SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 10, 993–1008 (1980)

The Evolution of the Operating System EMAS 2900

P. D. STEPHENS AND J. K. YARWOOD
Edinburgh Regional Computing Centre, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland

AND

D. J. REES AND N. H. SHELNESS
Department of Computer Science, University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, Scotland

SUMMARY

As a result of experiencing problems with manufacturer's software on the early 2900 machines Edinburgh University took the bold step of attempting to move the general purpose time-sharing system EMAS from an ICL 4-75 to an ICL 2970 computer. This paper describes the move together with the changes made and includes some preliminary performance figures from the new system. This implementation project has produced a major time-sharing virtual memory operating system with a fraction of the effort required to implement the original system.

KEY WORDS 2900 Series Operating system Multi-access Benchmarking Re-implementation

INTRODUCTION

By January 1974 the Edinburgh Multi Access System (EMAS)[13] had reached a high level of efficiency. When running on the ICL 4-75 machine whose configuration is shown in Figure 1 it was capable of supporting up to 32 simultaneous terminals with satisfactory response. The system was popular and soon overloaded and the Regional Computing Organisation* drew up an Operational Requirement for a further machine; this machine was to support 64 terminals (i.e. about twice those supported on the 4-75) and simultaneously process twice the batch throughput of the University 370/155. A four-part batch and interactive benchmark (the Glasgow benchmark)[11] was drawn up based partly on performance measurement of EMAS.

The Regional Computing Organisation accepted a tender from ICL for delivery of an ICL 2980 in September 1975 with an undertaking to pass all four parts of the agreed benchmark by June 1977. The configuration of the 2980 is shown in Figure 2. In fact the 2980 destined for the Regional Computing Organisation was diverted to the European Space Agency and a 2980 did not arrive until May 1976. A smaller machine—the ICL 2970 (configuration in Figure 3)—arrived in early summer 1975 for evaluation and software development. At this stage there was no intention of writing an Operating system and no relevant work was done apart from writing a compiler for the EMAS implementation language IMP.[12] By early summer 1976 most of Edinburgh's

* The Regional Computing Organisation comprises the Universities of Edinburgh, Glasgow and Strathclyde.

0038–0644/80/1210–0993$01.00

Received 25 March 1980
Revised 19 June 1980

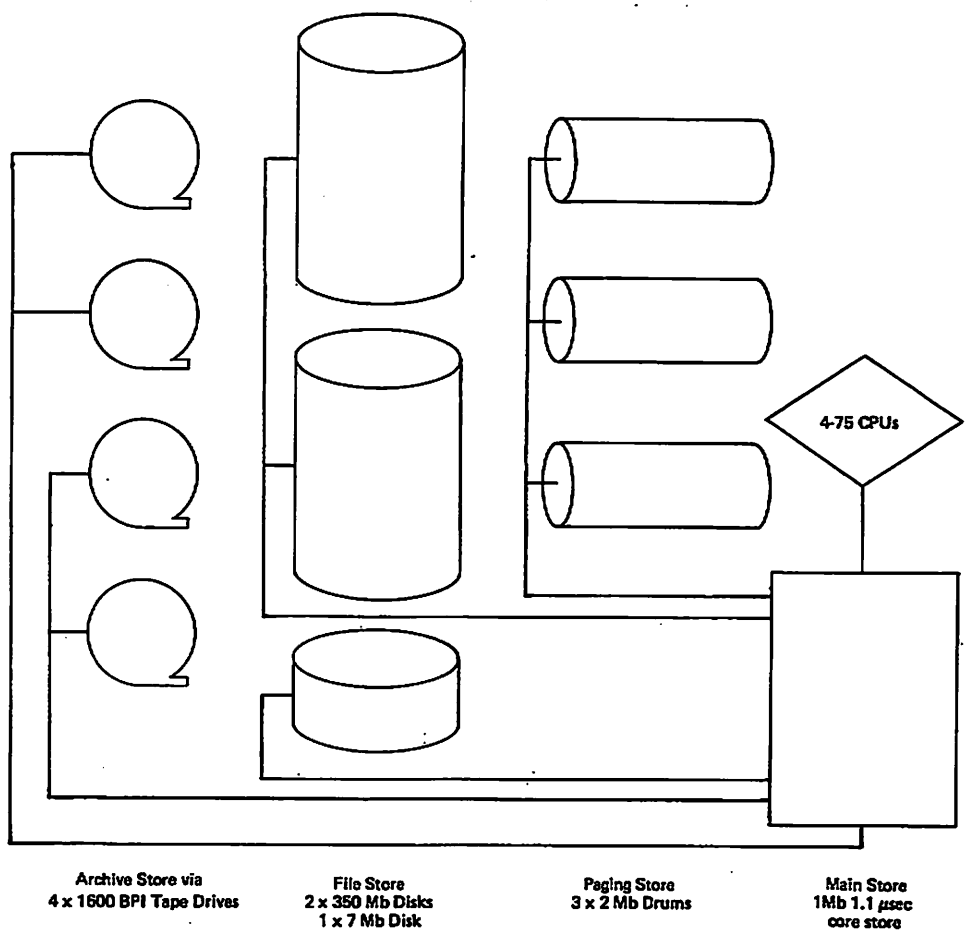

Figure 1. EMAS 4-75 configuration

system programmers were convinced that ICL would not pass the interactive benchmark and that the likely margin of failure was large; consequently they began to speculate idly on the possibilities of bootstrapping an operating system from a 4-75 to a 2970. ICL remained confident, the University authorities appeared to believe ICL and inaction followed. It was not until October 1976 that a serious attempt was made to produce an alternative system. A small team of four full-time people and about half a dozen part-time helpers set out to write the EMAS 2900* operating system. Progress was rapid. By spring 1978 an internal service for project members was working, and a few 'real' users were invited to use it. In October 1978 a service was offered on the 2970 although development continued. In January 1980 service was opened on the Regional Computing Organisation 2980 and also on the University of Kent 2960.

