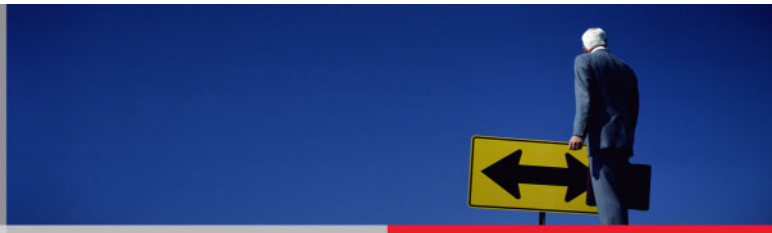
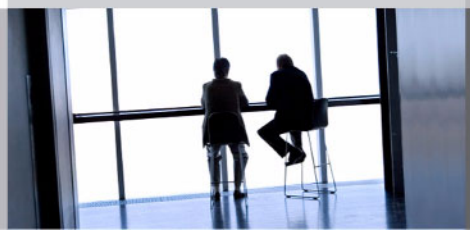
IBM z/Transaction Processing Facility: Overview and Enterprise Integration using SOA





Redguides
for Business Leaders

Bruce Armstrong
Barry Baker
Jonathan Collins

- Build efficient and highly scalable solutions on System z
- Discover the business and IT benefits of z/TPF
- Learn about the key features and functions of z/TPF

Redbooks

Executive overview

IT executives and decision makers are under constant pressure to deliver simultaneously near-term cost savings while also modernizing IT architectures and systems to allow the business to respond more rapidly to, and to take advantage of, changing market conditions.

In the past, some IT executives were given a budget and the latitude to pursue “green field” development of entirely new IT solutions to replace existing systems with a goal of providing, at a minimum, a solution that is more flexible and amenable to future changes. However, today’s global economic conditions have significantly curtailed this approach to modernization. IT executives are challenged to show a return on investment significantly sooner which is causing many companies to take a more incremental, evolutionary approach to modernizing their systems.

Certainly new IT architectures, systems, and solutions continue to be developed, but their focus has shifted away from replacing existing solutions to implementing solutions to new problems and providing new capabilities to the business. A common architecture that is used for these incremental modernization efforts is one based on service orientation and the adoption of industry-standard frameworks. This architecture allows IT executives to extract the most value from existing assets by prescribing standardized ways to achieve application and system interoperability in a non-proprietary manner. IT executives can then build out new capabilities that either augment, or make use of, existing solutions faster and with greater flexibility and choice.

IBM® Transaction Processing Facility (TPF), with its latest version being IBM z/Transaction Processing Facility (z/TPF) Enterprise Edition v1.1, has a lineage that dates back to the 1960s. It is used by many Fortune 500 and multiple Fortune 100 companies, typically providing the infrastructure that enables one of their main sources of revenue. Said another way, the underlying TPF architecture, which remains unchanged with z/TPF, combined with TPF-based applications, provides a proven solution to some of the most demanding transaction-based workloads—workloads that require a globally consistent view of the business’s critical data that is undergoing near-constant change, while also providing extremely high levels of availability. Beyond this core capability, z/TPF also provides for a modern, open run time that allows it to be an active participant in the larger enterprise architecture. z/TPF relies on an open Linux® and Eclipse-based development environment that expands the z/TPF talent pool.

This IBM Redguide™ publication describes the business drivers and their impact on IT. This guide also discusses how z/TPF, in conjunction with the System z® hardware platform, can

play a significant role in an enterprise's overall IT architecture by providing flexibility, responsiveness to market and customer needs, and cost effectiveness. It also discusses the features and capabilities in z/TPF that provide for greater application insight, allowing z/TPF-based assets to be modernized through service orientation and the adoption of industry-standard application frameworks.

Business drivers

As the rate of change occurring in the business environment increases, IT executives are challenged to deliver on goals that seem to be in conflict. They are challenged to provide a foundation for innovation to enable their companies to respond to changes on the horizon, while at the same time to raise the level of the return on investment (ROI) from IT, and to realize that ROI sooner. Beyond these conflicting goals, CIOs and IT executives are asked to enable the corporate strategy as well as to influence and inform the corporate strategy based on the value added capabilities that they think they can provide.

These executives are working to drive the business and IT areas closer together in an effort to make working together easier and to develop visionary plans that enhance the competitiveness and reach of their companies. In short, they focus on the bottom line of the balance sheet and on helping create top-line revenue growth.

From the report on the 2009 IBM Global CIO study, *The New Voice of the CIO*, it is clear that the responsibilities and focus of the CIO and other IT executives is changing. Based on the more than 2,500 CIOs interviewed, today's CIOs are spending more than half of their time on "activities that spur innovation," which also includes managing non-technology business issues, while spending the remainder of their time on the more traditional tasks associated with the CIO role related to "managing the ongoing technology environment." These new responsibilities, as well as the traditional pressures placed upon the CIOs, are pushing CIOs to explore new approaches to providing new capabilities with speed and flexibility.

The business and IT landscape is constantly changing

Businesses today need to be more nimble and dynamic to ensure that they are responsive to market needs. Depending on the strategy of the business—be it cost leadership, differentiation, or focus—IT executives are feeling a variety of pressures. If the business strategy is focused on cost leadership, then you expect the IT executive to be focused on closely managing the costs of IT. If the business strategy is focused on differentiation, you expect the IT executive to focus on the balance between cost-cutting and providing the infrastructure needed to continually separate offerings from other offerings in the marketplace.

Regardless of the business strategy in place today, IT executives must cope with rapid changes to the business strategy in response to the changing customers' needs and the market. In addition to the changing business strategy, IT executives have to deal with the constant changes in the IT landscape and be acutely aware of the "hype cycle", coined by Gartner, to know when a new technology has moved beyond the hype and has started to truly provide practical benefits.

In response to these challenges, IT executives are working to establish an IT architecture that is flexible, responsive, and cost effective.

Flexibility

As the demands of the market change, companies need to respond. They need to be flexible with their products and their lines of business. In this environment, continuing to rely on isolated business silos and processes will not provide the level of flexibility needed to respond.

Today, businesses are focused on integrating processes and composing new processes from smaller processes that are either unique to a line of business or were reused from other business units. The IT executive is challenged to integrate the IT architecture throughout the lines of business to help achieve this integration. Beyond maximizing the value of a company's processes, business process integration drives an increase in the overall understanding of a company's capabilities and allows for an analysis of which portions of the business model are strategic, competitive, or basic. With this knowledge, the business can initiate changes to processes and can also decide to outsource the basic portions of the business model, leaving resources to focus on those areas where they can differentiate themselves from their competitors.

With this in mind, the IT executive is challenged to put in place an IT architecture that allows for the re-use of existing resources and IT systems, while enabling new process capabilities, including the outsourcing of portions of the business model without disrupting the overall process portfolio.

Responsiveness

Closely related to flexibility is responsiveness. Companies are looking to respond to customer needs, to respond to rapid increases or decreases in demand, and to do so very quickly. A missed new sale because of an IT problem can be a missed opportunity at a long-term, repeat customer. IT executives are challenged to put in place solutions that can scale to meet demand spikes.

Responsiveness also deals with the time it takes for the business to recognize that a process needs to change in response to changing customer needs and with the time that it takes to have that process change instantiated in the supporting IT solution. To increase the level of responsiveness, IT executives are challenged to provide the right level of IT solution monitoring and insight that maps to the business results with the corresponding tools needed to rapidly change the existing IT solutions.

To achieve this responsiveness, IT executives focus on unlocking the value of the exploding amounts of data that is generated by IT solutions and on providing standardized means to access that information from the operational systems. IT executives must also isolate those parts of the IT solutions that tend to change rapidly and move away from implementing those things in hand-written application code. Instead, they must adopt solutions such as business rules management systems to increase the responsiveness to changing requirements for the IT solution.

Cost effectiveness

The IT executive who is tasked with enabling a low-cost strategy focused on the cost effectiveness of IT architectures is also asked to be a "cost-cutter." To this end, IT executives are looking to put in place the IT solutions that are optimized for a particular process, or they are making use of virtualization technologies to map many different processes to one particular technology platform. The virtualization trend of today is primarily to cut costs by consolidating hardware platforms. The next wave of analysis is likely to be IT process differentiation, which determines what processes are commodity functions that are eligible for commodity hardware and software and what processes are strategic to the business and require special considerations.

If a process is deemed to be strategic and is critical to the functioning of the business, it can require a more purpose-built IT platform to meet the unique needs of the workload, such as high levels of availability, scalability, reliability, low TCO, and so forth. If the business requirements make such a system necessary to achieve the strategy, then one of the key requirements of such a solution is to ensure that it is open and interoperable with the remainder of the IT architecture.

