History of the Cal Timesharing System

Paul McJones
paul@mcjones.org

David Redell
dave.redell@gmail.com

***Abstract*—The CAL Timesharing System (CAL TSS) was developed at the University of California at Berkeley between 1968 and 1971 to provide interactive computing for research and instruction. It ran on a mainframe computer, the Control Data Corporation 6400, and was one of the earliest systems to use capabilities for protection. We discuss the origin of the project, development of the software, and the brief experience using the system before it was shut down in May 1972.**

Background

IN 1966 the Computer Center of the University of California at Berkeley was once again outgrowing its computing resources. Since its founding in 1956, the Computer Center had operated a series of International Business Machines (IBM) scientific computers, the latest of which was a 7094-7040 Direct Couple System (with the much slower 7040 as an Input/Output front end for the 7094).[1] Computer use was becoming widespread in courses and for graduate and faculty research across many departments. System load projections predicted saturation within a year, and users with large data sets were already frustrated with the 7094's memory size of 32,768 36-bit words (about 147,000 8-bit bytes). The Direct Couple System was primarily batch oriented, running jobs submitted via punched cards or remotely via IBM's Scientific Terminal System. Limited interactive service, in the form of IBM's Quiktran, was available three hours a day.

Acquiring the hardware

The Computer Center staff and advisory committee studied the available large-scale computers, including the IBM 360/67 and 360/75, the General Electric GE-635, the Univac 1108, and the Control Data Corporation (CDC) 6000 series. One consideration was support for interactive as well as batch computing.

The GE and Univac offerings were eliminated in the first round of selection. IBM and CDC each submitted proposals. IBM proposed the 360/67 and indicated that a timesharing system was under development; they promised delivery of the hardware in June 1967. CDC proposed a 6400[2] with Extended Core Storage (ECS), which was a large core storage unit that supported high-speed (ten 60-bit words per microsecond), low-latency (4 microseconds) transfers to or from main memory[3] but was not addressible for instruction fetch or data load/stores. They promised delivery of a basic system that fall.

In April 1966, Computer Center Director Abraham Taub[4] sent Chancellor Roger Heyns a

[1]The full series was: 701 in 1956, 704 in 1959, 7090 in 1962, 7094 in 1964, and DCS in 1965. [43] [66]

[2]The 6400 was instruction set compatible with the record-setting 6600, but executed instructions sequentially, lacking the 6600's parallel functional units and instruction stack.

[3]Thus a program of 10,000 words or about 30,000 instructions could be swapped into central memory in one millisecond.

[4]Abraham Taub (1911–1999) was a mathematician and physicist. At the University of Illinois from 1948 to 1964 he was involved with the ORDVAC and ILLIAC I computers and later headed the Digital Computer Laboratory. [43]

memorandum describing the planning that had been done and requesting approval of proposed budgets for academic years through 1969–1970; these budgets hypothesized a large National Science Foundation (NSF) grant for Computer Facilities. [64] The memo described both the IBM and CDC proposals in some detail; the decision to purchase a CDC system seems to have been made shortly thereafter. In May, CDC wrote a letter to Taub detailing a proposed configuration, with a delivery date in November 1966. [50]

The next six months were filled with intense negotiations between the Computer Center, CDC, NSF, and the University itself. The Computer Center worked with CDC to configure an appropriate system, settling on two 32K-word 6400 CPUs, a 500K-word ECS, a disk drive, tape drives, printers, card reader and punch, etc. Selecting the disk drive was complicated by the fact that CDC's new 6638 wouldn't be available by November, so Berkeley agreed to rent an older 6603 in the interim. The second CPU and the ECS were to be shipped in July 1967, and the 6638 disk was to ship in September 1967.

The Berkeley Chancellor's Office made presentations to the Regents of the University of California and obtained permission to acquire the full set of CDC equipment, to borrow $970,000 to pay for the November deliverables, and to apply to NSF for a grant. Interestingly, part of the justification to the Regents for buying the CDC system was that it "offers us an opportunity to pursue a novel approach to time-sharing"—namely, the use of ECS for fast transfer of programs to and from main memory. This was felt to make NSF more likely to support Berkeley's grant proposal. [39] Section VI.3 of the grant proposal was "An Outline of the Projected Time-Sharing and Batch Processing System Using the C.D.C. 6400/6400 Equipment." [66] It pointed out that a large program filling the entire central memory could quickly be transferred to or from ECS, whereas the rotational latency and lower transfer rate of a swapping drum made it advisable to optimize CPU utilization by dividing central memory between two smaller programs: one being executed and one being transferred in or out. The anonymous author (probably Taub or Associate Director Martin Graham[5]) also discussed the lack of paging or segmentation hardware. While this meant it was not possible to have a larger virtual address space than the size of physical memory, it was felt that traditional overlay techniques, implemented via fast, direct transfers between central memory and ECS by the user program, would make up for this.

The Chancellor's Office also negotiated with CDC on the price. CDC had initially proposed a 20% educational grant and a 10% research grant to help fund Berkeley's development of timesharing software for the 6400. They retracted the research grant offer [17] and instituted across-the-board price increases, but offered Berkeley a compute time buyback deal that preserved the prices informally agreed upon in May. [65]

Taub's NSF grant proposal [66] requested $2,010,384 over 4 years in order to purchase the CDC equipment, "develop and maintain a novel time-sharing system for the campus using the dual C.D.C. 6400 computers and the extended core storage," and maintain and operate the equipment for a period of 4 years. Instead, NSF eventually issued Award 67P7635 in the amount of $1 million over 3 years starting in 1967; it was restricted to "Purchase of Computing Systems." Later it was amended to allow grant money to be used to subsidize educational and research users who lacked funds to pay for computer time.

