OS/390 from an NT Developer's Perspective

James Antognini

2 December 1999

Overview

- Introduction.
- S/390: The hardware.
- Address space.
- Addressability.
- Authorization.
- PC routines.
- The repositories.
- Dispatching and scheduling.
- Units of work.
- The I/O model.
- Recovery.
- Serialization, coordination.
- Development of systems-level programs.
- Miscellany.

Introduction

- Many, many similarities between OS/390 and NT –
  - Whilst the hardware isn't quite proprietary, the software is.
  - A "mainframe" can be quite small, even portable.
  - The dispatching mechanism is one interrupt-driven, preemptive multithreading.
- Orientation of presentation is to the NT kernel-mode developer who has the concepts and wants to know "How do I do it?"

S/390: The Hardware

- PSW (Program Status Word, like EIP+flags) contains some state information, including “kernel-mode” indicators. There are defined places (a one-level stack) for “old” and “new” PSWs to contain state.
- There are 16 32-bit general-purpose and access registers, as well as control and floating-point registers.
- There are page tables, including higher tables. For each physical page there are bits affecting access.
- There are storage keys, something of an historical artifact but a feature useful for protecting a subsystem’s information from its clients (e.g., CICS).
- There are PC-entry tables to call routines via hardware. SVC tables are primarily software (like dispatch routines).
- And there is addressability.

Address Space

- “Address space” is roughly equivalent to “process,” being a unit of management (dispatchability, identity and accounting) as well the framework of virtual addressing.
- There are 2G in an address space (history again).
- At any moment, a program may address 2G or merely the low 16M of an address space.
- System code and data areas straddle the 16M line (except for page 0).
- An address space may be a unit of execution or merely of data (the “data” space). The PSW points to the former; GPRs and ARs point to both types, up to a combination of 15 different ones at one moment.
- The management:
  - The address space has its own dispatchability, under which “threads” run.
  - Swapping: Movement out or in affects use of real storage and, of course, eligibility for dispatching.
  - Identity = authority.
  - Accounting.
  - WLM can manage the address space individually or as part of an “enclave.”

Addressability

- A routine always has a primary and a secondary address space (for historical reasons) as well as a home address space.
- Home is the dispatched address space.
- The primary is whence come instructions. Most often the primary is the home address space.
- The secondary can be used for data.
- When an authorized component is invoked via a PC instruction, primary addressability may switch to the component (server) address space, and the caller's (client) becomes secondary. Dispatchability doesn't change.

Authorization

- The fundamental division is supervisor versus problem state (= kernel-mode versus user-mode).
- Running in “system” key allows access via software to supervisor state. Key change is available via hardware or software.
- APF (Authorized Program Facility) is like a superuser attribute, which gives access to other states via software.
- Identity is in itself unimportant for authorization.
- But for the setting of APF attributes (or restricted USS (UNIX System Services) attributes), identity affects permission.
- An authorized application involves having code in an APF library (≈ system DLL) and being invoked authorized. Usually the bulk of code is in PC routines.
- Note that kernel-mode components are not based on an I/O model. And there is no equivalent of a filter routine!

PC Routines

- These are established by an authorized component, typically running in its own address space. They are the usual way of “exporting” authorized services.
- The establisher loads the routines in the right place and defines who can call them and with what attributes they receive control.
- A routine can have the establisher’s address space as primary or the caller’s.
- The establisher must make the PC numbers (= indices) known to (and valid for) expected callers.
- A call is effected solely by hardware, via table lookup given the PC number.

The Repositories

- There is no analog of the NT registry. There is rather a multiplicity of repositories of information.
- SYS1.PARMLIB is employed for some system information, especially at IPL (“boot”) time.
- “Policy” and other information is kept in datasets (= files).
- Catalogs (master and user) keep dataset descriptors and, in some cases, state. They – the master catalog particularly, due to its crucial role – are a bit like registries.

Dispatching and Scheduling

- OS/390 is a preemptive, interrupt-driven, multithreaded system.
- WLM (Workload Manager) makes scheduling decisions based on policies, to affect dispatching priority, swapin status and working set.
- Given WLM decisions, the dispatcher selects address spaces and then their SRBs and tasks to be given control.
- Time slicing is not (at present) used.
- Local, CPU and higher locks are somewhat like IRQL in that they inhibit or prevent loss of the CPU.

