四．虚拟内存的性能

1．虚存管理和性能

RS/6000 虚存由实存和硬盘交换区组成，虚存空间划分为 256MB 的段，段又划分为固定大小的页，每页 4KB，这些页存在于实存或硬盘交换区上，对应地实存中的帧大小也为 4KB，虚存管理 VMM 的目的就是管理实存帧的分配和解决页面命中失败问题（程序访问的数据不在实存或虚存的情况）。从性能的角度出发，VMM 的目标是最小化处理器和磁盘 I/O 花在虚存上的开销，最小化命中失败的响应时间。

换页算法，即 VMM 如何进行页面的调度，和以下几个概念有关：

空页表 free list
VMM 维护一个空页表，当页面命中失败时，可马上将要访问数据调入空页表中的空页，不需先进行页面调出，以提高性能，换页算法使空页表中的空页数量保持一定水平。

固定段 persistent segments 和工作段 working segments
虚存段分为固定段和工作段，固定段在硬盘上有固定的存贮位置，数据文件和执行文件被映射到固定段。在页面表 page frame table 中，段类型为 C 和 P 的就是固定段，C 表示数据来自远程，如：NFS，P 表示数据来自本地。

工作段在硬盘上没有固定的存贮位置，当进程运行时产生工作段，由进程堆栈和数据区、核心文本段、核心扩展文本、共享库文本和数据段组成，当工作段页面从实存中调出时，也需要占用硬盘空间，这部分空间就是交换区。在页面表 page frame table 中工作段用类型 W 表示。

计算内存 computational memory 和文件内存 file memory
虚存段也可分为计算内存和文件内存，计算内存由工作段和程序文本段构成，其余的页构成文件内存。

新页和再访问页的命中失败
页面访问的命中失败分为新页和再访问页的命中失败，新页的命中失败指要访问的页从未被调入内存，再访问页的命中失败指要访问的页曾经调入内存，但由于某些原因又被调出，此种类型的命中失败率太高，说明换页算法需要调整，尽量减少因此产生的磁盘 I/O 等不必要系统开销。

VMM thresholds 参数
系统管理员可以通过 vmtune 调整 minfree 和 maxfree、minperm 和 maxperm 等参数影响换页算法。

空页表参数 minfree 和 maxfree，当空页表中的空闲实存帧数量少于 minfree 时，页面调出程序 page stealer 开始执行，直到空页表中的空闲实存帧数量达到 maxfree。

页面调出程序 page stealer

当可用实存帧在空页表中太少时，系统执行 page stealer 扫描页面表 page frame table，针对每个页面，如果：

访问位 reference 被置上的，说明最近有使用，该位被清除，但该页不被调出；

访问位未被置上的，说明上次清除以来无访问，该页被调出。

修改位 modify 为真，则

1．段类型为 W，写回交换区

2．段类型为 P，写回硬盘

3．段类型为 C，写回 NFS

如修改位为假，则直接释放。

通过 minperm 和 maxperm 可以平衡计算内存和文件内存的调出，这两个值是百分比数。

如果实存中文件内存的百分比低于 minperm，换页算法不考虑再访问页的命中失败率，同时调出计算内存和文件内存。

如果实存中文件内存的百分比高于 maxperm，换页算法只调出文件内存。

如果实存中文件内存的百分比介于两者之间，换页算法一般来讲调出文件内存，但文件内存的再访问页命中失败率高于计算内存，则同时调出计算内存和文件内存。

换页算法的这些设计，目的是为了在公平对待各种类型页面的基础上，给予计算内存略高一些的待遇，减少被调出的机会，从上文的定义我们知道，程序文本段属于计算内存，一个经常出现的情况是，连续调入大数据文件，导致一些程序文本段被调出，但这些程序文本可能马上要被再次用到，定义合理的换页算法弥补了这种不足。

颠簸 thrashing 问题

AIX 用换页算法将未来可能不用的页调出，以免新进程要求调入时造成 I/O 延迟，但如果活动进程太多，则没有未来可能不用的页，这时系统会发生频繁的调入和调出，由于过量的页面调度，系统处于难以做有效工作的状态，AIX 用内存加载控制算法监测颠簸的发生，通过挂起一些进程和延迟新进程的投入来解决问题。

2．标准 AIX VMM 工具

常用的 AIX VMM 工具有 vmstat 和 ps。

vmstat

vmstat 可以反映系统中活动虚存 avm（Active Virtual Memory）和实存空帧 fre 的情况。

语法：vmstat [-s] [interval [count]]　interval 每次采集间隔，count 次数。

我们以 1 秒的间隔采集多次，观察输出，如果 page 段的 pi 超过每秒 5 次，

说明换页太频繁，可能是内存不足；如果 page 段的 sr/fr 太高，说明系统中活动子系统太多，要扫描多个页面，才能释放一个空页；页面的调入和调出不仅发生在内存和交换区之间，而且发生在内存和硬盘之间，在用-s 的输出中，如果发生在交换区的调入调出和全部的调入调出接近，说明系统的交换区有问题，需要加内存或在不同的硬盘上建交换区。

ps

ps 根据参数的不同，提供多种内存使用的报告，最常用的是 ps v。

输出：

SIZE：进程占用的工作段大小，以 KB 为单位。

RSS：进程占用的实存数量多少，以 KB 为单位，包含工作段（私有数据）和固定段（代码），注意同一个程序的多个实例，其代码不会多次调入内存，因此这些实例的代码页实际占用的是实存的同一页面。