What are the options?

Very few IT executives have the luxury of being able to create the IT architecture of a large organization and treat it as a “green field,” meaning that they are not encumbered by what exists today in the environment. A more accurate view of the issues that IT executives deal with is a collection of disparate IT solutions that were most likely created years, and in some cases decades, ago and have undergone near-constant change to add new functionality and enhancements. In this environment, IT executives need to make decisions about which of the existing IT solutions require attention so that those solutions can better support the business strategy. The approaches to changing these IT solutions can vary in cost, complexity, and associated risk.

Complete rewrite

Over the years, for very large IT solutions, businesses tended to characterize them as brittle and obstacles to change. To a certain extent, this assessment is a fair characterization. Large complicated solutions are expensive to create because they often solve large complicated problems. Unless the nature of the problem changes significantly, most new solutions to the original problem are likely to be significant in size and complexity. This fact, coupled with the evolving capabilities of the existing IT solution in question, makes a complete rewrite a costly option that also comes with a high amount of risk that the new system might not provide the detailed functionality that the existing system delivers. The task is large because many of these IT solutions were created over decades and have a minimal amount of documentation. This lack of documentation leaves the new system to be created, in part, by reading through and understanding millions of lines of source code to understand all of the processing and data interdependencies that in most cases need to be re-created in the new solution. For example, if the IT solution conforms to a standard messaging format that allows for integration with business partners or suppliers, then the new IT solution is bound to have the same requirement.

Also, complete rewrites of solutions often do not consider the local factors present in the IT organization that enabled the existing solutions, in most cases, to meet the needs of the business and how these local factors might not translate easily to the new system. Local factors can include a broad set of issues, including adequate skill levels of the IT staff, levels of technology adoption, operations management capabilities, and so forth.

For example, assume that a business has a strict requirement for system availability and that a complete rewrite effort of this system is initiated. Figure 1 on page 5 shows how the raw capabilities of the platform are combined with local factors to determine the overall availability of the system. In this example, the boxes represent the range of availability a user can expect from a particular platform, where the terms *Industry Best*, *Industry Average*, and *Below Average* refer to the various levels of achievement that can be attained through local factors. In this case, to move to the new platform and to achieve the required level of availability might entail finding the highest skilled people in that technology. Alternatively, it might require the full adoption or exploitation of the underlying technology just to achieve the required level of availability that is met with the existing platform and average levels of local factors.

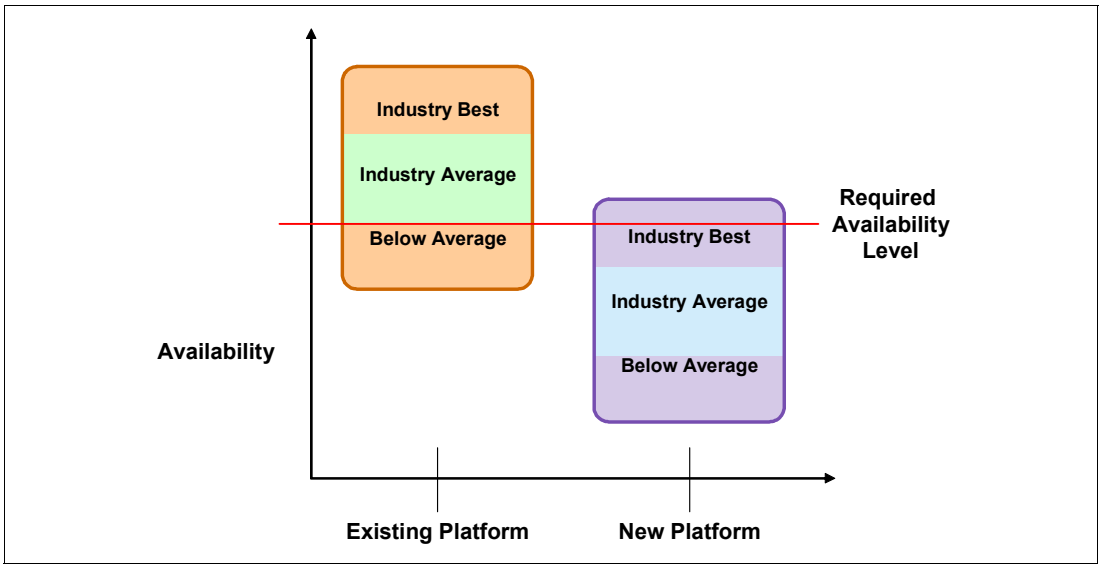


Figure 1 Impacts of local factors on platform selection

The complete rewrite approach brings with it significant challenges. However, that is not to say that this approach is without merits and that it should not be included in the IT executive toolbox. If the business need is extreme, then a complete rewrite might be the best option. However, it is important to be aware and to consider all of the challenges, costs, and likely outcomes that are associated with such an effort.

Status quo

The disadvantages of the complete rewrite has driven many IT executives to consider how to maintain the current path while trying to keep up with the flood of new requirements and a decreasing budget to address those requirements. While leaving the existing IT solution in place, IT executives can focus on extracting the most out of these application and data assets that have been refined over the years. Reuse can be achieved through the adoption of open standards-based ways of interacting with the applications and extracting value out of the data on the existing IT solution. This reuse can be achieved through defacto-standard messaging such as IBM WebSphere® MQ or through the adoption of Web services, either of which can be part of a service-oriented architecture (SOA). Beyond reuse, many IT executives are adopting standards-based messaging as a means of reducing the ongoing costs that are associated with operating and maintaining ad-hoc connectivity solutions that were created over the years.

As mentioned earlier, another technique is to isolate those portions of the IT solution that are changing rapidly and to expose them to IT staff and the business so that they can be analyzed, modeled, and updated in a way that does not require a significant amount of IT staff support and so that changes can occur rapidly. For example, introducing the IBM WebSphere iLOG Business Rules Management System to an existing IT solution can significantly improve its flexibility and responsiveness. Systems like this have multiple deployment options, ranging from embedding the rule execution engine in the existing IT solution to deploying it on a new stand-alone system whereby the exiting IT solutions can use standard messaging protocols to exploit it.

The “status quo” can bring with it some negative connotations. However, by using some of these techniques, IT executives can add targeted incremental capabilities to an existing IT solution with significantly less cost and risk when compared to a complete rewrite approach, while also demonstrating ROI sooner. The term “status quo” implies small, incremental

improvements. Work is accomplished, but that work might not advance the business priorities of the business.

Incremental transformation through service orientation and industry frameworks

Having reviewed the two common paths that are taken by IT executives to achieve higher levels of flexibility, responsiveness and cost effectiveness, it becomes clear that an alternative approach is required. An approach is needed where IT executives can establish a flexible IT architecture that allows for the ongoing usage of those portions of the existing IT solutions that are optimized and working well, while providing a framework to modernize, rewrite, re-platform, or even purchase those portions that are not currently meeting their requirements or that are difficult to operate, maintain, or enhance.

Providing for *incremental transformation* can be thought of as a blend of the *complete rewrite* and the *status quo* approaches. Incremental transformation can take advantage of the benefits of both approaches while leaving behind some of the disadvantages.

Figure 2 shows the goal of taking an incremental transformation approach, providing greater IT capabilities faster.

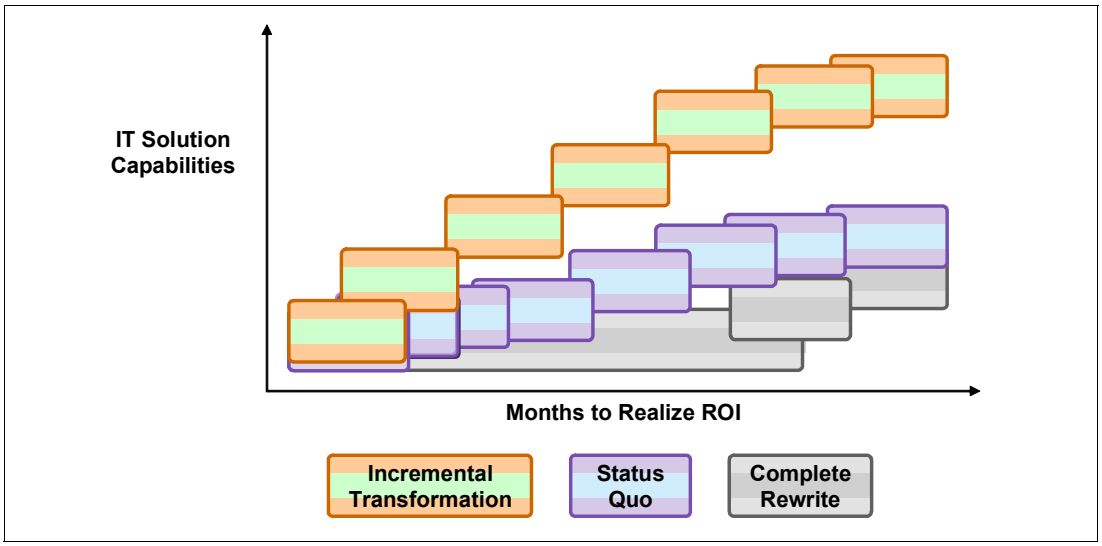


Figure 2 Modernization approaches and ROI

With the overall business strategy defined, an approach that can be taken to help effect strategic change is one in which a Component Business Model™ (CBM) is constructed. CBM is a logical representation of the various business functions that make up an enterprise and includes capability information about the existing processes, technology, and people for each of the components. From this work, IT executives can perform analysis to determine if the organization's capabilities align with the overall strategy and can make decisions with regards to identifying candidates for rewriting, enhancing, exploiting, and transforming. The value of viewing an organization's capabilities through the perspective of the CBM is that it helps to provide the right level of detail to help make high-level decisions about how an organization implements the overall business strategy. CBM also helps to highlight the components of the business that create differentiation.

