

What is this “zPDT”?

(and how does it fit in?)

C. M. (Mike) Hammock
Principle Executive, Hammock IT Services

The latest entrant into the IBM mainframe compatible world is really anything but “open”, but followers of the mainframe world, open or not, should be interested. Although IBM’s “System z Personal Development Tool” (zPDT) is the newest package to enter the world of “software-based systems” (as IBM puts it) or emulation (as most others refer to it), it actually has a long history in this field.

A (very) little history

Let’s take a quick look at the history of these “software-based” (or emulation) systems.

Although most readers already have a good understanding, it will be useful to make a couple of points that will be significant later.

Emulation in some forms has been around for a long time: you could emulate an IBM 1401 on an IBM S/360 model 30 and you could emulate an IBM 7010 on an IBM S/360 model 50. But most people today assume “emulation” to mean the use of a relatively small, inexpensive computer (such as an Intel-based PC) to provide many of the features and functions of a much more expensive “mainframe” computer. IBM tried to do this a couple of times in the 1980s, with the PC/370 and AT/370 products that implemented an actual hardware instruction processor in a PC and AT-based box. The first really successful attempt to do this was probably the IBM P/390 system, introduced in 1994 using a hardware-based processor in a PC Server chassis. Although the processor was hardware-based, all of the I/O and communications functions were emulated through OS/2 software using the PC devices.

The next big step was to go fully software-based, with no hardware instruction processor. The first system to successfully do this was FLEX-ES (originally Open/370) produced by Fundamental Software of Fremont, California. FLEX-ES actually started about the same time as the P/390 was being introduced, but it did not become a really viable product until the late 1990s. For a while IBM seemed to embrace FLEX-ES and both companies entered into a useful, but somewhat unsteady, partnership including IBM promoting the use of FLEX-ES for use by ISVs (Independent Software Vendors) developing software for the System z environment. About the same time as FLEX-ES was making an impact on the commercial world with IBM’s blessings, Roger Bowler and Jay Maynard introduced the Hercules mainframe emulator package as a completely open and “free” (open source QPL license) solution that could run on both Windows and Linux host systems. Although IBM would not license its software to run on a Hercules system, there was significant evidence that many people were doing so anyway. (Yes, there are many more details, but that is close enough for our purposes.)

Then, in the mid 2000s, Platform Solutions Inc. appeared with their product which they insisted was not an “emulator” because it ran on the bare metal (an Intel Itanium processor). The rest, as they say, is history. IBM then felt the need to crack down on emulation-based systems and essentially put Fundamental Software (FLEX-ES) out of business by refusing to license some critical patents and refusing to license IBM software to run on FLEX-ES, and bought PSI to close off that threat. (UMX was another player in the emulation world but never seemed to have a large impact, so we’ll leave them out of this.) Apparently IBM believes that Hercules is used mainly by hobbyists and others that do not present a significant threat as IBM has never taken any publicly acknowledged action against the Hercules developers or users.

Where the zPDT came from

With that as a background, where did zPDT come from and how does it fit into the mix?

Interestingly enough, zPDT is actually a descendent of the first of the successful emulation systems, the P/390. To a very large degree, zPDT appears to be a version of P/390 that uses a software-based instruction processor rather than hardware. The command names and many features have the same names as P/390 equivalents, even down to calling the definition of the emulated system a “dev-map”. There have obviously been a lot of new features and capabilities added and the system has been ported to Linux. (Another IBM Internal-Only version runs on AIX on System P hardware.)

zPDT was apparently a back room project for several years before it started seeing the light of day in 2006 when the author first got experience with it. At that time it was a very functional system, although still inferior in many respects to FLEX-ES. IBM came close to releasing it for use by ISV developers in 2007 but finally decided it was not quite “ready for prime time” and kept it limited to internal use for the next several years. In October 2009 it was officially made available to ISVs who are members of IBM’s PartnerWorld for Developers program. There have been many improvements, both functional and performance related, in zPDT over the past three years so it is now a very capable system with very good performance and it continues to be enhanced and improved via periodic new releases.

zPDT, in keeping with recent IBM System z processors, only supports the 64-bit mode of operation. 31-bit systems need not apply. For developers working only with the latest versions of z/OS, z/VM, etc., this is no problem, but some developers like to keep old operating systems around for supporting their customers still on such older releases.