Launching the timesharing project

Although the first hardware (the A machine) arrived on schedule in November 1966, the Computer Center wasn't immediately able to begin work on a timesharing system. For seven years (since the arrival of the IBM 704), users had become accustomed to IBM's customary smooth evolutionary improvement and backward compatibility despite a series of hardware upgrades. To squeeze out more performance, users had written many library programs in assembly language. Suddenly there was an incompatible (albeit much faster) CPU and immature system software. With unhappy users complaining, the Computer Center programming staff spent a number of person-

[5]Martin H. Graham (1926–2015) began his career at the Brookhaven National Laboratory, and then became a professor at Rice University, where he led a project to build the Rice R1 computer before coming to Berkeley in 1966.

years improving CDC's system software and helping users rewrite their applications. [27]

Two Computer Center programmers, Howard Sturgis[6] and Dave Redell,[7] had previously modified IBM's Scientific Terminal System (STS) to work with relatively inexpensive Teletype terminals. (STS allowed batch jobs to be submitted from a remote terminal, but did not offer services such as text editing or interacting with a running program.) This time they started from scratch on a Remote Terminal System (RTS) providing comparable functionality for the CDC system, building their application logic on what would now be called a thread scheduler, patterned after one that had been written for a PDP-5 as part of Project Genie. It's clear that Sturgis was also thinking about the challenges of building a timesharing system. As early as May 1967 he wrote a four-page memo describing a simple timesharing system that would be able to execute compilers and user programs written for CDC's SCOPE batch operating system via an adapter to a new file system. [52]

By November 1967 RTS was far enough along that Sturgis could think more about the timesharing system.[8] He wrote a memo describing "a basic time sharing system" that seemed more an experiment in specification style than a specific proposal. It used terse prose descriptions of objects (data block, user process, system process), their abstract state, and operations defined on each object type. [53] The actual system described was perhaps a hypothetical reinterpretation of RTS as a program running on the 6400 Central Processing Unit (CPU) rather than on a Peripheral Processing Unit (PPU), as in RTS and SCOPE. The PPUs were small computers that could directly access input/output (I/O) devices, transfer data to and from Central Memory, and control the Central Processing Unit, which could not directly access I/O devices. In Sturgis's design, one component of a process was a map that specified how to transfer words from data blocks in ECS to the process in Central Memory, and then optionally back to the data blocks. This was useful since the 6400 Central Processing Unit had only a single (base, bounds) register pair to allow relocation and protection of the running user program. System processes represented I/O devices, so I/O could be done with process synchronization primitives (block and wakeup) rather than interrupts. An example was given of reading and writing to a low-speed terminal, with code written in Fortran.

Around this time Taub resigned as Computer Center Director, returning to full time as a professor in the Mathematics Department. Graham agreed to serve as Acting Director on a temporary basis. Apparently he agreed with Sturgis about starting the timesharing project, because on February 20, 1968 Sturgis circulated a short project description with a cover note saying "This is the proposed body of a letter from Prof. Graham[9] to myself (HES). He wants approval or comments from Ken Hebert[10] [and] Gene Albright."[11] [54] The plan involved three phases: six months of design with 2 people,[12] one year to implement a limited system with 4-5 full-time people, and a better system involving another year or two with the same staffing. The limited system was to allow interactive editing and execution of user programs plus RTS-style support of remote printers, with all compilation, assembly, and linking taking place on the batch system (the other 6400). There would be a "preliminary permanent file system," although this was followed by a question mark. The better system would extend this to more users, execution of large programs including compilers, etc., batch service, and a better permanent file system. It seems this was intended to replace CDC's SCOPE operating system running on the A machine, because its requirements included both 6400s and all the ECS and disk storage. Starting a week later, Sturgis began a series of short design memos. Perhaps most interesting was one titled "Capabilities, basic objects" [55] since the use of capabilities—unforgeable references to operating system objects—was to characterize the design of

[6]Howard Ewing Sturgis (1936–1990) had been a graduate student in mathematics at Berkeley, but took an extended leave of absence to work at the Computer Center.

[7]David D. Redell (1946–), who is the second author of this paper, was in his final year of an AB in the new Computer Science field major.

[8]Redell had started graduate school that fall under a fellowship that precluded working at the Computer Center.

[9]But presumably actually written by Sturgis.

[10]Kendrick J. Hebert was Assistant Manager.

[11]Gene Albright was Chief Programmer.

[12]Starting "after completion of RTS".

the eventual system.

While Sturgis was eager to begin, Graham realized he needed support from the Chancellor's Office. He'd hoped that a new Director could take over, but with no appointment having been made by May, he took action. First he called a meeting with Butler Lampson[13] and Peter Deutsch, who had jointly developed the operating system for the Project Genie SDS 940 system. [30] The Computer Center was represented by Hebert (who would become Acting Director), Albright, Sturgis, and two younger programmers: Allen Ginzburg and Bruce Lindsay.[14] Next Graham wrote a memo to Vice-Chancellor for Research Loy L. Sammet, noting the willingness of Lampson and Deutsch to "collaborate with the Computer Center in designing the system." [19] Graham proposed to fund three full-time programmers for one year, and to investigate additional hardware needed up to $80,000. At that time the Computer Center's finances were excellent, with multiple income sources from state and federal sources matching the 1966 projections, and Sammet confirmed Berkeley's commitment to staff the project. [48] [40] This meeting led to further discussions between Lampson, Graham, Hebert, Sammet, and CDC over the next few months. CDC agreed to install without charge the Central Exchange Jump option on Berkeley's second 6400 (see below) and held out the possibility of additional hardware through a research grant if Lampson was "directly attached to the project." [51] That was not feasible, but he continued as co-designer and advisor. (Deutsch was not further involved.)

