IBM RS6000 的性能调整

性能问题的产生一般不引人注意，对运行速度不尽如人意的系统，用户往往倾向于更换为新型号的设备，但有些情况下，可以通过对系统进行调整来延长设备的生命期。曾经有一位证券行业的用户，日终的清算工作要长达 3 个小时，经过性能分析优化后，减少到 30 分钟。本文旨在通过介绍性能分析的步骤，如何使用性能分析优化工具，来帮助 RS/6000 系统管理员挖掘自己现有设备的潜能。

一．综述

RS/6000 的性能调整包含两方面的内容：系统管理员的工作和应用开发人员的工作。前者通过合理分配资源和调整系统设置来最大化系统吞吐能力；后者通过减少应用对系统资源的占用来最小化用户的响应时间。两者的目的有时会出现矛盾，比如：开发人员为减少响应时间，编程时希望系统为用户提供足够的资源，以尽快完成作业，而系统管理员的目标是平衡整个系统，希望尽可能多的用户提高满意度，因此，会限制个别用户对资源过度要求。本文的内容主要涉及和系统管理员有关的工作。

性能调整的核心围绕着资源的利用，这些资源分为物理的和逻辑的，物理的如：CPU、内存、输入输出设备（包括磁盘空间、磁盘读写臂、I/O 总线、通讯线和各种卡等），逻辑的如：逻辑卷管理器、虚存管理器、文件系统、队列和缓存等。

合理利用这些资源有许多途径，可以通过有效的程序代码和数据组织最小化资源要求；尽量使用合适的资源，如：用本地的而非远程的、用硬盘上的而非光盘介质上的资源；还可增加对并行资源的利用，如：磁盘的共同访问技术、使用更多的通讯线路和 CPU；以及控制资源分配，如：限制用户可使用的资源、限制优先级、使用批处理技术等方方面面。最后还能通过增加更新更快的关键资源来缓解资源的不足。

二．性能工具和调整步骤

1．和性能有关的工具

RS6000 的操作系统是 AIX，我们将 AIX 的性能工具，按所属软件包分为标准工具和高级工具，按使用目的分为用于分析报告的和用于优化的。

AIX 标准的性能工具：

用于分析和报告的：	用于优化的：
accounting	chdev
gprof/prof	nfso
iostat	nice
lsattr	no
lslv	renice

netstat	reorgvg
ps	schedtune
sar	vmtune
trace/trcpt	
vmstat	

AIX 高级性能工具：

需安装 Performance Toolbox 或 Performance Aide 特许程序。

用于分析和报告的：	用于优化的：
bf/bfrpt	fdpr
filemon	lvedit
fileplace	rmss
lockstat	
netpmon	
stem	
svmon	
syscalls	
trpof	

这些工具的使用说明，在 AIX 的随机帮助文档中有详细介绍，这里不再占用篇幅。

2．性能调整步骤

调整负载让系统有效的利用资源，首先要确定系统的主要应用达到怎样的性能目标才可以接受，然后通过以下步骤，找到影响整体性能的关键资源，加以优化直到达到目标，必要时增加额外的资源。

用 iostat 检测是否是 CPU 原因？

如果是 CPU 不足的问题，进一步可采用的工具和优化手段有：tprof、优化编译、重新安排作业运行时间表、优先级控制和升级 CPU 等等。

用 vmstat 检测是否是内存原因？

如果是内存不足的问题，进一步可采用的工具和优化手段有：svmon、rmss 和增加内存等等。

用 filemon 检测是否是硬盘原因？

如果是硬盘读写响应较慢的问题，进一步可采用的工具和优化手段有：重组织硬盘、增加交换区（又称为换页空间）、重新安排作业运行时间表、增加内存和增加硬盘等等。

用 netstat/nfsstat 检测是否是网络原因？

如果是网络响应较慢的问题，进一步可采用的工具和优化手段有：修改网络参数、优化 NFS 性能、远程数据本地化、重新安排作业运行时间表和增加内存等等。

最后还可用 trace 检测 CPU 在空闲状态，而等待进程又不投入运行的原因，以及是否发生颠簸(thrashing)。

三．分析优化 CPU 的使用

1．进程和线程的优先级

我们知道进程的优先级从 0-127，值越小，优先级越高，而 40 是分水岭，实时进程优先级在 0-40 之间，用户进程优先级在 40-127 之间，可以用 ps –l 命令看进程的优先级，可以在运行时，通过 renice 来修改。