Mapped directly from the CBM are the underlying business processes, which are implemented on top of an architecture that is based on service orientation. Establishing an IT architecture around the concepts of service orientation, which is a business-driven architectural approach, can create a tighter alignment between the business and IT, because

it is an architectural style that supports business process integration through the linking of services or repeatable business tasks. The promise of service orientation lies in the business flexibility that it enables by encapsulating existing and new IT assets as loosely coupled services that can be reused, changed, re-platformed, and combined easily to meet the changing needs of the business. Figure 3 depicts the various layers that are involved in this approach, which enables an IT executive to have targeted discussions that ensure that the business strategy is aligned with the overall IT architecture.

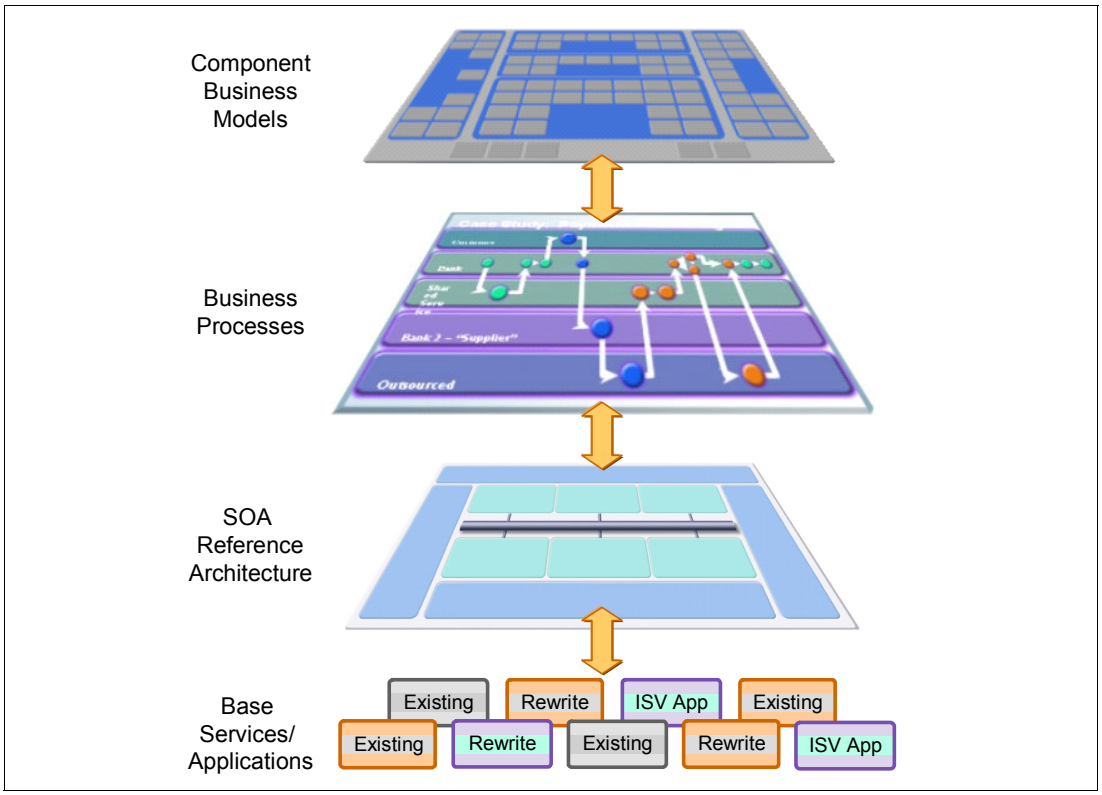


Figure 3 Mapping from the business strategy to the IT architecture

The first logical step for an IT executive is to move towards adopting standard messaging throughout the environment, which blankets the existing base level services and applications using service-oriented techniques. This task then informs the creation of the overall SOA Reference Architecture, as will the creation of the CBM and a clear understanding of the underlying business processes. After this structure is in place, business and IT can trace from the top-level Business Component down to the underlying Base Services or Applications that support it and make decisions about what, if any, actions to take against those Base Services based on the categorization of the Business Component (for example strategic, competitive, or base).

To help accelerate this overall approach to incremental transformation through service orientation is the creation and adoption of what are called *industry frameworks*. Industry frameworks are a predefined set of software and middleware usage patterns that have industry-specific extensions that provide an IT architecture that addresses the specific needs of a particular industry. For example, an industry framework builds on top of existing middleware offerings and provides value through industry-specific process and information models, design templates, re-usable code assets, and common (utility) services and support for relevant industry standards.

By establishing an IT architecture based on service orientation, an IT executive can provide the foundation for incremental transformation whereby asset reuse is minimized and the

existing assets are enhanced and extended. Simply stated, any modern platform must be able to effectively participate in an SOA. In the following sections, we discuss how IBM z/TPF Enterprise Edition v1.1 can effectively participate in an SOA while delivering on its overall value proposition.

Introducing z/TPF

z/TPF is the market-segment dominant transaction processor and operating system implemented for specialized high-bandwidth computing that exploits the IBM System z platform. z/TPF is optimized for:

- Maximum transaction rates (tens of thousands per second)
- Maximum networked user communities (hundreds of thousands concurrent users)
- Fast access to write intensive data in large, contiguous data bases
- Superior Reliability, Availability and Serviceability (RAS)

z/TPF originated from a joint development effort between IBM and airline customers. In the past, the airline industry and IBM shared source code and intellectual property. As the industry evolved, IBM became the provider of the operating system, leaving customers to focus resources on application development.

IBM continues the partnership with z/TPF customers by sharing the operating system source code, serving as a clearing house for new functions, and addressing the requirements of the z/TPF customers. Airline customers have taken the application code in a variety of directions over the years to address their individual needs and, at times, have contributed code for the common benefit of the industry. z/TPF is unique in the airline industry for its performance and availability while addressing the demands of unique data processing workloads.

TPF has progressed as System z hardware has progressed, including the exploitation of faster processors, expansive data storage, and huge amounts of addressable memory. TPF pioneered the design and implementation of clustering System z servers together for high availability and even higher processing power. More recently, z/TPF provides support for open source compliers and state-of-the-art memory addressing.

Benefits of z/TPF to your business

z/TPF is a unique technology in the IT industry. No other operating system accesses more of the raw I/O processing power of System z, minimizing the total cost per transaction in a highly available environment. The value of z/TPF is that it fully exploits what System z has to offer to deliver extreme transaction processing (XTP). With z/TPF, IBM maintains and extends the performance and total cost of ownership (TCO) leadership of TPF while also extending IBM product leadership in non-functional quality of service capabilities.

z/TPF delivers the following benefits:

- Faster time to market for new features and services

  z/TPF provides SOA support with an open runtime, open tooling, open interfaces to TPF data and applications, open source package porting, and better and faster problem determination.

- Higher quality of service

  z/TPF supports 64-bit processing, which improves capacity, error recovery, and file cache. z/TPF also offers improved system resource monitoring and program control, less down time for configuration updates, and system monitoring through enterprise tooling. z/TPF

has a proven track recover for providing high volume transaction processing resulting in extremely high levels of data availability and integrity.

- Data security

  z/TPF provides security support for data in flight, at rest, and in use. z/TPF also provides support for secure key management and PKI to meet various security guidelines, relying on the underlying security capabilities of System z.

- Minimize the risk of modernization

  z/TPF enables the move to service orientation with SOA support, open interfaces to TPF applications and data, and support for existing applications.