We do not propose in this paper to give a detailed introduction to the design or initial implementation of EMAS. An overview has been published,[13] together with detailed descriptions of specific aspects of the system.[2, 7, 8, 10] It is useful to note the features that give the system its current appearance:

* In this paper EMAS refers to the original system running on the 4-75 and EMAS 2900 to the re-implemented system.

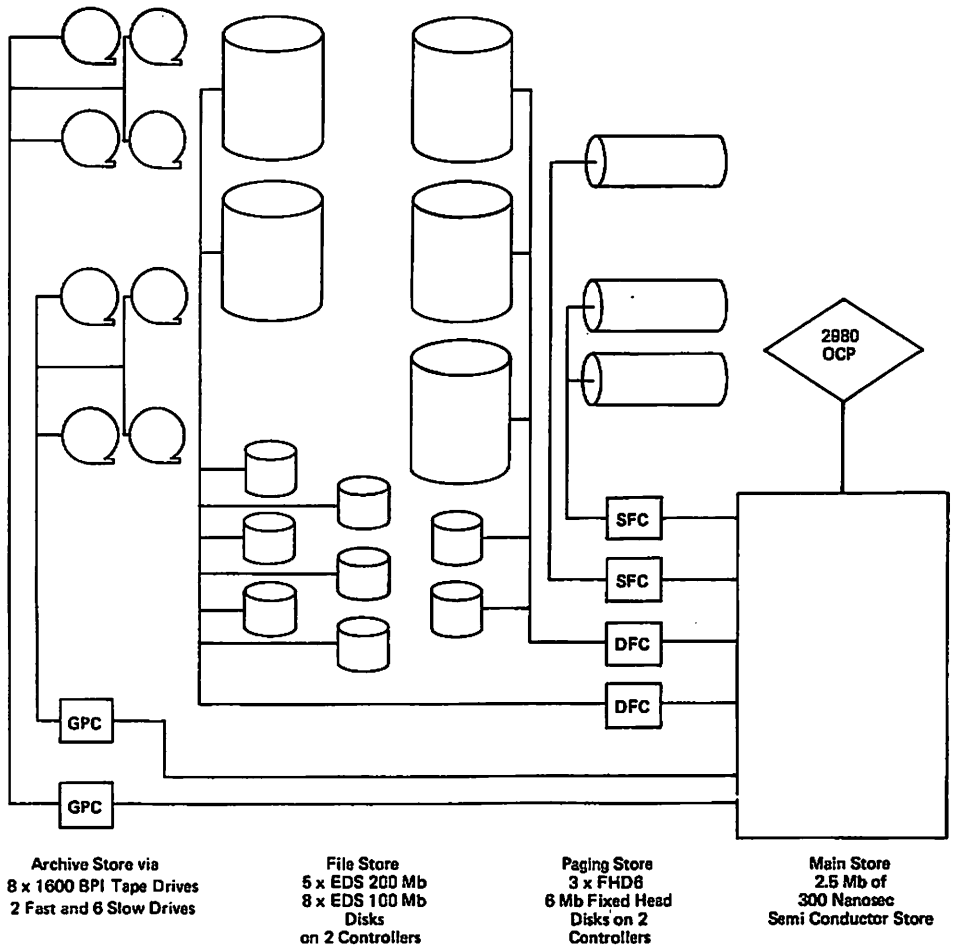

Figure 2. EMAS 2980 configuration

Interactive working

The system is normally accessed from an interactive terminal, which it uses as its major source of control information. This may be one of a large number of devices attached via a geographically distributed network.

Multiple virtual memories

Each user is allocated a 16 megabyte virtual address space.

Mapped files

Files are not accessed via a procedural interface (read record, write record, etc), but by being associated with a range of virtual addresses and accessed as if they were memory.

Controlled sharing of information

The system imposes no restrictions on how a user organizes his programs and data. If, however, at any instant two or more users are running the same program or