Designing the ECS system

In July 1968 the project got underway with Lampson as faculty sponsor and Sturgis and Lindsay as full-time staff. A series of design memos appeared starting in midmonth and running through September. The first two (by Sturgis, as were most of the series), were terse but already suggested the design that would emerge over the coming months: "Map, operation, capability, objects" [56] and "Process, subprocess." [57]

The design was described in terms of abstract object types and the operations (actions) that could be performed on them. Objects were stored in ECS and accessed via capabilities, which were stored in capability lists. A process was subdivided into multiple protection domains called subprocesses, each with its own capability list and map to control swapping of its address space to and from files. Processes could also read and write files, and could send and receive one-word messages on event channels. Operations were themselves objects, and so could be controlled by capabilities. Allocation blocks imposed an ownership tree on all objects, and controlled and accounted for the usage of ECS space and CPU time.

Many of these ideas, and especially capabilities, can be traced back to the work of Dennis and Van Horn [12].[15] Dijkstra's paper [13] on the THE system, with its emphasis on a hierarchical design, was another inspiration.[16] Lampson and Sturgis combined these ideas in an interesting way: the lowest layer, called the ECS system, would provide the basic facilities necessary to support the higher layers, including mechanisms such that the higher layers could be efficiently implemented in a distributed fashion as protected domains within each user process.[17] Many system calls could be handled directly by the ECS system, but infrequent cases would cause a so-called FRETURN (failure return, borrowed from SNOBOL [16]) from the ECS operation, to be handled by a subprocess implementing a higher layer. Several mechanisms were required to make this work. Operations could have multiple layers, typically starting with an ECS action, followed by one or more subprocess call layers that were only invoked after an FRETURN by the previous

[13]Butler W. Lampson (1943–) received an AB from Harvard University in 1964 and a PhD in EECS from the University of California at Berkeley in 1967. After a year as an Assistant Professor in EECS, he joined Berkeley's new Computer Science Department.

[14]Bruce G. Lindsay (1945–) received an AB from the University of California in 1967. He began working at the Computer Center with Professor Rene DeVogelaere on his Active Language Calculator.

[15]They in turn say: "Our notion of the capability list stems from the 'program reference table' idea first used in the Burroughs B5000 system." Interestingly, they credit the Rice Computer Project, [28] started by Martin Graham, as well as Burroughs for influencing their segmentation design.

[16]As secondary sources, Sturgis's thesis [63] also notes Robert Fabry's work on a capability-oriented system at the University of Chicago [15] and the Multics operating system's mapped address spaces, protection regions within processes (Multics rings), and distributed system code. [11]

[17]Designs like this would come to be known as microkernels.

layer. ECS files could have “holes” representing portions not currently resident in ECS; part of the representation of an open disk file was such an ECS file. So it was important for this initial design process to anticipate the needs of higher layers (disk files, directories, command processor, debugger, etc.) that would be designed later.

There had been a question of whether the existing hardware was sufficient. While CDC's SCOPE operating system ran in the PPUs, which could perform an Exchange Jump to switch the CPU from one user program to another, as much as possible of the new system would run on the CPU. Thus Berkeley wanted CPU programs to be able to initiate an Exchange Jump to make a system call; this required the CEJ/MEJ option that CDC had agreed to include for the second machine.[18] A serial line multiplexor for use with asynchronous terminals such as Teletype Models 33 and 35 had been designed for RTS and would be used with the timesharing system.[19]

In August 1968 the Extended Core Storage was finally installed on the A machine. (In December 1967 an additional 32K-word memory module and the 6638 disk subsystem had been installed.) In January 1969 the second 6400—the B machine—was finally installed. It had 32K words of central memory. Both machines were connected to the ECS (split 200K/300K words for the A and B machines) and the 6638 disk subsystem (17M words split evenly for the two machines).

At the end of October the complete design emerged as an elegant series of specification memos including all the object types and major functionality of the eventual ECS system. [58] This was an impressive accomplishment given the small number of person-months expended. It also represented a major escalation from the approach of Sturgis's May 1967 [52] and February 1968 [54] memos: rather than build a conservative system, he would apply and extend the latest ideas in operating system research.[20]

Implementing the ECS system

With specifications in hand, detailed design and implementation began. At this time the Computer Center system programming staff worked in an annex (no longer present) to South Hall, and design meetings were held in Lampson's office in Cory Hall. By October 8, 1968[21] Sturgis and Lindsay had been joined by Karl Malbrain[22] and Charles Simonyi.[23] Simonyi and Paul McJones[24] had been working since spring 1968 on CAL SNOBOL.[25] By fall they were finishing that project, and Lampson recruited Simonyi to join the timesharing project. Simonyi designed several ECS system structures, but quickly decided to pursue another job off campus and recruited McJones to complete his designs.[26]

Programs were written in CDC's COMPASS assembly language, keypunched by the programmers, and assembled and linked with batch jobs on SCOPE. Malbrain had previously written a CPU simulator as a debugging tool; it was combined with a PPU simulator written by Sturgis to allow simulation of the entire system, which was especially useful because the actual hardware had no traditional front panel allowing access to memory and registers.

Sturgis wrote the PPU code (initial load, master loop, I/O drivers) and low-level CPU code (ECS allocation, interrupts, initial process). Lindsay wrote the central CPU code (process object, scheduler, swapper, system entry/exit) and other objects (subprocesses, event channels, allocation blocks). McJones wrote more actions (capability lists, files, subprocess maps, operations). Malbrain wrote the interim command processor,

[18]CDC declined to provide two other options: a way for PPUs to directly access ECS and an additional read/write control unit for the 6638 disk subsystem.

[19]The multiplexor was designed by David J. Wheeler, on sabbatical from Cambridge University, with contributions by Graham.

[20]In a 2021 interview with the authors, Lampson said, “But there's no doubt what the inspiration was for many of the things in the system. And it was Howard. Not me.” [35]

[21]A design note from that day lists tasks for the four people.

[22]Karl Malbrain (1950–) was a second-year undergraduate in Electrical Engineering and Computer Science (EECS).

