Multics Performance 3/3/69

General Observations

1. Today's system performs well for only a small range of jobs (Particularly well for jobs you can't do elsewhere.) range
We are working to increase the range to include most computer jobs. We expect to accomplish this goal. We do not expect to satisfy all of them.
e.g., Number-crunching (CDC-7800 type).

2. Emphasis in performance measurement has been on understanding the causes, not on evaluating throughput understanding
or counting "number of users". Therefore we do have rough estimates of the latter.

also experiments are expensive

3. Performance generally has been high enough that it does not inhibit progress of system development. Therefore we and sound timing
have been able to concentrate on long-range performance program
improvements rather than short-range, hour-to-hour fixes which might engender later maintenance.

4. Overall outlook: Performance needs work, but we understand understanding
where the problems are and how to fix them; And have not no intrinsic bottlenecks
found any intrinsic bottlenecks preventing CTSS-like cost and performance results – going down.

Measurement / Analysis tools 3/3/69

1. Ready message gives real time, cpu time, # of page waits for each command.

2. System clock is read at beginning and end of each missing-page, missing-segment, linkage, wall crossing } count and interrupt.

and averages and distributions are maintained

3. System clock is used to generate an interrupt (every 10 ms) and segment being executed is noted. Provides a profile of where we spend our time.

4. Record kept of # of page faults which occur for each segment #.

* 5. Traffic controller maintains meters; time charged to each process —

idle time: true idle —

idle → no work desired — multiprogramming idle

→ work in queue → all queued processes waiting for pages — paging idle

→ multiprogramming will not permit queued work to proceed —

# of interactions —

average queue length —

6. PDP-8 computer used as a probe to watch Multics tables while system is running. (Display) Can watch
   - process execution states vs time
   - ready list activity
   - core memory activity
   - page fault time distributions

7. PDP-8 computer used as a driver to simulate a load of command-typing users; variable script allowed. (Includes think time)

8. Certify program creates n processes and starts them each following a command script. (No think time) Combined with 2, 3, 4, +5 above

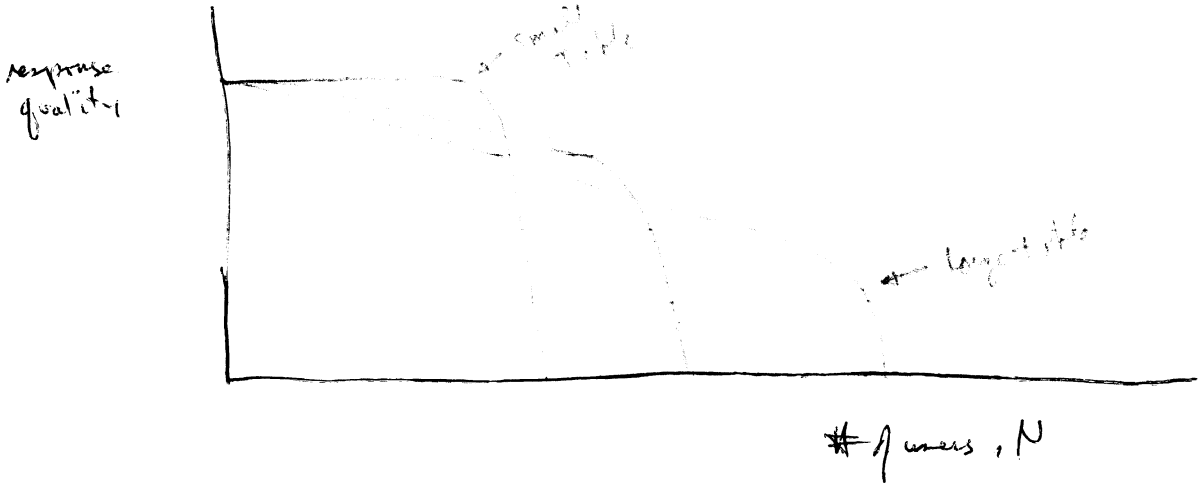
* 9. Per-primitive meters like * 2 above.

Also CTSS statistics as a basis for comparison and first-cut expectation of user load.

