Inter-Office Memorandum

To	Bill Lynch, Wendell Shultz	Date	August 2, 1977
From	H. C. Lauer	Location	Palo Alto
Subject	Multiple MDS's in Pilot	Organization	SDD/SD

XEROX

Filed on: <Lauer>MDS.memo, .ears

Following our discussions, I have looked into the problem of designing Pilot to use and/or support multiple Main Data Spaces (MDS's). There seem to be two different problems, which need to be treated separately:

> First, can we design Pilot so that some or all of its components occupy different MDS's from any application processes.
>
> Second, can we arrange things so that different major applications occupy different MDS's in such a way that one can be completely swapped out while another is *active* (i.e., supporting one or more processes which are either doing something are waiting to do something at short notice of, say, milliseconds rather than seconds or minutes).

The first problem imposes very different requirements from the second. In particular, if Pilot or some part of Pilot occupies a different MDS from the application, the problem of cross-MDS calls (or transfers of control) must be solved to permit high-bandwidth communication between MDS's. I.e., it must be possible for an application to call Pilot functions or otherwise interrogate Pilot data structures in another MDS without undue delay and loss of efficiency. The second case does not require this because communication between two applications is inherently more limited and can be resolved on an *ad hoc* basis.

The two problems have different kinds of solutions. In particular, a possible solution for the second problem is to copy Pilot in each MDS (its major data structures would probably be external to any MDS) and swap MDS's as whole units. This way, there would only by one MDS in real memory at any time (except, of course, when the swapping was actually taking place). This would pose no cost in resident memory while the MDS is actually active and would permit the application to call Pilot functions via the normal procedure mechanism of Mesa with no loss of efficiency. Swapping would be slightly cumbersome and would require fetching or creating a temporary place to stand while the swapping takes places, but this would be sufficiently infrequent that the display could be blanked in order to provide the memory required for this purpose.

The first problem is more interesting and more difficult. Several approaches are:-

> Include a copy of the interface to Pilot in each application MDS. This would use the interprocess communication and synchronization facilities to activate Pilot processes in another MDS.
>
> Invent a fast cross-MDS procedure call and return mechanism which allows an application process to switch MDS's on the fly. The problem of passing pointer parameters in this case can be finessed easily by making Pilot do explicit short- to long-pointer conversion, something operating systems have done for years.

Design a cross-MDS FORK instruction by which an application can spawn a process in the Pilot MDS in lieu of a procedure call.

Invent a fast message passing system capable of transmitting messages across MDS boundaries (the dual of the cross-MDS FORK).

Cost of multiple active MDS's

Any of the approaches to the first problem require that there be more than one active MDS is real memory at any one time. In particular, any MDS which is supporting processes in the ready queue has several pages of fixed information which either must be locked down in non-swappable memory or which are accessed so frequently that they will by default be not swapped out even if they could be. The cost, in terms of real memory, of supporting two MDS's at the same time is thus measured as the difference between the number of pages in the working sets of both the Pilot and the application MDS and the number of pages which would be in the working set if everything were rolled into one MDS. I have come up with the following numbers for this cost:

The fixed components of an MDS, namely the *system dispatch table, the allocation vector,* and *the global frame table* require two pages of real memory. It is remotely conceivable that this could be reduced to one, but I doubt it. In a single MDS system, all of the functions would share the same resident tables.

Each MDS must have its own *frame heap,* a portion of which is very warm if not 'always' resident. This is likely to cost at least one page, perhaps several, over the single MDS system, in which both Pilot and applications share the same frame heap.

Unless we have a cross-MDS procedure call, there will have to be global frames for Pilot in both the application and the Pilot MDS's. These are necessarily in different pages, and during any call upon Pilot will be resident. This probably costs at least one extra page of resident memory, perhaps more.

In addition to these, there may be other intangible real memory cost which we cannot identify at this time.

The total of all of this comes to at least *four* pages of real memory, perhaps more, to support an extra, active MDS. Although this does not sound like much, it represents 16% of our resident memory budget for Pilot, which is already too small.

Recommendation

I recommend the following position:

That Pilot not support multiple MDS's before release 4.0.

That individual workstation of the Janus product line not support multiple active MDS's at any time; multiple swapped MDS's with only one MDS active at a time be supported in release 4.0.

The larger system elements of the Janus line, including servers, gateways, and multiuser units, be configured with enough real memory to support multiple active MDS's.

We will specify the interface to Pilot to be independent, as far as possible, of the multiple-MDS problem so that in the future we can embed portions of it in separate MDS's if we choose.

Other issues to keep in mind

Multiple MDS's impose an interesting problem with respect to object style programming. If a record object, for example, is passed between MDS's, there had better be very stylized rules or practices about embedding procedure variables in it. In particular, the descriptor of a procedure declared in one MDS is of no value to a process in another MDS. As a matter of policy and style in the Pilot functional specifications, this should pose no problem (because Pilot will not be calling application or user provided procedures). But it is an area in which caution must be excercised.

c: Redell
McJones
Gifford
Crowther
Jarvis