Highly Automated Low Personnel System Administration in a Wall Street Environment

Harry Kaplan – Sanwa Financial Products Co., L.P.

ABSTRACT

Administering a small system running intensive financial applications with the demand for twenty four hour coverage is nearly impossible for a single person. Moreover, most solutions to automated system administration are not clever enough to pinpoint the problem and deliver the required action to the responsible person in a timely manner. Our system evolved by integrating small utilities which function as information gathering tools, notifying tools and action tools. Vital information to us means: the users on each system, load on each server, temperature in the computer room, remaining swap and UNIX file system space on each machine in the company, health of our market data ticker plant, and state of backup tapes. Notifying tools include electronic mail, electronic postit notes, a digitized dial-up voice messaging system, and alphanumeric pagers. Action tools include fairly standard UNIX utilities such as killing runaway application processes, removing core dumps, and, as such, require little further description.

Introduction

This paper focuses upon routine information gathering and notifying tools which help a Wall Street systems administration department monitor critical resources on each workstation and server worldwide. No news is good news, so most of these routines run silently 24 hours a day until a problem is discovered. Aside from alerting the system administrators to take action, one unique feature of our environment is to "empower" the end user with their own tools for monitoring workstation resources.

Motivation

The demands upon a system administrator in a Wall Street firm of some seventy-five employees worldwide are markedly different from that in other environments. Throughout various periods in the four year existence of the firm, I have been the sole administrator of one local and three foreign offices, designing everything from our wide area network to electronic post-it note facilities. Having a homogeneous environment of Sun Sparcstations running SunOS from the very start was a boon, but before we were able to grow the systems department to a staff of three, there was little time for complex programming tasks. Instead, the goal was to integrate as many standard, off the shelf products as we could purchase to save systems development time. Unfortunately, I discovered there is very little available to automate administration tasks, so we ended up creating tools which can serve as the thousand eyes and ears of our system. These tools help to identify trouble spots before they develop and even aid the the user herself in identifying problem areas.

This paper presents a description of the utilities we have developed to help us maintain a near-zero downtime record on the trading floor, as well as survive the onslaught of component cave-in, memory saturation and even air conditioner failure in the computer room.

Small tools are built upon other simple tools. For example, here is the source code for our shell-level memory checking tool called "memory", which handles both Solaris environments and error conditions fairly painlessly. It also depends on a small script to determine the operating sytem level. There are numerous ways of doing the latter (checking for /vmunix for example) under SunOS, we chose a fairly arbitrary one, see Figures 1 and 2.

Another small script checks all ufs (4.2) file systems for disk space. Yet another tests that a given database server is active and healthy.

Information Gathering Tools

System Monitor

This is a cron job which runs each morning and mails the output to sysadmin, including snapshots of recent logs of the various things that run overnight. We found that the difficult part was to do a last log of all logins on all machines for the last two "interesting" days. A day is interesting if a new person has logged on to that machine in the interim. Figure 3 is the coding fragment that performs this trick.

This tool began as a single script which ran on every machine in the company. As the number of machines grew, this particular piece of mail became a chore to scroll through casually even on that first strong cup of morning coffee. Over the years most of the subroutines have been broken out into separately labeled pieces of mail. The systems administrator must therefore browse through several dozen pieces of mail each morning to get a complete picture of the system status.

```
#!/bin/sh
if is_Solaris2
then
    MEM=`/usr/sbin/swap -s`
else
    MEM=`/usr/etc/pstat -s`
fi
OUT=`echo $MEM |
      awk '{ print $(NF-1) }'
    | sed 's/....$//'`
if [ "$OUT" -eq 0 ]
then
    echo 0
else
    echo $OUT
fi
```

Figure 1: Source for ‘‘memory’’

```
#!/bin/sh
case `uname -r` in
    5.1|5.2|5.3) exit 0 ;;
    *) ;;
esac
exit 1
```

Figure 2: Source for ‘‘is_Solaris2’’

```
rsh $host last |
awk '{
    lastday = day
    day = substr($0,37,3)
    if (day != lastday)
        past++
    if (past > 2)
            exit
    print }'
```

Figure 3: Fragment of ‘‘sys_monitor’’ source

We researched some of the packages available to create separate mailboxes into which these reports could be sorted, but the subject lines are not yet uniform enough to permit an automated distinction between system monitor mail and other messages intended for the system administration staff. In any case, speed reading mail messages becomes second nature for the well-seasoned system administrator, and we would not want to eliminate this vital skill from the growing portfolio of odd talents required for the job.