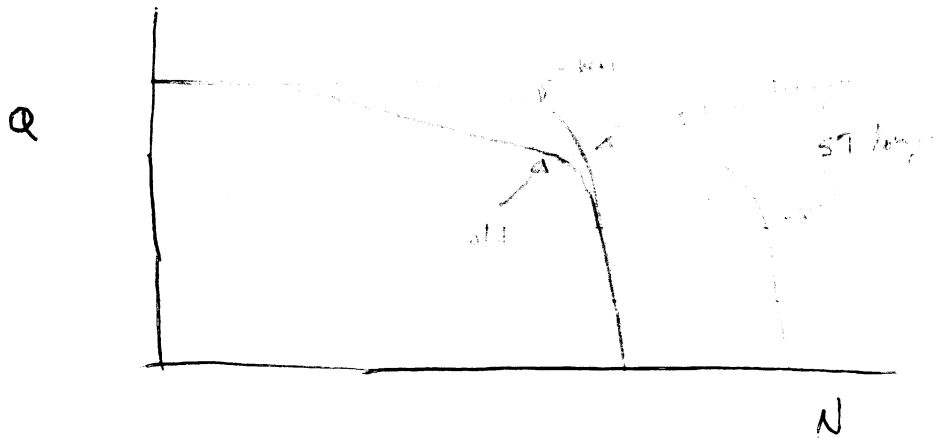
* Coming soon

Factors influencing Performance 3/3/6x

System Table size



Effect of a performance improvement if ST size not changed:



(This effect has been noticed most strongly in 384K → 256K case experiments
Performance for n small is made better; but max (N) is unchanged.

* # Users? 3/3/68

(Real question is what does it cost to support a user doing a particular job.)

1 CPU, 256K system

	~~Status~~	4/1/68	8/1/68	10/1/68	12/1/68 → 3/1/69
System Programmers:	≤1	3-4	6-8	10/12	12-15

1 page size; misc fixes; page table/AST coupling, Multiplying control; TW DM-80, IPC new, link strategy

Class of users	today (from scientific runs on CTSS & Multics)	10/1/69	future	
Heavy	≤8	≤15	≤30	(Users can be arbitrarily heavy)
CTSS	16	30	60	(Program debugging, Info retrieval)
Light	~~~~ 16 (← core bound)	~~~~ 40	90-120 ~~~~ (← CPU/core in balance)	(Basic; text editing; desk calculator)

a. All numbers can be multiplied by 2-3 by adding 3-4 CPUs and sufficient memory.

b. Future hardware improvements will probably multiply all numbers by 2 or so.

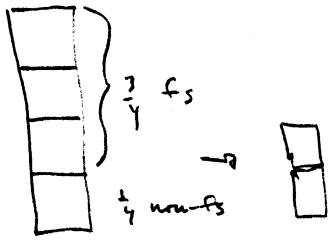
c. Present compilers place too many users in the "heavy class" and present system ~~do~~ offer no economy to "light" class. Both of these problems are the focus of attention.

d. "Today" figures can be improved (we haven't measured amount, mainly duration) by adding more core.

3/3/69

Major Multics Performance Problem areas:

1. File system code size and working set too large.
   Major revision in works, due to be ready to begin plugging in in Mid-April.



2. Per-process minimum working set too large (effects small (both calculation and I/O))
   Minor but effective revision planned

3. Compilers are large and slow
   Will require continuous attention; high speed user end to generate binary formats is high priority.
   Fortran IV
   AED

Most other potential problems are lost in the noise of the first three.

- Need to ~~rev~~ review command language interpreter code path.
- Need to review I/O system code path (GIOC interpreter needs almost complete rewrite - TWIO some work will slow this)
- Need to speed up process creation, so we can use it more.
- Need to reduce # of WAIT crossings; audit path.
- Need to produce a ~~super~~ better profiler

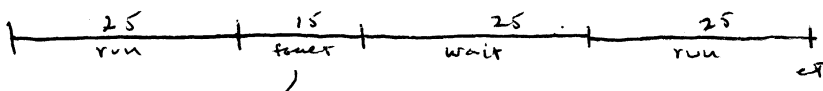
Running time between page faults. 3/3/69

Currently about 20 - 30 ms (Average page fault time) (recent exit from run)

Over for def.

c/w 25 ms average latency

c/w 15 ms page fault handling (CPU) time.

