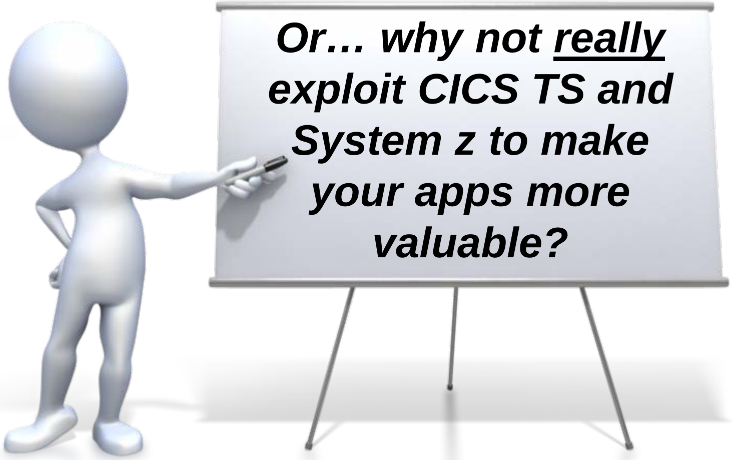
#SHAREorg

SHARE
Technology · Connections · Results

CICS Integration & Optimization:
Tales from the Trenches



Or… why not really exploit CICS TS and System z to make your apps more valuable?

Russ Teubner, CEO
HostBridge Technology

March 12, 2014
Session 14819

SHARE in Anaheim

Abstract

CICS users are loyal to their apps – and for good reason! However, they also need to <u>integrate</u> these same applications with an ever widening array of <u>web, cloud and mobile resources</u>. If that weren't enough, every year they are under pressure to support <u>new workload</u> and <u>reduce the cost of ownership</u>. That's a tall order.

Fortunately, IBM continues to deliver new versions of CICS that focus on operational efficiency and service agility. ISVs like HostBridge build upon these capabilities to help customers save time, reduce cost or generate revenue.

This presentation highlights tactics and strategies that customers are using to enhance the value of their existing CICS investments (and lower their cost).

HostBridge in Brief

Precision integration for CICS

- Founded in 2000 to invent a new breed of integration software by exploiting CICS TS
- Driven by customer requirements
- Objective: save time, cut costs, generate revenue
- ***Do the hard stuff***

Serving large organizations worldwide

- NISSAN, AEGON, Navy Federal Credit Union, Wells Fargo, Edward Jones, Harland Clarke, PACCAR, Aegon UK, State of AZ, NYC Department of Education, City/County of San Francisco, Los Angeles County

