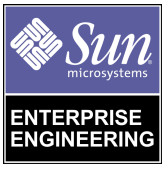
Processing Accounting Data into Workloads

By Adrian Cockcroft - Enterprise Engineering

Sun BluePrints™ OnLine - October 1999



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-3752-10
Revision 01, October 1999

Processing Accounting Data into Workloads

Solaris™ operating system accounting data records who ran what command when and how much resource the command used. The information is normally accumulated into two reports, a total for each user and a total for each command. Workload analysis breaks down groups of users and groups of commands into defined workloads. By processing the accounting data in this way workload based accounting logs can be produced that aid in the capacity planning process.

This article is based in part on the measurement chapter of the Resource Management BluePrint. It extends the information about Solaris operating system accounting to include code examples that extract the data in a usable format and pattern match it into workloads.

Measurements

If you want to manage a resource, you must measure its usage.

The generic types of measurements that you can obtain directly or indirectly via related measurements are throughput, utilization, queue length, and response time. These measurements are made at several levels, including business operations, application, user, system, network, process, and device level.

One problem in documenting details of Solaris operatng environment measurements is that far more measurements are available than the standard tools such as `vmstat`, `sar`, and `ps` display. Most commercial performance tools read the kernel directly and have access to the full range of measurements, although they do not use all of them. The Sun Enterprise SyMON product collects a large proportion of the available measurements. The most convenient way to explore this data is to use the SE Toolkit. This is a freely available but unsupported tool that provides full access to

all the raw data sources in the Solaris operating environment, and generates higher level processed data. It is used to prototype ideas that can then be promoted for incorporation in products, in particular for the Sun Enterprise SyMON 2.0 tool. In this article code that processes accounting information is provided as an add-on to the SE Toolkit, but the code can simply be converted for use in compiled C programs as well.

The SE Toolkit

The SE Toolkit is based on a C language interpreter that is extended to make available all the Solaris operating environment measurement interfaces in an easy form. All the code that takes metrics and processes them is provided as C source code to run on the interpreter so that it is easy to trace back and see where data comes from and how it is processed. You can then write your own programs in any language to obtain the same data. The SE Toolkit has been jointly developed by Richard Pettit and Adrian Cockcroft as a “spare time” activity since 1993. Richard worked at Sun but is currently at Foglight Software, and Adrian is the author of this article. The SE Toolkit can be downloaded from `http://www.sun.com/sun-on-net/performance/se3`. Detailed information and examples of how to write code that reads the kernel and the SE Toolkit itself can be found in *Sun Performance and Tuning, Java and the Internet*, by Adrian Cockcroft and Richard Pettit, Sun Press 1998.

Measurement Levels

Measurements can be classified into several levels. Data from lower levels is aggregated and merged with new data as more valuable higher level measurements are produced.

- Business operations

  Business workloads are broadly based and are not only computer oriented. Use a form that makes sense to managers and non-technical staff to represent the part of the business that is automated by the computer system.

- Application

  The business operation can be broken down into several applications such as sales and distribution, e-commerce web service, email, file and print. Application specific measurements include order entry rate, emails relayed, web server response time, and so on.

- User

  Each class of user interacts with several applications. The number of users and the work pattern of each class of users should be understood.

- Network

  Networks connect the users to the applications and link together multiple systems to provide applications that are replicated or distributed. Measure traffic patterns and protocol mixes for each network segment.

- System

  System level measurements show the basic activity and utilization of the memory system and CPUs. Some network measurements such as TCP/IP throughput are also available on a per system basis. Per process activity can be aggregated at a per system level then combined with network measurements to measure distributed applications.

- Process

  Process measurements show the activity of each user and each application. Current process activity can be monitored. Accounting logs provide a record of who ran what when.

- Device

  Devices such as disks and network interfaces are measured independently and aggregated at the system level. There are few ways to link the usage of a device to a process or a user automatically, so detailed information about the configuration of devices and the usage of file systems by applications is needed.

This article concentrates exclusively on improved processing of process accounting data so it can be integrated with other measurement sources.

Accounting

Who ran what, when, and how much resource was used?

Many processes have very short life spans. You cannot see such processes with `ps`, but they may be so frequent that they dominate the load on your system. The only way to catch them is to have the system keep a record of every process that has run, who ran it, what was it, when it started and ended, and how much resource it used. The answers come from the system accounting subsystem. While you may have some concerns about accounting because of the “big brother is watching you” connotation or the cost of additional overhead, the information is important and valuable. The overhead of collecting accounting data is *always present but is insignificant*. When you turn on accounting, you are just enabling storage of a few bytes of useful data when a process exits.