Getting access to zPDT

Access to zPDT is very limited, with only approved ISVs given the chance to use it. Software developers who are producing products that they will sell to run in the System z environment can apply for membership in PartnerWorld for Developers and then to purchase the hardware key to use zPDT. Eligibility for use of zPDT is essentially the same as for the PWD models of the FLEX-ES systems that were sold for that purpose.

The zPDT system is protected by a USB hardware key IBM refers to as a “token”. This token has been assigned a machine type of 1090 and is generally referred to as a z1090. Licenses are loaded or enabled on the token and are good for one year and must then be renewed. Licenses are available for 1, 2, or 3 enabled/emulated processors and these are known as the 1090 model type. Machine type 1090-L01 enables one processor on a Linux base while a 1090-L03 would enable three processors. Each z1090 has a unique serial number and the System z software running on a zPDT system will report that it is running on a 1090 with the assigned serial number.

The zPDT and z1090 are distributed in a rather unusual manner; you cannot purchase them from IBM. ISVs who are approved for zPDT use have two ways to acquire a zPDT-based system.

1. They can purchase a complete, configured, and customized system on either an IBM System x or Lenovo ThinkPad base from Information Technology Company (ITC) LLC (www.p390.com). These systems will be built on IBM recommended hardware/software platforms and will generally have support contracts included to provide first and second level zPDT support to the customer.
2. They can purchase only the z1090 token and license (again from ITC) then build and configure their own system. Although seemingly cheaper than purchasing a complete, ready to run system, the catch here is that there is no traditional IBM support for zPDT. If something does not work as expected, or the customer is not sure how to set up a good

configuration, there is no one to call. There is a [Yahoo User Group](#) that is monitored by an IBM person that users can post to for assistance.

There is also a series of three IBM Redbooks: [SG24-7721](#), [SG24-7722](#) and [SG24-7723](#) that provide very good information for developers trying to understand and use zPDT.

IBM publishes what hardware and software platforms they have tested and recommend, but there is no restriction on what x86-based hardware or Linux software developers can attempt to use.

System z Software for zPDT

The only IBM system z software currently authorized to be used on the zPDT systems are those obtained through the IBM ADCD (Application Development Controlled Delivery) process.

Developers must sign appropriate license agreements and are then provided copies of a z/OS development system (on DVDs) or allowed to download z/VM from an IBM provided site.

There is currently no provision for licensing other IBM software products for the z1090 machine type. Other software vendors will undoubtedly develop their own guidelines and procedures for licensing their software on a 1090. Linux for System z can of course be run on the zPDT, either “natively” or as a guest of z/VM, with no licensing restrictions.

The current zPDT system is directed squarely at the small ISV, many of whom previously used FLEX-ES-based systems. Since all FLEX-ES licenses have now expired, most of these developers have acquired accounts on IBM’s Dallas-based systems. Many of them did their final testing and verification on the IBM system but quietly did most of the real development work on Hercules-based systems in their offices. The zPDT systems give these developers the opportunity to re-combine their development work on a single, locally-controlled, reasonably-priced system.

Technical view of zPDT

We cannot, of course, provide a deep technical view of zPDT in a brief article such as this, but we will discuss some of the more interesting features, first from an “externals” viewpoint, then delving briefly a little deeper.