[23]Charles Simonyi (1948–) was also a second-year engineering undergraduate. He came to Berkeley from Hungary by way of Regnecentralen in Denmark, where he'd worked with Per Brinch Hansen on the RC 4000 and Peter Naur on GIER Algol.

[24]Paul R. McJones (1949–), who is the first author of this paper, was a second-year undergraduate in EECS.

[25]A SNOBOL4 dialect for the 6400; see [41].

[26]In December 1968 Simonyi joined Malbrain in a rushed cross-country drive to witness the Apollo 8 launch, foretelling Simonyi's later space travel.

called the Bead, and the SCOPE simulator that allowed SCOPE programs such as compilers to run on the ECS system. Keith Standiford,[27] who joined the project in the spring, began by writing a command program to send files to the printer.

Around this time two people from the new Computer Science Department "unofficially" joined the project: Jim Gray[28] and Jim Morris.[29] Gray's initial project was the line collector, which allowed the user to make corrections by typing control characters while the program was reading a line from the teletype. Gray also defined the standard representation for text, which was based on 7-bit ASCII (rather than CDC's 6-bit Display Code). Morris's initial project was a line-oriented text editor. While working on the editor's search command, he conceived of the initial idea for the Knuth-Morris-Pratt string-searching algorithm [29] as a way to avoid backing up in the search, which would have been awkward because he was only maintaining a single file buffer. [44]

The interim system

By the summer of 1969 enough of the ECS system existed to allow a public demonstration: editing, compiling and execution of Fortran programs from two teletypes simultaneously. Around that time the timesharing staff moved from crowded South Hall Annex to two apartments in an old university-owned building at 2515 Channing Way. A milestone was reached: development of the system on itself now began, using the SCOPE simulator to run the assembler and linker. The B machine was shared with the SCOPE system programmers on a fixed daily schedule, requiring dumping and reloading the ECS file system to magnetic tape.

Redell returned to graduate school after a one-year hiatus and joined the team, working with Lindsay—their first project was documentation. In October Vance Vaughan[30] joined the team, taking over completion and maintenance of the ECS system. Late that year Gene McDaniel[31] joined the team, initially working with Vaughan on tests and measurements for the ECS system.

In parallel with these activities, there was a behind-the-scenes effort to find a new academic sponsor. Lampson continued to advise the group until he left Berkeley to join Xerox Palo Alto Research Center around January 1970, but in addition to his teaching duties, the company he had co-founded[32] had been taking increasing amounts of his time. By July, Hebert had invited Gray to become project director. Gray's PhD advisor, Computer Science Professor Michael Harrison, wrote to Hebert on August 1 to recommend Gray, but warned of the need for the University to show its commitment to provide support for completing the timesharing system. [26] A month later Sammet wrote to Hebert, expressing pleasure at Gray's interest and assuring Hebert of his continued support of the project, mentioning "I have in mind that the work will also be directed toward providing documentation and personnel skills necessary to keep the program fully operative once it is developed." [49] (That summer the Computer Center had assigned Marianne Bentley[33] to assist with documentation.) In October, Gray's new role was marked by the publication of "An Overview of the CAL Time-Sharing System." [32] It combined a paper that Lampson had just presented at the second NATO software engineering conference [31] with Gray's first quarterly progress report. [20]

First users

In January 1970 Sturgis completed an interim facility for using the disk. [21] [59] This involved an I/O interface to the disk and a command-level program running under the Bead. I/O interfaces in general consisted of a PPU program that commu-

[27]Keith P. Standiford (1949–) was a second-year undergraduate in Electrical Engineering and Computer Science.

[28]James N. Gray (1944–lost at sea in 2007) received his BS and PhD from the University of Californa at Berkeley in 1966 and 1969, respectively. From 1969 to 1971, he was an IBM Post Doctoral Fellow in the Berkeley Computer Science Department.

[29]James H. Morris, Jr. (1941–) received a BS from Carnegie Tech and an MS and PhD from MIT. In January 1969 he became an Assistant Professor in the Berkeley Computer Science department.

[30]Vance Vaughan (1939–) came to Berkeley in 1957 as a freshman, received his BA in 1963 and his MA in 1966; along the way, he worked as an operator for the IBM 701 and then as programmer and research assistant in the Astronomy department.

[31]Gene A. McDaniel (1948–) was a fourth-year undergraduate in psychology and computer science.

[32]Berkeley Computer Corporation was building a large time-shared machine based on experience with Project Genie. [33]

[33]Marianne Bentley (1944–) received a BA in mathematics from Smith College in 1965. She began working at the Computer Center around 1968.

nicated with the device hardware and a pseudo-process (low-level CPU code) that interfaced between the PPU program (which could only read and write buffers in central memory) and processes (which could only send and receive events on event channel objects and read and write ECS files). [3] The interim disk program implemented a simple disk file system and allowed users to transfer complete ECS files to or from this file system.

At this point, the system could support "nearly 10" users, [63] who could be editing large text files and assembling them on the SCOPE simulator, although close cooperation by the users was necessary to manage their use of ECS space. Neither the Bead nor the interim disk program provided file access controls. But the increased capacity of the disk over the ECS eased further development of the timesharing system and allowed several people from the Computer Science Department to be invited to use the system on an experimental basis, taking advantage of the programming documentation (see [4] and [5]) recently completed by Bentley, Lindsay, and Redell. In 1969 Morris had ported the BCPL programming language to the 6400 running on the A machine, assisted by undergraduate Richard Aronoff. Now he made BCPL available as a native subsystem on the timesharing system. [46] As her introduction to the system, Instructor Laura Gould[34] wrote a program to convert a SCOPE text file to an ASCII text file. Later in the year, Morris's undergraduate student William Bridge[35] designed an implementation of the BASIC language. It was fully interactive on CAL TSS, and could also be used in batch mode under SCOPE on the A machine. Bridge used BCPL as the implementation language. [9]

Malbrain and McJones had taken a hiatus from timesharing development to design an assembly language programming system with PL/360-like syntax, a loader, and a debugger, but this project dragged out and was never finished. [37] SNOBOL was useful for writing text-processing programs, so McJones added teletype IN() and OUT() procedures to CAL SNOBOL, although it still had to be run under the SCOPE simulator.