Accounting data is most useful when measured over a long period of time. This temporal information is useful on a network of workstations as well as on a single, time-shared server. From this information, you can identify how often programs run, how much CPU time, I/O, and memory each program uses, and what work patterns throughout the week look like.

Basic System Accounting

To enable accounting to start immediately, enter the three commands shown below. Check out the accounting section in the *Solaris System Administration Answerbook* and see the `acctcom` command. Add some `crontab` entries to summarize and checkpoint the accounting logs. Collecting and checkpointing the accounting data itself puts a negligible additional load onto the system, but the summary scripts that run once a day or once a week can have a noticeable effect, so schedule them to run outside business hours.

```
# ln /etc/init.d/acct /etc/rc0.d/K22acct
# ln /etc/init.d/acct /etc/rc2.d/S22acct
# /etc/init.d/acct start
Starting process accounting
```

Your `crontab` file for the `adm` user should contain the following:

```
# crontab -l adm
#ident  "@(#)adm        1.5     92/07/14 SMI"   /* SVr4.0 1.2   */
#min    hour    day     month   weekday
0       *       *       *       *       /usr/lib/acct/ckpacct
30      2       *       *       *       /usr/lib/acct/runacct 2> /var/adm/
acct/nite/fd2log
30      9       *       *       5       /usr/lib/acct/monacct
```

You get a daily accounting summary, but the best one to keep track of is the monthly one stored in `/var/adm/acct/fiscal`. Following is an excerpt from `fiscrpt07`, which is the report for July on this desktop system.

```
Jul 26 09:30 1996                    TOTAL COMMAND SUMMARY FOR FISCAL 07 Page 1

                TOTAL COMMAND SUMMARY
 COMMAND  NUMBER        TOTAL    TOTAL      TOTAL      MEAN     MEAN     HOG         CHARS  BLOCKS
    NAME    CMDS     KCOREMIN  CPU-MIN   REAL-MIN    SIZE-K  CPU-MIN  FACTOR        TRNSFD    READ

  TOTALS   26488  16062007.75  3960.11  494612.41   4055.95     0.15    0.01   17427899648   39944

mae           36   7142887.25  1501.73    2128.50   4756.45    41.71    0.71    2059814144    1653
sundgado      16   3668645.19   964.83    1074.34   3802.36    60.30    0.90     139549181      76
Xsun          29   1342108.55   251.32    9991.62   5340.18     8.67    0.03    2784769024    1295
```

xlock	32	1027099.38	726.87	4253.34	1413.04	22.71	0.17	4009349888	15
fountain	2	803036.25	165.11	333.65	4863.71	82.55	0.49	378388	1
netscape	22	489512.97	72.39	3647.61	6762.19	3.29	0.02	887353080	2649
maker4X.	10	426182.31	43.77	5004.30	9736.27	4.38	0.01	803267592	3434
wabiprog	53	355574.99	44.32	972.44	8022.87	0.84	0.05	355871360	570
imagetoo	21	257617.08	15.65	688.46	16456.60	0.75	0.02	64291840	387
java	235	203963.64	37.96	346.35	5373.76	0.16	0.11	155950720	240
aviator	2	101012.82	22.93	29.26	4406.20	11.46	0.78	2335744	40
se.sparc	18	46793.09	19.30	6535.43	2424.47	1.07	0.00	631756294	20
xv	3	40930.98	5.58	46.37	7337.93	1.86	0.12	109690880	28

The commands reported are sorted by KCOREMIN, which shows the amount of CPU time used and the amount of RAM used while the command was active. CPU-MIN is the number of minutes of CPU time. REAL_MIN is the elapsed time for the commands. SIZE-K is an average value for the RSS over the active lifetime of the process. It does not include times when the process was not actually running. (In Solaris 2.4 and earlier releases, a bug causes this measure to be invalid.) HOG FACTOR is the ratio of CPU-MIN to REAL-MIN; a high factor means that this command hogs the CPU whenever it is running. CHARS TRNSFD counts the number of characters read and written. BLOCKS READ counts data read from block devices (basically, local disk file system reads and writes). The underlying data that is collected can be seen in the `acct`(4) manual page. The data structure is very compact—around 40 bytes, as shown in FIGURE 1:

```
DESCRIPTION
     Files produced as a result of calling acct(2) have records
     in the form defined by <sys/acct.h>, whose contents are:

    typedef ushort  comp_t;     /* pseudo "floating point" representation */
                                   /* 3-bit base-8 exponent in the high */
                                   /* order bits, and a 13-bit fraction */
                                   /* in the low order bits. */

     struct  acct
     {
             char    ac_flag;     /* Accounting flag */
             char    ac_stat;     /* Exit status */
             uid_t   ac_uid;      /* Accounting user ID */
             gid_t   ac_gid;      /* Accounting group ID */
             dev_t   ac_tty;      /* control tty */
             time_t  ac_btime;    /* Beginning time */
             comp_t  ac_utime;    /* accounting user time in clock */
                                  /* ticks */
             comp_t  ac_stime;    /* accounting system time in clock */
                                  /* ticks */
           comp_t  ac_etime;   /* accounting total elapsed time in clock */
                                  /* ticks */
             comp_t  ac_mem;      /* memory usage in clicks (pages) */
             comp_t  ac_io;       /* chars transferred by read/write */
```