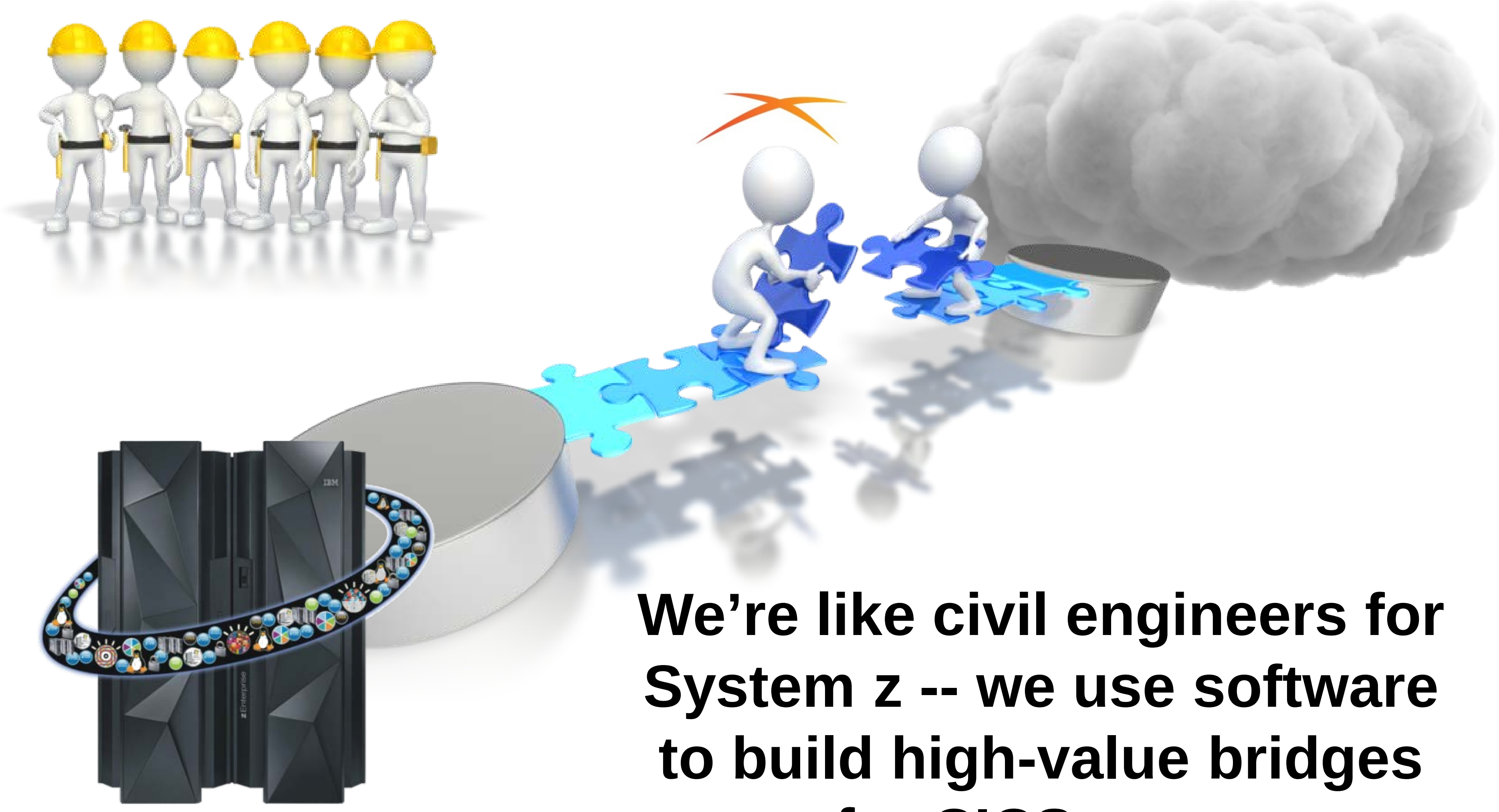
Strong technology partnerships

- Strong working relationships with IBM System z, zOS, LE and CICS product groups