These system monitor scripts which collect information generally operate in a passive mode: the output must be browsed in order to be useful. This is fine for certain types of information. Recently, however, we have added another layer of reporting which actively links certain log monitors (such as the database server dumps) to notification routines which search for errors of various kinds. One interesting error occurs when routines which are supposed to complete do not. A new procedure we have implemented watches the dozen or so databases as they are checked and dumped. If the database checking routine hangs or the dump does not complete, an appropriate notification routine is invoked. We have learned that in this specific case, trade entry may be jeopardized the following morning, so we would "like" to be woken out of bed by a buzzing pager when this occurs.

Up Watcher

Once an hour, every machine on our network at each branch office is combed by the Up Watcher, and problems or potential problems are reported via pager or email. A remarkable amount of applications or systems level errors can be caught just by monitoring disk space and swap space events. Three times a day it is run in verbose mode which reports that everything is ok (along with computer room temperature) even if no problems are detected. This helps to insure that we know if the pager interface has crashed, or if our paging message vendor has suddenly closed shop and gone out of business.

Figure 4 is an outline of its logic.

```
For each machine on the network
 If it is not known to be down
   [ no entry in sundown ]
 If it is alive
   [ can ping and run simple command ]
     Check swap space
        > MIN_SWAP (10 MB)
     Check each 4.2 file system
        < MAX_SPACE (92%)
     Check user load
        < MAX_UPTIME (8)
For each of the dozen local
database servers
   Check that each database is
    up and running
For each of the (two) foreign sites
 on high speed IP link:
   Check that the link is up
   (ping a remote machine)
   Check that the foreign database
   servers are up
For each remote uucp site,
   Check to see if last successful
    connection > 1 hour
   Check to see if > MAX_WAIT (10 pieces)
   of outstanding mail are waiting
```

Figure 4: Logic schema of ‘‘up watch’’

This system works well for us, except that you tend to get an annoying hourly message when the cleaning person has knocked out the ethernet plug at 8 p.m. on a certain workstation on the trading floor. To this end, we created the /usr/local/sundown directory. By touching a file name corresponding to the machine that has been downed, an annoying hourly message that the machine is down will be suppressed.

The sundown scheme is also useful during crunch time on the development server, where we do not wish to know each hour exactly how perilously close to 110% the developer partitions have become, or what new exponent of user load has been reached during heavy compiles.

One interesting feature of the Up Watcher is that users are alerted and can take actions themselves. If Up Watch determines that the machine is low on memory the following electronic message is sent to the user:

Your workstation $i is precariously low on memory. You have only $DOWNTO Megabytes of Virtual Memory Remaining. This may adversely affect any applications currently running. Please save and quit any LARGE applications you are not using. Thank you. This message was generated automatically. A system administrator has been alerted.

If the workstation is overloaded, the following message is sent to the user:

Your workstation $i is overloaded. There are currently an average of $LOAD process waiting for cpu time. One of them may be sick and need attention. Please contact a system administrator if you believe you are not running any heavy applications on this machine. Thank you. This message was generated automatically.

Feed Watcher

Similar to the Up Watcher, we run a feed watch three times a day at each office site where we are running our own in-house ticker plant to provide market data to the trading floor. It alerts us if any of the data feed provider machines have gone down or are not emitting any data. We even sample the type of data emitted to make sure that it is not garbage, and that it has been updating if the market is still open. Because we have distributed our market data feed system over many different server machines, Feed Watch also tells us which machine to log onto and which process to kill in order to restart a problematic market data daemon. It is often run manually if some problem in market data comes to the attention of the system administration staff, and helps to pinpoint exactly which component of the market data system is malfunctioning.

Adm Log Watcher

An adm log watcher on a designated workstation processes a tail -f on /var/adm/messages, and will alert us if an NFS server goes down for example, with the output

```
Apr 21 21:04:07 yukiguni vmunix: NFS
  server kendo not responding still trying.
```

In this case, an immediate trip to the office after hours is advised.

Wiztemp

Wiztemp is a digital thermometer we purchased[1], and its daemon was easily customized to interface with our pager system. The temperature is always appended to any error messages output by the Up Watcher, so that we are warned of any impending disaster due to a rise in temperature in the computer room. In London we even have a digital thermometer on the trading floor, because some of our vital equipment was at risk when the air conditioning was shut off at night.