FIGURE 1 Accounting Data Format

```
        comp_t  ac_rw;       /* number of block reads/writes */
        char    ac_comm[8];  /* command name */
};
```

FIGURE 1 Accounting Data Format

Processing Accounting Records

The accounting summarization destroys the raw data files after they have been processed. For more advanced processing the raw data accounting records must be extracted and processed as they are written. The first step is to expand the compressed accounting record into a more usable format, so you can pattern match the user and command names into defined workloads.

The 16 bit “floating point” data representation is the first stumbling block. The code for processing it into a more usable form follows:

```
timeval_t comp2timeval(comp_t ct) {
        int e;
        ulonglong f;
        timeval_t t;
        e = (ct >> 13) & 07;
        f = ct & 017777;
        while (e > 0) {
                f <<= 3;
                e -= 1;
        }
        t.tv_sec = f / 100;
        t.tv_usec = (f % 100)*10000;
        return t;
}
```

Using this routine, the data can be expanded into a data structure that uses more conventional types to hold the data. This is shown in the following example:

```
#define TTY_NAME_LEN    16

struct expanded_acct {
        char            ac_flag;
        char            ac_stat;
        ulong           ac_uid;
        ulong           ac_gid;
        char            ac_tty_name[TTY_NAME_LEN];
        long            ac_btime;
        timeval_t       ac_utime;
        timeval_t       ac_stime;
        timeval_t       ac_etime;
        ulonglong       ac_mem;
        ulonglong       ac_char_rw;
        ulonglong       ac_bloc_rw;
        char            ac_comm[8];
};
```

Displaying Raw Accounting Records

Some code was written using the SE Toolkit to monitor the `/var/adm/pacct` file, which contains newly written accounting records. It expands each record and prints out the data it finds every five seconds. Some example output follows.

```
% se show_acct.se
  UID   GID    start   usr_tm   sys_tm  elps_tm KB_mem   KB_rw blck_io      CMD flags
    0     0 22:23:04     0.00     0.00     0.00      0       0       0 cfgserve FORK XSIG
    0     0 22:23:04     0.01     0.00     0.01   1112       0       0 cfgserve FORK XSIG
    0     0 22:23:04     0.05     0.02     0.07   1148       9       0     xget XSIG
    0     0 22:23:04     0.00     0.00     0.00      0       0       0 cfgserve FORK XSIG
    0     0 22:23:04     0.01     0.00     0.01   1136       0       0     grep XSIG
    0     0 22:23:04     0.00     0.02     0.02    796       0       0       rm XSIG
    0     0 22:23:04     0.00     0.01     0.01   1128       0       0       rm XSIG
    0     0 22:23:04     0.01     0.06     0.55    824       4       0 cfgserve XSIG
 9506    10 22:23:08     0.00     0.01     0.01    952       0       0    uname XSIG
 9506    10 22:23:08     0.00     0.00     0.00      0       0       0       se FORK XSIG
 9506    10 22:23:08     0.00     0.01     0.02   1408       0       0      cut XSIG
 9506    10 22:23:08     0.01     0.00     0.01   1136       0       0    uname XSIG
 9506    10 22:23:08     0.00     0.01     0.03   1536       0       0     mach XSIG
 9506    10 22:23:08     0.02     0.01     0.03    973       0       0     expr XSIG
 9506    10 22:23:08     0.01     0.01     0.02    772       0       0     expr XSIG
 9506    10 22:23:08     0.00     0.04     0.14    790       1       0 se.sparc XSIG
```

The columns are the process user and group numbers, the process start time, the user and system time CPU consumed, the elapsed time between process start and exit, the average memory usage displayed as kilobytes, the total amount of read and write system call data displayed as kilobytes, the number of blocks of metadata I/O (i.e. directory and indirect blocks) the command name, and the flags set at termination.