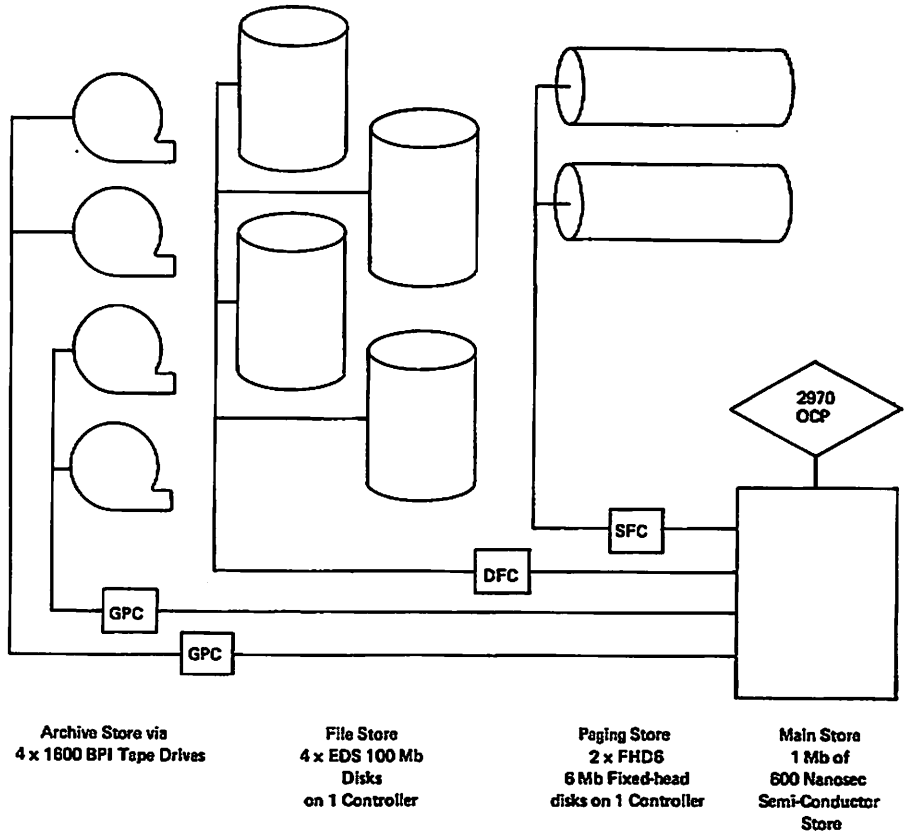

Figure 3. EMAS 2970 configuration

operating on the same data, the system notes this and arranges for a single copy of the relevant entity to be shared.

Transparent memory hierarchy

The system operates a three-level memory hierarchy consisting of

File store	(on disc)
Paging store	(on a drum or fixed head disc)
Main store.	

These levels are managed by the supervisor in a way that is transparent to the user, who is only aware of files and virtual addresses.

Minimal user constraints

The system attempts to restrain the user as little as possible in his use of files, languages, virtual addresses etc. There is a set of facilities provided by a standard subsystem, but the user may ignore it and easily provide his own subsystem if he so wishes.

Minimal information loss through crashes

If the system crashes (whether due to hardware or software) an attempt is made to minimize the information loss.

No system degradation under load

The system cannot allocate more resources than it possesses, but it should allocate as much as possible to the users at all times.

Repeatability and enforced fairness

The resources used by a job should be dependent on its requirements, and not on other demands on the system; therefore if a job is run again it should use the same resource. A user should get a fair share of the system determined by his own requirements and the number of users of the system.

DESIGN PHASE

The design phase was unlike the corresponding stage of any other project known to the authors. It consisted of 15 months of academic discussion between people who were sure that they could write a much better operating system than the one supplied. This stage was followed by a very short period of ruthless decision taking. Of the major decisions taken at this time only one was seriously wrong and that one proved relatively easy to reverse. In spite of this fairly high success rate we would hesitate to suggest that others follow this procedure. The major constraint on the design can be seen at a glance from the diagram of EMAS structure (Figure 4). The Kernel and Director comprise about 144Kb of code out of a total of nearly 1Mb of system code plus all the user written code and packages. In view of the small number of people available and the limited timescale the only feasible strategy was to implement the Kernel and Director so that most of the rest of the code could be transferred by recompilation with minor amendments. The principal problems of re-implementation are caused by the substantial hardware differences between the 4-75, which has IBM 360 architecture, and the 2900 Series.[4] These differences lie in three areas:

1. Stack support: the 2900 series is fundamentally a one accumulator, one index register machine with a hardware supported stack, whereas the IBM equivalent has 16 fast general registers. The 2900 tries to make up for the lack of fast registers by high performance 'slave' stores. These fast stores are invisible to the programmer but hold recently accessed variables, enabling the main store to be bypassed. This is a very attractive idea: register optimization—such a problem to the programmer or compiler writer—is performed by hardware. However the performance gains from slaving are typically small—10–15 per cent of execution time. Clever register optimization on a 4-75 can be much more effective.
2. There are substantial differences in paging. On the 4-75, paging was added as an afterthought and did not affect the privileged program states. On the 2900, the supervisor could be paged, and the large VM size (8196 Mb) and small page size (1 Kb) caused further problems: page and segment tables must be in contiguous real addresses but real store is not, in general, contiguous on the 2900 series.
3. The 4-75 had devices on simple channels which could execute one channel program at a time, whereas the 2900 had peripheral controllers which could optimize transfers from several devices at once.

