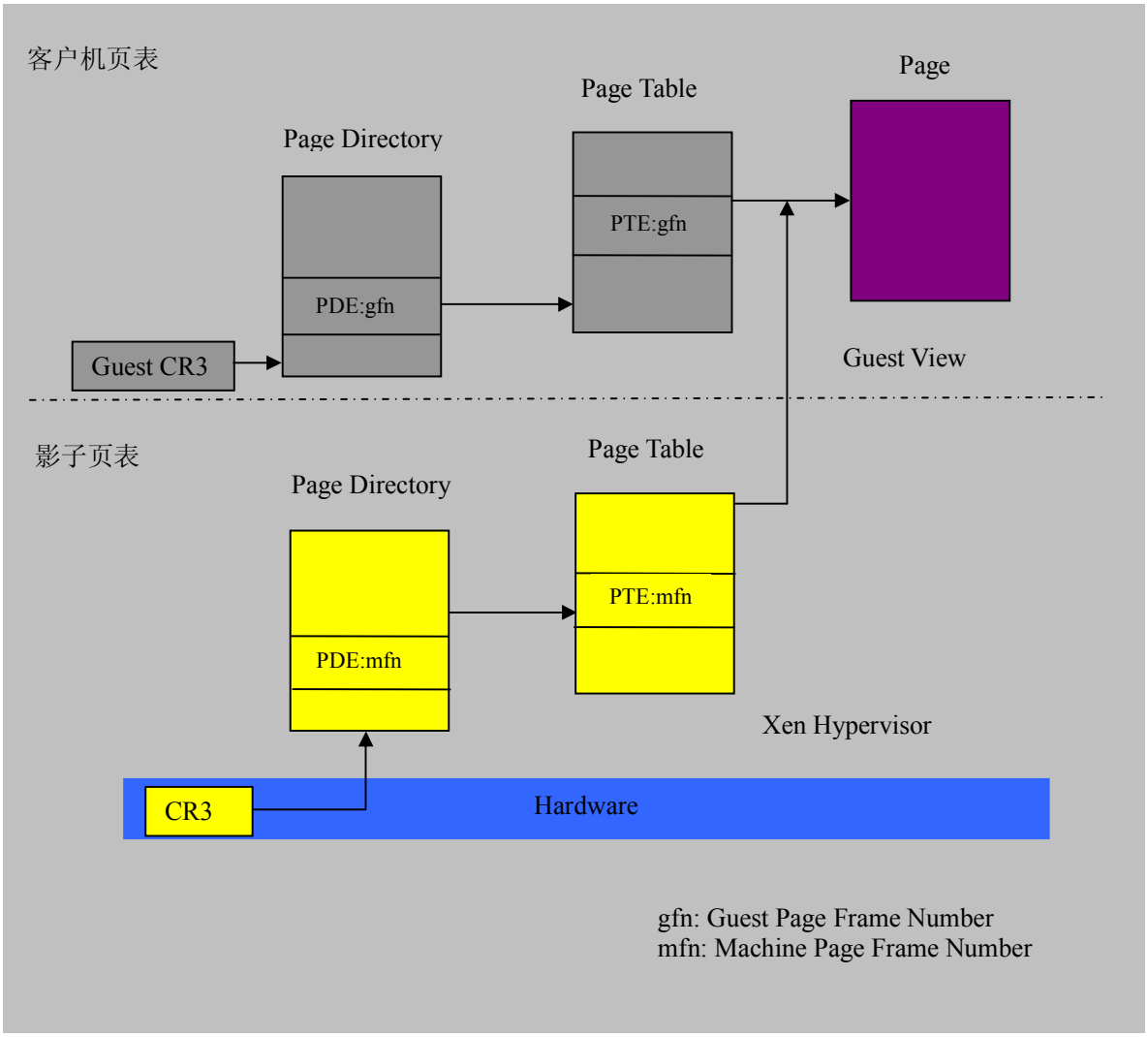

图 2 影子页表与客户机页表

3.影子页表机制在 Xen 中的实现

(1)虚拟 TLB 机制

在 Xen 中，监控程序利用影子页表不仅实现了客户机线性地址到宿主机物理地址的转换,还要用影子页表来保证客户机能如常执行内存管理的操作，这些操作可分为三类，包括 CR3 的更新、INVLPG 指令的执行及页表和页目录项的修改。要实现这一点，最直观的做法是要让监控程序能截获客户机对内存管理和访问的每次操作，这样影子页表就能和客户机页表严格保持一致，这种一致就意味着程序的正确执行。

