"FUNCTIONAL DISTRIBUTION OF THE WORKLOAD OF A LINDED COMPUTER SYSTEM AND ITS SIMULATION"

Constantine Lazos[1]
Department of Mathematics
University of Southampton
England

Abstract:
Consideration is given to a possible functional distribution of the workload over two linked computers with separate channel access to a large disc store, into the resource utilisation of the linked system achieved by simulation using a modified and re-entrant single processor simulator. Results suggest that the proposed distribution realises a high utilisation.

1. INTRODUCTION:

Hardware enhancement is one of the ways of relieving an overloaded computer configuration. In particular to increase the capacity (of the same computer) by adding another processor. Although this is an attractive way of solving the problem, it is physically impossible with many machines. In such cased the use of an additional but separate computer can solve the problem.

This was the case at the Computer Center of the University of southampton and stimulated a preliminary study of how two machines should be operated. This paper presents the design of the operation of two machines as well as the results obtained through simulation.

If there were two machines in the same Center, one could operate then as two independent computers; but, on the other hand, it might be wasteful to use the same computer to service ass activities.

(1) For example, character oriented work would waste computer power ifit were to be carried out by the processor with very powerful floating point facilities. Hence, a decision was taken to consider the two computers as a single unit (some sort of connection between them being provided) and therefore to design an operating system that would take into account the different properties of the two machines and would distribute the wouk load of the system in the best possible manner, while at the same time maintaining a good service for the user.

(2) Description of the proposed linked system. The present consideration concerns a computer configuration with the following restrictions and limitations:

(i) Two processors (1 processor = core and CPU). They would have different properties, e.g., different speeds, different size of core, one is powerful in arithmetic operation, the other is not, etc.... but both belong to the same family - thus permitting compatibility of such features as low level languages and comunication protocols.

(ii) They would be connected via a core-toc ore link.

(iii) They would have simultaneous access to a large number of independent disc units.

(iv) All I/O devices, Card Readers (CRs), Tape Readers (TRs), Line Printers (LPs), Magnetic Tapes (MTs), would be connected to one of the processors but they could be switched to the other in an emergency.

(v) One or more remote stations would be connected to the slow processor in the system, each consisting of a LP and a CR/TR and/or CP/TP of rather slow speed.

(vi) Teletypes would be connected through a communications processor which in turn would be connected to the more powerful machine.

[1] Present address: Department of Mathematics, University of Thessaloniki, Greece.

Looking at the hardware configuration it is obvious that we have two CPU units (one CPU = arithmetic and logical unit and control section) each connected to separate core memory. Such a system cannot claim to be a multi-processing system in its "pure" sense, (i.e., applying the same object program to the two CPU's simultaneously). Nevertheless, one can take advantage of the way the two processors are connected and apply the computers to different phases of the same problem.[4]

How then can we apply both computers to the same problem, since we do not look at it as a "true" multi-processing system?

We should take advantage of the fact that the two processors have different properties. Thus it would be wasteful to ask the powerful one to carry out the character oriented work which is associated, for example, with handling I/O operations, particularly with slow peripherals. Programs, on the other hand, go through several distinct and non-overlapping phases (steps) until their execution is complete. Thus it is possible for these phases to be carried out on different processors.

From now on we refer to the fast processor as the MAIN processor (MP), to the slow processor as the ASSISTANT processor (AS) and to the whole system as AS/MP.

2.1. Work load of the system.
Such a system is going to handle a large number of jobs (perhaps 1000 jobs or mare per day). Jobs will arrive at the main center or at the remote stations at random and with a wide spectrum of resource requirements; they will differ greatly in size, time, languages, etc.....
Users tend to judge the performance of a system and the quality of the service the installation offers by the characteristic known as turn-round time.[6] On the other hand every computer system consists of fast but expensive hardware which must be utilixed to the full.

Thus we have the "user" on the one hand who tends to be 'selfish' demanding faster service, and the "system" on the other whose resources (components) should be maximized.

It is these two basic concepts that were taken into account in distributing the work load and allocating the functions that every machine has to carry out. Each computer should be given tasks best suited to its capabilities, while still providing a good service for the user.