Against this background the original system was reviewed and revised to make the

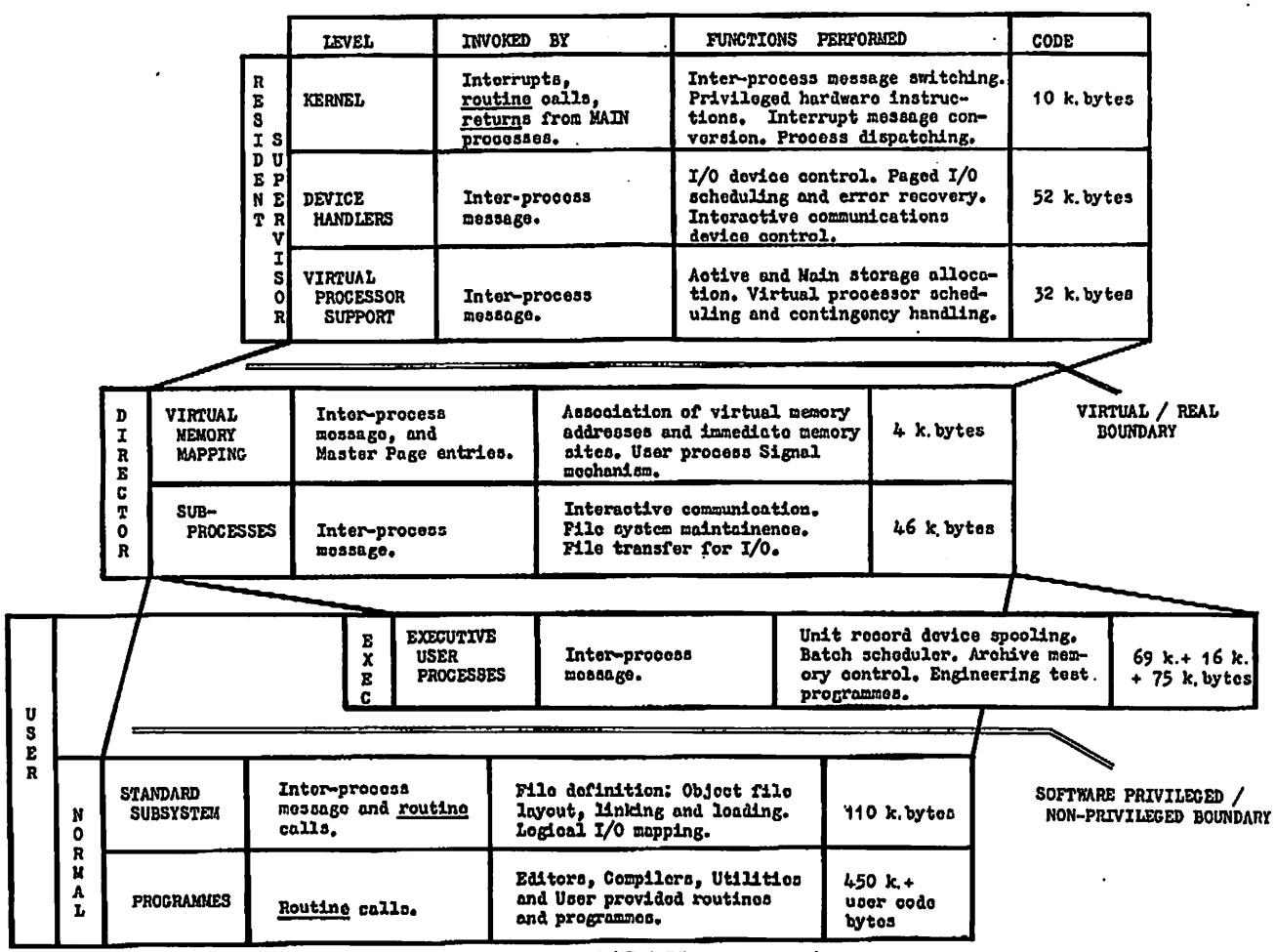

Figure 4. EMAS 4-75 component sizes

minimum changes necessary. The results of this review were as follows:

System structure

The basic structure of EMAS consisted of a small resident Supervisor and a paged supervisor process (Director) which together provided for each user a large virtual memory or address space. The low address part of the virtual memory is not accessible to the user and the various parts of the software are linked together by a message passing system and dispatcher.[13] These features have been transferred to EMAS 2900 with only one minor change: Director has been implemented as a set of privileged procedures running in the user's virtual machine, as in MULTICS, rather than a separate process associated on a one for one basis with each user process. It was necessary to use a separate process on the 4-75 as the protection mechanism could not otherwise prevent the user from corrupting Director. The excellent access control and System Call features of 2900 architecture have enabled the simpler and more efficient approach to be adopted.

File system

Central to EMAS is its File Store[8] together with its Backup and Archiving System.[14] This holds named files for all users, each file consisting of an unstructured sequence of bytes of arbitrary length. The File System is virtual in that files are accessed by becoming part of the virtual memory (connection) rather than by any record access mechanism. Director maintains the file store and has ingenious algorithms for allocation and deallocation of space to avoid fragmentation. EMAS 2900 has adopted the EMAS file system in its entirety, even maintaining the unit of disc allocation at one EMAS segment (64Kb) rather than one 2900 segment (256Kb). Having the unit of allocation at less than a segment involved small changes to the virtual memory management but this was preferred to a much larger unit of allocation which would cause fragmentation losses that would be unacceptable on the smaller discs ($<$100Mb). Further, a consistent block size simplified movement of the file system maintenance utilities and facilitated a common archive (tape) store for both machines.

Configuration independence

In EMAS, which was a prototype system, the configuration was built into the code. Flexibility was restricted to enabling the operator to mark unserviceable devices as not available. In EMAS 2900 all configuration information appears in a resident read only segment which can be considered part of the machine on which the system runs. This segment could be nominated at load time by the Operator (a procedure used by ICL on 2950 and 2956 machines) but the design aim was to construct this segment by means of a General Reconnaissance Of Peripheral Equipment (GROPE). This would involve a two-part system load. Initially a cut down supervisor is loaded. This performs the GROPE, constructs the read only segment, establishes communication with the Operator and loads the system. The design aim was to allow the system to run on any hardware configuration then announced without change.

Local control

An important principle of EMAS was that of local decision-making. Decisions—particularly those affecting page replacement—are taken on information relating to the behaviour of the relevant process and not on the basis of all processes currently in the

multiprogramming set. This ensures reproducibility of program behaviour and prevents a program that exhibits anti-social paging behaviour from affecting other well-behaved processes. The many beneficial effects of local control (particularly in terms of fairness and elimination of thrashing) are discussed at length elsewhere.[10] However this organization is not readily apparent in the original system which consisted of modules partitioned by function. These modules can make local decisions by reference to the process master segment[13] (a sort of run-time database) of fixed and complex structure.

