The UNIX System Activity Package

Tsyh-Wen Pao

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

This memo describes the design and implementation of the UNIX† System Activity Package. This package reports UNIX system-wide statistics including CPU utilization, disk and tape I/O activities, terminal device activity, buffer usage, system calls, process switching and swapping, file-access activity, queue activity, and message and semaphore activities.

It provides four commands to generate various types of reports: *sar*, *sag*, *sadp* and *timex* commands. Procedures for automatically generating daily reports are also included.

1. INTRODUCTION

The System Activity Package reports UNIX system-wide measurements including CPU utilization, terminal device activity, disk and tape I/O activities, buffer usage, system calls, system switching and swapping, file-access activity, queue activity, and message and semaphore activities. There are five functions:

— *sar* command: allows a user to generate system activity reports in real time and to save system activities in a file for later usage.

— *sag* command: displays system activity in a graphical form.

— *sadp* command: samples disk activity once every second during a specified time interval and reports disk usage and seek distance in either tabular or histogram form.

— *timex* command: a modified *time*(1) command, which times a command and also reports concurrent system activity.

— system activity daily reports: procedures are provided for sampling and saving system activities in a data file periodically and for generating the daily report from the data file.

The system activity information reported by this package is derived from a set of system counters located in the operation system kernel. These system counters are described in Section 2. Section 3 describes the commands provided by this package. Section 4 gives the procedure for generating daily reports. A description for each of the files used by the system activity package can be found in Attachment 1.

2. SYSTEM ACTIVITY COUNTERS

The UNIX operating system manages a number of counters that record various activities and provide the basis for the system activity reporting system. The data structure for most of these counters is defined in the **sysinfo** structure (see Attachment 2) in **/usr/include/sys/sysinfo.h**. The system table overflow counters are kept in the **_syserr** structure. The device activity counters are extracted from the device status tables. In this version, the I/O activity of the following devices is recorded: RP06, RM05, RS04, RF11, RK05, RP03, RL02, TM03 and TM11.

† UNIX is a trademark of Bell Laboratories.

In the following paragraphs, the system activity counters that are sampled by the system activity package are described.

— **cpu** time counters: There are four time counters that may be incremented at each clock interrupt 60 times per second. Exactly one of the **cpu[ ]** counters is incremented on each interrupt, according to the mode the CPU is in at the interrupt; idle, user, kernal, and wait for I/O completion.

— **Lread** and **lwrite** count logical reads and logical writes, that is, read and write requests issued by the system to block devices.

— **Bread** and **bwrite** count blocks transferred between the system buffers and the block devices. These actual I/Os are triggered by logical I/Os that cannot be satisfied by the current contents of the buffers. The ratio of block I/O to logical I/O is a common measure of the effectiveness of the system buffering.

— **Phread** and **phwrite** count read and write requests issued by the system to raw devices.

— The **swapin** and **swapout** counters are incremented for each system request initiating a transfer from or to the swap device. More than one request is usually involved in bringing a process into memory, or out, because text and data are handled separately. Commonly used programs are kept on the swap device and are swapped in rather than loaded from the file system. The **swapin** counter reflects these initial loading operations as well as resumptions of activity, while the **swapout** counter reveals the level of actual "swapping." The amount of data transferred between the swap device and memory are measured in blocks and counted by **bswapin** and **bswapout**.

— Counters **syscall** and **pswitch** are related to the management of multiprogramming. **Syscall** is incremented every time a system call is invoked. The numbers of invocations of system calls: *read*, *write*, *fork* and *exec*, are kept in counters **sysread, syswrite, sysfork** and **sysexec**.

  Pswitch counts the times the switcher was invoked, which occurs when:

  a. a system call resulted in a road block,
  b. an interrupt occurred resulting in awakening a higher priority process, or
  c. 1 second clock interrupt.

— Counters **iget**, **namei**, and **dirblk** apply to file-access operations. **Iget** and **namei**, in particular, are the names of UNIX operating system routines; the counters record the number of times that the respective routines are called. **Namei** is the routine that performs file system path searches. It searches the various directory files to get the associated i-number of a file corresponding to a special path. **Iget** is a routine called to locate the inode entry of a file (i-number). It first searches the in-core inode table. If the inode entry is not in the table, routine **iget** will get the inode from the file system where the file resides and make an entry in the in-core inode table for the file. **Iget** returns a pointer to this entry. **Namei** calls **iget**, but other file access routines also call **iget**. Therefore, counter **iget** is always greater than counter **namei**.

