The Hitchhiker's Guide to the Mainframe

By Ford Prefect – November 2002

The Basics

Introduction

In this article I will provide a brief history of the mainframe, with an emphasis on features that make a mainframe unique. I will not attempt to cover the architecture in any technical detail at this time; the intent is simply to set the stage for future articles. If you are in fact an experienced mainframe user then this article probably has no particular value for you, other than possible amusement. I will assume that the reader is familiar with general computer concepts and technologies, and will present the information in terms that should be familiar to PC or Unix geeks.

Why "Hitchhikers Guide"? I am a big fan of the works of the late great Douglas Noel Adams, especially his "The Hitchhiker's Guide to the Galaxy", a trilogy in 5 (6?) books. For those of you who have not read these books - DON'T PANIC - any references you don't understand can be safely assumed to be Somebody Else's Problem - they are not important - the answer is 42, and that's all you need to know. Of course if you really want a better understanding of the galaxy and how it will affect your next party, then maybe you should read the books. ;-) I was given the nickname "Ford Prefect" (one of the story's lead characters) many years ago by a group of friends who were also HHGG fans. So sit back, relax, and enjoy an Ole' Janx Spirit while I introduce you to the world of mainframe computer systems.

First, let's define a few terms:

What is a Mainframe?

The term mainframe was originally coined to refer to the very large computer systems that occupied very large steel framed boxes, and was used to differentiate them from the smaller mini- or micro-computers. While the term has been used in various ways over the years, it is most often used to describe the successive families of IBM computer systems starting with System/360. This would also apply to compatible systems built by other companies such as Amdahl and Hitachi Data Systems (HDS). This is the definition we will be using for the remainder of this series of articles.

Although some have used the term "mainframe" to refer to IBM's AS/400 or iSeries systems, this is an improper use of the term; even IBM considers those systems as mid-range servers rather than mainframes.

Another more humorous definition of a mainframe can be found in Isham Research's Devil's IT Dictionary: "an obsolete device still used by thousands of obsolete companies serving billions of obsolete customers and making huge obsolete profits for their obsolete shareholders. And this year's run twice as fast as last year's."

What is an I/O Channel?

A mainframe channel is somewhat analogous to a PCI bus, it is capable of connecting one or more "controllers", each of which controls one or more "devices" (disk drive, terminal, LAN port, etc.). One primary difference is that mainframe channels connect to controllers via either pairs of large "bus and tag" cables (for parallel channels) or, more recently, fiber optic ESCON (Enterprise System CONnection) cables (serial channels) and FICON (fiber channel). In the early days these channels were external boxes (about 6'x30"x5'H each), but are now integrated into the system frame.

These channels are a key part to one of the great strengths of the mainframe – its massive I/O capabilities. I will touch on the capabilities of the I/O subsystem later in this article.

What is DASD?

DASD is an acronym for "Direct Access Storage Device"; IBM coined this term to refer to any type of storage that was directly (randomly) addressable. Today the term is just what we call disk drives, but in the olde days it also referred to drums and datacells and such. What is a datacell? Well, before disk drives became cheap, fast, and popular IBM had a device that was basically a drum around which it would wrap one of many different magnetic strips (cells), then read/write information on data tracks on the spinning cell. The method of accessing the data was identical to that for a disk, but seeks involving cell changes were obviously measured in seconds. <G> Datacell devices also had a nasty habit of jamming when trying to dismount a cell into its storage slot, sometimes causing physical damage to the media – we've come a long way baby !

What is an LPAR?

An LPAR (Logical Partition) is a type of virtual machine implemented by PR/SM (Processor Resource/System Manager) a hardware/firmware feature on all recent mainframes. In each partition, a separate system image can be run, providing complete software isolation. This is similar in some ways to "domains" on UNIX systems, but IBM's approach is more granular, allowing both CPUs and I/O sub-systems to be shared between logical partitions. PR/SM allows you to run up to 15 LPARs on a single system, each having dedicated real storage (RAM) and either dedicated or shared CPU(s) and channel(s). Since the functions most important to performance are implemented in the CPU, there is very little performance cost. IBM has announced its intention to expand this support to more than 15 LPARs in the near future.

