Saltzer
8/5/68

Dist list?

Identification

Supervisor Performance Measurement for Initial Multics
J. Gintell

Purpose

The Supervisor Performance Measurement facility described herein enables the gathering of system-wide information about supervisor performance in a multiple process environment. It is implemented by means of a modified Fault Interceptor Module (FIM), an alarm clock handler to process alarm clock interrupts used to "sample" procedure usage, a set of metering procedures, and a system-wide data base containing gathered data. One of the major design objectives of the Supervisor Performance Measurement facility is to minimize its effect upon system performance.

Summary of Information Recorded

1. Time distributions (histograms) of times spent processing page, segment, linkage, and wall crossing faults.
2. Count of procedure segment usage (i.e., number of alarm clock interrupts) for each segment whose segment number is the same for all processes.
3. Count of number of page and segment faults received for each segment whose segment number is the same for all processes.

A pre-selected combination of page, segment, linkage, and wall crossing faults (or no faults) can be considered for metering. The time recorded for processing a fault will be adjusted to discount the time attributable to any subsequent faults temporarily disrupting its processing.

The per-segment information (items 2. and 3.) can be recorded during the processing of a pre-selected combination of page, segment, linkage, and wall crossing faults or during the time when no faults are being processed.

Metering can be done during initialization only, after initialization only, or both. It can also be stopped and restarted.

Discussion

Nested and Recursive Faults

This metering facility must handle selective metering by fault type, and must accomodate the situation where processing of one fault can be suspended while another fault is processed. Defined here is a notion of "states" of a process and a stack in which to maintain information regarding faults which have not had their associated processing and metering completed. These concepts must be implemented if time distributions for nested and recursive faults are to be recorded.

Nested and recursive faults will be handled as follows. When a fault which is to be metered occurs, the previous fault metering is arrested. When processing of the new fault completes, the metering of the previous fault is continued. This type of metering discipline allows the total time for all fault processing to equal to the sum of the times for each fault and treats recursive faults as if they were independent. Conversely, since metering can be inhibited, it is possible to include the processing time of a relatively "inferior" fault (e.g., a page fault) with the processing time of a relatively "superior" fault (e.g., a segment fault) simply by choosing to meter the superior fault and to not meter the inferior fault.

The state of a process

Any running process can be considered to be in one of five states:

R1. Processing a page fault

R2. Processing a segment fault

R3. Processing a linkage fault

R4. Processing a wall crossing fault

R5. None of the above

At any given time a processor is being used by some running process which is in one of the 5 states, hence the processor can be considered to be in one of the five states (R1, R2, R3, R4, R5).

A processor switches from one of its five states to another (or back to the same) when one of the four faults occurs, when the dbr is swapped or when the control unit is restored by the FIM upon the completion of processing of one of the four faults. Each process will maintainits current state in a per-process stack by updating a "state word" whenever the process explicitly changes state as by receiving a fault of the above type or by completing processing of such a fault.

At each instance of possible action by the metering procedures, the current state is determined by comparing the per-process state word for the currently running process with the appropriate entry of the system meter data base to determine whether the action is to be taken.

Fault Time Distributions

A distribution of times for each type of fault processing is maintained. For each fault type an array of processing integrators is maintained, where

each entry contains the number of faults and the total elapsed time for all faults whose processing time falls into the range associated with the given entry. At the time that the processing of a fault is completed, its time value is used to determine which entry to use.

The Fault Meter Stack

Because it is desired to know the value of the processing time for the complete processing of a fault before recording it and because a fault's processing can be interrupted prior to its completion, it is necessary to maintain a stack of elapsed times and fault types for all those faults whose processing is not completed. Each process has such a stack of incompletely processed faults. The stack must be wired down since it can be referenced during page fault processing. Since a limited amount of space is available and overflow is possible, the stack maintainence code has to be prepared for such overflow. In the event of overflow the metering will continue to operate, even though the results are meaningless, so that the amount of space that would be needed to avoid stack overflow can be recorded and used to prevent subsequent stack overflows.

The "Process Clock"

Durations will be measured by computing the difference between successive readings of the "process clock". The process clock is a conceptual clock which counts off time only while a process is running. It is implemented and kept current by using process metering information maintained by the traffic controller and current timer register readings:

Process clock time = trs-t + te

where trs = value which was last loaded in the timer register

t = current reading of timer register

te = elapsed time spent by process since it was created.