The design aim for the new system was to provide a module—the Local Controller—on a one for one basis with each process, to run in the process's address space. This method would reflect more accurately the structure of the system and would mean that the run-time database could consist of ordinary IMP variables and arrays. Thus the database layout, which is essentially a matter private to the Local Controller, would be determined by the compiler each time the Local Controller was compiled, and could be altered by recompilation. A further advantage of this arrangement was that the sizes of arrays could be dynamic, based on information discovered at GROPE time. The database would still page with the process and would not be in main store when the proces was in the wait state. These advantages, particularly that of avoiding a fixed and unalterable database format, were considered to outweigh a quite serious disadvantage. 2900 architecture[4] provides only one software stack switching interrrupt, the OUT interrupt, which would have to be used for Director–Local Controller communication and also for Local Controller–Supervisor communication. Such double use is possible, but expensive, since the Interrupt Steering Table is constantly updated.

Virtual memory control

Great thought was given to the design of virtual memory management software and tables since system overhead was mostly directly related to the amount of paging traffic. In those considerations much weight was given to various performance measurements made on the original EMAS system. [1,2,9]

The first major decision concerned page size. System 4 had 4Kb pages while 2900 had 1Kb pages which could be grouped by software into extended pages (epages). The performance measurers were certain that a 1Mb 4-75 would perform best with 2Kb pages but software changes would be minimized by keeping to a 4Kb epage. The decision was to program with a potentially variable epage size but to start with a 4Kb page.

Secondly, the EMAS memory hierarchy was critically scrutinized. In this each process has a core working set which is a true subset of the larger Active store (Drum*) working set which is itself a subset of the process's virtual memory. If no drums are serviceable there is no interactive service; this was reasonable on System 4 where the Discfiles are very slow, but unreasonable on 2900. EMAS 2900 must be able to run without any fixed head paging device and also to run effectively with much less than the optimal amount of such storage.

Thirdly, the paging strategy was examined. Should EMAS 2900 use the EMAS method of preloading a process's working set of pages when that process enters the multiprogramming set (often called the Swopped Working Sets or SWS strategy) or should the simpler demand paging strategy be adopted? SWS was markedly superior

* The paging device is referred to throughout as a Drum, although on 2900 it is in fact a small fixed-head disc.

on EMAS and has recently acquired powerful theoretical support[5] so the decision here was to retain the SWS strategy.

Fourthly, program and data sharing was reconsidered in the light of 2900 hardware assistance, viz a second (public) segment table for all processes and also cascaded indirection of segment table entries. Although System 4 provided no hardware assistance, EMAS provided extensive program and data sharing in a manner transparent to users. The sharing was a most successful feature in that it enabled the operating system to effect substantial economies in core and drum space as well as a more modest saving in page transfers. These savings are particularly valuable since they increase with load; thus sharing helps performance most when the system is overloaded. Sharing via public segments is obviously possible but such sharing introduces a problem of selection; using indirect local segment table entries to point at a shared segment table does not remove the selection problem. Both methods have the serious side effect of combining the page use markers for shared material, thus preventing the accurate maintenance of the process's working set. EMAS 2900 ignores these hardware aids and uses the public segment table only as a local segment table for the supervisor.

Lastly, a minor change in the handling of unused page frames was incorporated. A quirk of implementation had resulted in EMAS losing the contents of a page frame when it had been successfully written out after use. If the page was wanted again it could not be extracted from the list of unused page frames on those occasions when the frame had not yet been re-used. This inconsistency was never removed since the free page list had typically 20–50 entries as against a total of 1500–2000 'active' pages and it seemed likely that the gains from 'recapturing' the page would be small. However in view of the large store sizes available on 2900 series, EMAS 2900 would recapture pages.

Communications

The communications on EMAS had been overtaken by events during the eight-year life of the system. The design used ICL's hardware multiplexor (The Multi-Channel Communications Control Unit) which necessitated fixed resident buffers and a 'Buffer Manager'. Later the MCCCU was replaced by a Front End Processor but the essentials of the software design were unchanged. The design worked well when all terminals were operating at ten characters per second. What had not been foreseen was the great increase in terminal speed and hence I/O in a terminal session; a further surprise was the attraction of EMAS's secure file system. Programmmers using machines in Newcastle, Cambridge, Harwell and Manchester kept their programs and data on EMAS and transmitted them daily. The amount of paging traffic on EMAS concerned with trivial buffer filling and emptying rapidly became insufferable. A radical rethink was required, complicated by the inability of ICL to provide a communications device that seemed likely to fulfil the demands EMAS 2900 would make on it.

The elegant solution adopted will be fully described in a subsequent paper. The main features are as follows:

1. A Digital PDP11 as Front End Processor interfaced to ICL's 2900 Application Module interface via locally built hardware.

2. All control functions to reside in the FEP.

3. Transfers between processes and the FEP to be made using large buffers in *virtual* memory. This would involve the software in halting transfers at page boundaries, arranging page replacement and restarting the transfer.

Two features would enable the virtual buffering to be incorporated. Firstly, the address translation facility of Peripheral Controllers; secondly, the software structure of EMAS 2900 necessary for local control. Just as a Local Controller manages the paging of the working set of a user process, so a modified Local Controller—the Communications Controller—can manage the paging of all virtual buffers involved in transfers. Indeed, since both Local and Communication Controllers could use the full range of Supervisor services, it seemed likely that very little new code would be required. The data sharing of EMAS 2900 would operate on such buffer pages without any amendment.