Pattern Matching Workloads

A workload can be defined by specifying a pattern of user names and command names. The user name can be simply obtained from the UID. The existing process based workload class code from the SE Toolkit was used as a basis. This code uses environment variables to configure the workload class. The same environment variables can be used so that one workload specification will be used to match both live processes and accounting data. The resulting tool is similar to the process-based workload analyzer `pw.se`. But instead of giving a snapshot of live processes on the system, it captures all the processes that have exited.

The command can be configured using a wrapper script like `acctw.sh`, shown in the following example:

```
#!/bin/csh

setenv PW_CMD_0 'httpd'
setenv PW_CMD_1 'se.sparc'
setenv PW_CMD_2 'dtmail'
setenv PW_CMD_3 'dt'
setenv PW_CMD_4 'roam'
setenv PW_CMD_5 'netscape'
setenv PW_CMD_6 'X'
setenv PW_USER_7 $USER
setenv PW_USER_8 'root'
setenv PW_COUNT 10
exec /opt/RICHPse/bin/se acctw.se 10
```

When it runs, the workload specification and matching process load is shown as follows:

```
% ./acctw.sh
12:40:10
wk  command    user procs    usr    sys   elps   ramK ch_rwK  bufio
 0    httpd             0   0.00   0.00   0.00      0      0      0
 1 se.sparc             0   0.00   0.00   0.00      0      0      0
 2   dtmail             1   0.14   0.07   0.76   5340     17      2
 3       dt             0   0.00   0.00   0.00      0      0      0
 4     roam             1   0.16   0.07   1.15   4729     18      4
 5 netscape             0   0.00   0.00   0.00      0      0      0
 6        X             0   0.00   0.00   0.00      0      0      0
 7          adrianc     0   0.00   0.00   0.00      0      0      0
 8             root    31   0.23   0.35   1.22 131270     40      0
 9                      0   0.00   0.00   0.00      0      0      0
10    Total       *    33   0.53   0.49   3.13   9122     75      6
```

This data can be collected throughout the day, then plotted or processed on a per workload basis to see the trends in activity levels of each workload.

Solaris Resource Manager™ Software Accounting

Solaris Resource Manager™ (SRM) software provides the ability to generate accounting data based on resource usage. This is made available by the accumulation of resource information in the lnode tree. Although the SRM software does not provide resource accounting, it does provide the data and data-extraction tools used to develop a system to generate accounting (or billing) information. See the SRM chapter of the Resource Management Blueprint for more information on SRM Accounting.

Network Accounting—NetFlow

Solaris Bandwidth Manager software has built-in support for Cisco NetFlow™ software. This feature allows for detailed network measurements that can be sent to other software packages that can process and analyze this data.

Besides Solaris Bandwidth Manager software, NetFlow data collection is supported on a variety of Cisco devices, such as the Cisco 7000 and 7500 series routers.

QEach NetFlow datagram record contains detailed flow information, such as how many bytes were in the flow, how many packets, the duration, source and destination IP address, and so on.

Cisco offers applications that read and process NetFlow datagrams. NetFlow FlowCollector™ is a NetFlow datagram consumer for one or more NetFlow devices. These devices simply point to the host and port number on which the FlowCollector software is running. The FlowCollector will aggregate this data, do preprocessing and filtering, and provide several options to save this data to disk (such as flat files). Other applications such as network analyzing, planning, and billing can use these files as input.

NetFlow FlowAnalyzer™ is an application that uses the output from NetFlow FlowCollector. It provides elaborate processing, graphing, and reporting options for network analysis, planning, trouble shooting and more.

You can run the Solaris Bandwidth Manager software in statistics mode, where it will process packets but not provide the priority scheduling. By using the NetFlow output of the Solaris Bandwidth Manager software, you can obtain detailed network measurements that are not easily measured otherwise. For example, if you want to know how much bandwidth is used by a specific application or user, NetFlow data analysis can provide this. The flow data can also provide latency information of specific applications. Trend data can show what the long term impact is of the deployment of certain applications.

For more information regarding Cisco NetFlow, see `http://www.cisco.com/warp/public/732/netflow/index.html`.

Summary

There are several sources of accounting information, and in particular the system accounting log is an under-utilized source of useful performance information. The code that implements the extended accounting processing is available as an add-on package that works with existing releases of the SE Toolkit, and it will be included in future releases. [`http://www.sun.com/sun-on-net/performance/se3`]

Author's Bio: Adrian Cockcroft

The author of Sun Performance And Tuning, Adrian is an accomplished performance specialist for Sun Microsystems and recognized worldwide as an expert on the subject.