As mentioned previously, the zPDT can be thought of as a P/390-based system running on top of Linux with a software-based instruction processor. It uses “Device Managers” to talk to various I/O devices, such as the AWSCKD manager for CKD disk devices. The AWSCKD format is the same as that used by P/390s with one exception we will note shortly. Device managers are provided for tapes (AWSTAPE for emulated tape on disk or AWSSCSI for SCSI attached tapes), AWSOSA for LAN connectivity (either QDIO or LCS mode), AWS3174 for 3270 console support, and several others. The device configuration is defined in a “profile” or “Device map” that also includes system information such as the memory size and number of processors.

```

[system]
memory 6144m
3270port 3270
processors 3

[manager]
name aws3274 0002
device 0200 3279 3274
device 0201 3279 3274
device 0202 3279 3274 cms
device 0203 3279 3274 cms
device 0204 3270 3274

[manager]
name awsckd 0001
device 0120 3390 3990 /zdisk/540res.ckd
device 0121 3390 3990 /zdisk/540w01.ckd
device 0122 3390 3990 /zdisk/540w02.ckd
device 0123 3390 3990 /zdisk/540spl.ckd
device 0124 3390 3990 /zdisk/540pag.ckd
device 0125 3390 3990 /zdisk/prodpk.ckd

[manager]
name awsosa 0009 --path=f0 --pathtype=OSD
device 500 osa osa --unitadd=0
device 501 osa osa --unitadd=1
device 502 osa osa --unitadd=2

[manager]
name awsosa 0019 --path=A0 --pathtype=OSD --tunnel_intf=y
--tunnel_ip=10.1.9.1 --tunnel_mask=255.0.0.0
device 504 osa osa --unitadd=0
device 505 osa osa --unitadd=1
device 506 osa osa --unitadd=2

[manager]
name awsosa 0029 --path=f1 --pathtype=OSD
device 508 osa osa --unitadd=0
device 509 osa osa --unitadd=1
device 50a osa osa --unitadd=2

[manager]
name awstape 004
device 590 3490 3490
device 591 3490 3490

```

Simple zPDT Profile for a z/VM system

Since zPDT runs on a 64-bit Linux base, very large System z memory configurations are possible and some systems have exceeded 40GB. The hosting hardware and Linux system must have sufficient real memory to contain the defined zPDT images. The hosting Linux system should not be allowed to swap so the defined System z instances should total at least 1 – 2 GB less than the real memory size of the hosting system.

zPDT really provides a realistic emulation of an LPAR within a System z processor complex. This is compared to FLEX-ES which presents more of a full processor complex view, complete with “LPAR-like” instances. Although multiple zPDT instances can be run, they each have to

run under a different userid (UID) and processors cannot be shared across these instances. If a user has a 1090-L03 and starts one instance with three defined processors, no additional instances can be started. Disk devices can be shared, although it is through a somewhat awkward mechanism.

The enabled processors can be defined as either normal (CP), IFL, zIIP or zAAP processors. A processor defined as an IFL, for example, can only be used to run Linux or z/VM to host Linux. Unlike on standard System z systems (z9, z10, etc.) there are no cost or performance advantages to defining specialty processors, but it may be very convenient for developers to test their products ability to exploit the specialty processors.

Because of the limitations in sharing processors, many developers are choosing to use z/VM to facilitate better resource sharing among multiple guests rather than defining multiple zPDT instances as they might have done with FLEX-ES.

zPDT has implemented several interesting features, one example of which is “disk versioning”. An AWSCKD disk can be defined to support versioning and some point in time a command is issued to “checkpoint” that disk volume. From that time on any changes to the disk volume are saved in a separate area. At a later time another command can cause the changes to either be backed out or made permanent. This is another potentially very useful tool for the development environment. The use of disk versioning will break compatibility with P/390 AWSCKD files. No external channels (parallel, ESCON, FICON) are supported by zPDT on Linux, so no channel attached devices can be connected and used by these systems.

Any details of the internal structure of zPDT are very sketchy and are obtained chiefly by reading “between the lines” in various manuals and presentations. Early, IBM Internal Use Only versions of zPDT used an “interpretive only” mode of execution. This is how Hercules works today, examining each System z instruction, determining what it is supposed to do, and then executing the equivalent x86 instruction(s). More recent versions of zPDT have added a “Just-In-Time” (JIT) compiled mode to this. Some algorithm determines whether a section of code should be interpreted or whether it would be better to invest some more initial cycles to compile the System z instructions into equivalent x86 instructions (to simplify the process somewhat). This interpreter plus JIT compiler is what FLEX-ES used to achieve its high performance. FLEX-ES also cached the compiled sections of code for later reuse. I have not been able to verify that zPDT does this caching also, but I suspect so.

How does it compare?

So how does the zPDT compare to other systems that developers have used in the past? Will former FLEX-ES users feel at home? Will Hercules users doing development work (yes, we know you are out there) have a steep learning curve? What are the differences and advantages of zPDT?

Host Platforms:

All three systems (zPDT, FLEX-ES, and Hercules) will run on a Linux-based system. Hercules will also run in a Windows environment. (I consider this a disadvantage but will be kind and ignore this option.) Both zPDT and Hercules seem to be less sensitive to the specific Linux distribution used than FLEX-ES appears to be. I believe this is because FLEX-ES has more low-level code that is likely to be affected by changing Linux distributions. We'll give a slight advantage zPDT and Hercules for this category.