The designers were aware that basing the communications on virtual buffering with paging of buffers during transfers was a novel strategy. However, the disadvantages of the alternatives made them uninviting.

TOOLS

A brief description of the tools used in developing EMAS 2900 seems necessary. No new tools were constructed as part of the project—the most important were already available and the others were transfered at an early stage from the original system to the new.

The least sung but most valuable tool was the original EMAS system in spite of being severely overloaded. Its terminal access, editors, compilers and debugging features were invaluable. Above all its secure file store preserved and protected the source code. No source code was lost or destroyed from the start of the project in October 1976 until January 1978 when development was moved to the new system. In spite of the load on the system it was usually possible to edit a module, recompile it, link a new test supervisor and write it to magnetic tape for testing on the 2970 in about half an hour of elapsed time.

The second tool was the same dialect of the Edinburgh Implementation Language (IMP)[12] as was used in the original system. We do not claim that IMP is technically superior to more modern languages such as Concurrent Pascal,[6] but there were compelling reasons to use it. Firstly, it existed—the language and three compilers (for System 4, 2900 and also a System 4 to 2900 cross compiler), together with a large selection of source code formatters, diagnostic packages etc. Secondly, all the implementers understood every nuance of the language, and thirdly, we wished to transport source code from the old system.

In retrospect two aspects of IMP were particularly valuable. In the early stages the ability to obtain source language diagnostics from any failure saved many hours of debugging time. Later the ability to find out how often each source statement had been executed enabled the small amount of effort available for tuning to be deployed effectively.

The third tool available was system monitoring code. Much effort had been spent on monitoring the early EMAS system but two particular areas had proved of lasting value. The ability to monitor the message passing mechanism in a highly selective way has proved invaluable in disentangling curious interactions between components. Also

of value was the ability to write the system to magnetic tape and later examine the tape with a utility which could reconstruct, in source terms, complex tables and linked lists.

Fourthly, there was the Edinburgh Remote Terminal Emulator—ERTE.[1] This could produce between 8 and 64 interactive terminals for testing, performance measurement and benchmarking.

IMPLEMENTATION

The EMAS 2900 project was informal in structure, short in duration and small in size. It began in October 1976 with the aim of providing a service to invited users in January 1978 and a full service in October 1978. The project started with the four authors augmented by a group of six or seven individuals who made specific contributions in an area of expertise (e.g. device handlers). The contributions of this second group were limited in time. The project increased in size when the system was able to support superstructure and had reached eight full-time and four or five part-time members by September 1978. The total effort applied up to the start of full service was about 15 man years—very close to the prediction of Whitfield and Wight.[13] During the two-year project about 1000 hours of dedicated 2970 time was used. This total does not include terminal access to the initial user service or any time used on System 4.

The implementation was mostly free from unpleasant surprises. GROPEing was very difficult to implement since it involved using the hardware at a basic level, where there were great differences between the various machines. Had we foreseen the pain involved or the changes needed when the hardware was upgraded—a fairly frequent occurrence in the early years of the 2900 series—we would not have attempted this extension. Nevertheless now that all is working and understood, it greatly simplifies software maintenance to have a single self-configuring software product. It is clear in retrospect that we underestimated the number of occasions on which peripheral equipment would be unplugged for maintenance purposes and the difficulties of re-incorporating such peripherals into the system.

The early communication software was shaken by the discovery of a curious and then undocumented omission in all 2900 Peripheral Controllers. These Controllers perform address translation with full protection just as does the processor—this is a very desirable hardware feature. Having done all this the controllers fail to update the read and write history bits in the page tables so that the Local Controller could miss seeing that the virtual communications buffer had been used or updated. A software bodge managed to cure this but at the cost of writing out pages that have not been updated.

In spite of these minor tribulations the implementation timescales were met. Although the initial user service was not started until May 1978 the full service started in October as planned with rather more than the promised facilities. The final code size of the Supervisor was greater by about 30 per cent than for the original system. Almost all the increase was in driving the more complicated 2900 peripherals. The drum driver was markedly more complicated since the 2900 Sector File Controller operates with 1K sectors. Each EMAS 2900 transfer involves initiating four requests and fielding four terminations. The data areas have also increased in size partly because of the larger VM and segment sizes on 2900 but mostly because the absence of 16-bit integers has resulted in 32-bit entries being required. Nevertheless, about 200 4K page frames are available for user pages out of 1Mb store on both systems. This is partly because there

are no communications buffers in EMAS 2900 and also because the local segment table is now part of the Local Controller stack and pages with the process.

PERFORMANCE

The user view of EMAS 2900 performance is very satisfactory. With 32 terminals, the 2970 provides a much livelier service than the 4-75. It is necessary to have about 45 terminals active before the service degrades to a level that will not pass the interactive part of the Glasgow benchmark. However, since the 2970 is about twice as fast as the 4-75 it should be expected to support 64 terminals. It is instructive to compare the EMAS 2970 timing measurements with the figures for 4-75 published by Whitfield and Wight.[13] Exact comparison is difficult but it is clear that a higher proportion of time is used in the Supervisor on 2900 than on 4-75. It is also clear that device driving is responsible for much of this. In Whitfield and Wight, 1 million drum transfers have taken 973 s—i.e. about 970 μs of 4-75 CPU time each—whereas in Table I 314,000 drum transfers and corresponding interrupts have used 514 s, i.e. about 1630 μs of 2970 time or 3260 μs of 4-75 time per transfer.