To simplify the computation of elapsed time the notion of "pseudo-start-time" for the processing of each fault is used. Pseudo-start-time is that value which would have been read from the "process clock" had the processing of the current fault not been disrupted. It is computed each time the processing of a fault is resumed by subtracting from the process clock reading the elapsed process time used so far by the disrupted fault.

Method of metering for alarm clock interrupts

The alarm clock is set to interrupt after a certain interval. When the interrupt occurs, if the current process is in a state for which segment usage counting is to occur and the segment number is within the range of segment numbers which are the same for all processes, a count for the interrupted segment will be incremented.

Method of metering for fault metering

If the fault is the type to be metered, any current metering is suspended and a new meter for the new state is started.

When the handler for a metered fault returns control, the elapsed time is computed and accumulated in the system data basis.

Implementation

Introduction

Supervisor Performance Measurement is accomplished by two modules and a data base containing all metering results. The modules are the alarm clock handler <alarm_> which is used to initialize the alarm clock and to process alarm clock interrupts and the metering segment <system_meter> which does all fault metering. The data base is the System Metering Table (<smt>). Segments <alarm_> and <smt> must be wired-down segments included in the desriptor segment template and accessible in every ring so that the alarm clock interrupt can be handles rapidly and at any time. <alarm_) must be mastermode. <system_meter> must be cal lable from both rings 0 and 1 because it is called for both page and linkage faults and it must be wired down because it is called during page fault processing.

Entries and Calling sequences

1) call system_meter$init;

   called by initializer

2) call system_meter$multics;

   called by initializer just prior to calling Multics

3) call system_meter$page_fault;

   called by fim upon detection of page fault

4) call system_meter$segment_fault;

   called by fim upon detection of segment fault

5) call system_meter$linkage_fault;

   called by fim upon detection of linkage fault

6) call system_meter$wall_in;

called by fim upon detection of inward wall crossing fault

7) call system_meter$wall_out;

called by fim upon detection of outward wall crossing fault

8) call system_meter$stop (error);

dcl error fixed(35);/* ≠ 0 if did not stop*/

called by intrp_stat prior to using data base

9) call system_meter$start(error);

dcl error fixed (35); /*#0 it couldn't start*/

10) call system_meter$unlock;

called by intrp_stat at end of interpretation to unlock metering

11) call alarm$set;

called by system_meter$init to start the alarm clocks interrupting

Description of Metering Procedures

1. Action taken when a fault occurs
   a. go to step h. if this fault type is not to be timed
   b. read the "process clock"
   c. using "process clock" reading and pseudo_start_time found in current stack frame, store the elapsed time for processing thus far in the current stack.
   d. allocate a new frame, updating pointer to current frame, fmeter_ptr
   e. store a pointer to the previous frame in the new frame
   f. store the current state in the new frame
   g. store the "process clock" reading in the new frame
   h. continue

2. Action to be taken when fault processing is completed
   a. go to step h if this fault is not to be timed
   b. read the "process clock"
   c. using the pseudo_start_time found in the current frame, compute the elapsed time used for this state. (elapsed time = "process clock" - "pseudo_start_time").
   d. record this time in that entry of the array of time distribution for this fault which is to be used for time of the computed value. The recording is done by incrementing the total time by the just computed time, and the fault count by 1.
   e. pop the stack by setting the current frame pointer fmeter_ptr to the previous frame using the current frame to obtain the value.
   f. free the "popped" frame
   g. compute the "pseudo_start_time" and store in the current frame ("pseudo_start_time" = "process clock" - elapsed time)
   h. continue

3. Special Action upon page or segment fault only
   a. go to step d if per_segment page or segment faults are not to be counted for this state
   b. access segment number for this fault
   c. increment the page or segment fault count in the entry for this segment in the per_segment fault_count array
   d. continue

4. Action taken upon alarm clock interrupt

   a. increment clock_count, read and record time, and set clock

   b. determine whether segment usage counting is to be done for the current process state and return if the state is one for which no counting is to be done

   c. if segment number is out of range update out_of_range count and return

   d. increment segment count entry

   e. return

5. Special entries in metering module

   a. init:

      1) initialize the meter stack space

      2) interpret switches to set state switches

      3) call <alarm>|[set] to set alarm clock

      4) start metering if so indicated

      5) return

   b. start:

      1) set error if data base is locked and return, else

      2) initialize all meter_data tables

      3) start metering

      4) return

   c. stop:

      1) set error if unable to lock and return, else

      2) stop all metering

      3) lock metering

      4) return

d. unlock:

1) unlock metering

2) return

System Metering Data Base

The System Metering Data Base contains switches showing the global characteristics of the metering run and some times indicating when the run started and stopped.