2.2. Allocation of functions in the system.

The allocation of functions among the two processors is as follows:

<u>The ASSISTANT Processor (AS) will:</u>

(i) Carry out the character oriented work associated with servicing all input/output devices.

(ii) Organise the input/output stream via disc store.

(iii) Compile the jobs and store the resultant code on disc store.

(iv) Handle all pre-jobs set-up requirements (magnetic tapes, discs, etc....).

(v) Schedule the jobs by keeping various tables in its core memory and back-up copies on disc.

(vi) Feed the main processor with 'cooked food' (i.e., jobs already compiled).

(vii) Copy MT files onto disc and vice-versa.

(viii) Execute small jobs (if there is enough time for it) - lowest priority.

<u>The MAIN Processor (MP) will:</u>

(i) Load and execute jobs.

(ii) Carry out all I/O requirements during the running of the object program (off-line I/O from/to disc).

Thus from the "system" point of view this is likely to:

(i) Maximize the utilisation of MP for arithmetic tasks and of AS for housekeeping tasks.

(ii) Increase utilisation of the AS CPU because of a more deterministic workload, i.e., primarily compilation.

(iii) Minimize the waiting time for setting- up magnetic tapes and eliminate the restriction on the decks needed for effective multi-programming (of magnetic tape jobs) by off-lining techniques.

(iv) Reduce operator intervention.

From the 'user' point of view it would offer:

(i) Aprobable improvement in throughput and thus turnaround time.

(ii) Jobs with fatal compilation errors will get a fast turn-around no matter if they are small or large, high or low priority. This can be of great value particularly in a University environment where the majority of the users are not professional programmers and they have to write, punch and correct their own programs.

(3) Simulation of the AS/MP system.

In the preceding sections we outlined the organisation of the work-load of the AS/MP system, the allocation of functions among the two machines and how each computer would participate in the processing of each job. Though this approach seems to be quite attractive, it also creates certain problems that need to be investigated. Firstly, can the AS processor cope with the compilation task, which is to secure a steady flow of semi-compiled jobs on to the ready-list for the MP processor? Depending on the environment (University, Company, Research Group,...) jobs are subject to compilation errors thus never reaching the ready-list for the MP. This factor can be very critical, especially in a university environment where many jobs are subject to compilation errors. Another problem is the traffic that would be created in the two disc channels connection the disc with the two processors. CPU utilisation is also of great concern together with the utilisation of the peripheral devices.

Simulation presents a possible way of attempting to answer these problems but would require considerable effort to effect the simulation of AS/MP system, i.e., to build a specifec simulation model for two CPU's, two core memories, etc. This would constitute an iconic reflection of the system, and it had to be avoided.(4) The goal of model building is not to construct a faithful reproduction of the system but rather to obtail the desired information and results with accuracy, with a minimum of effort. Models constitution iconic reflections of a system take a very long time to build and can be very costly.

Nielsen (7) has reported that in one case a computer company came to the conclusion that simulation would cost more than the actual system. In another case the actual system was in the midst of its implementation, while the simulation had not been completed. Furthermore, this type of simulator ezecutes very slowly. It was, therefore, necessary to find a vay of simulating the AS/MP system without writing a tailoe-made simulator for it. One possible solution was to consider making use of a single-computer simulator which had been written and tuned to the ICL 1907 - COOP 3 operation system at Southampton and in which the comparison of simulated with actual results proved highly stisfactory(5).

3.1. In-process/out-process:

Dijkstra et al(3) in building their multiprogrammed operation system for a Dutch machine, the EL,XS, introduced a hierarchical structure in the system with five levels of control (L0 to L4). These levels are presented graphically in Fig 1. This structure is sometimes referred to as an in-process architecture. One may observe now that this structure may be found in almost every comper-hensive operating system though the levels may not be defined wxplicitly. This is true with the COOP3 operating system, where level 3 for example corresponds to what is known in that system as member-one, level 4 is the same, etc... It is this structure that was followed in the construction