Table I. 2970 performance measurements (4.1.1980)

EMAS 2900 SUP26A timing measurements						
Service	Calls	Time (s)	Average† (ms)	% of total	Av. inst.† /call	Notes
Idle time*	82 723	1090·121	13·178	9·2	1	
Nowork time*	64 528	4662·809	72·260	39·5	1	
Paging	1 535 920	856·038	0·557	7·3	215	
Disc	3 766	2·511	0·667	0·0	331	
Disc transfers	215 211	113·373	0·527	1·0	227	Disc driving
Disc interrupts	138 060	194·097	1·406	1·6	584	
Move/1	1 030	2·070	2·009	0·0	1065	File system
Move/2	6 507	3·772	0·580	0·0	262	maintenance
Drum transfers	314 520	289·527	0·921	2·5	369	Drum
Drum interrupts	172 526	224·787	1·303	1·9	499	driving
Slow peripherals	251 036	263·465	1·050	2·1	402	
Communications	301 077	255·517	0·849	2·2	347	
Local control	212 570	473·558	2·228	4·0	923	Working set maintenance
Foreground users	598 005	2929·841	4·899	24.8	0	
Background users	17 034	161·125	9·459	1·4	0	
Other	84 301	3003422	3·563	2·1	402	

Drum size = 12 Megabytes.
Recaptures = 37 per cent.
Shared pages = 16 per cent.
* 'Nowork time' is idle time when all users are voluntarily in the wait state.
'Idle time' is all other idle time.
† These figures refer to an optimized supervisor. When the supervisor is recompiled with checks the times and path lengths are increased by 30–40 per cent.

The other revelation from Table I is that more than 50 per cent of page exception interrupts can be satisfied without a transfer—16 per cent because the page is being used by another process and 37 per cent by recapturing the page from the free list.

These figures show the advantages of abandoning paging devices and using very large main memories.

The virtual buffering technique used in communicating with interactive terminals is very successful and terminals of all speeds have been attached. A buffer of one epage (4Kb) is sufficient for slow and medium speed terminals although the software will accept any size up to 64Kb. The virtual buffering is so trouble-free that the local printers and card readers have been attached as pseudo-remote devices. Using the maximum virtual buffer size it is possible for the spooling manager—a paged process—to read or print 64Kb without any intermediate transactions. This is a great improvement on the original system, which required the spool manager to page in every 4000 bytes read or printed. The other design aim of the communications software—that of protocol independence—seems to have been achieved judging by the experience of the University of Kent, who run EMAS 2900 on their ICL 2960. Kent operate a very different local network from Edinburgh but they are able to run the EMAS 2900 software unchanged. All necessary alterations are confined to the FEP and these were accomplished and tested in a few months. The CPU overheads of the communications software at up to 5 per cent of the CPU on 2970 are higher than on 4-75, where they were held to 1·5 per cent of the CPU. The extra paging cost of the 4-75 communications strategy is not readily apparent from the figures in Whitfield and Wight.[13] Subsequent studies have suggested that 20 per cent of the page turns on the 4-75 would be eliminated by the paging strategy used on the 2900 series.

Two desirable attributes have survived the re-implementation intact. The first is robustness—the mean time between software crashes on EMAS has always been high and it tended to infinity once development ceased. EMAS 2900, although still at an early stage, already survives hundreds of hours between software crashes. The second is the user interface as provided by the Standard Subsystem.[7] This feature together with compatible compilers makes transferring work from the old to the new system relatively painless.

We still regard EMAS 2900 as at an early stage in its life and expect the programme of tuning, measurement and analysis to yield a steady improvement in performance over the next two or three years. As part of these experiments it is hoped to try various page sizes in an experimental system.

BENCHMARKING

The presence in Edinburgh of the Remote Terminal Emulator (ERTE), used to establish the interactive part of the Glasgow benchmark, proved very useful. The 32 terminal scripts were subdivided into groups of 8 and duplicated, thus enabling 8, 16, 24, 32, 40, 48, 56 or 64 terminals to be emulated. (Strictly it is not possible to subdivide the benchmark as all scripts are different.) These emulated users could be used for testing the infant EMAS 2900 system to find bugs and measure performance. Having eliminated a number of bugs ERTE was then used to time the system and immediately produced a very surprising result. EMAS 2900 performed as well or better with a simple demand paging algorithm as it did with the SWS algorithm which had been so successful on System 4 EMAS. This conclusion seemed true across a range of configurations, including those with no drums. The designers found this result extremely surprising and further performance tests will be required before it can be fully explained. It is clear however that the high recapture rate has increased the cost of

preloading an unnecessary page: to the cost of the transfer must be added the overwriting of some page frame that might otherwise have been recaptured. Fortunately it proved easy to correct the faulty design decision and simplify the software by deleting the preloading of working sets. Tables II and III give figures

Table II. EMAS 2970 benchmarking (31.1.79 and 3.3.79)

	2970 1Mb core 4 EDS100s on 1DFC			
Experiment	A	B	C	D
Mbytes of drum	12	12	12	0
Batch streams	0	0	0	0
Interactive users	16	32	47	32
All commands				
Average reaction	2·2	7·2	13·9	15·8
Average response	10·5	15·4	22·9	23·6
Commands/min	40	64	71	53
Editing				
Average reaction	0·5	1·4	2·5	4·9
Average response	3·7	4·9	6·5	8·2
% reactions <2 s	84	65	47	23
CPU usage				
% in supervisor	22·5	42·0	43·2	36·2
% in user	37·7	56·5	51·7	46·4
% idle+no work	39·8	1·5	5·1	17·4
Paging				
% recaptured	43	27	23	16
% shared	19	20	20	34

measured by ERTE for benchmark runs. The technique used was to start the benchmark, wait ten minutes so that all terminals were active and then take measurements for a 30-minute window. This was much less time-consuming than running the entire benchmark. From such complete runs as were made it seemed necessary to achieve 50–55 commands/minute on the 32-terminal version and 100–110 commands/minute on the 64-terminal version if the benchmark was to be completed in the 2-hour period specified, the average amount of work done per simulated user being kept constant. When examing these tables one should realize:

(a) That the reaction time is measured from the last character of the command being typed until the first character of the reply.