但实际上在某些情况下，影子页表可以与客户机页表不一致。Xen 采用了类似 TLB 的机制来保证客户机正确访问内存，我们称之为虚拟 TLB。当然，监控程序在客户机要访问影子页表与客户机不一致的部分前必须重新同步这两种页表，保证程序正确执行。

影子页表可以看成是客户机页表的巨大的 TLB。客户机建立的页表不直接控制对内存的访问，而影子页表和宿主机的 TLB 合起来就可执行与客户机页表对应的 TLB 的功能。

如果客户机页表的访问权限比相应的影子页表高，因为监控程序实际访问的是影子页表，所以监控程序会截获一个缺页异常（Page Fault），这就类似一个 TLB miss。在处理该缺页异常时，监控程序会参照客户机页表来更新影子页表，这就对应了一个 TLB fill。
如果客户机页表的访问权限比相应的影子页表低，通常表现为客户机修改了页表表项，并且这修改把权限降低了。此时要使这一修改生效，客户机会通过重写 CR3 或执行 INVLPG 指令来刷新 TLB。这两种操作都会被监控程序截获，从而修改影子页表中相关的表项。这就对应了 TLB flush。
实际上，影子页表机制的具体实现有几种方法，但它们或多或少都利用了虚拟 TLB 的机制。图 3 描述了虚拟 TLB 机制。