Designing the disk system

To complete the timesharing system, several more layers were needed: permanent disk files coordinated with the ECS file system, directories, and an executive layer with a command processor, basic debugging, and accounting; collectively this was referred to as "the disk system." Sturgis was the system architect. Lindsay and Redell were responsible for disk files, McJones for directories, and Sturgis for the command interpreter and other facilities. Standiford began work on a display driver that allowed use of the dual-display operator's console to examine and modify central memory and ECS; he also virtualized the display to allow a user process to access the display and keyboard. Vaughan and McDaniel continued finishing up details of the ECS system. [22]

By summer of 1970, the disk and directory system designs were completed and implementation had begun. The ECS system was now complete except for some final I/O interfaces. Based on the accumulated experience, some redesign and reprogramming of the ECS system began. Sturgis began working in earnest on the executive layer design. [63]

At this time there was another personnel change: Gray resigned as director. [23] Gray's quarterly progress reports (see [20]–[23]) show a steady shift from optimism ("We are confident that TSS will gracefully support 100 student users when it is complete.") to pessimism ("The job is thankless, draining, mundane, and unpleasant.") over a nine-month period. In his defense, he was dealing with a programming team made up largely of volunteers and students who juggled work with their studies in an atmosphere colored by political protests.[36] Sturgis gamely assumed the directorship and served in that role for the remaining eighteen months of the project.

First student users

Gray's final progress report announced the intention of releasing initial versions of the new

[34]Laura E. Gould née Lehmer (1932–) was a researcher and instructor.

[35]William H. Bridge (1948–) received his AB in Computer Science at Berkeley in 1970 and went on to graduate school, also joining the Computer Center.

[36]President Nixon's invasion of Cambodia that spring had led to widespread demonstrations and the Kent State shootings.

disk file and directory systems by fall. [23] Internal memos cited this "September system," listing features including a TSS batch system for student "load and go" jobs and enumerating required tasks. [69] [38] [68] Completion of the "September system" stretched into the following year (and the batch system was never written), but an important milestone was reached that fall. As Sturgis noted in his first progress report: [60]

> As an experiment, in the Fall of 1970, 8 students from an elementary programming class were allowed the use of the system for one hour a day, 5 days a week. In these 5 weekly hours the 8 students were able to do the week's assigned programs, usually with time to spare. The 8 students placed so little load on the system that it is suspected we could have handled 16 students.
>
> The limitations of this temporary system stem from the fact that if a program is manipulating a file, the entire file must reside in ECS. Thus ECS rapidly fills up as more users attempt to use the system simultaneously. Other limitations are: the Bead command processor provides no protection between users, any user may access and modify any other user's files; no accounting information is collected; finally, there is no facility for forcing large programs to reside on the disk part of the time.

The course was Gould's CS120A, Computers in the Humanities, taught using CAL SNOBOL. [7] Another limitation not mentioned by Sturgis was that users were expected to manually request and release ECS space as they entered and exited from subsystems. For example, to edit a file, the user would type:

```
TRIM
SPACE,5000
C,EDITOR,S,<fname>,<user>
```

and then after exiting the editor would invoke the `TRIM` command again to return any unused space to the common pool.

Morris, pining after MIT's much more polished Compatible Timesharing System, [42] [72] wrote "The Idiot's Guide to TSS."[37] [45] The next quarter, Gould again used the timesharing system for CS120B; this time there were 12 students. And in the spring 1971 quarter, she taught 60 or more CS1 ("Computers and Data Processing") students using Basic. [18] Two other professors used the system to teach additional courses using Basic and Fortran. An evaluation by Vaughan published in August 1971 (see Appendix B of [61]) indicated that both students and teachers found timesharing to be preferable to batch computing. Students spent less time, and teachers found they could give more advanced assignments. This was in spite of the poor documentation, complex commands, limited access to teletypes, etc.

The final system

1971 began auspiciously for the Computer Center with a move to the second floor of newly finished Evans Hall, consolidating administration from Campbell Hall, the batch system from South Hall Annex, and timesharing from Channing Way. The computers would not arrive (from the Campbell Hall basement) until October. The brutalist 12-story concrete structure was conveniently located and the move seemed to signal that timesharing was becoming mainstream.

Another change doesn't seem to have registered with the timesharing project members: the Computer Center was reclassified as a service unit, losing the status as an Organized Research Unit that it had enjoyed since its founding in 1959.[38] Although there was no immediate outward impact, it meant the Computer Center could no longer initiate projects like CAL TSS.

By this time many parts of the new disk, directory, and executive layers were running, and utilities were available for using the card reader and (single) tape drive. On February 27, Sturgis and Hebert met with the Subcommittee on Time Sharing of the Chancellor's Advisory Committee on Computing, promising that "after March 15 and before the start of the Spring Quarter," the updated timesharing system would be available to general users from 2pm to 6pm each day. [73] Approximately ten teletypes would be made

[37]The title, inspired by a popular how-to guide for Volkswagen repair, was a barb directed at the CAL TSS designers. [44]

[38]This was a University-wide change; see [67].

available in the Computer Center, and users could could have their own teletypes connected to the system (hardwired rather than via dial-up lines). A financial analysis that assumed the system could support 100 teletypes[39] estimated that the cost would be $2 per user hour. The Subcommittee concluded: "Therefore, the proposed time sharing system appears to be very economical compared to commercial systems."