Key motivation for z/TPF

TPF 4.1 was designed, and originally delivered in the early 1990s, and while TPF 4.1 continues to support System z, it does not fully exploit the latest System z hardware. z/TPF fully utilizes System z hardware to provide the basis for modernizing the TPF environment through the adoption of service orientation and industry frameworks, through its standards-based open runtime, and through the use of standards-based open development and tooling environments. These improvements enable your organization to better align IT systems with business objectives while maximizing the ROI of existing TPF-based assets.

Like many sectors of the IT industry, specialized TPF skills can be scarce. With the introduction of z/TPF, the development environment utilizes a Linux platform along with open tooling based on Eclipse technology. This environment allows you to use a common skill set throughout the IT environment for TPF application programming, systems programming, and operational skills. With this greater flexibility, you now have more options to hire entry-level resources.

What makes z/TPF unique

As an IT executive you might ask “why do I need another unique platform in my environment? Wouldn’t I be better served by simplifying my IT architecture down to one platform and operating system?” These are fair questions for a number of reasons. First, IT solutions gain inertia. That is, when a solution is in place, it tends to be hard to remove. Second, maintenance and operations costs are ever increasing, and consolidating down to one standard platform can reap significant cost savings.

This section explains why z/TPF is unique and necessary for particular workloads and how it can easily coexist with other operating systems when taking a consolidation approach with IBM System z.

In one way, z/TPF is “just” an operating system. We say “just” because, like any other operating system, it alone does not provide all the functions needed for a business. Alternatively, just like other operating systems, applications use many functions to perform certain tasks. An application using these functions in some programmed combination can provide a needed business function, such as booking a reservation or performing a credit card authorization.

Purpose-built versus general purpose

You can highly customize z/TPF to meet the unique needs of your business. z/TPF provides an extensive suite of application and system services that provide a great deal of flexibility, allowing for the creation of highly-tuned IT solutions.

Drawing analogies is never without pitfalls, but it can help to illustrate a point. If z/TPF was an automobile, it would likely be a Formula 1 race car. A race car needs to be fast (high performance), it needs to be able to be fast for long periods of time with out stopping (highly available), and it needs to do a few things really well (purpose-built versus general purpose).

The design of TPF is optimized around the workloads needed primarily by reservation applications. However over the years, other industries (such as financial services) have taken advantage of the strengths of z/TPF. The attributes of a z/TPF run time environment are high availability and extremely low response times. The functions included in z/TPF range from software that manages transactions competing for an airline seat, to dealing with massive flight schedule changes (when flights are cancelled due to weather at a major airport, for example).

z/TPF is a special purpose operating system for unique workloads. It is optimized to be very good at specific processing tasks, as opposed to a more general purpose operating system such as Linux.

Linux, for example, is a general purpose operating system that focuses on meeting the processing needs of a majority of the marketplace. (Granted there are Linux derivatives that are optimized for some specific environments but these still strive to maintain the core, general purpose nature of Linux.) The term *purpose-built* when used to describe z/TPF describes an operating system that performs select functions exceedingly well. These functions are designed to be optimized to the demands of the workload and are not focused on running a variety of workloads.

Efficiency

Efficiency is a term commonly used in the automobile industry or for electrical appliances but has applicability to computer systems as well. For example, say you simply wanted to write a piece of data to disk (and do it as quickly and efficiently as possible). Operating system #1 is designed assuming that applications are trusted to write the correct data and that the burden of backing out the data in case of an error is the responsibility of the application.

For the sake of comparison, operating system #2 is designed assuming that an application can not be trusted and that there were multiple applications executing in the environment, each with the authority to write data. This operating system might have additional software functions included, for example, to write data to disk but also to save a copy of the prior data to provide a back-out service in case of error. Thus, operating system #2 might trade some performance (more software instructions using more CPU and, therefore, longer response time) in exchange for protection from an errant application irreparably changing data. As another example, because there are multiple applications with update capability to data, this operating system might also provide comprehensive data-locking processing to prevent one application from overlaying the writes of another application. This operating system with locking functions is also enforcing data integrity over performance.

Which operating system is more efficient? If the end goal is to write data to disk and if the application can be trusted, then operating system #1 can be said to be more efficient. Operating system #1 does not incur all the extra steps of copying data for backout of data written in error and does not need to lock data from competing applications.[1]

As mentioned before, analogies are never perfect, but we have described z/TPF as being *purpose-built* (like a Formula 1 race car) and able to provide high availability (like a long endurance vehicle). Now, we describe it as an efficient platform for special purpose functions, so that z/TPF can be considered a fuel efficient Formula 1 race care as well as an endurance

[1] z/TPF has the option of locking data. We make the point made here for illustration purposes only.

vehicle. Note, z/TPF is efficient only to the extent that the condition of trusted applications are acceptable to the application developer.

Responsiveness

Efficiency takes several forms in data processing. In the case of z/TPF, the response time of specific tasks can be very fast when compared to other, more general purpose systems. What is meant by the term “very fast”? Quoting performance numbers for any environment is difficult because there are so many variables involved. General rules to keep in mind include:

- Reading data is typically faster than writing data, assuming that the data to be read can be cached in memory and not constantly changing.
- Caching of data to be read is typically a good thing for performance because it prevents the need to access Direct Access Storage Device (DASD). (Accessing most DASD is a relatively time consuming process compared to memory speeds.)
- Writing data typically takes longer than reading cached data because systems today need time to persist the data to mechanical DASD devices.

> **Note:** There are new generations of DASD that are not mechanical, such as Solid® State DASD (SSD). However, those technologies are outside the scope of our discussion. Even SSD devices take more time to write than read data.

Thus, predicting performance numbers depends on many, many factors. With that said, numbers are helpful in defining “very fast.” The experience of multiple z/TPF customers has shown response times for transactions can be in 1-10 millisecond range (1 1/1000th of a second) for reads and writes. Estimates of other operating systems environments on similar hardware have ranged from 5x to 10x more or higher. The design point for z/TPF is to provide a streamlined execution environment to provide, where possible, an order of magnitude improvement in performance.

Another consideration for a responsive system is how variable the response time becomes as the system comes under stress. The term *stress* can mean several conditions, but for the sake of this discussion, we consider it to mean above normal processing demand (such as rebooking thousands of passengers due to flight cancellations). z/TPF is optimized to provide consistent response time under these situations of high processor utilization. Whether the environment is running at 60% utilization or 98% utilization, the user of the system sees the same response time. z/TPF also provides highly efficient and effective clustering technology so that workload can be balanced across multiple systems for both performance (responsiveness) and availability.

Scalability

Efficiency can also assist with the scalability of a z/TPF environment. If a function can be accomplished in very small intervals of time, more work can be performed per unit of time than common in a general purpose environment. Because more work is needed to be accomplished, additional processing hardware is usually required. The efficiency of z/TPF allows more work to be accomplished per hardware than most general purpose alternatives.

To benefit from z/TPF efficiency, it is important to understand its strengths and to ensure that your business applications fit the environment. z/TPF has many strengths, but at the core it is a very efficient environment for write intensive workloads such as those found in reservation systems. Good IT solutions come from picking the right tool (hardware, operating system, programming language, application architecture, and so forth) for the right problem to be solved. z/TPF is a specialized tool for challenging IT problems.

z/TPF can scale linearly to process over a million transactions per second. As we will discuss later, this fact alone is not enough to do a proper comparison to other processing architectures but makes the point of the possible scalability of z/TPF for selected workloads. There are other architectures that scale to this numerical claim. The devil is in the details of the processing and data model.

Costs

Finally, efficiency can contribute to reduced cost for the specific workload in question. A special purpose operating system, designed for a specific workload can provide a cost effective runtime environment for that workload. The fact that the runtime environment can be responsive and can scale easily can lead to a competitive cost per transaction when compared to general purpose alternatives.

Granted there is more to the cost of the solution that just the runtime environment, because there is a cost of maintaining the special purpose environment. There is also the cost of applications that are designed with the appropriate assumptions to achieve efficiency and scalability.

Beyond the hardware and software cost of a z/TPF environment, the cost of labor is often an area of discussion. It is difficult in the scope of this discussion to provide all the cost considerations when considering z/TPF workloads. It is especially challenging to make comments about labor costs in z/TPF environments, because every business and industry is unique. What we can provide here are some comments on the advances made by z/TPF in recent years as well as the challenges of the workload that z/TPF is designed to address:

- The z/TPF development environment was designed specifically to address concerns of the availability of skills for businesses today.

  z/TPF has made great progress in recent years in supporting the open development environment that is common today, primarily Linux environments. In prior releases of TPF, the preferred programming language was Assembler on System z. That environment has changed considerably over the last several years. z/TPF has changed to focus on C and C++, using the Eclipse development environment and open system compliers that are common with Linux to increase the availability of skills for z/TPF applications.