Business Partner IBM

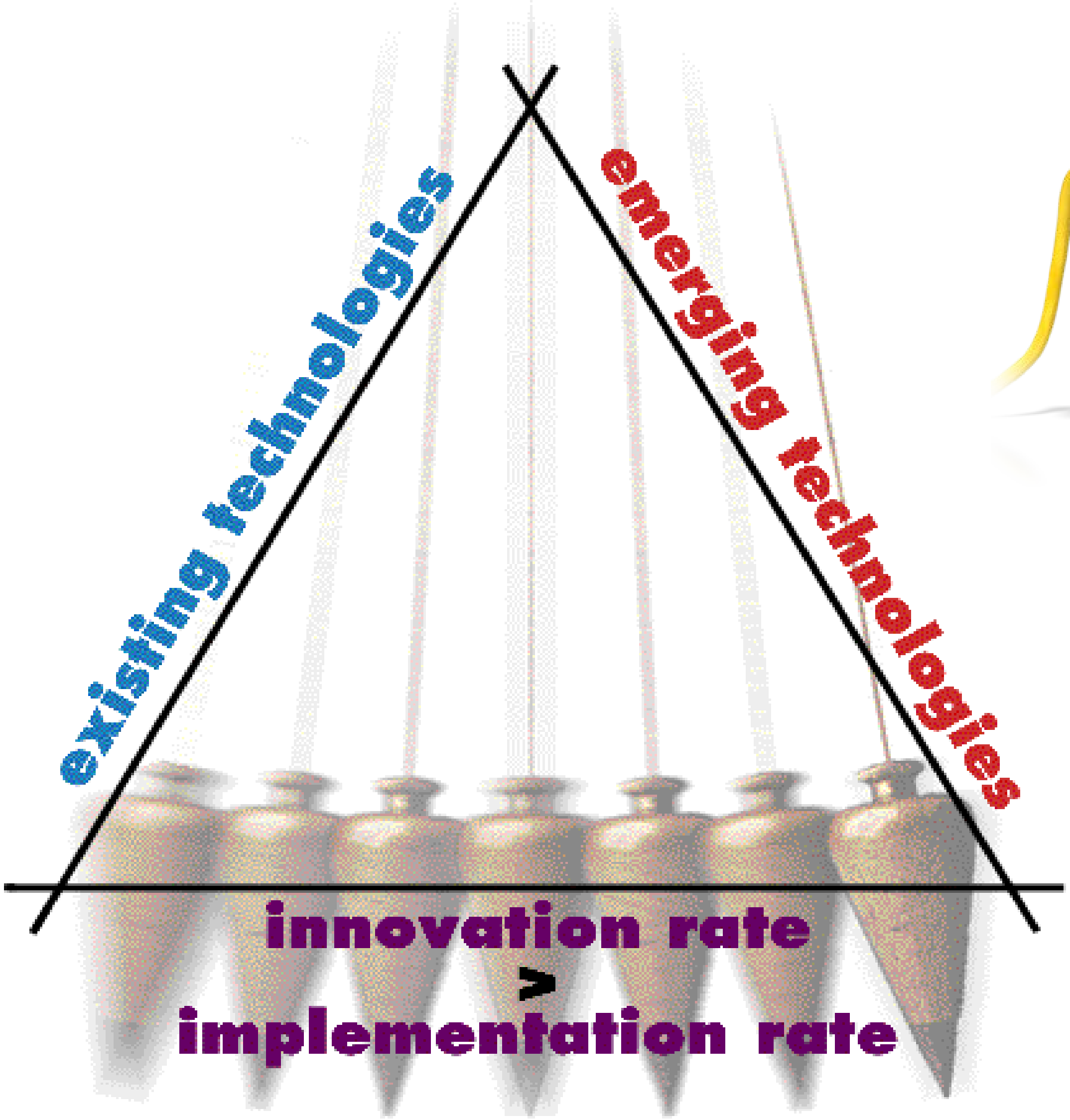
Ready for
Rational software

Stated Differently…

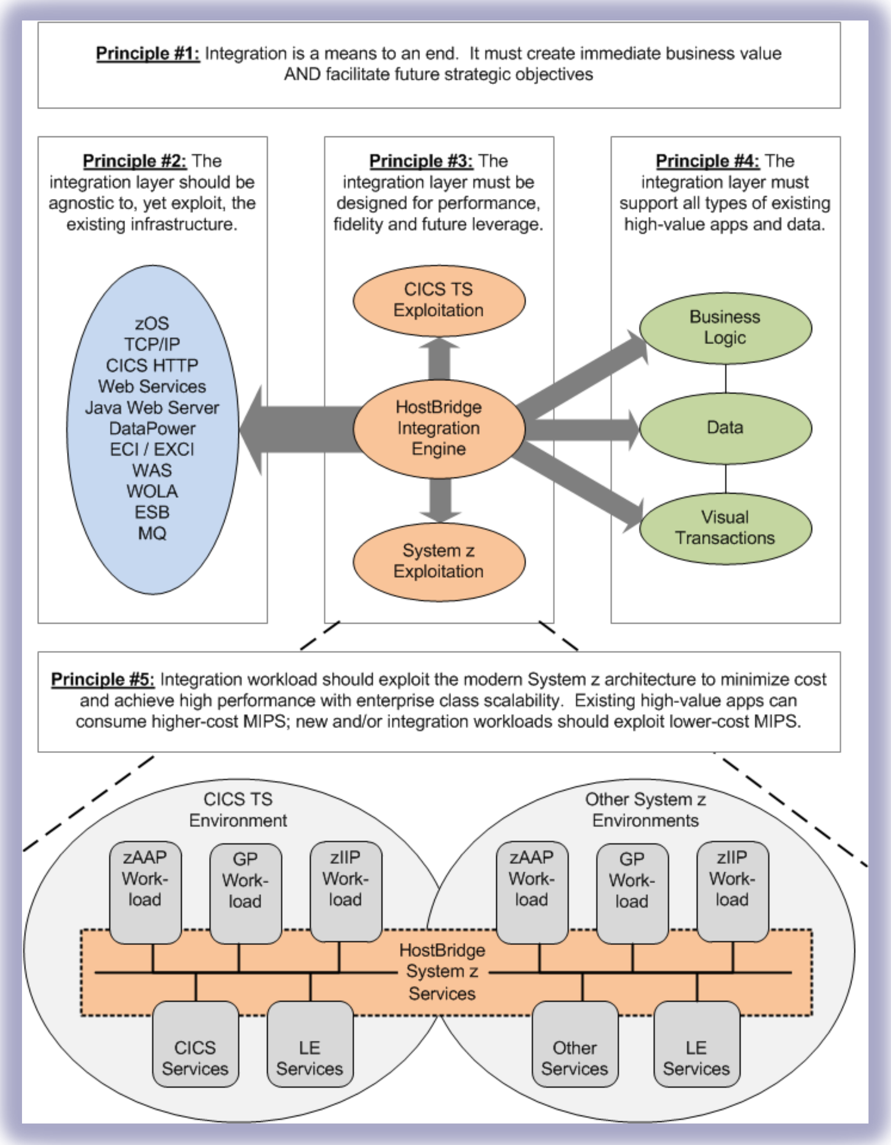


We're like civil engineers for System z -- we use software to build high-value bridges for CICS apps.

What Drives Integration?



Integration Principles



Principle #1: Integration is a means to an end. It must create immediate business value AND facilitate future strategic objectives

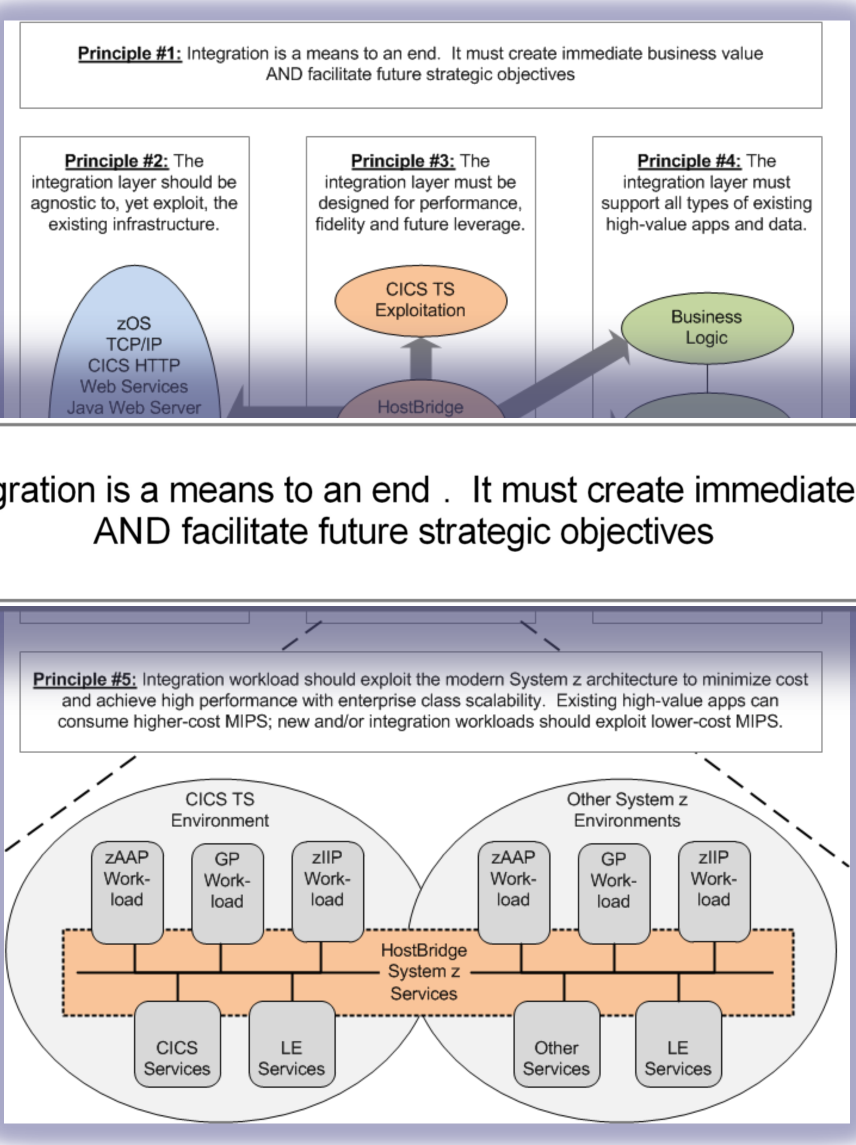
Principle #2: The integration layer should be agnostic to, yet exploit, the existing infrastructure.