There was a mad dash to straighten out last-minute problems, [70] and the system was made available on March 16. In May a major revision of documentation was released. [6] An introductory document featuring annotated command sessions shows the complexity faced by users. [8] Work continued over the following months, adding important features and measuring and tuning the system, with roles shifting to address the highest priorities. Bridge and McDaniel began working with Sturgis on the executive layer. Vaughan continued to maintain the ECS layer, but also assumed the role of "user interface," talking to potential users, creating additional documentation, and troubleshooting the entire system. Another milestone occurred at the end of July, when disk space quotas were completed. This required remaking all the directories; once this was done, the timesharing programmers moved their files over from the Bead-based interim system to the new system, sharing it with the users (but during off-prime hours). [61]

The critical resource was ECS, only 300,000 words of which were available to the timesharing system—the equivalent of about 2,250,000 bytes. This modest storage device held all the system code and tables as well as the active parts of all the user files. In May, Sturgis and Vaughan had done detailed measurements of ECS usage, which they divided into three categories: system overhead shared by all processes; per-process fixed overhead, and per-process space for active parts of open files. In a report on the state of the system at the end of July, Sturgis noted it could support a maximum of 15 teletypes (users); he extrapolated that with improvements expected to be implemented by September, and using the entire 500,000 words of ECS, that number could increase to 50 teletypes. [61] The report contained projections of the impact of planned changes on ECS usage, showed the feasability of the available disk storage (about 8 million words—the equivalent of about 60 megabytes) for a base of 500 users sharing 50 teletypes over a 10 hour day, and enumerated a number of additional tasks to provide a "polished" system.

By October 25 the changes to support accounting had been made and an initial policy established for pricing the different resources: $2/hour for connect time, $130/hour for CPU time, $0.43 per kiloword-hour for ECS space, and $0.07 per kilosector-hour for temporary disk space,[40] with another $9.40/kilosector-month for permanent disk space—the A machine was billed at a flat $400/hour. [62] The system operator's instructions now included a daily step to write the accounting information to a magnetic tape, where it could be transferred to the A machine to be punched into cards suitable for input to the Computer Center's accounting system.

Another important task was underway: changing the disk file system to support "forced disk swapping." As Sturgis noted: [61]

> Programs exist which will compute for long times between teletype interactions. These programs will hold large amounts of ECS while computing, thus preventing more interactive programs which have released space from continuing. The forced disk swap is the system's method of preventing this situation. Work on this facility will begin this summer and should be completed late this year.

Redell (working alone) designed the complex logic via "pseudo-code" written in VERS, [14] a new programming language that had recently become available on the system. But before he could finish the actual implementation in assembly language, time—and money—ran out.

Although the Computer Center's finances had been excellent in Spring 1968 when the project was given the go-ahead, that turned out to be a point of inflection, with revenue from state and federal sources flattening or dropping from

[39]The actual number was about a dozen at that time.

[40]A kilosector was about 500,000 bytes or roughly one 1980s floppy disk.

that point onward.[41] By 1971, the Computer Center was running a deficit. While the project members diligently worked on the timesharing system, the Computer Center management and Chancellor's Office had been privately contemplating its fate. By November, they decided to terminate development of the timesharing system, sell the B machine, and dedicate the full ECS and 6638 disk drive to the A machine, in the hope of increasing throughput and thus bringing in additional revenue. [40] This was communicated to the timesharing project staff on November 29, 1971. [71]

Aftermath

Development ceased immediately; Hebert's request that the state of the source code and documentation be preserved was not heeded.[42] [43] The timesharing system continued to run until May 1972, when the B machine was sold. Berkeley wouldn't have timesharing on campus until the arrival of Unix a few years later.

Gray, during a bleak winter in New York, [25] wrote an IBM technical report describing aspects of the system. [24] Sturgis went to work at Xerox PARC, where his first project was to write his dissertation, "A postmortem for a time-sharing system". [63] In 1976 he and Lampson published a paper, "Reflections on an operating system design." [34] Redell and Lindsay also wrote dissertations based in part on their experience with the system. [47] [36] All agreed it had been a formative experience.

CONCLUSION

By the mid 1960s the idea of interactive computing was becoming attractive to sophisticated users of computing, but the necessary timeshared operating systems were not readily available. Berkeley's decision to develop its own timesharing system followed similar efforts at MIT, Dartmouth, and Berkeley itself (Project Genie). The resultant CAL Timesharing project was torn between conflicting goals of delivering low-cost service and exploring interesting research ideas that could provide advanced functionality. While the project seemed to be on track to delivering on both goals, economic conditions forced an early end to the project. A few later projects experimented with capability-based operating systems, but the idea has not caught on.

[41] Accurately tracing the causes would require additional research, but would likely involve Ronald Reagan at the state level and Project Apollo, the Great Society, and the Vietnam War at the federal level.

[42] Perhaps Hebert had reviewed the original 1966 CDC sales agreement, which stipulated that the University would develop a timesharing system. CDC was to be assigned a technical contact and to be provided with final copies of all software documentation and all reports, but there was no mention of receiving the actual source code. [10] (This was before IBM's 1969 unbundling, when software was not generally considered to be intellectual property.) In any case CDC never showed any interest in the project.

[43] Various project members kept their personal files and a few magnetic tapes; McJones has been collecting this material since 1980 for donation to an archive. Starting in 2018, Terry Heidelberg has managed to run snapshots of the interim and final systems on an emulator.

ACKNOWLEDGMENTS

We thank the project members, the three advisor/directors, Professors Taub and Graham, Ken Hebert, and the Chancellor's Office. We also thank the Bancroft Library and the Charles Babbage Institute. The original project was funded by the University of California through the Berkeley Computer Center; additional funding was provided by Award 67P7635 of the National Science Foundation and an educational allowance from Control Data Corporation.