Stand and Stare

This is not an automated information gathering utility per se, although at times it appears to be so. This particular monitor occurs at various random points in the day when a user will stand over near the systems administration area and stare off into space as though in a trance. After some coaxing, it can usually be determined that a workstation has "frozen" up in a manner devastating to their normal business demeanor. Often a large application has been core dumping, and by the time the "stand and stare" monitor has sounded, the situation is back to normal. In the case where this occurs often, the user is often able to skip the trance state and announce the problem with little or no coaxing from the systems personnel.

Notifying Tools

Electronic Mail

The backbone of any system diagnostics will always be electronic mail. We adhere to the general UNIX philosophy that if things are good, we don't want to hear about it. Unfortunately, some of our diagnostic jobs are not smart enough to tell what is bad in a given log, and we are inundated with over 100 system-related e-mail messages a day. As mentioned above, retrofitting all of these messages with uniform and intelligent subject lines which announce GOOD or BAD results would ease the burden immensely.

Some logs and diagnostic routines are written only to files, and not sent through the mail, or are sent through the mail needlessly. For example, each day we run a search to find every file on the system which has been modified in the last 24 hours. This output is stored in hierarchical directories so that it is easy to figure out which tapes to use to restore files at a later date. The output is still sent to us via electronic mail, however. I actually find it profitable to skim through the contents of this mail message now and then because it alerts us as to areas in the company growing rapidly and possibly in need of additional disk space soon.

SFP Postit Notes

Early on, we wrote a simple Open Look widget to deliver one-line postit notes to an end-user work station. The message appers in a yellow rectangle with a date at the top, along with its sender and destination and a single QUIT button. To send SFP postit notes, you simply mail to alias "postit" and the user(s) you wish to send to are entered on the subject line. The postit program is then triggered through the standard mechanism of a mail alias pipe.

Over the years, various enhancements have been added to the postit system. If the user is not logged on at any workstation, or her X-windows security disallows posting one of these widgets, the sender is informed through electronic mail that it was unable to post for one of these reasons. All postit messages are also relayed through regular mail as well, so they may be picked up remotely. Few systems monitoring routines actually use the postit notification mechanism, because serious problems tend to occur when one is away from the workstation, or when the workstation is not operable in the first place. Notifications for failure of minor systems procedures like a failure in the daily dump tape management system can, however, be brought to the attention of the relevant party through the postit note mechanism.

SFP Alphanumeric Pager Interface

The pager alias works identically to that of postit, except than in this case a group of programs actively take hold of a dedicated modem and deliver the mail message directly to our pager vendor's computer interface[2]. Our paging filter converts all newlines to spaces, and will create continuation chains if the message is longer than the eighty characters allowed per pager screen. The user id of the sender is prepended in parenthesis to the message for convenience. A "plus" indicates that the message will continue over more than one pager screen

The pager interface is used extensively to convey error messages which are flagged by the Up Watcher and Feed Watcher programs. Typical pager messages will look like Figure 5.

```
(Pager) zazen /data 96% temp is 64F

(dragon!pager) UV: All machines are up
     and running  [ Hong Kong office ]

(Pager) kaoru is down temp is 65F

(Pager) UK DB is DOWN 60.8F

(Ha Kaplan)+ Cannot get ahold of our
     service vendor, typical, left a

(cont'd) message with mbar to reboot
     tea tomorrow
```

Figure 5: Typical Pager Messages

The pager notification mechanism is vital to alerting us of system problems 24 hours a day. All systems administrators as well as our office manager and the developers in charge of the data feeds are required to carry their pagers with them at all times. During especially critical work periods one often sleeps with the pager next to one's pillow. I am capable of sleeping through my pager buzzing, but it wakes the dog who in turns wakes me. Sound sleepers are advised to find some suitable proxy method of responding to system emergencies.

SFP Computerfone

The computerfone[3] interface provides many diagnostic services from any touch tone phone. This tools consists of the vendor-supplied hardware, a shell script and directories of digitized voice prompts and responses. Of course, with this system, the system response to a query must be fully anticipated, because the shell script uses a simple word matching algorithm to map error messages to enunciation. It is prudent to record the phrase "unanticipated data" as a default when no phrases are matched, rather than falling silent.