- z/TPF works to address the environment skills (languages, compliers, and so forth) available in businesses today.

  There still remains the challenge of domain knowledge for the workloads that z/TPF is designed to address. Reservations are a unique workload in the IT industry. The performance and scale of credit card authorization is a unique workload in the IT industry. In addition, skills are still a consideration with these workloads. This challenge is similar to other IT challenges, whether it be retail systems, manufacturing automation, super computing, or avionics. z/TPF provides the tools necessary to address unique workloads; however, domain expertise is still needed in designing solutions.

Availability

As previously stated, z/TPF is *purpose-built* and is efficient for the specific workloads for which it was architected, as is shown in the responsiveness, scalability, and reduced runtime costs when using applications built on z/TPF. z/TPF can also provide a highly available environment. Some of the high availability services are available to all applications, regardless of their design, and some services require the applications to be written to take advantage of the high availability services.

z/TPF is designed to be a 24x7x365 environment. Many environments claim high availability, but on closer examination require multiple systems that are clustered together with schemes of taking systems in and out of the cluster for maintenance. z/TPF is a high availability

environment in a single system implementation including clustered configurations. At first consideration, this design might not sound significant; however, z/TPF implementations have maintained availability for years. The availability functions of z/TPF are not limited to the operations of the runtime environment but also include change management.

For example, one of the most challenging computer science problems is how to introduce a software change on a running system. Software changes for applications and most system changes, except for a very few core functions, can be done without any scheduled downtime. One of the inherent advantages of a centralized server is that changes are managed in one place. Yes, this design does put a lot of responsibility on testing and planning a change, but there are advantages to this approach when compared with changes across dozens, hundreds, or thousands of servers.

Inherent data-duplexing

Inherent in z/TPF is the ability to *duplex data*, that is to write two instances of the data per every single write instruction from the application. This function serves two purposes:

- The duplicate data provides a high availability component to the data.
- z/TPF takes advantage of the dual data to enhance the performance and availability of the data.

z/TPF accesses the copy of the data that has the fastest access. If a primary record is lost, a duplicate copy of that record is always present. For performance reasons, both instances of the data are online in a normal environment and z/TPF reads from the instance that is most readily available to enhance the performance of the application reads.

Loosely coupled technology

While z/TPF on System z is used predominately in a scale-up fashion using multiple CPUs, to achieve higher levels of availability and scalability, multiple z/TPF systems can share a common set of DASD in a controlled fashion. This is called *loosely coupled* systems from a z/TPF perspective, where two or more systems share a common set of DASD and synchronize access to the DASD records.

Persist data to provide fast recovery

z/TPF has the ability to write extremely large volumes of data to DASD. This ability provides a capability for fast recovery times that other, more general purpose operating systems typically do not provide. Although z/TPF is designed to be highly available, accidents and failures (hardware or software) do occur. z/TPF can persist state changes of control blocks of in-flight transactions to allow the recovery of transactions in a very short amount of time, on the order of tens of seconds versus minutes or hours for general purpose systems. These recovery times are usually fast enough that users of the system do not notice them.

Processing characteristics well suited for z/TPF

z/TPF is a special purpose operating system and is not suited for every workload. The question is, what workloads are appropriate for the z/TPF environment? What are the characteristics to determine whether z/TPF is the right tool for the job?

Shared data with high update rate

z/TPF is efficient at writing and reading data records. This high performance feature makes it well suited for shared data (shared concurrently by multiple applications) that requires update access to the data. z/TPF can support additional updates per second to shared data than most general purpose operating systems, which is a significant reason why z/TPF is used commonly in the travel and transportation reservation industry with its ability to support the

writing of large quantities of data per unit of time. The efficiency of z/TPF to write data is critical to reservation systems.

Inter-related data

The term *inter-related data* describes data where there are dependencies between one data value and other data values that need to be kept closely linked. A classic example used in reservation systems in inventory data (the number of seats on an airplane available to sell) and passenger reservation records. Keeping the integrity of the inventory and the number of reservations is important. (Modern reservation systems do allow for over-booking, that is selling slightly more seats than physically available on the plane; however, at departure time the number of passengers cannot exceed the number of seats.) A strength of z/TPF is the ability to manage updates to large amounts of inter-related data. z/TPF provides flexibility on the types of data that can be inter-related and how transactions are managed—all while providing optimum performance.

High volume

What processing characteristics are well suited for z/TPF? Shared data (multiple, competing applications) with high update rates (write intensive workload with more writes than reads), inter-related data (multiple pieces of data needing to stay synchronized in some way), and the demand of high volumes of transactions (thousands to millions of request per second). z/TPF is designed to provide a high performance environment when all of these attributes exist.

Response time sensitive messaging

z/TPF provides memory caching capabilities that allow very efficient reads of data for some applications. When response time is critical for an application, z/TPF is key to satisfying demanding response time needs.

For example, credit card or debit card processing can be a time sensitive process. A customer is likely standing at register to make a purchase or has clicked the purchase button on a Web page triggering a credit card authorization. z/TPF can accept an authorization request and and process it in milliseconds. This can be attributed to a combination of high-bandwidth network connectivity, minimal path-length to process the request, and the ability to cache large amounts of data in memory and, if data is not cached, then the ability to read the needed data quickly.

z/TPF positioning with Extreme Transaction Processing

Extreme Transaction Processing (XTP) means different things to different individuals as one consistent definition does not exist. First, the term *extreme* is not easily quantified. Is there a particular number of transactions per second that qualifies as being extreme? One would argue that a million transactions per second qualifies as being extreme, but that alone is not a complete picture.

What is meant by a transaction? The word *transaction* is generally conceptually well understood but is a poor unit of measure for most data processing. For example, a transaction can be a simple read of a piece of data or can be a complex set of reads and writes (for example, credits and debits) between bank accounts. Saying that a system can do a million transactions a second—when those transactions are reading the same piece of data over and over—is very different from a million transactions a second of writes to multiple, and unique, banking accounts. When the word *transaction* is used, it is best to drill into the topic a bit further to understand what the activities are included in the scope of the transaction (reads and writes). Until there is a universal definition of the term *transaction*, it is wise to treat it as an important concept but not as an effective unit of measure of data processing work.

For some, XTP requires some number (hundreds or thousands) of networked systems (also known as *distributed systems*) performing some set of transaction workload. Some people require those systems to be considered “commodity” hardware, implying a particular (low) price point. One can argue that XTP should focus more on the results for a given workload than on a prerequisite infrastructure.

Partitionable and non-partitionable data

Data usage and organization matter when considering extreme transaction processing.

Partitionable data is data that is not dependent on, or interrelated to, other data in the environment. Partitioning data can be difficult in practice because all data in some way is related (sales are related to revenue, projects are related to staff, and so forth). What needs to be considered is the processing dependencies between data. If one piece of data is updated, is there a related additional piece of data that also requires change (for example, airline seat inventory and passenger reservations)?

If the data can be partitioned into subcategories and the application kept relatively simple, then very high transaction rates are achievable. Figure 4 shows an illustration of this environment. The term “simple” can be misleading. “Simple” here equates to a single function, reads can be cached, and data writes can have advantages due to there being a single owning application. Complexity starts to grow as soon as Application 1 needs access to Application 2’s data. If this complexity occurs, data locking and unlocking between applications needs to be considered.

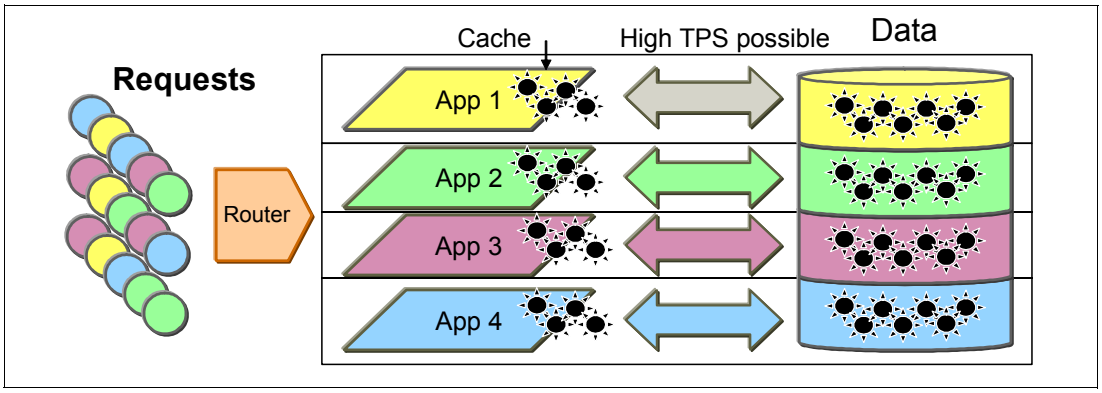


Figure 4 A simple application with partitioned data

The strength of z/TPF is when data cannot be partitioned, and very high rates of update are needed with very low response times, as illustrated in Figure 5.