Units of Work

- Tasks (TCBs) are equivalent to threads.
- SRBs are lightweight threads (but putting on weight over time) and always authorized.
- Disabled routines – I/O completion routines (SLIHs), DIEs (principally timers) – are similar to ISRs and DPCs in not being preemptible, restricted in available services and running independently of an address space.
- There are numerous exits (= callbacks) for things like task- and address-space termination, JESx events, I/O events and so forth.

The I/O Model

- The model comprises a front end, independent I/O hardware (channels) and back end.
- I/O is initiated by an application or by an IOS driver (installable and thus roughly analogous to a device driver).
- I/O is passed from IOS driver to IOS (≈ HAL), which starts physical I/O (SSCH).
- I/O completion causes an interrupt in a random address space.
- An IOS backend fields the interrupt and calls the appropriate IOS driver back end in disabled state.
- The IOS driver back end will employ an SRB if the originating address space is needed.
- There is no equivalent of a filter routine!

Recovery

- The MVS philosophy is “keep on truckin’”: Each authorized routine is responsible for trying to continue operation, to limit damage to itself and others and to log errors.
- The aim is, at minimum, to degrade gracefully. Stopping the system is strongly discouraged.
- Thus, exceptions like disabled page faults and, in general, errors in restricted system state can be handled.
- Recovery is done through FRRs and ESTAE routines, the former being more versatile.
- Terminology: “Recovery” = _except filter expression/ routine. “Retry” = _except block/routine.

Serialization, Coordination

- Local, CPU and various spin locks are available.
- Wait/post is equivalent to wait/notification event and signalling.
- ENQ/DEQ are similar to mutexes or semaphores.
- Memory can be shared between address spaces.
- Cross-address-space applications – whether authorized or not – are not common and necessitate a shift in the thinking typical in developers. This, even though mechanisms are plentiful.

Development of Systems-level Programs

- PL/X is like C but not generally available outside IBM. Assembler is often used outside IBM. Neither has a run-time environment.
- C, C++ and PL/I are other systems-oriented languages.
- Simple (= rudimentary) debugging tools, at best akin to SoftICE: VICOM (for IBM only), and VM and TEST in TSO. Dumps are very commonly employed. (But the machine environment cannot be trashed.)
- Two or three main IBM systems-programming centers: Poughkeepsie, Santa Teresa and (?) Raleigh.
- Few “system” ISVs (for example, units of Computer Associates).
- There are no books in print to introduce one to the endeavor. It’s more artisanal (learning by example and experience), if you will.

Miscellany

- In general, no stack, only heap. (Peter Relson's ASA services are an exception.)
- JCL = .bat file, but uglier and with more warts.

```
//MAKEJABL JOB ANTOGNI,'Compile and assemble',
//    MSGLEVEL=(1,1),MSGCLASS=H,
//    TIME=1440,REGION=32M,NOTIFY=ANTOGNI,USER=ANTOGNI
/*JOBPARM L=100
//PLX      EXEC PGM=AKEEPLX,
//         PARM='LC(100),DOBARS,MACPARM("YKTVSE")'
//SYSIN    DD DSN=ANTOGNI.PGM.PLX(MAKEJABL),DISP=SHR
//SYSPRINT DD SYSOUT=*,
//            DCB=(RECFM=FBM,LRECL=133,BLKSIZE=13300,BUFNO=15)
//SYSUT1   DD SPACE=(CYL,(10,10)),UNIT=SYSALLDA,
//            DCB=(LRECL=80,BLKSIZE=32000,RECFM=FB)
//ASM      EXEC PGM=ASMA90,COND=(4,LT),
//         PARM='LINECOUNT(100),NOOBJECT,XREF(SHORT),NODECK,RENT'
//SYSPRINT DD SYSOUT=*,
//            DCB=(RECFM=FBM,LRECL=133,BLKSIZE=13300,BUFNO=15)
```

- No GUI: Only a TSO or Unix-like command window.
- File systems: MVS-proprietary (primarily sequential, directory-like and random access) and POSIX-branded (USS).
- To get the comparison of NT and OS/390, ftp as ANONYMOUS to w3.s390.ibm.com, then 'cd os390/xmemsrvc, binary, get OS390andWinNT.zip' (observing case).