Using the SFP Computerfone interface, a trader in our Hong Kong office is able to check the health of a deal database in New York if a problem is suspected. More frequently, the computerfone is used to check the temperature in the computer room from an ordinary telephone on weekends during summer air conditioner inconstancies. By working through the touch-tone menu system we created, you can access the functions shown in Figure 5.

```
Leave audiotext e-mail for a sysadmin
Page a sysadmin
Check system functions:
   1) Check to see when last e-mail
       went out to foreign offices
   2) Check temperature in
      the New York computer room
   3) Check to see if all database
      servers are up and healthy
   4) Check to see if all T1 links are
      up and running
   5) Check health of selected critical
      machines on the New York network
      as well as important machines in
      the foreign offices
```

Figure 6: SFP Computerfone Menu

Of course, a security code must be entered before the computerphone will provide data to an outside party. Care must be taken that no vital company information should be available through any dialup method, including voice.

User Front Ends

SFP MemoryTool

Putting system admin power in the hands of the end user is an idea which on the surface may seem dangerous or undesirable, but in our environment has saved many hours of systems department time. We designed the SFP MemoryTool to be a simple and effective way for the end user to monitor their own memory usage, to advise us if they feel they need more physical memory added to their system, or if they are dangerously close to exceeding the available swap space.

MemoryTool is an Open Look widget which looks like a thermometer when open. The top of the thermometer indicates 100% of the available swap space, and the physical memory limit is scaled automatically to the proportion of RAM to total swap. If the sum of all memory currently in use (pstat -s) is within physical RAM, the color of the thermometer is green. Once it exceeds this mark, as the mercury climbs up the memory tool it turns yellow. Within 10% of total available swap space it turns red. When iconified, the memory tool still changes color when crossing any of these thresholds. There is a button at the bottom of the thermometer which the user may click to get a detailed popup explaining all about how workstation memory works and how much her workstation is currently using. Figure 7 is an example explanation.

```
Your machine is loaded with 40M RAM
     (FAST ACCESS) MEMORY
You have a total possible memory usage
     (RAM AND SWAP DISK) of 117M
You are currently utilizing 12M actual
     memory
This means that you are utilizing
30% of your RAM (FAST ACCESS)
memory and that you are currently at
10% of your total possible memory
capacity
```

Figure 6: SFP Memorytool Sample Explanation

Utilizing the Memorytool, users monitor the memory their applications consume and can determine on their own when it is necessary to eliminate one app before starting up another.

Topmem

Some users on our system have never opened a shelltool except to run the "topmem" command. We teach them to identify and kill their own processes. Topmem is merely a filter we wrote which pipes the output of one invocation of the top[4] command and sorts it with regard to the resident application memory used by programs, avoiding displaying any processes using less than 200KB of memory. Programs which are heavily consuming memory resources are listed at the top, so the user can view at a glance the application memory map on a particular workstation.

To Do

We plan to isolate at least one workstation from both NIS and the automounter so that if any NIS or NFS servers are down it can tell us about it instead of freezing up and then telling us about it after the problem has been fixed. It would be nice to set up a queuing system for outgoing pages, but the collision rate has so far been less than the random message deletion rate of our pager vendor anyways, so this hasn't been a priority. It would also be nice to find a pager vendor who assigned various priority levels to system problems. It is annoying to be woken up only because a midnight compile triggered by a cron job has set off a disk space warning.

We would like to extend the Up Watcher in a few new directions. Monitoring ethernet load would be desirable, as would any slowdown in the speed of database transactions.

Author Information

Harry Kaplan has designed UNIX systems on Wall Street for eight years, after having graduated from Berkeley before anything interesting was happening and having received a Ph.D. from Harvard in nothing relevant. He is currently the Systems and Technology Manager and Vice-President of Sanwa Financial Products, a limited partnership with The Sanwa Bank. He can be reached at 55 East 52nd Street, 26th Floor, New York City, New York 10055, or electronically at harry@sanwaBGK.com.

End Notes

[1] Wiztemp is available from Networks Wizards, PO Box 343, Menlo Park, CA 94026.

[2] An alphanumeric pager interface program called "tipx" is available by anonymous ftp from gatekeeper.dec.com: /.b/usenet/comp.sources.misc/volume13. We did not use this program, but wrote our own.

[3] Computerfone is available from Suncoast Systems, Pensacola, Florida.

[4] The ''top'' program is available by anonymous ftp from ecs.nwu.edu:/pub/top.