Note: This figure shows a single system instance of z/TPF. System z can be loosely coupled for data and message or transaction sharing across multiple systems.

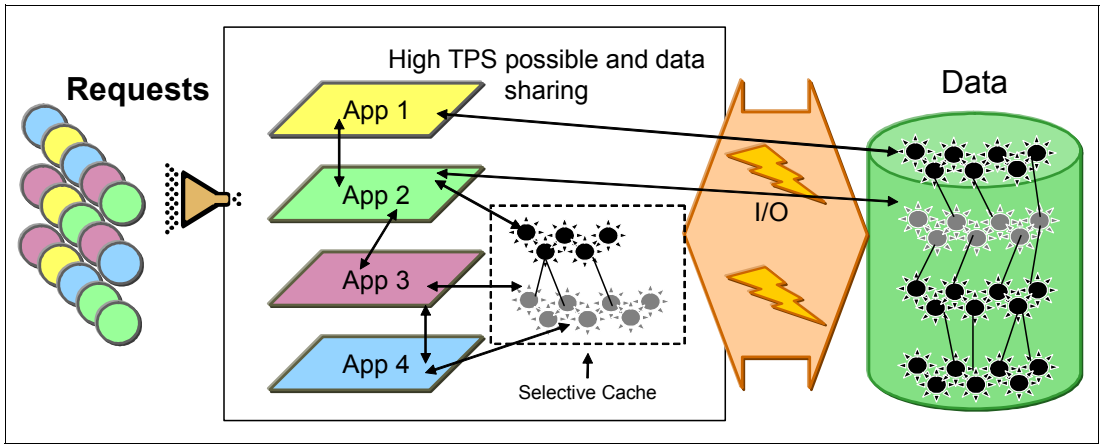


Figure 5 A more complex application where data is not partitioned

This environment is unique because all data is accessible by all applications. The data considered a contiguous data store with data sharing is allowed across applications.

General purpose operating systems can manage non-partitionable data as well, but what sets z/TPF apart is the ability to provide very high (extreme) update rates to the shared data. With fast response times for updates, shared data can be updated efficiently by z/TPF compared to other operating systems, because z/TPF requires relatively few number of instructions to execute an update.

In summary, extreme transaction rates can be achieved several ways. It is important to consider the data usage patterns and how best to achieve extreme transaction processing.

Centralized data

Data usage goes hand in hand with data placement. There are reasons why data should be centralized. It does not mean that data cannot be partitioned. It just means that there are advantages to having data centralized for some workloads.

One reason for centralized data is high update rates. If data is updated by the second, it is more effective and efficient to access the data where it resides than to try to make constant copies of the data to other servers. The laws of physics still apply to computer science, and to update a piece of data and copy that updated data to other servers takes time. This time and delay in replicating the data can create inconsistent views of the data across different applications

Other reasons for centralized data include:

- Ensuring security of data is easier if there is only one copy (excluding backup considerations).
- Having the data in one location can reduce the complexity to access the data if users who access the data are mobile.
- Reporting is easier with centralized data.

Brewer's CAP Theorem

Another way to discuss the advantages of centralized data is to look at the alternative of having distributed data. How to manage distributed data and maintain highly available system is a hotly debated topic in computer science.

Eric Brewer is a professor at the University of California at Berkeley. In 2000, he made a speech on the Principles of Distributed Computing that evolved into a theorem that carries his name. In short, what Brewer states is there are trade-offs to be made in any distributed processing scheme.

Brewer's CAP Theorem, states:

> "When designing distributed Web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three."

The word choices of *consistency*, *availability*, and *partition tolerance* might seem odd at first reading. Let us explain the words a bit more and try to illustrate the point that Eric Brewer is making:

- *Consistency* implies that all requesters of the data see the exact same data, which is of significance if there is more than one copy of the data. The goal is to always have data be identical between the two (or more) systems. Ideally, if the data is updated on one system then the copy (or copies) are updated instantaneously.[2]
- *Availability* addresses data availability as seen by the user of the services and data that is offered by a system. If you have two or more sets of data, the goal is to have that data available for both reading or updating at all times.
- *Partition tolerance* addresses the survivability of the environment to outages. Outages will always occur eventually (for example a communication failure in a network, such as a cable break).

So how does all of this information apply to our discussion? Brewer's CAP Theorem states that any distributed system can provide two of the three desired goals but not all three simultaneously, which is an inherent fact of distributed data processing environments.

For example, to provide data consistency for multiple systems, the systems need to be connected synchronously so that an update on one system is propagated to the other systems. However, this synchronous connection between systems is not conducive to high availability of the data. Data will likely be unavailable to applications during the window of data update across all the systems. Conversely, if data was made available at all times and updates were replicated asynchronously, there is a risk that an application on one system will read data locally that might have been updated on another system and will not get replicated. The data is available but not consistent across the environment.

An example of partition tolerance is a bit more involved. Simply stated, for partition tolerance, you need all data and functions to be redundant and spread throughout different networks and servers. The requirement of having redundant data is counter to the requirement of data consistency, because there is more than one copy of the data.

As the technologists in the industry deal with Brewer's CAP Theorem, be sure to consider whether centralized, contiguous data is the appropriate architecture for your environment, especially if you anticipate very high update rates to the data.

---

[2] The term instantaneously is the ideal. It is important to understand that networks, CPUs, memory reads and writes, accessing DASD all cause delays, however small, in updating data that resides in more than one place.

z/TPF key features and functions

Today's global business is demanding and complex. It requires vital, mission-critical data to move at split-second speeds, regardless of whether that data is moving down the street to a supply-chain partner or around the world among international corporations. In addition, extraordinary demands for around-the-clock operations require rock-solid speed, availability, reliability, resiliency, scalability, and performance.

z/TPF is an efficient and reliable processor of transactions. This function, however, is only one part of a solution required in any industry. The modern IT solution has to incorporate and integrate many functions and, therefore, multiple products to address the requirements of the business process.

Figure 6 shows the key z/TPF functions and features that will be examined in this section.

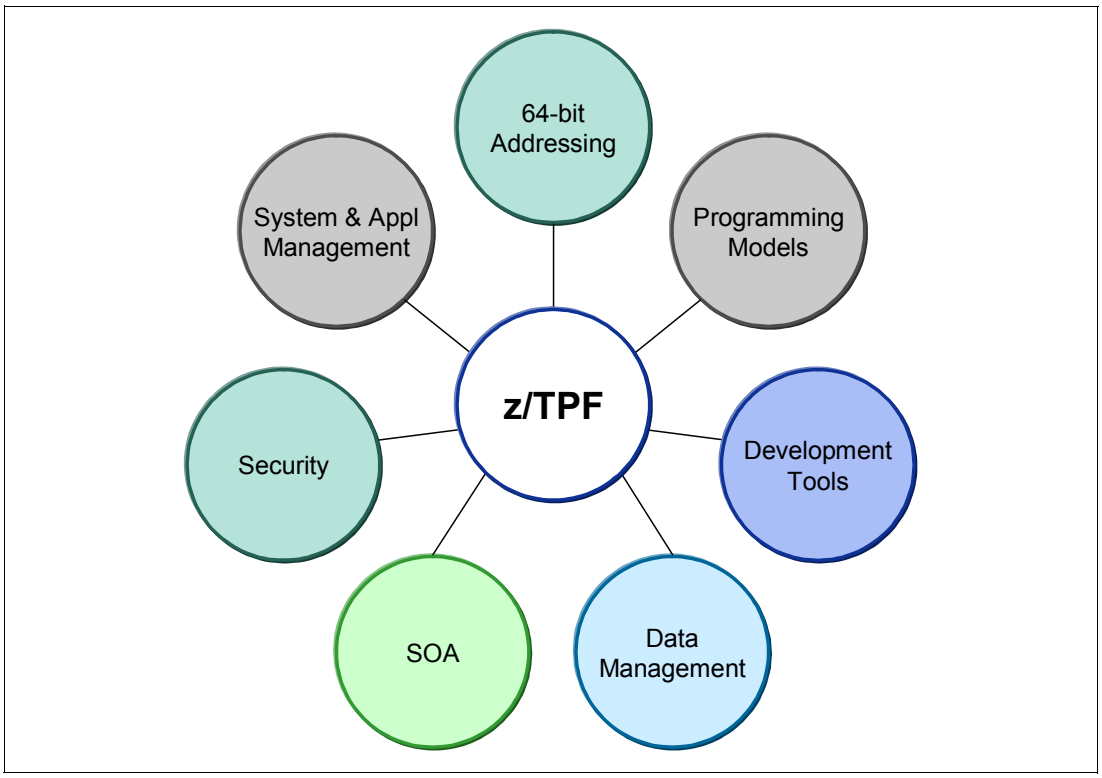


Figure 6 Key z/TPF functions and features

64-bit addressing

Traditional TPF transactions are short-lived and have a very small memory footprint. This dates back to original airline workloads of querying flight information, making reservations and related activities. Figure 7 shows a traditional TPF transaction.