Principle #3: The integration layer must be designed for performance, fidelity and future leverage.

Principle #4: The integration layer must support all types of existing high-value apps and data.

Principle #5: Integration workload should exploit the modern System z architecture to minimize cost and achieve high performance with enterprise class scalability. Existing high-value apps can consume higher-cost MIPS; new and/or integration workloads should exploit lower-cost MIPS.

Integration Principles



Principle #1: Integration is a means to an end. It must create immediate business value AND facilitate future strategic objectives

Principle #2: The integration layer should be agnostic to, yet exploit, the existing infrastructure.

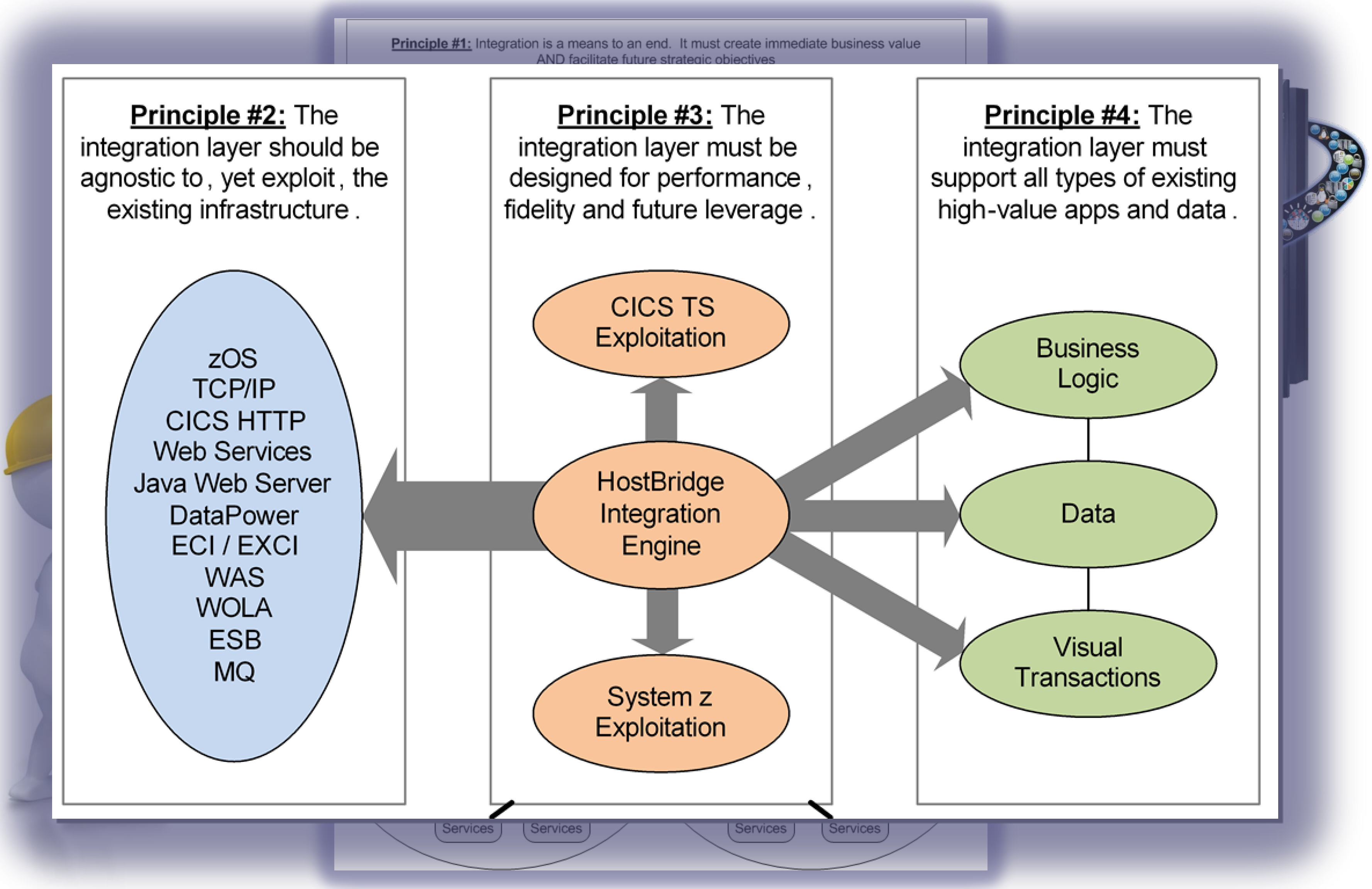
Principle #3: The integration layer must be designed for performance, fidelity and future leverage.