■ REFERENCES

1. Records of the Office of the Chancellor, University of California, Berkeley, CU-149, The Bancroft Library, University of California, Berkeley.
2. Records of the CAL Timesharing System, Lot #2022.0154, Computer History Museum. https://CalTSS.computerhistory.org
3. —, "TS Interrupt System," September 2, 1969, [2], 690902-int-sys.
4. —, "CAL Time-Sharing System Users Guide," November 1969. Actually, this is the programmers guide, [2], 6911-users-guide.
5. —, "CAL-TSS Internals Manual," November 1969, [2], 6911-internals.
6. —, "Time-Sharing System Manual," Part 3 of Volume III (The 6400 Computer System) of *The Cal Computer Center Users Guide*, May 1971, [2], 7105-tss-manual.
7. —, "General Catalogue, 1970–1971," University of California, Berkeley, p. 220, May 15, 1970. https://digicoll.lib.berkeley.edu/record/1651
8. Marianne Bentley and Vance Vaughan, "Introduction to CAL TSS," July 1971; updated November 1971, [2], 7111-intro-mab-vv

9. [William H. Bridge,] “BASIC,” Section E of Part 4 (Languages and Processors) of *The Cal Computer Center Users Guide*, May 1971, [2], 7105-cal-basic-whb.

10. Control Data Corporation. “Agreement for the sale of Control Data Equipment,” November 1, 1966, [1], Box 77, Folder 1.

11. F. J. Corbató and V. A. Vyssotsky, “Introduction and overview of the Multics system,” In *Proceedings of the 1965 Fall Joint Computer Conference, part I*), Association for Computing Machinery, pp. 185–196. https://doi.org/10.1145/1463891.1463912

12. Jack B. Dennis and Earl C. Van Horn, “Programming semantics for multiprogrammed computations,” *Commun. ACM* vol. 9, no. 3, pp. 143–155, March 1966. https://doi.org/10.1145/365230.365252

13. Edsger W. Dijkstra. “The structure of the ‘THE’-multiprogramming system,” *Commun. ACM* vol. 11, no. 5, pp. 341–346, May 1968. https://doi.org/10.1145/363095.363143

14. Jay Earley and Paul Caizergue, “VERS Manual Version 4”, Computer Science Department, University of California, Berkeley, October 1971, [2], 7110-vers-manual-v4-je-pc.

15. Robert S. Fabry. “A User’s View of Capabilities,” ICR Quarterly Report no. 15, The Institute for Computer Research, The University of Chicago, Nov. 1967.

16. D. J. Farber, R.E. Griswold, and I. P. Polonsky, The SNOBOL3 Programming Language, *The Bell System Technical Journal*, vol. XLV, no. 6, July-August 1966, pp 895-944. https://doi.org/10.1002/j.1538-7305.1966.tb04224.x

17. J. W. Faricy, “Special Allowance - 6400 System: University of California, Berkeley”, memo to R. R. Burns, July 28, 1966. Control Data Corporation Records, Marketing, Sales and Public Relations (CBI 80), Charles Babbage Institute, University of Minnesota, Minneapolis.

18. Laura E. Gould, “Instructions for running BASIC on the CAL Time-Sharing System,” May 1971, [2], 7105-instructions-for-basic-leg.

19. M. Graham, memo to Vice-Chancellor Loy L. Sammet, May 16, 1968, [1], Box 121, Folder 9.

20. Jim Gray, “Progress Report on 6400 CAL Time-Sharing System,” October 11, 1969, [2], 691011-progress-jng.

21. Jim Gray, “(Lack of Visible) Progress Report: CAL-6400-TSS,” January 15, 1970, [2], 700115-progress-jng.

22. Jim Gray, “CAL Progress Report,” April 15, 1970, [2], 700415-progress-jng.

23. Jim Gray, “Progress Report—CAL-TSS,” June 1, 1970, [2], 700601-progress-jng.

24. James N. Gray, Butler W. Lampson, Bruce G. Lindsay, and Howard E. Sturgis, “The control structure of an operating system,” Technical Report RC 3949, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, July 1972, [2], 7207-control-structure-jng.

25. Jim Gray. Oral History Interview with Jim Gray. Charles Babbage Institute, University of Minnesota Digital Conservancy, 2002. https://hdl.handle.net/11299/107339

26. Michael Harrison, memo to Ken Hebert, August 1, 1969, [1], Box 121, Folder 9.

27. K. Hebert, “CDC 6400 System Development”, memo to Vice Chancellor Loy L. Sammet, July 18, 1968, [1], Box 121, Folder 9.

28. J. K. Iliffe, Jane G. Jodeit, “A dynamic storage allocation scheme,” *The Computer Journal*, vol. 5, no. 3, pp. 200–209, November 1962. https://doi.org/10.1093/comjnl/5.3.200

29. Donald E. Knuth, James H. Morris, Jr., and Vaughan R. Pratt, “Fast pattern matching in strings,” *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977. https://doi.org/10.1137/0206024

30. B. Lampson, M. Pirtle and W. Lichtenberger, “A user machine in a time-sharing system,” *Proc. IEEE* vol. 54, no. 12, pp. 1766–1774, December 1966. https://bwlampson.site/02-UserMachine/Abstract.html

31. Butler W. Lampson. “On reliable and extendable operating systems,” presented at 2nd NATO Conference on Techniques in Software Engineering, Rome, September 1969; reprinted in *The Fourth Generation, State of the Art Report*, no. 1, pages 421-444. Infotech, 1971. http://bwlampson.site/07-ReliableOS/07-ReliableOSAbstract.htm

32. Butler W. Lampson, “An Overview of the CAL Time-Sharing System”. October 10, 1969. Includes versions of [31] and [20], [2], 691010-overview-bwl.

33. B. W. Lampson, “Dynamic protection structures,” in *Proceedings of the Fall Joint Computer conference,* Association for Computing Machinery, New York, pp. 27–38. https://doi.org/10.1145/1478559.1478563

34. Butler W. Lampson and Howard E. Sturgis, “Reflections on an operating system design,” *Comm. of the ACM*, vol. 19, no. 5, pp. 251-265, January 1976. https://doi.org/10.1145/360051.360074

35. Butler W. Lampson, interviewed by Paul McJones and Dave Redell, April 25, 2021.

36. Bruce Gilbert Lindsay, “Exception Processing in Computer Systems, PhD dissertation, Dept. of Computer Science, University of California, Berkeley, CA, 1977, [2], 77-thesis-bgl.

