TC-DATA METERS

offset (8)	length	Meter
10	1	# of running processes
11	1	# of ready processes
12	1	# of waiting processes
13	1	# of blocked processes
14	1	# of stopped processes
16	2	Total system time
20	2	Total idle time
22	2	idle + ready processes.
24	2	idle + waiting (+ no ready)
26	2	idle + no waiting + no ready

APT ENTRY 14	2	Process' virtual clock
		# of interactions
		# of loadings

Want to get average running time between waits

between blocks

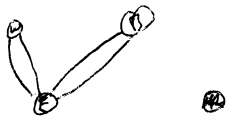
total time sold

* of waits

* of blocks

* of restarts

* of idle blocks, to subtract out



Saltzer

TO: MPN distribution

FROM: Michael J. Spier, Robert L. Rappaport

DATE: 12.1.68

SUBJ: Metering of new Traffic Controller

Introduction

The new Traffic Controller as specified by LIM.5.10 (excepting preemption and conversion to slave mode) and including the interfaces with the new IPC as specified by OIM.8.4 has been integrated into a system 1.8 environment.

The new TC includes the following features:

1. Multiprogramming control, with parameters set to max-eligible=2 and max-loaded=3.
2. New loading scheme, where an active process' 4 crucial segments have their page tables wired down and where the function of loading a process merely causes the process' PDS to be read into core and be wired down.
3. A wired down loader daemon process.
4. Modified block/wakeup interface, including a wired down Interprocess Transmission Table which is manipulated by the TC exclusively.
5. Built-in system meters to reflect,

   each process' total execution time

   current distribution of execution states (idle process excluded)

   total system time (meter initialized in tc_init)

   total idle time

   idle time distribution; i.e. idle while at least one other process is ready, idle while at least one other process is waiting, idle while all other processes are either blocked or stopped.

Tests performed

Two series of experiments have been made with the new system on 11.28 and on 11.30. There is a marked improvement in load/unload response time compared with the current (1.7) system; in the new system, the response of an unloaded process appears to be just about as fast as that of a currently loaded process.

The difference between the two experiments is in the environment. On 11.28 the system had a small-page pool of 272, in the later experiment the size of the pool has been increased to 400 small pages; the reason being that in the new system (temporarily, until pimu is recoded) hardcore process PDS segments are wired down in small pages, the number of which is estimated at 125. If we add to this 12 pages per active process we come up with a total loss of about 200 small pages (for a 4-console system) which leaves very little space for segmentation.

In both experiments, processes were created and allowed to reach the point in procedure <multics> where they print out the initial message. Segment <tc_data> was then dumped in order to read the processes' virtual clock, located in the APT entry. Then a process was created and normal metering was done up to the point where the teletype started to respond, but before the typing of the initial message.

Results of 11.28 metering

1. The first process (TTY194) was created and metered:

Total process execution time			30 sec
During this time, the process experienced:			
196	linkage faults	costing	14.2 sec
107	segment faults	costing	4.6 sec
197	page faults	costing	2.3 sec
502	wall crossings	costing	1.4 sec
		total	22.5 sec

It took the loader daemon 38 milliseconds to load this process.

2. Three more processes (TTY128, TTY196, TTY198) were then created as fast as the initializer process could respond.
Only their execution times were metered. TTY196 had a hangup and was redialed which must account for its excessive creation time.

Total process execution time TTY128	39 sec
Total process execution time TTY196	54 sec
Total process execution time TTY198	37 sec

The total loading/unloading time during the creation of all four processes which includes loading all 4 new peocesses + unloading the initializer and TTY194 + possibly loading/unloading another pair of processes to allow the initializer to type its ready message 375 milliseconds

Results of 11.30 experiment

1. The first process (TTY194) was created and metered:

Total process execution time 32 sec

During this time the process experienced:

173	linkage faults	costing	13.0 sec
104	segment faults	costing	4.5 sec
226	page faults	costing	2.7 sec
408	wall crossings	costing	1.2 sec
		total	21.4 sec

2. Two more processes (TTY196, TTY198) were created sequentially

Total execution time TTY196	37 sec
Total execution time TTY198	31 sec

Total load/unload time 110 milliseconds

Traffic Controller measurements

1. Per-process cpu usage (to be kept in Active Meter table on a per account usage also.)

2. Idle time
   a. Multiprogramming idle: all idle time accumulated while # of eligible processes ≥ max. eligible.
   b. True idle: all idle time accumulated while # eligible < max eligible.

3. Total cpu time, idle + per-process, and total

4. Per process virtual clock (for metering) (may be same as (1)).

5. Total number of calls to
   a. Block with interrupt switch on.
   b. Block with interrupt switch off.
   (calls by idle process loop excepted)
   c. Wakeup
   d. Restart
   e. Wait
   f. Notify

(also per process)

6. Total # of process loadings

7. # of ~~users~~ processes in each queue (have idle processes)

8. Recent average of # of processes in interactive queue.
(Updated 1/time-burst)

Problems with elimination of idle process.

1. Accounting - must not charge previous user with idle time.
2. Must be able to accept call to destroy - pass for the process which was playing idle loop.
3. Must be sure scheduling gets idle time scheduled out.
4. Must work with 2+ processors.
5. Dynamic reconfiguration - how do I remove a processor.