DRAFT: 2/6/67

## Identification

The Active Meter Table

T.H. Van Vleck

## Purpose

The Active Meter Table (AMT) is a wired -down system-wide data base which is sued for metering resource usage at times when page faults are not permitted. This section describes the format of the AMT.

## Description

An entry is made in the AMT for each account number which may have resources metered to it when the system cannot accept a page fault. It contains

processor usage - metered at process switches and interrupts

core residence - metered on page-accounting faults

secondary storage use - metered by segment control

disk and drum I-O traffic -metered by the DIM's

A new entry is made at a time when page faults are possible; specifically, when a process is added to the Active Process Table by the Process Activation~~s~~ Module or when a segment is added to the Active Segment Table by the Segment Initiate Module, and the desired AMT entry does not exist already.

In the AST or APT, an index number is entered which points out the appropriate AMT entry, so that processor cycles used, for example, may be accumulated in the AMT entry whenever a process suffers a timer runout.

The information contained in the AMT includes the segment name of the user's Account Data Segment, which identifies an Account Data Segment which will receive the "scratchpad" information from the AMT entry whenever the user calls the Block-interceptor in ring 1, or whenever the use count in the AMT entry becomes zero. (At either of these times, the system <u>can</u> accept a page fault.)

Each entry in the Active Meter Table contains:

1. Account number - the unique identifying number for the user's account. It can be used to construct the tree name of the user's Account Data segment, which will receive the scratchpad information when "update_accounting" is called. See BO.2.01.

2. Processor usage meter - a count of the number of memory cycles taken by processors in execution of the processes charging to the account, as described in BO.1.01.

3. Core residence meter - the number of word-seconds metered to the account as a result of references to pages ~~with the~~ intercepted by the page-accounting fault. See BO.1.02.

4. I-O transmission meters - for each device used by the file system, a count of the number of words transmitted for this process. See BO.1.03.

5. Secondary storage residence meters - as described in section BO.1.06, for each device used by the file system, the following 4 items:

    a) time last computed

    b) number of words used

c) maximum number of words which may be used

d) number of word-seconds charged

6. Use count- the number of Active Process Table and Active Segment Table entries holding a pointer to this entry. The use count is increased by calls to "start_sgt_meter" (BO.1.06) and "start_cpu_meter" (BO.1.01), and decreased by calls to "stop_sgt_meter" and "stop_cpu_meter". Whenever the use count decreases to zero, the AMT entry is updated to its Account Data Segment by a call to "update_accounting" and then freed.

### References to the AMT

Entries are made in the AMT whenever it is known that a scratchpad will be needed to accumulate usage figures while page faults cannot be tolerated - specifically, by calls to "start_sgt_meter" from the file system, and to "start_cpu_meter" from the process activation module. These calls supply an *account number*, a 36-bit quantity, which is looked up in the *AMT hash table* in order to determine whether a scratchpad already exists for this account. If the account number has an entry, the value associated with it in the hash table is the *AMT index*, an 17 bit integer ~~18-bit stri~~ng which is the relative address of the proper AMT entry. This AMT index is stored in the APT or AST for use in the following calls:

meter_cpu

meter_ss_io

meter_core

stop_cpu_meter

stop_sgt_meter

meter_length

meter_move

start_page

If the account number is not found in the AMT hash table, then no scratchpad exists for that account, and one must be created. An empty slot is picked up from the AMT free list and initialized, and a hash table entry is made for it.

When an AMT entry is released, its contents are updated to the ~~associated ADS~~ Accounting Update Table, it is placed on the AMTfree list, and the hash table entry is removed.

Locking of the AMT

Many processes may attempt to reference an AMT entry simultaneously, in order to

a) put information into the entry, as a result of metering calls.

b) take information out of the entry, when the information is updated ~~the~~ to paged storage.

Only two of the metering calls, "meter_length" and "meter_move" (BO.1.06), perform operations more complicated than simply increasing a cell in the AMT entry. These Two calls require locking of the AMT entry, a "loop-lock" call; all other metering calls can be implemented by use of the (read-alter-rewrite instructions of the) 645. An AMT entry must also be locked if it is being created or deleted, since its account number does not make sense at these times.

The updating calls are performed when the system can take faults, so that the information in the AMT is unstable, but the entry cannot be locked. Therefore, any process attempting to read the AMT when faults are possible marks its process id in a cell in the entry and checks the AMT lock to make sure it is off before copying the entry. All metering procedures which ~~update~~ ~~lock~~ stet. the AMT entry are required to zero this cell after setting the lock. Then, if the cell still contains the process id of the copying process after the AMT entry has been copied, we can be sure that a consistent copy has been snatched. ~~If snatching fails,~~ the strategy used when ~~the~~ snatching fails depends on the purpose of the update. For example, on updates arising from calls to Block, snatching is reattempted a variable number of times and snatching failures are recorded in the AMT ~~entry~~ entry. This strategy is described in detail in BQ.2.01.

Format of the AMT

The entire AMT is declared as controlled storage containing several variable-size tables. The table is declared as follows:

```
dcl 1 amt ctl (amtp),

2 entrs fixed bin 17,          /* entry count */
2 ilock bit (~~36~~ 72),       /* table interlock */
2 buckets fixed bin 17,        /* number of hash buckets */
2 hasht_ptr bit (18),          /* pointer to hash table */
2 frelist bit (18),            /* pointer to free list */
2 amtvar area (SIZE);          /* table area */
```

The AMT hash table is laid out by the following declaration:

```
dcl 1 amt_ht (amtp→amt.buckets) ctl (p1),

    2 vs bit (1),              /* vacant switch */
    2 ds bit (1),              /* deleted switch */
    2 indx bit (~~18~~ 17);    /* amt index */
```

The following EPL statement describes the format of an entry in the Active Meter Table:

```
dcl 1 amte ctl(p)                       /* the system active meter table */
      2 processor fixed, bin 36         /* count of processor cycles */
      2 core fixed, bin 36              /* count of core traps */
      2 account bit (36),               /* account number */
      2 ss(2 /* number of devices in file system */),
            3 time bit (72),            /* time last updated */
            3 trans fixed, bin 36       /* GIOC cycles transmitted  */
            3 wds_used fixed, bin 36    /* current occupancy */
            3 max_use fixed, bin 36     /* record quota */
            3 wd_secs fixed, bin 36     /* total word-seconds used */
      2 using bit (72),                 /* pseudo-interlock, for ~~call time~~ while updating*/
      2 urgency fixed,                  /* number of times could not update*/
      2 lock bit (~~36)~~ 72),          /* real interlock, for trap time */
      2 nusers fixed,                   /* number of processes and segments using
                                           account */
      2 next_free bit(18);             /* pointer to next free bucket*/
```

When an AMT entry is on the free list, account, using, lock, and nusers must be zero, and ~~lock~~ next free contains a chain pointer to the next free entry. The rest of the entry is meaningless.