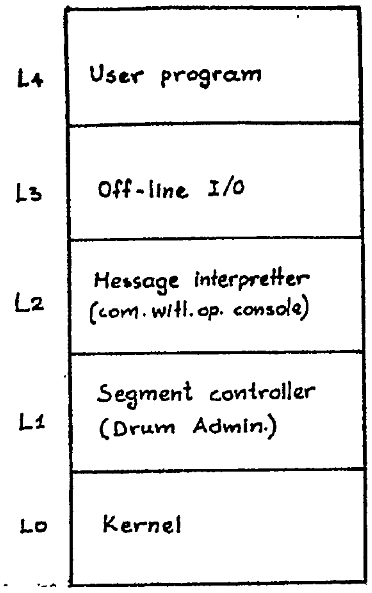

Fig. 1. System Hierarchy

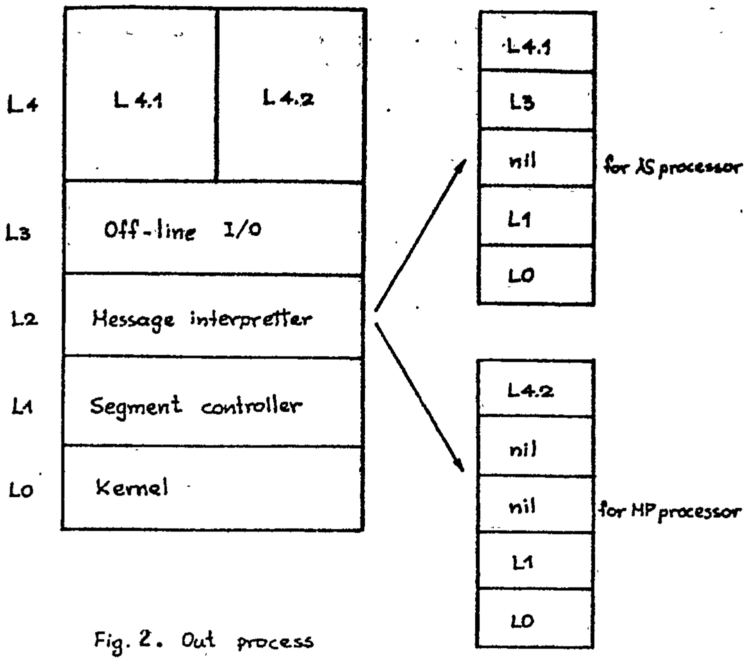

Fig. 2. Out process

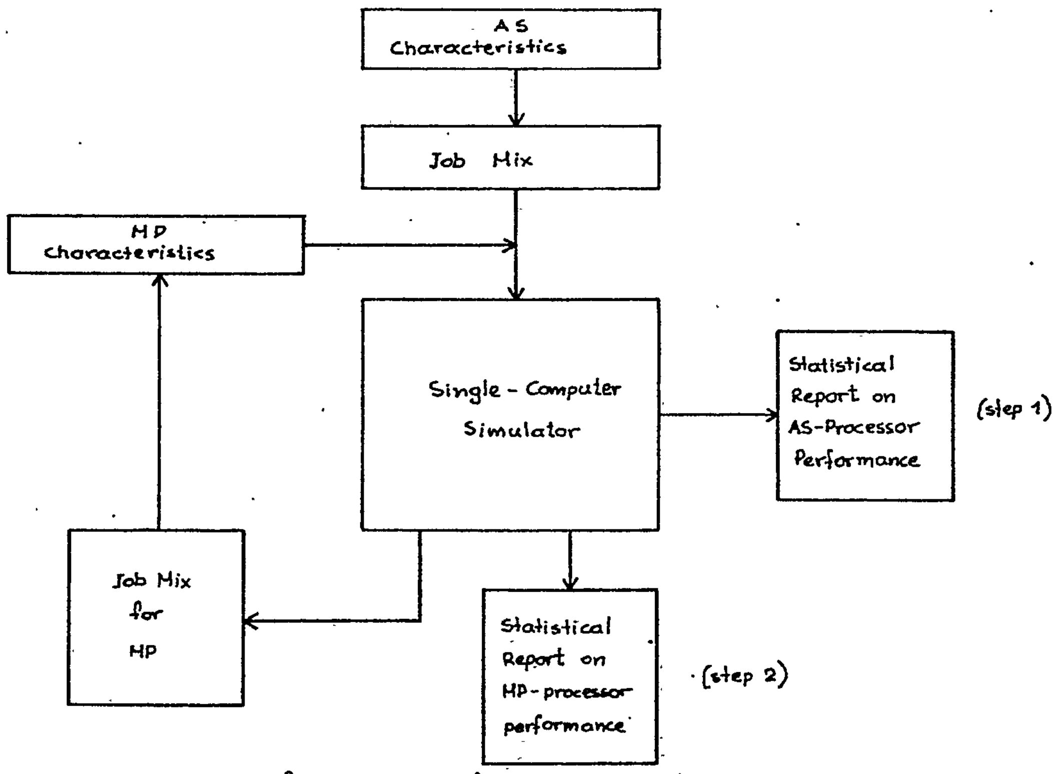
AS Characteristics
Job Mix
HP Characteristics
Single - Computer Simulator
Statistical Report on AS-Processor Performance
(step 1)
Job Mix for HP
Statistical Report on HP-processor performance
(step 2)