  Counter **dirblk** records the number of directory block reads issued by the system. It is noted that the directory blocks read divided by the number of **namei** calls estimates the average path length of files.

— **Runque, runocc, swpque** and **swpocc** record queue activities. They are implemented in the **clock.c** routine. At every one second interval, the clock routine examines the process table to see whether any processes are in core and in ready state. If so, the counter **runocc** is incremented and the number of such processes are added to counter **runque**. While examining the process table, the clock routine also checks whether any processes in the swap device are in ready state. The counter **swpocc** is incremented if the swap queue is occupied and the number of processes in swap queue is added to counter **swpque**.

— **Readch** and **writech** record the total number of bytes (characters) transferred by the *read* and *write* system calls respectively.

— There are six counters monitoring terminal device activities. **Rcvint**, **xmtint** and **mdmint** are counters measuring hardware interrupt occurrences for receiver, transmitter and modem individually. **Rawch**, **canch** and **outch** count number of characters in the raw queue, canonical queue and output queue. Characters generated by devices operating in the *cooked* mode, such as terminals, are counted in both **rawch** and (as edited) in **canch**, but characters from raw devices, such as communication processors, are counted only in **rawch**.

— Counters **msg** and **sema** record message sending and receiving activities and semaphore operations, respectively (refer to manual entries *msg*(2) and *sema*(2)).

— As to the I/O activity for a disk or tape device, four counters are kept for each disk or tape drive in the device status table. Counter **io_ops** is incremented when an I/O operation has occurred on the device. It includes block I/O, swap I/O and physical I/O. **Io_bcnt** counts the amount of data transferred between the device and memory in blocks. **Io_act** and **io_resp** measure the active time and response time of a device in time ticks. The device active time includes the device seeking, rotating and data transferring times, while the response time of an I/O operation is from the time the I/O request is queued to the device, to the time when the I/O completes.

— Counters **inodeovf**, **fileovf**, **textovf** and **procovf** are extracted from **_syserr** structure. When an overflow occurs in any of the inode, file, text and process tables, the corresponding overflow counter is incremented.

3. SYSTEM ACTIVITY COMMANDS

The System Activity Package provides three commands for generating various system activity reports and one command for profiling disk activities. These tools facilitate observation of system activity:

— during a controlled stand alone test of a large system,

— during an uncontrolled run of a program to observe the operating environment, and

— during normal production operation.

Commands *sar* and *sag* permit the user to specify a sampling interval and number of intervals for examining system activity, and then to display the observed level of activity in tabular or graphical form. The *timex* command reports the amount of system activity that occurred during the precise period of execution of a timed command. The *sadp* command allows the user to establish a sampling period during which access location and seek distance on specified disks are recorded and later displayed as a tabular summary or as a histogram.

— *Sar* command:
It can be used in two ways:

- When the frequency arguments *t* and *n* are specified, it invokes the data collection program *sadc* to sample the system activity counters in the operating system every *t* seconds for *n* intervals and generates system activity reports in real time. Generally it is desirable to include the option to save the sampled data in a file for later examination.

  The format of the data file is shown in *sar*(8). In addition to the system counters, a time stamp is also included. It gives the time at which the sample was taken.

- If no frequency arguments are supplied, it generates system activity reports for a specified time interval from an existing data file that was created by *sar* at an earlier time.

A convenient usage is to run *sar* as a background process, saving its samples in a temporary file, but sending its standard output to /dev/null. Then an experiment is conducted, after which the system activity is extracted from the temporary file. The *sar*(1) manual entry describes the usage and lists various types of reports. Attachment 3 gives formulae for deriving each reported item.

— *Sag* command:
Sag displays system activity data graphically. It relies on the data file produced by a prior run of *sar*, after which any column of data, or the combination of columns of data of the *sar* report can be plotted. A fairly simple but powerful command syntax allows the specification of cross plots or time plots. Data items are selected using the *sar* column header names. The *sag*(1G) manual entry describes its options and usage.

The system activity graphical program invokes *graph* and *tplot* commands to have the graphical output displayed on any of the terminal types supported by *tplot*.

— *Timex* command:
The *timex* command is an extension of the *time*(1) command. In addition to giving the time information, it also prints a system activity report derived from the system counters.

The manual entry *timex*(1) explains its usage. It should be emphasized that the **user** and **sys** times reported in the second and third lines are for the measured process itself including all its children, while the remaining data (including the **cpu user %** and **cpu sys %**) are for the entire system.