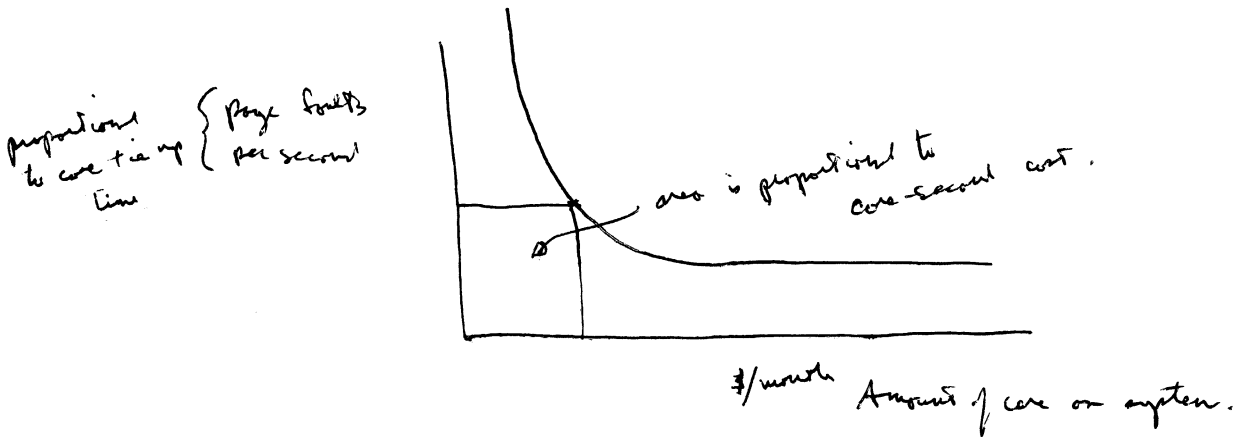


observations

a. 15 ms will go to 5 ms with new file system. 5000 instr → ~2000 instr. 7000

b. System is core memory bound. 384k memory system runs about 100 ms between page faults.

c. High page fault rate increases cpu cost
increases core time (cost)
reduces core quality (cost)



i. One does not necessarily minimize the page fault rate.

ii. Cost may be fairly insensitive in intensive use, but delay (response) times are probably not.

missing page fault handling.

1. Establish which page was referenced.
2. Establish secondary storage location of that page.
3. Allocate a core space for the page.
   (Requires throwing a page out)
   i. Find a suitable page
   ii. Find a suitable secondary location
   iii. Initiate a write request.
4. Initiate read request drum → core.
   notify other users if their pages are in use.
5. Give CPU away to a waiting process.

Each check ~ 400 machine language instructions, about 40 EPL statements.

Ring 0 TW Dim performance:

1. Overall running time of "typical user" script (cold flush) is down 15%.

2. Process creation time is up 15%. (Why?)
   does not affect overall time too much.

3. Half of performance improvement is in page fault handling.
   half of that is because page faults are
   mysteriously shorter (!)
   15.3 ms → 14.1 ms.

4. Print command (11 line file) used to take 4.2 seconds average.
   Now takes 1.8
   saved 2.4 seconds.
   saved ~ 50 page faults.

   Sorting ~ 5 page faults/line of output
   ~ 200 ms comp/line
   ~ 6 wall crossings/line (Half of all wall crossings)

BELL TELEPHONE LABORATORIES
INCORPORATED

FEB 18 1969

SUBJECT:

DATE: February 14, 1969

FROM: E. E. David, Jr.

MESSRS. D. P. LING: (Whipp Mu Sys Engineering)
R. C. FLETCHER: Whipp Mil Sys Research (Director)
R. W. KETCHLEDGE: (Indian Hill)
J. R. PIERCE: ← Exec Exec Dir ESS
V. M. WOLONTIS: (40) Data Sys Eng., Exec Dir
H. S. McDONALD:
S. P. MORGAN:
M. C. McILROY:
W. S. BROWN:
R. W. HAMMING:
E. N. PINSON:
R. H. CANADAY:
A. D. HALL:
MRS. G. J. HANSEN: !
R. MORRIS:
P. G. NEUMANN:
W. H. NINKE: (MH graphics)
J. F. OSSANNA:
D. M. RITCHIE:
C. S. ROBERTS:
J. N. STURMAN:
K. L. THOMPSON:
A. W. WINIKOFF:

On Wednesday, March 5, 1969, Professors F. J. Corbató, J. H. Saltzer, and Mr. R. C. Daley of Project MAC/MIT will visit Bell Laboratories. In the afternoon beginning at 2:00 p.m. in Room 3A-112/116 at Murray Hill, they will review some of the technical issues which are central to the MULTICS philosophy, including measurements and data from the operating system. You are cordially invited to attend this review.

E. E. David, Jr.

E. E. DAVID, JR.

Copy to
Mr. W. O. Baker - MH

w/ McIlroy

for next

Feb. 28, 1969

proposed subjects would like to hear

— various people using

1. (most sig.) what can be done to make 1-level store, preserve data when crash
2. unlinking: need prove. now
3x (minor annoyance): persistence of KST entries; e.g. remove a file, rename access
4x performance prospects
5x only way to op. now is to shutdown freq.; price ~ 1/2 hr. because warm boot ~ 12 min!
6x problems of cross-talk (better files being created + being used) text + links, starting, Someone happens to read in a new version
7x "old files never die" — often "scrogged" — hard to ensure a compatible set of programs
8. will an unlinking sol. invalidate dynamic linking
9. (McIlroy only) When protected ring crossing (w/ arg valid), how expensive will rings be?
10. Small computers BCPL, Ftn, ESPOSA use a lot of machine time