How Did All This Start?

Well, far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun ... Oops ... Sorry, wrong story, let's try again:

Back in the early 60's, IBM came out with a new line of computers to replace their various scientific and commercial models. They had a new idea: to create a line of computer systems that shared a common architecture, with a full range of models, all capable of running the same software. They called this new product line System/360. Although the physical implementation details of these systems varied significantly, the ISA was consistent throughout, except for some specific features (packed decimal and/or floating point instructions were optional on certain models, depending on whether you needed to do scientific or commercial work). Of course those first systems had about the same processing power as today's digital watches, and we all know how neat digital watches are. <G>

Since that time, IBM has released many additional models of mainframe systems in quite a few succeeding product lines, all based on the same basic system architecture, with vast improvements in performance and scalability. The end effect is that a load module (binary executable) written way back in the 60's can still execute today on a current mainframe with exactly the same outcome, except for the amount of execution time required.

Let's quickly summarize the most significant architectural changes made over the last 40 years, I'll cover these features in more detail in the future:

- System/360 had 32 bit general registers and 24-bit memory addressing, with a maximum of seven (parallel) I/O channels.
- System/370 added more sophisticated I/O channel functionality and virtual storage addressing (on certain models).
- System/370-XA (eXtended Architecture) extended addressability to 31-bit real/virtual, while maintaining a 24-bit mode for backward compatibility, and a completely revamped I/O subsystem architecture capable of using up to 256 independent channels.
- System/370-ESA added architectural features allowing programs to access multiple 31-bit address spaces concurrently.
- System/390-ESA added higher speed serial (fiber optic) I/O channels.
- z/Architecture, the most recent and largest change, expands the general registers and real/virtual addressing to 64-bit while preserving full backward compatibility for 24/31-bit applications.

So it's now possible to run a 32-bit application in 24-bit addressing mode under a 31-bit operating system running on 64-bit hardware, any questions? Good.

So What Makes These Mainframe Thingies So Special?

Well, the main characteristics that most contribute to the mainframe's refusal to die are its RAS, its I/O ability, and the ISA.

RAS

RAS (Reliability, Availability, and Serviceability) is a term IBM often uses to describe its mainframes. By the early 70's IBM had realized that the market for commercial systems was far more lucrative than that for scientific computing. They had learned that one of the most important attributes for their commercial customers was reliability. If their customers were going to use these machines for critical business functions, they were going to have to know they could depend on them being available at all times. So, for the last 30 years or so IBM has focused on making each new family of systems more reliable than the last. This has resulted in today's systems being so reliable that it is extremely rare to hear of any hardware related system outage. There is such an extremely high level of redundancy and error checking in these systems that there are very few scenarios, short of a Vogon Constructor fleet flying through your datacenter, which can cause a system outage. Each CPU die contains two complete execution pipelines that execute each instruction simultaneously. If the results of the two pipelines are not identical, the CPU state is regressed, and the instruction retried. If the retry again fails, the original CPU state is saved, and a spare CPU is activated and loaded with the saved state data. This CPU now resumes the work that was being performed by the failed chip. Memory chips, memory busses, I/O channels, power supplies, etc. all are either redundant in design, or have corresponding spares which can be can be put into use dynamically. Some of these failures may cause some marginal loss in performance, but they will not cause the failure of any unit of work in the system.

Serviceability comes into play in the rare event that there is a failure. Many components can be replaced concurrent with system operation (hot swapped); even microcode updates can often be installed while the system is running. For those components, such as CPUs, that cannot be replaced concurrently, the existence of spares allows the service outage to be scheduled at the customer's convenience.

