Published: 10/31/66

Identification

Processor Usage Metering
T. H. Van Vleck

Purpose

Section BO.3.01 describes the method to be used for metering the number of processor memory cycles used in the execution of a process.

Description

Processor usage metering is driven by the processor interval timer in each processor, a hardware register which can be loaded and stored, and which counts down by one each time the processor makes a memory reference.

The basic idea in the implementation of processor usage metering is to record the timer when an account becomes responsible for the processor and again when it relinquishes control. The difference is taken to be the number of cycles which should be metered to the account. Now, an account can become responsible for a processor's execution in 3 ways:

1. a process which is charging to that account can come to the top of the ready list and begin execution.

2. during the execution of some process, a system interrupt may occur which causes work to be done for a different process, and which is charged to the account of that different process.

3. during the execution of some process, a call may be made to the disk or drum DIM, and while in the DIM work may be done for many different processes, charging to many different accounts.

In the first case, processor-metering requirements are satisfied by having subroutine Swap-DBR, in the Traffic Controller, call a program which executes a small sequence of instructions to compute the time since the processor was last paid for and to meter this amount to the process which is just finishing execution. The sequence looks like this:

```
call store_timer(timenow);

cycles = time_last_paid - timenow;
```

```
call meter_CPU(cycles, index);

time_last_paid = timenow;
```

The quantity "time_last_paid" is kept in the Processor Data Block (BK.1.02) for the particular processor, and "index" is an index into the AMT for the entry for the process's account (BO.3.07).

For the second case, that of system interrupts, it is sufficient to require every interrupt handler to save the timer at the beginning of its execution, and at the end of the interrupt service, to compute the number of cycles "stolen", meter it to the account of the process responsible for the interrupt by calling Meter_CPU, and reload the timer with its original value.

The third case, that of the disk and drum DIM's, can be treated more or less like the second. That is, the modules is question must keep their own accounts and report the usages to accounting procedures, by a call to "meter_CPU".

In all three cases, usages must be recorded for accounts at times when page faults are not permitted. Therefore, any process which may have cycles metered to it must have a pointer to an entry in the Active Meter Table.

Implementation

When a process is entered into the Active Process Table by the Process Activator, the following call is made:

```
call start_CPU_meter(account, AMTindx);
```

this call

1. searches for an entry for the given account in the AMT. If none is found, one is created. If an entry already exists, its use count is increased by one.

2. returns an index into the AMT for storage in the APT.

Whenever a process switch is made or an interrupt service for a process occurs, the following call is made:

```
call meter_CPU (cycles, AMTindx);
```

this call adds cycles, the number of CPU cycles used, to the appropriate part of the AMT entry for the account specified by AMTindx.

When a process is deactivated, this call occurs;

```
call stop_CPU_meter (AMTindx);
```

the call decreases the use count in the AMT entry by one. If the use count goes to zero, updating of the AMT to the ADS is done and the AMT entry is freed.