Fig. 3. Simulation of the AS/HP system

		No. of Jobs = 153 Time = 3982 secs	No. of Jobs = 225 Time = 3338 secs
		%	%
CPU:	Busy	87.9	87.4
	Idle	12.1	12.6
Disc-Channel:	Busy	80.1	82.6
	Idle	19.9	17.4
Card Reader-1:	Working	34.8	34.8
	Waiting for disc	12.8	13.3
	Idle	52.4	51.9
Card Reader-2:	Working	37.7	33.8
	Waiting for disc	13.8	12.8
	Idle	48.5	53.4
Line Printer-1:	Working	58.3	66.1
	Waiting for disc	25.5	29.8
	Idle	16.2	4.1
Line Printer-2:	Working	68.5	66.7
	Waiting for disc	29.3	29.5
	Idle	2.2	3.8
Card Punch	Working	9.5	15.2
	Waiting for disc	0.8	1.5
	Idle	89.7	83.3
Core memory utilisation		0.851	0.720

	No. of Jobs = 153				No. of Jobs = 225			
	CRs	Core Memory	LPs	Disc	CRs	Core Memory	LPs	Disc
Current queue length	0	0	43	0	0	0	98	0
No. of entries	150	111	153	74.9%	194	139	225	77.7%
Maximum queue length	103	62	48	7	102	50	116	8
Maximum waiting time	1280	3422	1917	566 }msec	687	2328	2132	481 }msec
Mean waiting time	583	1127	345	65.4	359	466	479	71.2
Mean queue length	22.0	31.4	23.0	1.0	20.9	19.4	74.4	1.2

	No. of Jobs = 137 Time = 3982 sees						No. of Jobs = 183 Time = 3338 sees					
I/O buffers	CPU		Disc-channel		CMU	Jobs unpro-cessed %	CPU		Disc-channel		CMU	Jobs unpro-cessed %
	Busy	Idle	Busy	Idle			Busy	Idle	Busy	Idle		
256 words	69.2	30.8	79.7	20.3	0.893	21.1	72.8	27.2	88.9	11.1	0.890	27.2
512 words	85.4	14.6	50.2	49.8	0.859	11.7	90.6	9.4	57.3	42.7	0.850	13.1
1024 words	88.3	11.7	29.9	70.1	0.855	11.0	96.8	3.2	33.6	66.4	0.885	8.8

Queues for:	CRs	Core Memory	LPs	Disc	CRs	Core Memory	LPs	Disc
1) I/O buffers= 250 words								
Current queue length		29		2		50		2
No. of entries		137		71.5%		183		79.0%
Maximum queue length	nil	30	nil	4	nil	78	nil	4
Maximum waiting time (secs)		1359		300		2273		299
Mean waiting time (secs)		259		61.9		494		64.1
Mean queue length		17.4		0.9		49.5		1.2
2) I/O buffers = 512 words								
Current queue length		16		0		24		0
No. of entries		137		42.1%		183		46.1
Maximum queue length	nil	16	nil	4	nil	60	nil	4
Maximum waiting time (secs)		920		242		1796		285
Mean waiting time (secs)		187		35.7		422		55.9
Mean queue length		8.3		0.2		30.4		0.4
3) I/O buffers = 1024 words								
Current queue length		15		0		16		0
No. of entries		137		22.7%		183		31.7
Maximum Queue length	nil	16	nil	3	nil	46	nil	4
Maximum waiting time (secs)		552		178		1728		234
Mean waiting time (secs)		102		32.8		308		41.7
Mean queue length		5.4		0.1		24.5		0.1