Principle #4: The integration layer must support all types of existing high-value apps and data.

Principle #5: Integration workload should exploit the modern System z architecture to minimize cost and achieve high performance with enterprise class scalability. Existing high-value apps can consume higher-cost MIPS; new and/or integration workloads should exploit lower-cost MIPS.

Principle #1: Integration is a means to an end . It must create immediate business value AND facilitate future strategic objectives

Integration Principles



Principle #1: Integration is a means to an end. It must create immediate business value AND facilitate future strategic objectives

Principle #2: The integration layer should be agnostic to, yet exploit, the existing infrastructure.

Principle #3: The integration layer must be designed for performance, fidelity and future leverage.

Principle #4: The integration layer must support all types of existing high-value apps and data.

zOS
TCP/IP
CICS HTTP
Web Services
Java Web Server
DataPower
ECI / EXCI
WAS
WOLA
ESB
MQ

CICS TS Exploitation

HostBridge Integration Engine

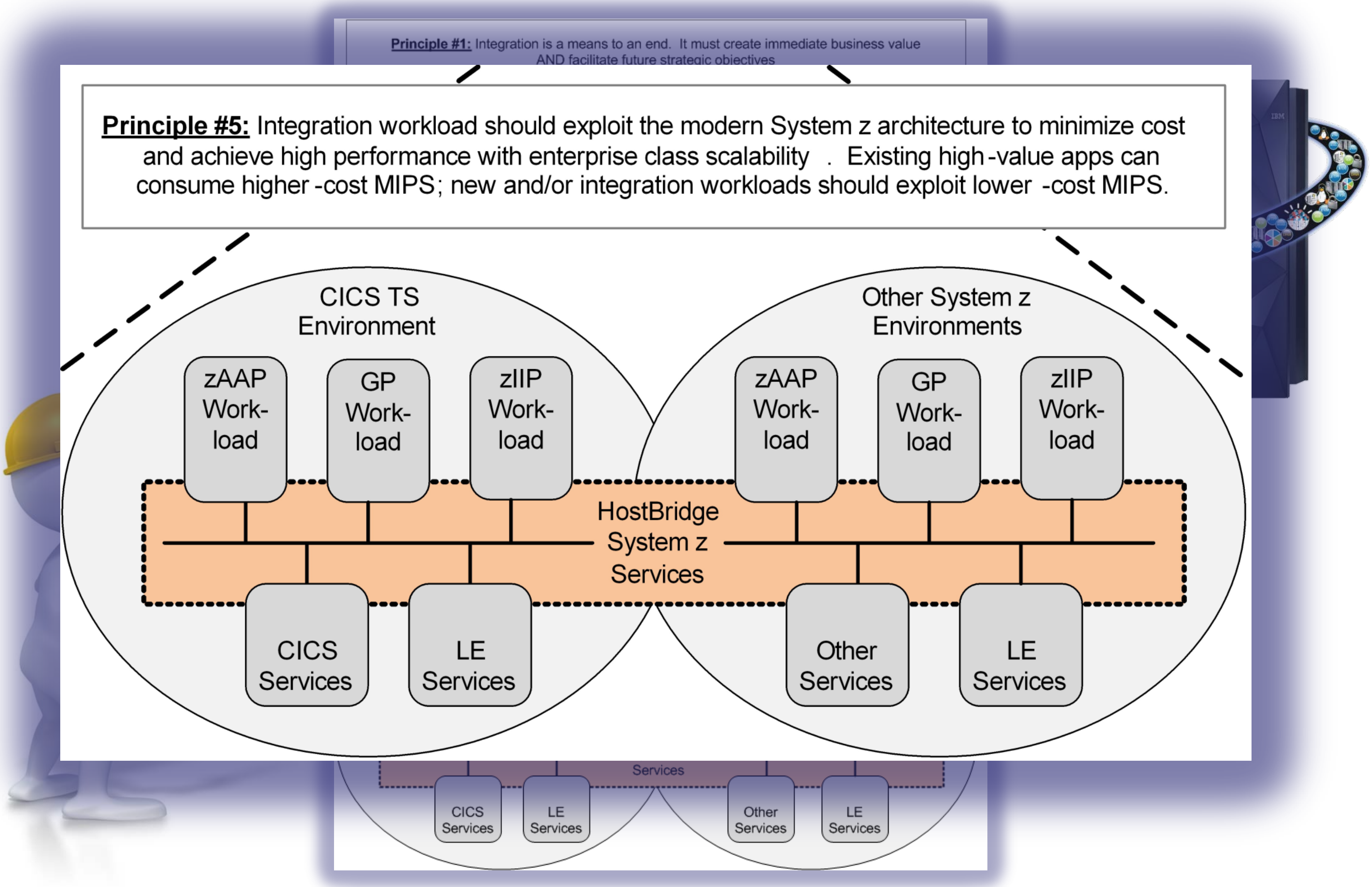
System z Exploitation

Business Logic

Data

Visual Transactions

Integration Principles



Principle #1: Integration is a means to an end. It must create immediate business value AND facilitate future strategic objectives