TRS：进程固定段（代码）占用的实存数量多少，以 KB 为单位。

%MEM：进程占用实存的百分比，即 RSS 除实存大小再乘 100%。

3．高级工具

要使用下面这些工具，也必须安装 perfagent.tools 包。

svmon：

提供包括全面的、进程级的和段级的内存使用情况报告。常用-G 和-P 选项。

-G：对整个系统的内存和交换区使用情况汇总。

-P：显示一个或多个进程对内存的使用情况。

-i：采集次数和时间间隔。

输出：比较 svmon –G 和 vmstat 的输出，memory 段的 free 和 vmstat 的 fre 含义相同，pg space 段的 inuse 和 vmstat 的 avm 含义相同。

假设某个进程 PID 为 xxxxx，比较 svmon –P xxxxx 和 ps v xxxxx 的输出，我们会发现，前者描述为私有数据 private 的工作段占用的页面数乘以 4KB 等于后者的 SIZE 值；而该值再加上描述为代码 code 的固定段占用的页面数乘以 4KB 等于后者的 RSS 值。

bf：

bf 是 Bigfoot Utility 的简称，可以反映进程对各种段的页面和独占页面的使用情况，反映多少页被一个应用访问，多少页被子函数或进程共享等信息，运行 bf 会极大地增加系统开支，不能在生产环境中使用。

bf 的输出默认到文件_bf.rpt，需要用 bfrpt 转换成文本格式，默认文本文件名是__global.rpt。

rmss

和 bf 一样，rmss 也不能在生产环境中使用，rmss 提供一种途径，不用实际移去内存，就可以模拟比实际内存少的系统环境，来运行程序，测试性能。它解答了这一类问题，在 AIX 上运行某个应用至少需要多少内存，其性能才可以接受？

rmss 有两种调用方法：
模拟多种大小的内存情况，最少至 4MB。
#rmss –c 16　　内存变为 16MB
#rmss –p　　显示当前内存
#rmss –r　　恢复成实际内存大小
让进程在各种内存大小的情况下运行，并统计换页和运行时间。
语法：
rmss [-s memsize][-f memsize][-d memsize][-n num][-o output] command
-s：开始内存大小　　-f：结束内存大小　　-d：间隔多少递变
-n：运行几次　　-o：统计输出文件
输出：统计在各种内存大小下，运行 command 命令的平均换页数、响应时间和换页率。

4．优化技术

针对内存不足的系统，有以下优化手段可采用：

应用优化：
用 svmon 检查是否有某一个应用程序一直在申请内存，却不释放，对其代码做优化，及时释放内存，减少不可释放的内存申请量。
应用申请内存（虚存地址空间）的分配方法有两种，延迟分配和提早分配，前者是默认的方法，即应用要求分配内存，但只当应用使用到这部分内存，系统才提供，这样做有一定风险，如果多个应用同时使用到以前分配的内存，则可能耗尽交换区，有的应用不能正常继续。提早分配，即应用申请内存，系统就全额提供，不管应用是否真的马上就要这些空间，让系统采用提早分配，要先运行命令：export PSALLOC=early，一般情况下，交换区大小是实存的两倍，采用提早分配，交换区大小应是实存的四倍。

系统优化：
可以通过以下两个命令进行调整。

vmtune
该命令位于/usr/samples/kernel/vmtune，不在搜索路径中，不带参数执行可以显示和虚存管理有关的参数当前值，都是以页为单位。带参数执行可以修改。
当 vmstat 的输出中，发现 fre 经常低于 minfree 时，进程会因等待换页而被延迟，可以用 vmtune –f minfree 和 vmtune –F maxfree 调高和空页表有关的两个参数。这两个值的单位是页面数。
当用 iostat 监测发现一些文件经常被重复调入内存，可以用 vmtune –p minperm 和 vmtune –P maxperm 调高和固定段有关的两个参数，减少代码页被调出内存的机会，这两个值的单位是百分比。

schedtune
该命令也位于/usr/samples/kernel/schedtune，不带参数执行可以显示当前内存控制参数值。

参数 h 默认值是 6，其用途是判定系统当前是否陷入颠簸状态，是否需要采取挂起进程的手段。系统每隔一秒钟检测刚过去的这一秒内，换出并写回硬盘的页面数和换出总数的比值，如果大于 1：6，即每换出六页就有一页要做写操作，那么系统很可能陷入颠簸，如果比值大到接近 1：1，那么可以肯定发生颠簸。h 值可以调整以适应不同的系统环境，用 schedtune –hn 修改该值（其它参数修改均类似），h 值调为 0，表示不进行检测。另外，结合 vmstat 的输出，如果 po/fr 大于 1/h，说明内存不足。

参数 p 默认值是 4，如果某进程的再访问页的命中失败率大于 1/p，说明该进程发生颠簸，作用和 h 值类似，只是针对某个进程。

参数 w 表示挂起的进程等待几秒后再激活，默认是 1 秒。

参数 e 表示挂起的进程投入运行后，拥有几秒的豁免权，不会再挂起，默认是 2 秒。

参数 m 表示最小活动进程数，核心进程、固定优先级小于 60 的进程、不可换出内存的进程和等待事件的进程不算在内，因为这些进程不能挂起，默认值是很保守的 2，多用户大内存的环境建议改为 4 或 6，该值也是为了防止活动进程太多系统发生颠簸，特定的配置下，只要不发生颠簸，可以试着调高该值。

参数 t 表示时间片，上文已做介绍。

参数 f 表示 fork 子进程失败时，等待多久再试，默认是 10 秒。

schedtune –D 可以恢复默认值，schedtune –h0 将关闭内存加载控制算法，在运行 rmss 命令前，建议先关闭控制算法，运行后再恢复默认值或指定的参数值。