There are four arrays containing statistics for page, segment, wall crossing and linkage faults where each array contains 25 entries, the n^{th} of which contains the number of associated faults which required between 2^{n+5} and 2^{n+6} microseconds to process and the total processor time for all such fault processings.

The System Metering Data Base also contains an array whose n^{th} entry indicates the number of times segment number n was being executed when the alarm clock was interrupted and another array, whose n^{th} entry indicates the number of segment and page faults taken for the n^{th} segment.

Two locks associated with the data base are in <s t>. The first lock is a local lock to be used when particular entries are being updated while metering is operating. It is used to avoid multiprocessor race conditions and need not be used when add-to-storage instructions are used. The second lock is global and indicates that the state words and meter control information cannot be changed. This will be set during metering and also during the processing of the data base by intrp_stat (B_.__.__).

Space for the perprocess stacks and a pointer to a list of free stack frames as stored in <smt>.

Specification for the System Metering Data Base <smt>

Name	Contents
segcnt	states during which segment usage counting should be done bit 0 = 1 if should count during none of PF,SF,LF,WCF 1 = 1 if should count during page fault processing 2 = 1 if should count during segment fault processing 3 = 1 if should count during linkage fault processing 4 = 1 if should count during wall crossing fault processor
psfcnt	states during which per_segment page or segment faults should be counted bits 0-4 as in segcnt
segcnt_init	initial value for segcnt
psfcnt_init	initial value for psfcnt
metflt	faults which should be timed bit 1 = 1 if should time page faults 2 = 1 if should time segment fault 3 = 1 if should time linkage faults 4 = 1 if should time wall crossing fault

Name	Contents
metflt_init	initial value for metflt
meter_init	bit 35 = 1 if should start performance analysis during initialization
meter_multics	bit 35 = 1 if should start (or continue) metering after <multics> is called
lock	unstable values lock = 0 if stable
stable	= 0 if valid to change state of metering will ≠ 0 if metering is running or if interpretation of table is occurring
begin_time	clock time at which initialization started
multics_time	time at which <multics> was called
latest_time	time at which last interrupt occurred
time_interval	time interval in microseconds for alarm clock
clock_count	total number of clock interrupts
clock_meter	number of clock interrupts for which metering was done
out_of_range	number of segment meter attempts for which segment number is out of range of table
seg_high	segment number of highest segment which is the same for all processes (in bits 0-17)

Name	Contents
multics_table	n^{th} entry contains number of times segment n was being executed when alarm clock interrupt occurs
perseg_psflt	n^{th} entry contains in bits 0-17 number of seg faults, 18-35 number of page faults for the segment whose number is n.
p_stat	array of pairs whose n^{th} entry contains data for page faults taking between 2^{n+5} and 2^{n+6} microseconds to process word 1 contains count word 2 contains total time for faults in the time range
s_stat	array for segment faults as above
l_stat	array for linkage faults as above
w_stat	array for wall crossing faults as above
stack	space in which meter stack frames are allocated
stack_size	size of total stack area in bits 0-17
stack_free	relative ptr (in bits 0-17) to beginning of free stack frame list
stovfl	running count of number of stack frames by which stack area has overflowed
stovfl_max	maximum value for stovfl during metering

Per Process Data

1. Per Process Fault Meter Stack

Each process has a stack of fault meters. In the top entry is the meter for the current state and in all other frames are the meters for the temporarily stopped states. The stacks are all contained within the system-wide data base. All unused frames are kept as a list of free frames. The top of a given process' stack is identified by a pointer in per/process address space, the PDB. The bottom frame represents the state of not processing page, segment, linkage, or wall crossing faults.

Each frame has two words:

word 1 bits 0-17 - process state
bit 0 = 1 not in PF,SF,LF,WCF
1 = 1 processing page fault
2 = 1 processing segment fault
3 = 1 processing linkage fault
4 = 1 processing wall crossing fault

bits 18-35 - rel ptr to previous frame
= 0 if end of frame

word 2 for top frame - pseudo_start_time of this state

for all other - amount of time spent thus far for this state frames

For the list of free frames, the relative pointer is used to point to the next free entry. A relative pointer whose value is 0 marks the last entry in the list.

2. Pointer to the fault meter stack

In the process data block of <pds> of each process is a relative ptr to the fault meter stack.

```
dcl  fmeter_ptr    fixed bin(18);
```