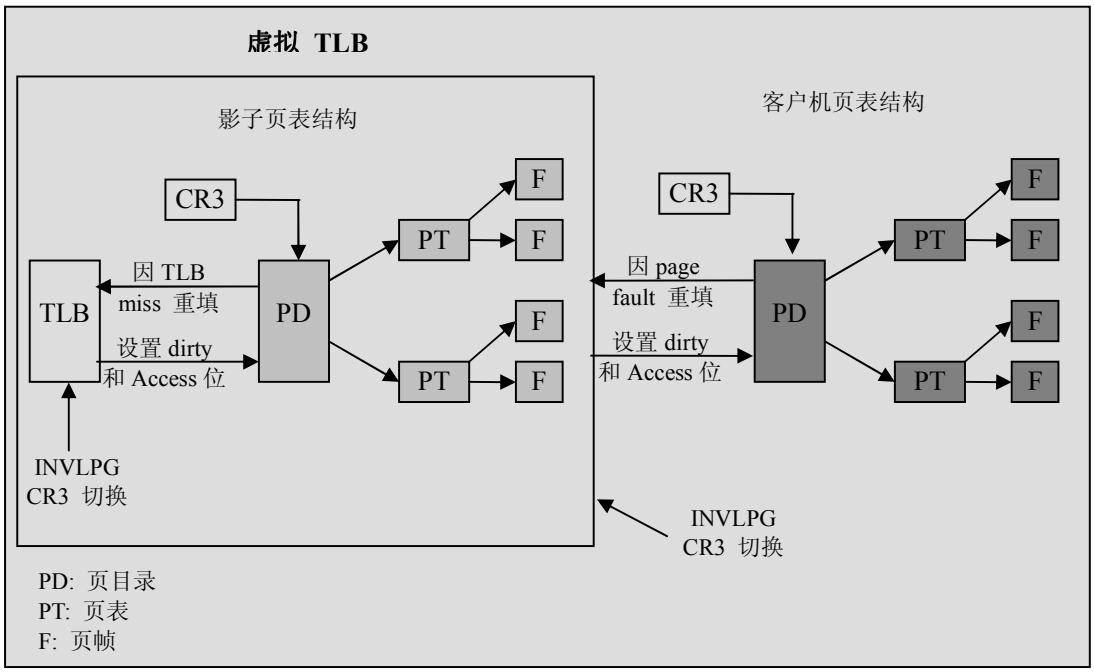

图 3 虚拟 TLB 机制

(2)影子页表的实现

在 Xen 中,影子页表支持几乎所有客户机使用的页表格式。这意味着 Xen 在 IA-32 位宿主机上可支持 32 位的客户机，在 IA-32 PAE 宿主机上可支持 32 位和 32 位 PAE 的客户机，而在 Intel® 64 位宿主机上，可支持 32 位，32 位 PAE 和 64 位客户机。
影子页表会使用几个基本的数据结构，一个是 P2M table 和 M2P table，这两张表存储了客户机物理地址到宿主机机器物理地址的一一对应关系；一个是主控表（Monitor table），在主控表中存储了所有客户机公用的 Xen 的地址映射的信息；还有一个就是哈希表。对某一个非最低级客户机页表中的页表项而言，以该客户页表项中的物理地址和该页表的类型作为哈希表的键值，在该哈希表中可以查找得到相应的影子页表项中的机器物理地址。
当客户机刚开始在宿主机上启动时，监控程序要为影子页表的建立作初始化的操作，主要是这三种数据结构的初始化。
当客户机启动分页机制时，影子页表还是空的，就像一个空的 TLB。随着客户机的执行，监控程序会截获客户机对内存访问的操作，一旦发现对应于客户机页表的影子页表不存在，监控程序会立即分配一个新的物理页，并参照客户机页表填充相应内容。

◆缺页的处理

当监控程序截取到缺页异常后，第一步要查找客户机的页表以确定指向发生缺页异常的线性地址对应的物理地址所在的页表项，该页表项的[0:11]位表明了该物理地址所对应的页的被访问权限，监控程序再根据此权限对照缺页异常产生的错误码，以确定该缺页异常是客户机本身的缺页异常，还是由于影子页表与客户机页表不一致而产生的错误（实际上，这是在非硬件虚拟机上不会发生的错误）。后面一种错误称之为影子错误。

对于客户机本身的缺页异常，监控程序不作任何处理就直接返回给客户机。客户机会解决该缺页异常。而对于影子错误，监控程序会根据出错的客户机页表项的内容来生成或更新相应的影子页表项。

实际上，会有多种出错的情况，但是归根结底，监控程序要把从最高级影子页表项到最低级影子页表项的页表结构建立起来。以最低级客户机页表中页表项内容的页表物理地址为索引，可以从 P2M table 中找到相应的机器物理地址，监控程序就把该机器物理地址填入最低级影子页表中相应位置的页表项中。这就是最终客户机线性地址到宿主机机器物理地址的映射。对于其它各级影子页表项中的页表物理地址的生成，监控程序则提供了一张哈希表和一个哈希函数。以客户机页表项中的页表物理地址和该页表项所在的页表的级别作为键值，通过该键值查询，若结果不为空，则表示该对应的影子页表项中物理地址已经存在并且所指向的影子页表已经生成。若结果为空，则需新生成一张影子页表，监控程序将获取指向该影子页表的机器物理地址，并把该机器物理地址填充到相应的影子页表项内容中，同时以相应的客户机页表物理地址和页表所在的级别生成键值，在代表该键值的哈希桶中填入该机器物理地址，以备查询。

对页表项的内容而言，除了要填充页表物理地址以外，还要填充该页表项所指向的页表的权限。而影子页表的权限，还决定了以后是否会产生被监控程序截获的缺页异常。

如果最低级的客户机页表项是被标记为不存在，监控程序会把相应的影子页表项填成空。如果最低级的客户机页表项是存在的，并且它的 Access Bit 和 Dirty Bit 都已设置，监控程序也会在相应的影子页表项中设置 Access Bit 和 Dirty Bit。如果最低级的客户机页表项中的 Access Bit 没有设置，则意味着该页表项所指向的页从未被访问过，这时监控程序会把相应的影子页表项标记未不存在。当下次要访问这个页时，监控程序会截获一个缺页异常，并且这是个读错误，这时会重新填充影子页表项，并将客户机页表项中的 Access Bit 设好。监控程序会检查客户机页表项的 Dirty Bit，如果没有设置，表示该页还没有被写入过，监控程序就会清除该影子页表项中 R/W Bit，这样若客户机要写该页，那么监控程序就会截获一个“写”缺页异常，这时就可以将影子页表项中的 Access Bit 和 Dirty Bit 设好，并同时将客户机页表项中的 Access Bit 和 Dirty Bit 也回设好。

◆INVLPG 指令仿真

操作系统可以执行 INVLPG 指令来刷新 TLB 中的表项。该指令以线性地址为参数，目的是要刷新该线性地址在 TLB 中的物理地址，使之无效。这样当处理器再访问该线性地址时，就必须遍历页表，并重新填充 TLB。该指令通常用于修改单一的页表表项。当然，此时监控程序的设置应允许截获客户机对 INVLPG 指令的执行。

执行这条指令就意味着在虚拟 TLB 即影子页表中相应的页表项应该为空，这样在客户机再访问该线性地址时，就会发生不存在的影子错误。监控程序会截获这个影子错误，然后会执行上节中对缺页异常的处理，重新使影子页表与当前的客户机页表同步。