Ease of Use:

There are many facets to "ease of use" and the three systems exhibit different characteristics. If someone was given copies of the three systems and their documentation and told to install and make them work I believe they would get zPDT operational first. FLEX-ES has the steepest

initial learning curve while both zPDT and Hercules are initially simpler. zPDT is simpler than Hercules because it has fewer options to choose from and basic networking is simpler to implement.

As the system configuration gets more complex, with more networking, multiple instances, and shared DASD and other devices, the picture changes a bit. FLEX-ES does a much better job of simulating an LPAR environment and this makes a multiple instance configuration easier to implement and operate than either zPDT or Hercules. I give zPDT the overall advantage in ease of use while FLEX-ES comes in a close second because of the relative ease of implementing more complex multi-instance configurations. Hercules suffers some because the documentation is not as complete or clear, especially when compared to zPDT.

Flexibility

Ease of use frequently comes at the expense of flexibility and we see that in play here. Both FLEX-ES and Hercules offer more options for things like architectural mode (S/370, ESA, 64-bit, etc.) and other processor options. FLEX-ES also offers very powerful configuration options, like multiple instances (very similar to LPARS), shared DASD and other devices, network access to remote devices and possibly most important, true channel attach capability with parallel and ESCON channel hardware. The ability to share processors across multiple instances also provides significant added flexibility for FLEX-ES. Hercules also has device sharing capabilities, although not quite as clean and flexible as FLEX-ES, it also allows remote (network) access to physically remote devices. Here I think FLEX-ES comes out on top, with Hercules close behind in the middle and zPDT trailing. We should keep in mind, however, this is probably exactly what IBM intended. IBM only wants developers using the more current System z operating systems.

Completeness

In terms of a complete implementation of the System z (or System/390) environment, each system has some advantages. Both Hercules and FLEX-ES offer more architectural modes than zPDT. FLEX-ES offers a much more complete simulation of the LPAR environment, including processors shared across multiple instances. The availability of parallel and ESCON channels also makes for a more complete implementation for FLEX-ES, although the lack of Fiber/FICON channels negates the channel capability somewhat. However, zPDT offers more of the current System z functions, such as providing full QDIO capabilities. Network adapters emulate OSA Express and can be configured accordingly. For example, with z/VM we can make full use of a Vswitch interconnecting multiple guests and then connecting to the outside network via OSA/QDIO emulation on the network adapter. Overall I believe FLEX-ES is more complete for users who need to run older operating systems while zPDT may be better for those running current level operating systems.

One important aspect that I'll put under "completeness" is how well the emulator actually adheres to the defined architecture. Both FLEX-ES and zPDT have been subjected to IBM's architectural conformance tests; the same tests to which standard System z hardware is subjected. This goes a long ways toward assuring developers that their system behaves just as a real System z would behave when running their software in a customer's shop. For some developers this may not be an issue while it is a major concern for others. Hercules has had years of use and is unlikely to have many "bugs" remaining, but there is no way to verify or guarantee this. (IBM is unlikely to allow Hercules developers access to the architectural tests any time soon.)

Performance

Performance is not just pure CPU speed in MIPS, but also I/O performance, and general efficiency. For example I believe that zPDT is now slightly faster than the most current FLEX-ES (I say that "I believe" because I have not been able to completely benchmark the most current FLEX-ES Version 8 system). Running in 64-bit mode on 64-bit hardware, zPDT can get a few

(perhaps 5%) more MIPS out of a given processor or core. However, FLEX-ES does I/O much more efficiently. Heavy I/O workloads can easily consume 50% of a processor on zPDT while FLEX-ES rarely uses more than 5 - 10% of a processor for I/O. I also believe that FLEX-ES's disk caching is more effective and controllable than zPDT's or Hercules' use of the Linux filesystem caching. Both FLEX-ES and zPDT will provide about 30 - 50% better performance than Hercules. So the advantage goes to zPDT for pure CPU performance or to FLEX-ES for better I/O performance/efficiency. Either FLEX-ES or zPDT, running on a contemporary Quad-core processor, can achieve significantly more than 100 MIPS per core, or as much 300 – 350 MIPS for a three processor-enabled configuration. (Note: certainly not all quad-core processors will achieve or exceed 100 MIPS per core, but some definitely will.)