每个进程由一个或多个线程构成，线程继承父进程的优先级，线程优先级=40+NICE+CPU/2，NICE 是父进程启动时的优先级，默认是 20，线程每分到运行时间片，则 CPU 值增加，而每等待 1 秒，该值减少。系统以此来调整各线程的合理投入运行。线程的优先级用 ps –emo THREAD 命令列出。

针对每个优先级有一个运行队列，线程根据其优先级决定在哪个队列中排队。管理员可以通过优先级的调整，控制关键应用享有较高的优先级，提高系统的性能。

2．CPU 活动的监视

可以用 iostat、vmstat、sar 等命令发现系统是否存在 CPU 不足的问题，当输出的%user+%sys>80%，说明 CPU 处理能力不足。用户时间 user 指程序自己和它调用的库函数占用的时间，系统时间 sys 指程序执行的系统调用占用的时间。如果两者之和为 100%时，即%idle 和%wait 为 0，说明很可能有无限循环的程序在执行。

接着，要定位有问题的进程，找到大量占用 CPU 的进程，对其优化。如果计帐系统在运行，可以用/usr/sbin/acct/acctcom 分析历史数据；也可用 ps au 或 ps vg 观察哪个进程的%cpu 和 TIME 值较高，%cpu 表示该进程存活期内占 CPU 时间的百分比，可衡量进程需要的资源是否主要是 CPU，TIME 表示占用的实际时间几秒。

如果您有该进程的源码，还可以用后面介绍的分析工具（profilers），进一步找到是程序的哪一部分代码效率低下，虽然程序的的执行由应用代码、库函数和核心调用组成，但往往都有一些我们称为热点的语句或函数被频繁执行，它们是影响性能的关键。

3．标准的分析工具

在安装 OS 时，有一个可选的包 bos.adt，选择安装后，系统提供了下面这些标准的分析工具，可对应用程序监控，找到耗时多，被频繁调用的函数，但对应用程序要用特别参数重编译，编译会加入一些监控函数，程序的性能可能降低 3-4 倍。

time：

语法：time program 或 timex program，注意 program 参数不能是一系列用管

道符连接的命令。

目的：time 是一个简单但有用的工具，用来监测单个程序的运行性能。

输出：time 以分钟秒钟格式，timex 以秒钟格式，表示该 program 运行时，占用的实际（real）、用户（user）和系统（sys）的运行时间。程序从开始运行到结束，这段时间就是实际运行时间。用户和系统时间前面已介绍过。

观察：如果 real 大于 user+sys 很多，说明可能系统 I/O 能力不足，运行时要等待 I/O。如果 real 只略大于 user+sys，说明可能系统 CPU 能力不足。注意，两种情况只反映了一种可能性，比如系统较忙，而监测的程序优先级低于同时运行的其它程序，real 也会大于 user+sys 很多，此时并非等待 I/O，而是优先级低，分不到运行时间片。

prof：

语法：prof [-optional|flags] [program]

目的：检查应用程序的子函数占用 CPU 时间、调用次数和调用频率的情况，可以分析 C/Fortran/Pascal/COBOL 程序，但程序要用-p 重编译，执行文件名做为参数 program。

输出：Name 函数名，%time 占时百分比，msec/call 多少毫秒一次调用等。

观察：注意%time 和 msec/call 较大的函数。

gprof：

语法：gprof [-flags] [program] [gmon.out]