◆CR3 切换仿真

为了进程间切换或其他目的，客户机会重写 CR3，有时对 CR3 的重写意味着要刷新当前处理器 TLB 中的所有项目，使之无效。这种重写通常意味着当前客户机使用的页表的改变。

监控程序截获客户机对 CR3 的每次切换，最直观的作法就是模拟真实的 TLB，删掉所有的影子页表，使之为空。但实际上从性能的角度考虑，某些影子页表的实现会令监控程序采用一些优化措施，仍然允许使用老的影子页表，但会让影子页表与客户机页表保持同步。不过，不管怎样，监控程序都要把新的指向最高级影子页表的指针即与客户机页表的 CR3 对应的机器物理地址保存下来，等到客户机真正执行时填入宿主机的 CR3 中。只有这样，客户机才能正确执行。

（3）性能优化

影子页表的实现是影响客户机性能的最关键的因素，其热点在于客户机切换 CR3 时要求 TLB 全部刷新。这个时候监控程序对影子页表的操作会对性能有较大影响。因为客户机切换 CR3 的动作会非常频繁。如果每次都把影子页表全部删掉重建，就会有不小的开销，而实际上刚被删掉的页表可能很快又被客户机使用到。如果监控程序此时知道客户机页表中哪些表项被更新了，就可以只更新相应的影子页表中的表项，旧的影子页表就可以重用。这样就可以尽量避免 CR3 切换时的开销。

有一种影子页表的优化算法是这样的：在影子页表第一次被分配时，就把指向该影子页表的上一级页表项中的页表被访问权限置为只读（read-only），这使得客户机下次想修改该客户机页表时，就会触发一个缺页异常。但是因为一个客户机页表可能对应了多个影子页表，监控程序在这时必须保证该客户机页表所对应的所有影子页表的权限都是只读。

客户机试图写入该页表时，监控程序就会截获该页错误，会生成一个称之为快照的新页，然后监控程序会复制客户机页表的内容到快照页中，并把对应的影子页表的权限增加为可写。同时，监控程序会把与这客户机页表相对应的信息如客户机的页表物理地址，对应的机器物理地址，快照的机器物理地址，出错的虚拟地址，以及该客户机页表此时对应的影子页表项的机器物理地址等信息组成一个数据结构，该数据结构被放入叫做 out-of-sync 的链表中。此后，客户机访问该页时就不会再触发任何缺页异常了。只要客户机修改了该页，相应的影子页表的内容和客户机页表的内容就会不一致。

在一段时间内，可能会有多个客户机页表页与其对应的影子页表页内容不一致，在 out-of-sync 链表中就会有多个链表结构。但等到客户机执行到一个在这些客户机页表地址范围内并且需要刷新 TLB 的内存操作时，比如一个新的影子错误，发生缺页异常的虚拟地址或 INVLPG 指令执行的虚拟地址所在的页表结构中，有页表在 out-of-sync 链表中，并且该虚拟地址所在的页表项与在 out-of-sync 链表中该页表的快照相应位置中的页表项不同，此时，监控程序会作一系列的动作，我们称之为同步。

在同步时监控程序就会遍历这个 out-of-sync 的链表，把链表结构中所有与影子页表不一致的客户机页表的内容与其快照相比较，只要发现客户机页表的内容自上次允许写操作时发生了变化，就参照客户机页表的内容更新相应的影子页表的内容。

在真正作影子页表的更新动作之前，监控程序还要执行一个重要操作。即删除所有在out-of-sync 链表中的客户机页表对应的影子页表的写权限。这是防止在客户机页表与影子页表同步之前，该缺页异常未处理之时，客户机对页表继续执行写操作。

当影子页表的更新操作完成后，影子页表就与客户机页表完全同步了。此后，当客户机想继续写页表时，就会发生新的写缺页异常。监控程序就作周而复始的动作。在 CR3 切换时，监控程序会主动执行上文所述的同步操作。

总结

内存虚拟化是虚拟化技术中最至关重要的技术之一，对影子页表性能的任何改进直接提升客户机的总体性能，这也是当前 Xen 社区中最热门的话题之一。对影子页表的改动，将直接影响其他模块的功能实现。

作者简介：
辛晓慧，现任职于Intel开源软件技术中心（Intel Open Source Technology Center），
主要开发方向为虚拟化技术和内存管理，目前参与Xen项目开发。