While the normal use of *timex* will probably be to measure a single command, multiple commands can also be timed; either by combining them in an executable file and timing it, or more concisely, by typing:

```
timex sh −c "cmd1; cmd2; ... ;"
```

This establishes the necessary parent-child relationships to correctly extract the user and system times consumed by *cmd1*, *cmd2*, ... (and the shell).

— *Sadp* command:
Sadp is a user level program that can be invoked independently by any user. It requires no storage or extra code in the operating system and allows the user to specify which disks are to be monitored. The program is reawakened every second, reads system tables from **/dev/kmem**, and extracts the required information. Because of the one second sampling, only a small fraction of disk requests are observed, however, comparative studies have shown that the statistical determination of disk locality is adequate when sufficient samples are collected.

In the operating system, there is an *iobuf* for each disk drive. It contains two pointers which are head and tail of the I/O active queue for the device. The actual requests in the queue may be found in three buffer header pools: system buffer headers for block I/O reguests, physical buffer headers for physical I/O reguests and swap buffer headers for swap I/O. Each buffer header has a forward pointer which points to the next request in the I/O active queue and a backward pointer which points to the previous request.

Sadp snapshots the *iobuf* of the monitored device and the three buffer header pools once every second during the monitoring period. It then traces the requests in the I/O queue and records the disk access location and seek distance in buckets of 8 cylinder increments. At the end of monitoring period, it prints out the sampled data. The output of *sadp* can be used to balance load among disk drives and to rearrange the layout of a particular disk pack. The usage of this command is described in manual entry *sadp*(1).

4. DAILY REPORT GENERATION

The previous section described the commands available to users to initiate activity observations. It is probably desirable for each installation to routinely monitor and record system activity in a standard way for historical analysis. This section describes the steps that a system administrator may follow to automatically produce a standard daily report of system activity.

— Facilities:

- *sadc* — the executable module of **sadc.c** (see Attachment 1) which reads system counters from **/dev/kmem** and records them to a file. In addition to the file argument, two frequency arguments are usually specified to indicate the sampling interval and number of samples to be taken. In case no frequency arguments are given, it writes a dummy record in the file to indicate a system restart.
- *sa1* — the shell procedure that invokes *sadc* to write system counters in the daily data file **/usr/adm/sa*dd*** where *dd* represents the day of the month. It may be invoked with sampling interval and iterations as arguments.
- *sa2* — the shell procedure that invokes the *sar* command to generate daily report **/usr/adm/sa/sar*dd*** from the daily data file **/usr/adm/sa/sa*dd***. It also removes daily data files and report files when they are over 7 days old. The starting and ending times and all report options of *sar* are applicable to *sa2*.

— Suggested operational setup:
It is suggested that the *cron*(1M) control the normal data collection and report generation operations. For example, the sample entries in **/usr/lib/crontab**:

```
0 * * * 0,6 su sys -c "/usr/lib/sa/sa1"
0 18-7 * * 1-5 su sys -c "/usr/lib/sa/sa1"
0 8-17 * * 1-5 su sys -c "/usr/lib/sa/sa1 1200 3"
```

would cause the data collection program *sadc* to be invoked every hour on the hour. Moreover, depending on the arguments presented, it writes data to the data file once or 3 times at every 20 minutes. Therefore, under the control of *cron*(1M), the data file is written every 20 minutes between 8:00 and 18:00 on weekdays and hourly at other times.

Note that data samples are taken more frequently during prime time on weekdays to make them available for a finer and more detailed graphical display. It is suggested that *sa1* be invoked hourly rather than invoking it once every day, this ensures that no matter when the system crashes, the data will be collected within an hour after the system is restarted.

Because system activity counters restart from zero when the system is restarted, a special record is written on the data file to reflect this situation. This process is accomplished by invoking within **/etc/rc** when going to multi-user state:

```
su adm -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

Cron(1M) also controls the invocation of *sar* to generate the daily report via shell procedure *sa2*. One may choose the time period at which the daily report is to cover and which groups of system activity are to be reported. For instance, if:

```
0 20 * * 1-5 su sys -c "/usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uybd"
```

is an entry in **/usr/lib/crontab**, *cron* will execute the *sar* command to generate daily reports from the daily data file at 20:00 on weekdays. The daily report reports the CPU utilization, terminal device activity, buffer usage and device activity every hour from 8:00 to 18:00.

In case of a shortage of the disk space or for any other reason, these data files and report files can be removed by the super-user. The manual entry *sar*(8) describes the daily report generation procedure.

5. ACKNOWLEDGEMENTS

L. A. Wehr is responsible for the set of system activity counters incorporated in the UNIX Time-Sharing System. The author wishes to acknowledge his discussions and help in providing a test environment during the development of the *sar* command. The output format of the *sadp* command is adopted from *iostat* of CB/UNIX. Thanks are due to T. Cook and D. DeJager for their cooperation in making the verification of the result generated by *sadp* possible. Finally, the author gratefully acknowledges D. A. DeGraaf's contribution to this package and appreciates A. Petruccelli's efforts in making this memo in its printable format.

ATTACHMENT 1

The source files and shell programs of the system activity package are in directory **/usr/src/cmd/sa**.

sa.h	the system activity header file which defines the structure of data file and device information for measured devices. It is included in **sadc.c, sar.c** and **timex.c**.
sadc.c	the data collection program that accesses **/dev/kmem** to read the system activity counters and writes data either on standard output or on a binary data file. It is invoked by the *sar* command generating a real time report. It is also invoked indirectly by entries in **/usr/lib/crontab** to collect system activity data.
sar.c	the report generation program that invokes *sadc* to examine system activity data and generate reports in real time, and save the data to a file for later usage. It may also generate system activity reports from an existing data file. It is invoked indirectly by *cron* to generate daily reports.
saghdr.h	the header file for **saga.c** and **sagb.c**. It contains data structures and variables used by **saga.c** and **sagb.c**.
saga.c & sagb.c	the graph generation program that first invokes *sar* to format the data of a data file in a tabular form, and then displays the *sar* data in graphical form.
sa1.sh	the shell procedure that invokes *sadc* to write data file records. It is activated by entries in **/usr/lib/crontab**.
sa2.sh	the shell procedure that invokes *sar* to generate the report. It also removes the daily data files and daily report files when they are a week old. It is activated by an entry in **/usr/lib/crontab** on weekdays.
timex.c	the program that times a command and generates a system activity report.
sadp.c	the program that samples and reports disk activities.

ATTACHMENT 2

```
struct sysinfo {
        time_t          cpu[4];
#define CPU_IDLE        0
#define CPU_USER        1
#define CPU_KERNAL      2
#define CPU_WAIT        3
        time_t          wait[3];
#define W_IO            0
#define W_SWAP          1
#define W_PIO           2
        long            bread;
        long            bwrite;
        long            lread;
        long            lwrite;
        long            phread;
        long            phwrite;
        long            swapin;
        long            swapout;
        long            bswapin;
        long            bswapout;
        long            pswitch;
        long            syscall;
        long            sysread;
        long            syswrite;
        long            sysfork;
        long            sysexec;
        long            runque;
        long            runocc;
        long            swpque;
        long            swpocc;
        long            iget;
        long            namei;
        long            dirblk;
        long            readch;
        long            writech;
        long            rcvint;
        long            xmtint;
        long            mdmint;
        long            rawch;
        long            canch;
        long            outch;
        long            msg;
        long            sema;
};
```

ATTACHMENT 3

The derivation of the reported items of a report is given in this attachment. Each item discussed below is the data difference sampled at two distinct times *t2* and *t1*.

— CPU utilization:

%-of-cpu-x = cpu-x / (cpu-idle + cpu-user + cpu-kernel + cpu-wait) * 100

where cpu-x is cpu-idle, cpu-user, cpu-kernel (cpu-sys) or cpu-wait.

— Cached hit ratio:

%-of-cached-I/O = (logical-I/O − block-I/O) / logical-I/O * 100

where cached I/O is cached read or cached write.

— disk or tape I/O activity:

%-of-busy = I/O-active / (t2 − t1) * 100;
avg-queue-length = I/O-resp / I/O-active;
avg-wait = (I/O-resp − I/O-active) / I/O-ops;
avg-service-time = I/O-active / I/O-ops.

— queue activity:

avg-x-queue-length = x-queue / x-queue-occupied-time;
%-of-x-queue-occupied-time = x-queue-occupied-time / (t2 − t1);

where x-queue is run queue or swap queue.

— The rest of system activity:

avg-rate-of-x = x / (t2 − t1)

where x is swap in/out, blks swapped in/out, terminal device activities, read/write characters, block read/write, logical read/write, process switch, system calls, read/write, fork/exec, iget, namei, directory blocks read, disk/tape I/O activities, message or semaphore activities.

January 1981