目的：是 prof 的高级形式，提供更多的功能，源码编译要用-pg 参数。

输出：除类似 prof 的输出外，还有调用关系图，反映父函数所属子函数的调用情况。

4．高级分析工具

要使用下面介绍的几个高级分析工具，必须安装 perfagent.tools 包，用 lslpp –lI perfagent.tools 命令检查该特许程序包是否已安装。

tprof：

tprof 的工作原理是每秒 100 次中断去记录进程对地址的访问，来反映程序的哪些部分被频繁调用，因此，对编译没有特殊要求，但如果有-g 编译，tprof 能指出是程序的哪一行被频繁调用。只能分析 C 和 Fortran 程序，因每秒采集 100 次，所以对短程序不准确。

语法：tprof [-s][-k[-e][-v][-p program][-t pid][-x cmd]

输出：在文件[program name].all 中有 CPU 的使用和函数调用情况，如用-g 编译，还有一些文件产生，直观描述源码中的某个函数或某行的访问情况。

例如： #xlc –g myprogram.c –o myprogram

#tprof –p myprogram –x myprogram

将有__myprogram.all 和__t.myprogram.c 等文件产生，从中可以发现程序的热点。如果想了解共享库的哪些函数频繁调用，可以用：

#tprof –v –s –k –p myprogram –x myprogram

比较 tprof 和 prof/gprof 的功能有如下不同：

tprof	prof/gprof
函数级描述不要重编译	都要重编译
语句级描述（重编译）	不支持语句级描述
没有增加用户 CPU 时间	增加 10%-300%
可以接受优化代码	不接受优化代码
所有 CPU 使用汇总	单个程序对 CPU 使用
无函数调用关系图	后者有调用关系图
无函数调用记数	有函数调用记数

stem：

用命令#/usr/bin/stem –p myprogram 对 myprogram 做预处理，可以将每个函数的入口和出口用 stem 的标准入口和出口替代，在/tmp/EXE 下会生成新的可执行文件，运行该文件，产生 stem_out_001 报告文件，这个文件可能很大，反映了每个函数的执行顺序。

5．优化技术

针对 CPU 能力不足的系统，有这些优化手段：

代码优化

程序中采用有效的代码结构，应用编译选项-O、-O3 和-Q 等，产生优化的执行码，使用专用库函数提升向量计算的效率，如 Basic Linear Algebra Subroutines 库/lib/libblas.a 等等。

优化工具

fdpr：

fdpr 是 feedback-directed program restructuring（直接反馈程序重构）的简称，也需要安装 perfagent.tools 包，且只能在 64 位的机器上使用，经 fdpr 优化的程序运行的更快，对内存的使用更有效，调用的例子如下：

#fdpr –p program –R3 –x test.sh

test.sh 是调用 program 程序的一个 shell 程序，包含了这个程序运行需要的参数等运行环境，–R3 是最高的优化级别。fdpr 的执行分三步：

创建可执行程序

执行并采集数据

根据运行搜集的数据，进行分析，重新构造可执行程序

但是，这种重构有可能使程序的运行结果和重构前不一致，因此，必须对重构的程序全面测试。

系统优化

通过系统管理实现优化的手段有：

用 nice 和 renice 控制进程的优先级，关键应用享有较高的优先级，nice 用于程序启动时，renice 用于程序运行中；

调整运行计划，将较大的批处理工作安排在系统不忙的时候运行，涉及到 at、batch 和 cron 命令；

为使用户登录效验 UID 更快，可用 mkpasswd 命令，使/etc/passwd 生成基于 hash 算法排列的文件/etc/passwd.dir 和/etc/passwd.pag，这两个文件生成后，用

passwd、mkuser、chuser 和 smit 命令修改/etc/passwd 都会自动修改它们，但如果使用编辑器或 pwdadm 命令修改/etc/passwd 文件，则要重建 hash 文件，重建 hash 文件的语法：#mkpasswd /etc/passwd;

用/usr/samples/kernel/schedtune –t increase 可增加时间片（timeslice）的长度。increase=0，则用默认的长度 10ms，increase=3，则长度为 30ms，在修改后的系统中，不需要或不应占用全部时间片的应用线程，会通过 yield 调用，释放 CPU 资源给其它高优先级的线程。