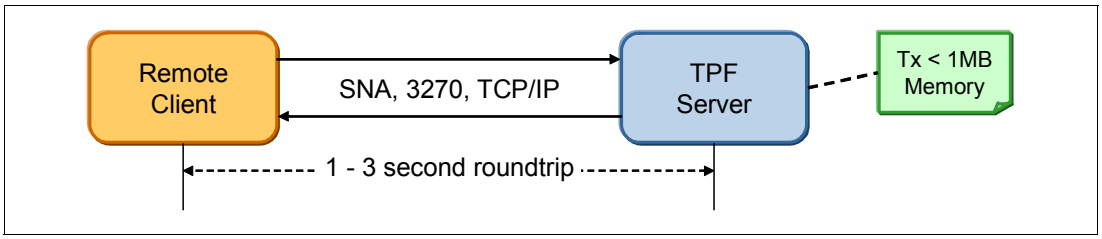


Figure 7 Typical TPF transaction

Newer technologies (such as SOA) lead to transactions that are longer-lived and have much larger memory requirements. With 64-bit addressing, z/TPF is a technology enabler to process these larger messages. Figure 8 shows a typical modern z/TPF transaction.

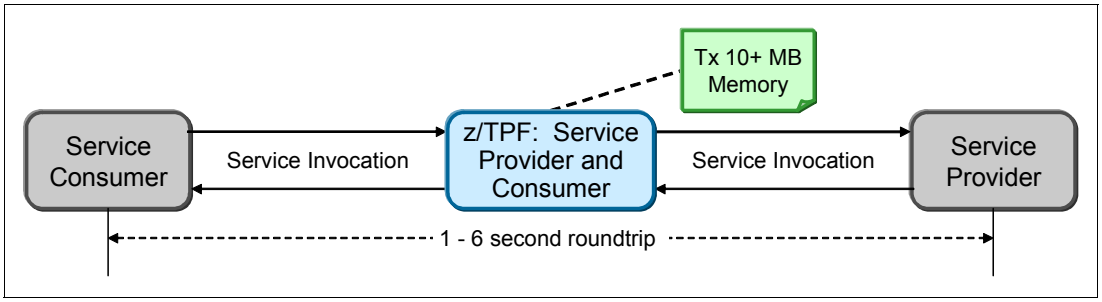


Figure 8 Typical z/TPF transaction

TPF4.1, the precursor to z/TPF, relied on 31-bit addressing, which limited its addressable storage to 2 GB. Now, with z/TPF, which relies on 64-bit addressing, the addressable storage limit is 16 EB (exabyte). Thus, the system has considerably more memory to scale and can process more work with greater efficiency.

Let us put the size of 64 bit addressing into perspective. If a 24-bit address space equates to 2.8 inches, then a 31-bit address space would equate to 30 feet. A 64-bit address space equates to over 48 million miles (the distance from Earth to Mars).

These generations of addressing architectures are what the System z platform and TPF has provided over the years. Your applications will also benefit from the increased memory. With this increased access to memory, applications can be enhanced and restructured to rely on more data being available in memory versus on disk. In a number of areas, z/TPF has exploited this increased access to memory to reduce or defer accesses to physical storage, be it DASD or tape, to provide increase levels of performance and availability, in a transparent way, to TPF applications and systems.

Programming models

Earlier releases of TPF supported C/C++ language programs but were designed and optimized for Basic Assembler Language (BAL), causing applications based in C/C++ to pay a performance penalty on TPF. This design is no longer the case with z/TPF, which was designed from the ground up to support both BAL and C/C++ language programs in a high performance fashion.

z/TPF supports an open, POSIX compliant runtime environment, which is beneficial for porting applications and utilities to the platform and for taking advantage of generally available software developer skills that are familiar with developing applications using POSIX-based standard libraries. This design provides improved performance for z/TPF and reduces the TCO of the platform with regard to the programming skills needed. However, this design does not eliminate the need for some degree of special skills for the unique workloads that are the design point of z/TPF, but it can help to reduce a majority of the specialization that is needed to develop and maintain applications for the platform.

Development tools

You can improve developer productivity with open tooling. The modern development environment of z/TPF enables you to use the following tools:

- Linux and open source tooling instead of z/OS® based tools (which was required in prior versions of TPF)
- GNU Compiler Collection (GCC)
- Seamless assembler language and C/C++ program interaction

The TPF Toolkit (packaged separately from z/TPF and z/TPFDF) is a complete, extensible Eclipse-based integrated development environment (IDE) that enables programmers to effortlessly code, build, and test in one tool. In addition, using the Toolkit helps attract and retain new talent because the Toolkit is Windows®-based and easy to use.

In z/TPF, an improved and integrated source-level debugger is available for assembler language and C/C++ source-level debugging.

A number of tools are available for creating better applications on z/TPF. With these tools, developers can achieve faster and easier problem determination in the following key areas:

- Core corruption detection
- Resource utilization understanding
- Core dump enhancements
- Program display and tracing
- Enhanced program control

Data management

There is no single database option that is right for all types of data. You must select the correct data format and access method based on the how the data will be used, which is based on the following conditions:

- Access frequency
- Users of data
- z/TPF versus remote platforms
- Ported code versus user-written code
- Distribution of data
- Functionality required

Service Data Objects

z/TPFDF and Service Data Objects (SDO) are a new model of data access that is complementary technology for SOA, developed jointly by IBM and BEA. The benefits include:

- Convenient and generic way to access data on a remote platform
- Universal model for business data

- Common unifying format for exchanging data between services
- Includes dynamic interfaces
- Not tied to the data organization, like SQL to relational databases
- Object-oriented, thus maintenance is easier

Figure 9 shows the SDO architectural diagram.

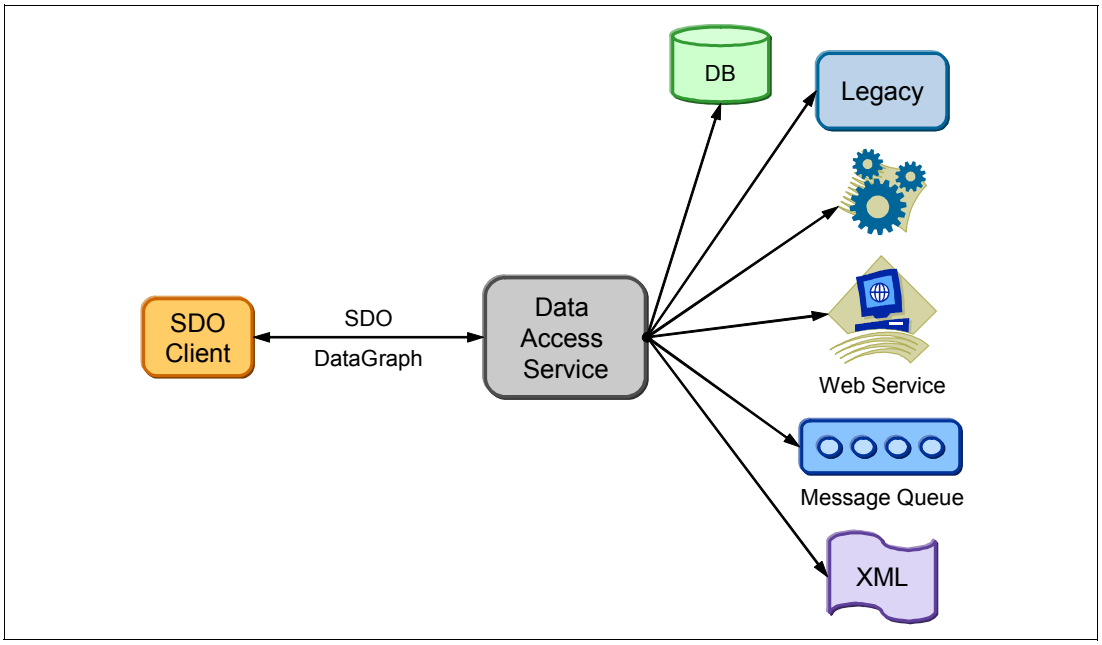


Figure 9 SOA architectural diagram

Externalizing data

Often, z/TPF systems are the system of record for critical business data. Data owned by z/TPF might be required for use by remote platforms (such as data warehousing and analysis). Several options exist to make data externally viewable to other systems in your enterprise:

- Direct access via SDO z/TPFDF.
- Data replication or transformation performed off the z/TPF platform.
- Data replication or transformation performed on z/TPF using MySQL and a relational database.