(b) That the response time is measured from the last character of the command being typed to the first character of the next prompt reaching the terminal. Thus listing 1000 characters to a 10 character/s terminal (the normal speed in this benchmark) must have a response time of at least 100 s.

(c) That some commands in this benchmark compile and run very large programs of many thousand statements, and that a small number of such commands have a large effect on average response times.

Table III. EMAS 2980 benchmarking (6.4.79)

	2980 2·5Mb core 18Mb of drum on 2 SFCs 8 EDS100s on 2 DFCs	
Experiment	X	Y
Batch streams	8	8
Interactive users	48	63
All commands		
Average reaction	3·0	4·3
Average response	10·2	11·6
Commands/min	118	145
Editing		
Average reaction	0·7	0·9
Average response	4·2	4·5
% reactions <2 s	97	92
CPU usage		
% in supervisor	22·0	25·4
% interactive users	48·7	52·1
% batch work	29·3	22·5
% idle+no work	0·0	0·0
Paging (batch and interactive work)		
% recaptured	29	26
% shared	43	45

Experiment Y shows that EMAS 2900 can support 64 terminals on a 2980 along with a lot of simultaneous batch work while maintaining a very good response—more than 90 per cent of editing reactions were less than 2 s. This is sufficient to pass the 64-terminal benchmark by a large margin.

Experiment B shows that a 2970 can pass the 32-terminal version of the benchmark (as specified at Southampton University) with good response when using 12Mb of drum. Experiment D shows that it is just possible to pass the Southampton benchmark with no drums at all but that response is sluggish with only 23 per cent of editing reactions in 2 s.

CONCLUSION

The relatively uncommon practice of re-implementing a system has produced a major operating system using a fraction of the effort that was required to produce the original software or other comparable systems. The self-restraint of the designers in restricting changes to the minimum has resulted in the new system retaining the user interface, reliability and high performance of the original. With the use of relatively machine-independent high-level languages, re-implementation may become more widely considered as a technique for moving to new hardware with minimum disruption.

AVAILABILITY

EMAS 2900 is currently running on 2960, 2970 and 2980 machines. It has run on a 2976 and will soon be mounted on a dual processor 2972. It will not run on a 2950 or 2956. The University will make the software available in source code form on request. There will be no charge for this in the case of educational or research institutions. The University is also able to supply the special hardware developed for the communications interface.

ACKNOWLEDGEMENTS

We would like to thank Professor H. Whitfield and the other designers of EMAS for producing a system that it has been a pleasure to re-implement. The following, among others, contributed ideas, suggestions and that most precious commodity, reliable, tested code: A. Anderson, W. Laing, C. D. McArthur, R. R. McLeod, J. Maddock, P. Robertson, A. Shaw, F. Stacey and A. Waller. B. A. C. Gilmore designed and implemented the FEP software. We also thank the Post Office Data Processing Service for seconding J. Maddock and A. Waller to work with us.

REFERENCES

1. J. C. Adams, W. S. Currie and B. A. C. Gilmore, 'The structure and uses of the Edinburgh Remote Terminal Emulator', *Software—Practice and Experience*, **8**, 451–459 (1978).
2. J. C. Adams and G. E. Millard, 'Performance measurement on the Edinburgh Multi-Access System', *Proceedings of the International Computing Symposium*, Antibes, 105–112 (1975).
3. J. C. Adams, 'Performance measurement and evaluation of time shared virtual memory systems', *Ph.D. Thesis*, Edinburgh University, 1977.
4. J. K. Buckle, *The ICL 2900 Series*, Macmillan, 1978.
5. P. J. Denning, *Working Sets Past and Present*, Purdue University Report CSD-TR-276, 1978.
6. P. B. Hansen, 'The programming language Concurrent Pascal', *IEEE Transactions on Software Engineering*, **1**, (2), 199–207 (1975).
7. G. E. Millard, D. J. Rees and H. Whitfield, 'The Standard EMAS Subsystem', *Computer J.*, **18**, 213–219 (1975).
8. D. J. Rees, 'The EMAS Director', *Computer J.*, **18**, 122–130 (1975).
9. A. L. Schoute, 'Modelling of virtual memory computer systems with multiple classes of process', *Ph. D. Thesis*, University of Groningen, 1978.
10. N. H. Shelness, P. D. Stephens and H. Whitfield, 'The Edinburgh Multi-Access System, scheduling and allocation procedures in the Resident Supervisor', *RAIRO (Informatic, Computer Science) B*, **3**, 29–45 (1975).
11. J. G. Sime, *Benchmark for the ICL 1906S Computer System*, University of Glasgow, 1973.
12. P. D. Stephens, 'The IMP language and compiler', *Computer J.*, **17**, 216–223 (1974).
13. H. Whitfield and A. S. Wight, 'EMAS—the Edinburgh Multi-Access System', *Computer J.*, **18**, 331–346 (1973).
14. A. S. Wight, 'The EMAS Archiving Program', *Computer J.*, **18**, 131–134 (1975).