Overall

As you might have guessed by now, there is no clear overall "winner." Both FLEX-ES and zPDT excel in some areas and not in others. In most categories Hercules comes in last, but if we include a category of cost or code availability, it would certainly come in first there. Unfortunately, we no longer have a choice of using FLEX-ES, so the logical choice, for those who have the choice, and the dollars, will be zPDT.

A note about Linux on System z

Most of the discussions above, especially the comparison of zPDT with FLEX-ES and Hercules have really been about running the IBM System z operating systems (z/OS, z/VM, and z/VSE) on these machines. In this discussion, Hercules is obviously handicapped by the inability to license these operating systems on Hercules, no matter what the relative performance, ease of use or functionality. When the discussion turns to Linux on System z (which we will now just call zLinux) the tables turn somewhat. Since zLinux can be run on any platform, including Hercules, the very low cost and good availability of Hercules offers a more significant advantage. Hercules offers almost as good a zLinux platform as zPDT; the performance is not quite as good, but it may be “good enough.”

There are two issues which do discourage the use of Hercules for development of zLinux applications or to actually run zLinux applications on Hercules.

1. It seldom makes sense to run applications in “production” mode with zLinux under an emulator such as Hercules. Why run zLinux within an emulator and suffer the necessary performance/capacity impact, rather than just running it on a “first level” Linux running on the base hardware? For most applications, zLinux offers few advantages over an Intel-based Linux. Most zLinux users run it on System z because of the flexibility and reliability of the underlying hardware. If the underlying hardware is still an Intel PC, nothing has been gained by going to zLinux.
2. Assuming that Hercules users will abide by IBM’s licensing restrictions, there is one major disadvantage of using Hercules for zLinux development: the lack of z/VM. Most zLinux users and developers utilize z/VM to host multiple (many) zLinux guests to take better advantage of the System z hardware. Again, this is a licensing issue, not a technical one, but it is an issue.

Where does zPDT go from here?

The obvious question is whether IBM will allow zPDT to be used in additional environments, besides the pure ISV software development world. The environment or use that immediately comes to mind is the commercial, end-user world but that is a huge leap for IBM at this point. There are also some shortcomings in zPDT for use in the commercial small System z environment. Let’s first look at some other possibilities that might be considered reasonable

“baby steps” then look at what would should be done to make zPDT more suitable for commercial use.

One non-development use for zPDT might be “packaged” solutions, especially ones that would be distributed at multiple sites. Almost 20 years ago a major insurance company installed thousands of small, modified IBM 9370 systems (running VSE) in local offices. zPDT systems could be packaged and distributed in a like manner, putting a z/OS (or z/VM, or z/VSE or zLinux) under a desk in thousands of branch offices of some company. The local people would not even realize (or care) that they were running z/OS on an emulation-based system. Another possibility would be for mobile systems; z/OS-based (or z/VM, etc...) systems running on a hardened laptop that could go “on the road” or into potentially dangerous environments. The primary advantage of both the above scenarios is that a standard IBM System z mainframe would not normally be considered, so there is no chance of the zPDT system impacting traditional mainframe sales. This is a very important consideration for IBM mainframe planners. If IBM was to make the huge leap and consider selling zPDT for use in traditional commercial environments, what would need to be done to make it more attractive? Probably the biggest factor is some form of traditional “channels” or at least support access to equivalent devices. Well-supported tapes are still important, although actual ESCON or FICON channels are not required, some form of reasonably-priced and supported (by zPDT and the operating systems) is necessary. This could be as simple as supporting SAS connected LTO tapes. “Channels” themselves are not nearly as important as they used to be, but supported access to equivalent devices is important. Another factor is support and use of back-level operating systems. Many small mainframe customers are still running ESA mode systems because they stayed on them too long and have effectively gotten trapped there. IBM would need to offer some encouragement, technical or financial, to help these old/small customers move up to current software on a zPDT.

Sixteen years after IBM announced the original P390 products some customers are still using them in everyday business. Although the P390 family has been somewhat of a headache for IBM (mainly because of the software licensing it introduced) it has really turned out to be a successful and long-lived product. It will be very interesting to see if the P390’s apparent offspring, the zPDT, sees similar success.