SOA

SOA is becoming increasingly the industry standard for building robust, flexible IT infrastructure, because developers can use this model to create applications in a way that is independent of a particular network protocol or operating system. Breaking a complex business function into smaller, individual services creates a more-flexible application that can react quickly to changing market conditions. The idea is to create individual services that provide a well-defined function and are designed with reusability and ease of use in mind.

The primary SOA support with z/TPF is through standards-based Web services technologies (such as XML, SOAP, WSDL, UDDI, and WS-I). Through this support, z/TPF-based services and data can be accessed from other platforms using these standard technologies. Likewise, z/TPF-based applications can access services and data on other platforms using these standard technologies.

The Web service provider support in z/TPF is a transport-independent mechanism for processing Web service requests using the SOAP standard that provides the following benefits:

- Relies on a highly efficient XML scanner for parsing SOAP requests
- Exists to implement Web service extensions
- Separates business logic from invocation method with the componentized processing model
- Provides Web service provider tooling

The Web service consumer support consists of a set of z/TPF unique APIs that can be used by applications to send SOAP-based Web service requests to service providers easily. The characteristics of this support include:

- Support for multiple transport mechanisms (HTTP client, WebSphere MQ, and so forth)
- The ability for applications to set transport-specific options (and override default options)
- Web service consumer tooling

The goal of both the Web service consumer and provider tooling is to make the development, deployment, and management of these services on z/TPF, in both procedures and required skills, similar to those needed on other platforms. This tooling is included in the IBM TPF Toolkit, which helps simplify the steps required to make use of Web services on z/TPF.

Opening existing applications to a new user interface using Web services enables you to modernize core business systems as well as adopt an SOA approach to the overall IT infrastructure. Using Web services on z/TPF results in improved usability, greater flexibility, and faster time-to-market for applications.

The z/TPF SOA Value Enhancement allows you to use workload charging (WLC) to cost effectively enable SOA capabilities running on System z hardware. This enhancement removes the monthly charges for the software Million Service Units (MSUs) that are associated with some SOA functions.

Note: z/TPF SOA Value Enhancement helps z/TPF users move to SOA by removing the “technology penalty” of implementing modern business solutions

Security

Data security is paramount in an electronic world. Data in flight, at rest, and in use in memory needs protection from unauthorized parties. For example, a main target of electronic data thieves is credit card numbers. Without data security during Internet credit card transactions, credit card numbers could be easily stolen.

A driving force behind the security approach for z/TPF is PCI (Payment Card Industry) compliance, which is a set of requirements designed to ensure that all companies that process, store, or transmit credit card information maintain a secure environment.

Data *in flight* is sensitive data that is encrypted when transmitted over a network. Data *in use* is sensitive data that needs to be secure when it is being used by applications, and data *at rest* is sensitive data that is encrypted when saved on DASD or on tape.

z/TPF uses many methods to protect data, centering on the following principals:

- *Data encryption*: Converts data into a form that only the sender and intended recipient can understand; the data is unintelligible to unauthorized parties.
- *Data integrity*: Ensures that data has not been altered.
- *Authentication*: Establishes that you are who you claim to be; that you are authentic.

The key security concepts involve:

- Cipher algorithms and keys
- Symmetric cryptography
- Public key cryptography
- Data integrity and message digests
- Digital signatures

By adopting these approaches, z/TPF security can play a vital part of your organization's enterprise security strategy.

System and application management

Several tools exist to help analyze the performance and health of your z/TPF system. Each tool is targeted at providing specific types of information for analysis:

- *z/TPF Data Collection Reports*: Provides information about system performance during a defined collection interval. Useful for identifying trends on overall system performance and for gathering data required to tune various system configuration settings.
- *IBM Tivoli® Monitoring for z/TPF*: Provides real-time monitoring of system resource utilization as well as historical querying capabilities and integrates with other IBM Tivoli Monitoring solutions in your enterprise. Useful for coverage staff to identify potential problems before they occur and to monitor the general health of the z/TPF system during continual operation.
- *z/TPF Software Profiler*: Provides a means for gaining a deeper understanding of the runtime characteristics for a particular application or system function. You can use the information gathered to modify and tailor the performance of specific parts of the z/TPF system.

Tools are available in z/TPF that enable your group to gain insights into what applications are doing. Developers can understand how applications are functioning while performing problem determination and resource monitoring, which can help analysts, developers, architects, and operations teams cut through the complexity of applications and interdependencies so that they are more responsive and make changes with less risk. These tools include:

- *Trace Log Facility*: Provides the capability to trace application function and macro calls to a file or real-time tape so that application debugging is not disruptive.
- *New OWNER Parameter*: Identify the culprit who might be abusing system resources, resulting in improved resource tracking and problem determination, which might improve system stability.
- *Dump Enhancements*: Ability to group programs in dump groups to make it easier to define overrides for programs that require the common areas to be dumped, which allows developers to associate dump overrides on an application-wide basis instead of having to define an override for each program.
- *Integrated Performance Analyzer*: The Performance Analyzer client available in the IBM TPF Toolkit supports automatic downloading of trace files from z/TPF, loading them

directly into an easy-to-use interface where you can profile applications and better understand how programs are functioning.

IBM is continually working to take advantage of new technologies that IBM has built or acquired for use with z/TPF. For example, IBM offers tools that can analyze source code written in languages that z/TPF supports, presenting an opportunity to apply such tools to z/TPF applications.

Summary

TPF evolves continually to meet the demanding and ever changing needs of customers, as it has done for decades. With z/TPF Enterprise Edition V1.1, IBM continues this commitment by providing a modern, scalable, highly available platform that can fully participate in heterogeneous enterprise architecture. z/TPF provides the platform that is necessary to protect and extend the value of TPF-based application and data assets by providing a flexible, responsive, and cost effective platform to help IT executives meet the changing needs of the business.

Other resources for more information

For more information, consult the following resources:

- IBM z/Transaction Processing Facility

  http://www.ibm.com/software/htp/tpf/
- The New Voice of the CIO

  http://www.ibm.com/services/us/cio/ciostudy/
- Gartner Hype Cycles

  http://www.gartner.com/it/products/hc/hc.jsp
- *The IBM Mainframe Today: System z Strengths and Values*, REDP-4521

  http://www.redbooks.ibm.com/abstracts/redp4521.html?Open
- IBM Component Business Model™

  http://www.ibm.com/services/uk/igs/html/cbm-bizmodel.html
- IBM Industry Frameworks information

  http://www-304.ibm.com/jct01005c/isv/tech/validation/framework/

The team who wrote this guide

This guide was produced by a team of specialists from around the world working with the International Technical Support Organization (ITSO).

Bruce Armstrong is a Strategy and Design Manager in IBM Software Group. He is based in Research Triangle Park near Raleigh, North Carolina. He has been leading a 2 year project on z/TPF architecture and design plus the study of Travel and Transportation reservation systems. He is a frequent speaker on IBM networking technology and strategy, especially for z/OS environments. Bruce has 30 years in the IT industry ranging from being an IBM customer, IBM consultant, sales, and product development.

Barry Baker is a Senior Engineer and Manager in the TPF Systems Lab in IBM Software Group. He has 10 years of experience working on the development of the TPF family of

products and is currently focused on TPF Support, Strategy, Design, and Test. He frequently meets with the C-level and LOB executives of the TPF customer set to ensure that TPF continues to meet their unique and demanding transaction processing needs.

Jonathan Collins is a Product Line Manager for the TPF, z/TPF, and ALCS products in IBM Software Group. He is responsible for financial management, acquisitions, pricing, marketing, new business opportunities, partnerships, strategy, and product direction. Jonathan has over 9 years of experience working in the TPF organization, ranging from product development, executive customer interfacing, and project management.

Thanks to the following people for their contributions to this guide:

- Martin Keen, Consulting IT Specialist, IBM ITSO

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

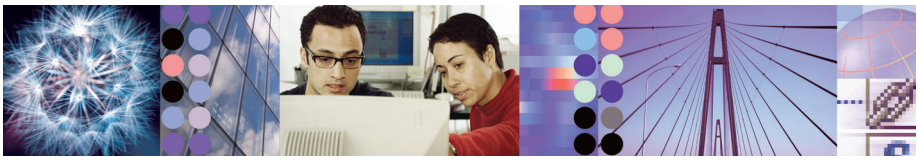
Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Contact an IBM Software Services Sales Specialist



This document, REDP-4611-00, was created or updated on January 26, 2010.

Trademarks