June 1, 1970

Progress Report - CAL-TSS

Jim Gray

This is a status and projected progress report on CAL-TSS, the ECS-based time-sharing system on the CDC 6400. The project is approximately two years old now and has about ten man years invested in it (~ $80,000 in salaries). Currently there are five paid employees on the staff; 3/4 Programmer IV, 2-1/4 Programmers II, 1-1/2 Programmer I, and 1/2 Senior Coder. For the past two years several people have worked gratis, notably Butler Lampson, Jim Morris, myself and Laura Gould. Jim and Laura have joint appointments with the Center and the Computer Science Department.

The main thrust of the project has been to develop an operating system capable of supporting up to 100 student users at teletypes plus a background batch system. When the project began the promise was made that "something would be running by the Fall quarter of 1970". We still plan to deliver a system capable of supporting 100 students by the fall. However this system will not be polished and will have almost no interactive software. Two things should be noted:

a. we are meeting our schedule
b. the cost of the code is 50¢ per line while the standard for industrial operating systems is $5 or more per line.

Current Status of TSS

Currently CAL can support 20 student users. There is an interactive editor, a SCOPE simulator (capable of running COMPASS, FORTRAN, SNOBOL, ALGOL and other SCOPE systems), a printer driver, a CDC-console display driver, a two-level disk file system, and a driver for ASCII-terminals (e.g., teletypes and datapoint displays).

CAL is currently up about six hours a day. The staff is using it to write the rest of the system; there are no users. A typical staff programmer's session consists of:

logging on
getting his files from the Disk
editing his files using the editor
calling SCOPE
assembling his files
leaving SCOPE
printing the assembler output on the printer
executing his program
debugging it using an octal debugger
saving his files on the disk

The programmer in this cycle consumes one sixtieth of the machine. Students and people in edit mode create a lighter load.

The system crashes once every 40 hours (on the average over 3 months). Almost all of these crashes are due to hardware errors. Software crashes occur at intervals of over 200 hours.

The shortcomings of the current system are that:

(a) all information resides in ECS and there is no ECS-Disk swapper ($\Rightarrow$ at most 20 small users).

(b) the File directory system is unprotected.

(c) the command processor and SCOPE simulator give cryptic diagnostics adequate for systems programmers but not for users.

(d) there are no interactive programming languages (except SNOBOL under SCOPE and the editor).

Projected Status of TSS

During the next four months (a) and (b) above will be corrected. In addition, a simple background batch system is under construction. Unfortunately, we do not have the manpower to work on (c) or (d).

Projects that are to be completed are an ECS-Disk swapper, a directory system on the disk which protects users from one another and guarantees privacy, a card reader driver, a simple batch monitor, and maintenance of the current system.

Pending Problems

I am resigning as director of CAL effective today. Howard Sturgis who, along with Butler Lampson, was the principle architect of TSS is replacing me. The job is thankless, draining, mundane, and unpleasant.

The most critical problem currently is morale. The two-year old project is suffering from boredom. It was especially hard hit by Nixon's latest actions and now is the home of the reconstituted Computer Science Department.

The second problem is that CAL is being designed from the inside out by systems programmers. Whenever something is difficult, it is finessed (passed to a higher level of the system). This has led to the comment that one of the metaprinciples of the system is buck passing. For example, to rename a file one creates a new file with the new name, copies the old file to the new one, then destroys the old one. Fighting such battles (which I lost) is painful and there are an infinity of them. The sooner users confront the implementors of CAL the better.

The third problem is that no interactive languages are being written for CAL. There has been no support from any academic department; neither the CS Department nor the EECS Department has encouraged its students, RAs or professors to contribute to CAL. Both of these departments could make valuable contributions in this area.

Another problem is that the software seems to be several orders of magnitude more reliable than the hardware. The teletypes are especially bad. ECS is unreliable, but we feel we can live with one crash every two days.

Proposed Solutions

If CAL is to be made available to users, these users will need a consulting service. To expect five people to develop an operating system and to provide consulting for users, all in two years, is a little too much. The Center should move as quickly as possible to assimilate new people into the TSS staff before the fall... It takes about three months of real time and one month of current-staff time to assimilate one person.

Jim Morris should be given a mandate and any support necessary to produce a BASIC for CAL. BASIC has been coded and the micro-code interperter is debugged (in BCPL). He should also direct the APL project (the interpreter is debugged).

I am working (slowly) on an interactive PL/1.

Karl Malbrain is working (half-heartedly) on a debugging system for a compiled PL/1 and on Cool Aid (the assembler-loader-debugger).

Concluding Remark

The only thing that keeps CAL alive is Ken Hebert. He has been very pleasant to work with, and I admire his quixotic attempts at keeping the Computer Center alive, although I feel he is doomed to failure.