In addition to the inherent reliability of the system design, IBM also has created a tightly coupled clustering technology called Parallel Sysplex, which allows up to 32 systems to be operated as a single system image. In a properly deployed Parallel Sysplex, even the complete catastrophic loss of a single system (like those Vogons flying through the datacenter) can be tolerated without any loss of work. Any work that was being performed on the failed system can be automatically restarted on a surviving system. One additional advantage of Parallel Sysplex is that one (or more) system(s) can be removed from the Sysplex for hardware/software maintenance (for instance during off-hours), while the remaining systems continue processing the workload. When the maintenance activity has been completed the

system(s) can then be brought back into the Sysplex. One of the ways this can be exploited is the ability to upgrade the software levels on the entire Sysplex (one system at a time), without ever causing any application outage.

With all these capabilities, a true 100% system availability is very practical, and is being achieved at many sites.

I/O Throughput

These channels are actually I/O processors, they are given "channel programs" to execute, which contain chains of I/O commands including primitive branch capability. This greatly reduces the CPU's involvement in I/O operations, allowing the CPU to work more efficiently. These channels are capable of handling many concurrent I/O operations and thousands of devices each.

One typical attribute of a mainframe configuration is that there are multiple different paths to each device. A DASD device will normally be accessible through as many as four independent control units each connected to different channels. In fact, it is not unheard of for a given control unit to be connected to as many as 32 separate systems. One of the concepts that may sound very foreign to PC/Unix users is that it is quite normal for a single DASD device to be physically connected, online, and in use by many systems concurrently. There are significant hardware and software protocols in place to ensure data integrity across multiple systems to allow this to happen.

In the 360 and 370 architecture the operating system would create a channel program and attempt to execute it on a channel with a connection to the desired device. If the channel or control unit was busy the SIO (Start I/O) instruction would fail and the OS would attempt to start the channel program on another channel connected to a different control unit. If all paths were busy the OS would have to queue the request to be retried later. One of the significant changes introduced in XA was the idea of the channel subsystem, which coordinated and scheduled the activity on all the channels on the system. Now the OS only has to create the channel program and send it to the channel subsystem, which then handles all channel/control unit and queuing issues. This enabled even greater I/O throughput, while allowing the CPU to be even more efficient because it is only interrupted when the entire I/O operation has been completed.

The total I/O throughput capacity of the current z900 mainframes is no less than 24GB (that's bytes, not bits) per second. I have not personally had the opportunity to benchmark the performance on these latest systems, but while theoretical numbers can sometimes be misleading, I wouldn't be at all surprised to see a z900 performing as many as 100,000 I/O operations per second.

The ISA

While there have been significant changes in the overall instruction set of IBM mainframes over the years, IBM has maintained an amazing amount of backward compatibility for applications. Many of the most significant architectural changes have affected facilities (such as the I/O subsystem) that are only directly invoked by the OS, and are not available to application programs. IBM has gone to great pains to ensure that their customers will not have to re-write or re-compile their programs to run on newer systems. This makes it much easier for customers to adopt new hardware, making it possible for those customers to simply swap out an older system for a newer model, without having to do extensive software testing. It is very common for a company with a single mainframe to simply replace it in a span of a few hours, without any testing on the new system before putting it into production. Typically this is done with the customer remaining on the same OS version, only upgrading to the newer version as needs allow. For instance, a customer can install a new z900 system while still running a 31-bit OS, then install and test a 64-bit version of the OS in a separate LPAR before moving production onto the 64-bit OS.

That's it, For Now…

I hope you found this article to be informative and thought provoking, but due to its brevity I'm sure that there will be questions. I would like to invite everyone to head on over to the message boards and pose any questions you may have. If things get too deep over there for me I may call on a friend or two to help me answer your questions, so go ahead and ask away. Also, feel free to let me know what subjects you might like to see me cover in future articles.

You had better appreciate this Guide entry more than "Mostly Harmless", it certainly took more research! Anyway, I'm a bit worn out from all this work so I think I'll nip off to Milliways for a few Pan Galactic Gargle Blasters and watch the Universe explode for my pleasure, anyone know where that frood Zaphod's gotten off to? I'll see you at the message boards when I get out of recovery.

Photo #1 is a zSeries 900 mainframe. Photo #2 is a 1956 RAMAC 305 5 MB hard drive. Photo #3 is a zSeries 800 mainframe. Photos except #2 courtesy of International Business Machines Corporation. Unauthorized use not permitted.