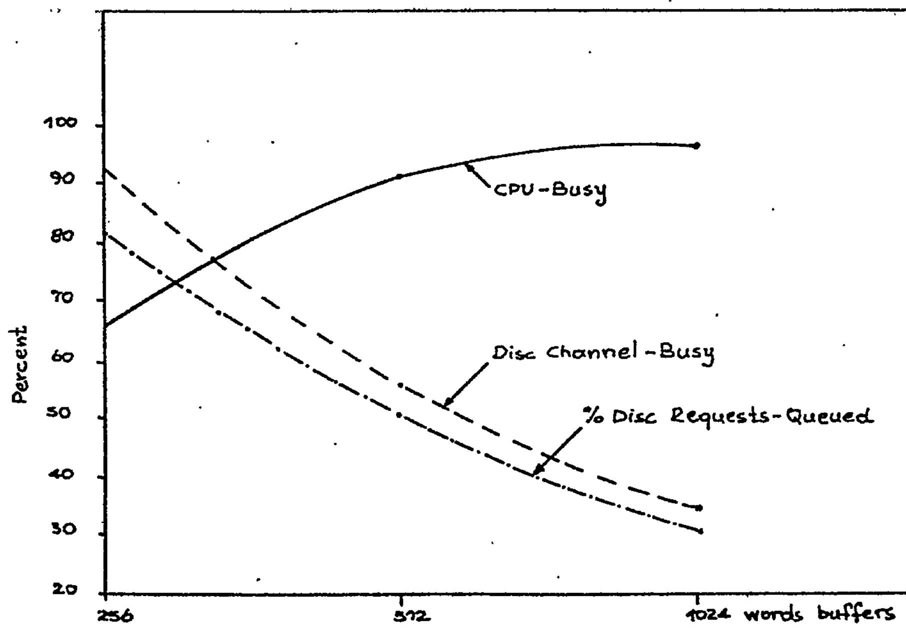

Fig. 5. MP-Processor performance (Run II)

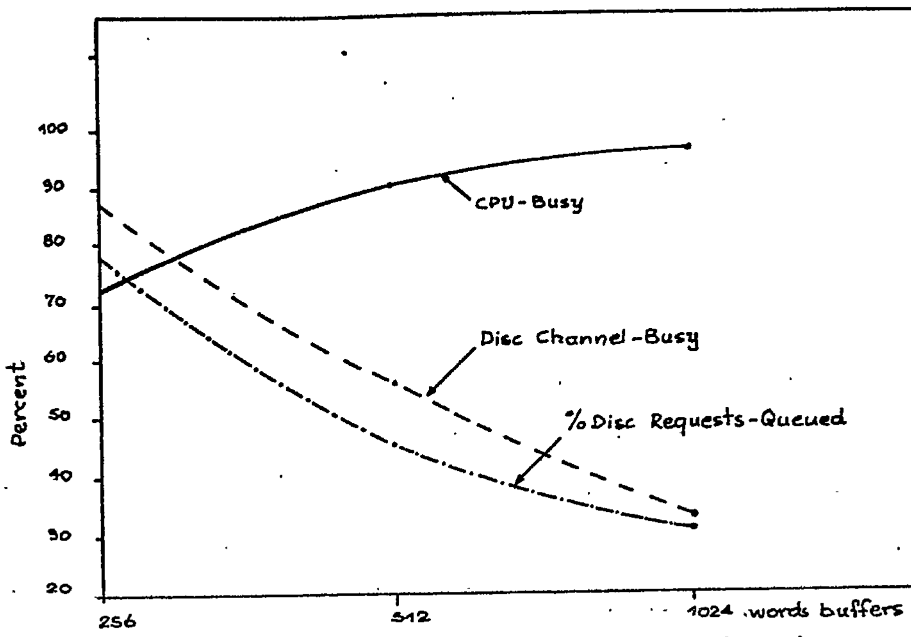

Fig. 4. MP-Processor performance (Run I)