37. Karl Malbrain and Paul McJones, “COOL-AID,” 1969, [2], 69-cool-aid-km-prm.

38. Karl Malbrain?, “Batch system,” [June 1970?], [2], 7006-batch-system-km.

39. E. W. Mauchlan, “Note for Regents’ Committee on Finance, July 15, 1966: Upgrading of Computer System and Authorization to Submit Grant Application for Computer,” [1], Box 77, Folder 1.

40. E. W. Mauchlan, L. L. Sammet, and D. R. Willis, memo to Chancellor Bowker re Computer Center funding, November 12, 1971, [1], Box 121, Folder 12.

41. Paul McJones, “CAL SNOBOL archive,” web site, 2021-2022. https://www.mcjones.org/CAL_SNOBOL/

42. Special issue on time-sharing at MIT, *IEEE Annals of the History of Computing,* vol. 14, no.1, January/March 1992.

43. Calvin C. Moore, *Mathematics at Berkeley: A History*, Wellesley, MA: A K Peters, 2007.

44. James H. Morris, Jr., *Thoughts of a Reformed Computer Scientist: On the Nature of Real and Artificial Intelligence*, Amazon, ISBN 9798492652494, 2021.

45. James H. Morris, Jr., “The Idiot’s Guide to TSS,” Fall 1970, [2], 7009-idiots-guide-jhm.

46. [James H. Morris, Jr.,] “BCPL,” Section F of Part 4 (Languages and Processors) of *The Cal Computer Center Users Guide*, May 1971, [2], 7105-cal-bcpl-jhm.

47. David D. Redell, “Naming and Protection in Extendible Operating Systems,” PhD dissertation, Dept. of Computer Science, University of California, Berkeley, CA, 1974, [2], 7409-thesis-ddr.

48. L. L. Sammet, memo to Professor Butler Lampson, September 17, 1968, [1], Box 121, Folder 9.

49. L. L. Sammet, memo to Acting Director Hebert, September 3, 1969, [1], Box 121, Folder 10.

50. R. N. Schuhmann, letter to Prof. A. H. Taub, May 5, 1966, [1], Box 77, Folder 1.

51. R. N. Schuhmann, letter to Prof. A. H. Taub, October 11, 1968, [1], Box 121, Folder 9.

52. H. Sturgis, memo on an approach to timesharing for the 6400 with ECS, May 4, 1967, [2], 670504-timesharing-hes.

53. H. Sturgis, memo on a basic timesharing system described as an abstract machine, November 15, 1967, [2], 671115-basic-timesharing-hes.

54. H. Sturgis, project plan with cover letter, February 20, 1968, [2], 680220-mhg-to-hes.

55. H. Sturgis. “Capabilities, basic objects,” February 29, 1968, [2], 680229a-basic-objects-hes.

56. H. Sturgis, “Map, operation, capability, objects,” July 14, 1968, [2], 680714-map-oper-capab-hes.

57. H. Sturgis, “Process, subprocess,” July 16, 1968, [2], 680716-proc-subproc-hes.

58. H. Sturgis, ECS layer specification, November 1, 1968, 44 pages. Introduction and 8 numbered sections, [2], 681101-ecs-layer-hes.

59. H. Sturgis, “Interim Disk System: Preliminary Manual,” [January 1970,] [2], 7001-interim-disk-sys-hes.

60. H. Sturgis, “About Cal TSS,” January 6, 1971. Sturgis’s first status report, [2], 710106-about-hes.

61. H. Sturgis, “CAL TSS Report,” August 1, 1971. Sturgis’s second status report, [2], 710801-cal-tss-report-hes.

62. H. Sturgis, “A note on charging,” August 17, 1971, [2], 710817-charging-hes.

63. H. Sturgis, “Postmortem of an operating system,” PhD dissertation, Dept. of Computer Science, University of California, Berkeley, CA, 1973, [2], 7401-thesis-hes.

64. A. Taub, “Plans for the Berkeley Computer Center,” memorandum to Chancellor R. Heyns, April 14, 1966, [1], Box 77, Folder 1.

65. A. Taub, memo to Vice-Chancellor Alan Searcy, September 16, [1], Box 77, Folder 1.

66. A. Taub, National Science Foundation Computer Facility grant proposal, undated (circa fall 1966), [1], Box 77, Folder 1.

67. Angus Taylor, memo to Chancellor Heyns, January 18, 1971, [1], Box 121, Folder 12.

68. V. Vaughan, “September system,” [June 1970?], [2], 7006-sept-sys-vv.

69. V. Vaughan. “Reconstituted list of things to be done on the ECS level of Cal TSS,” June 1, 1970, [2], 700601-reconstituted-list-vv.

70. V. Vaughan, “Facilities available,” March 15, 1971, [2], 710315-facilities-avail-vv.

71. V. Vaughan, “CAL Time-Sharing System Status,” November 29, 1971, [2], 711129-status-vv.

72. David Walden and Tom Van Vleck, eds, *Compatible Time-Sharing System (1961–1973): Fiftieth Anniversary Commemorative Overview* (PDF). IEEE Computer Society, 2011. https://multicians.org/thvv/compatible-time-sharing-system.pdf

73. Edward L. Wilson, “Report to Chancellor’s Advisory Committee on Computing from Subcommittee on Time Sharing,” February 27, 1971, [2], 710227-subcommittee-on-time-sharing-elw.

Paul McJones is retired in Mountain View, California. Contact him at paul@mcjones.org.

David Redell is retired in Redwood City, California. Contact him at dave.redell@gmail.com.