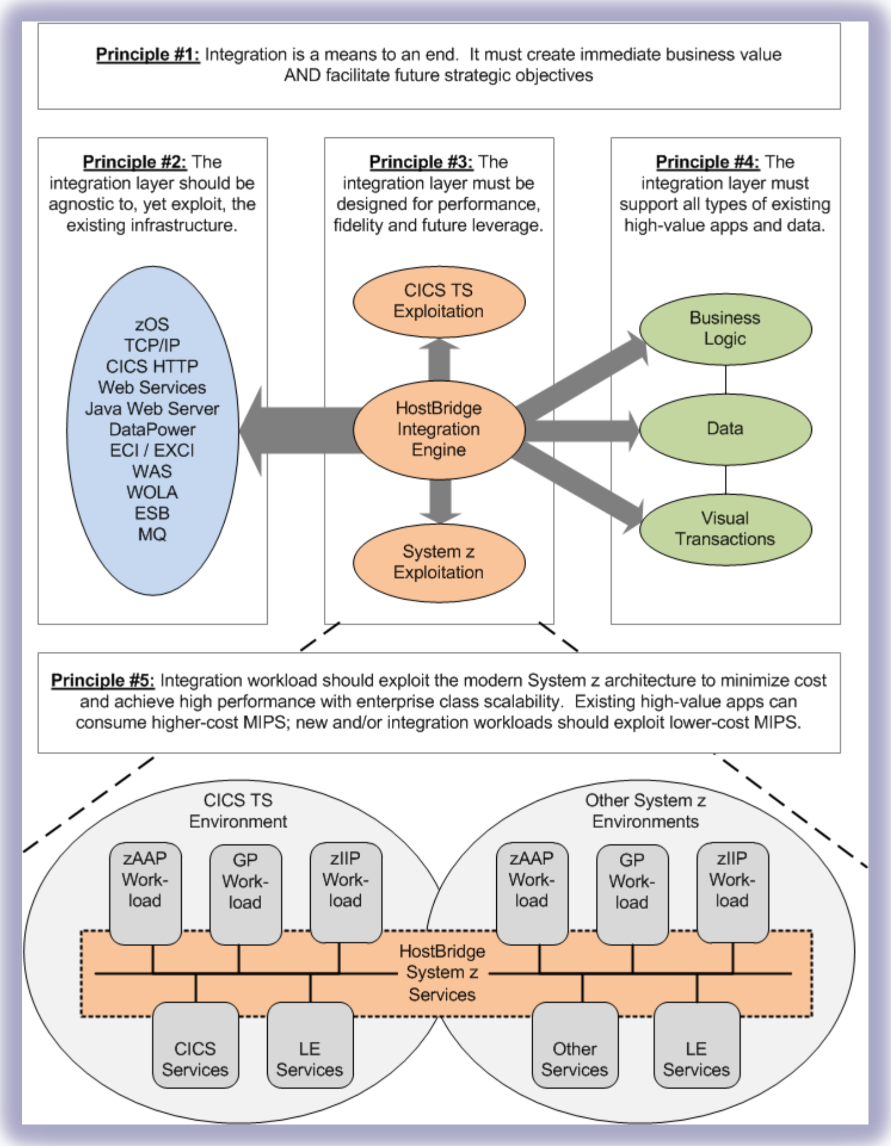
Principle #5: Integration workload should exploit the modern System z architecture to minimize cost and achieve high performance with enterprise class scalability. Existing high-value apps can consume higher-cost MIPS; new and/or integration workloads should exploit lower-cost MIPS.

Thus… Our Perspective

Whether your business objective is to reduce costs or generate revenue,

<u>HOW</u>

you approach integration matters.



<u>Principle #1:</u> Integration is a means to an end. It must create immediate business value AND facilitate future strategic objectives

<u>Principle #2:</u> The integration layer should be agnostic to, yet exploit, the existing infrastructure.

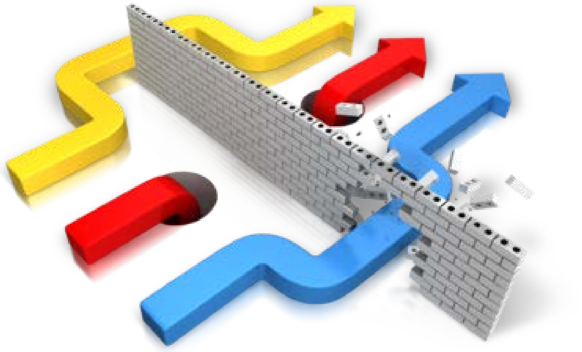
<u>Principle #3:</u> The integration layer must be designed for performance, fidelity and future leverage.

<u>Principle #4:</u> The integration layer must support all types of existing high-value apps and data.

<u>Principle #5:</u> Integration workload should exploit the modern System z architecture to minimize cost and achieve high performance with enterprise class scalability. Existing high-value apps can consume higher-cost MIPS; new and/or integration workloads should exploit lower-cost MIPS.

Tactics and Strategies

- **Expand Specialty Engine Usage**
  - **Only high-value apps should be running on GPs**
  - **Everything else should be on zIIP/zAAP**
- **Expose Flexible Service Interfaces**
  - **HTTP, REST, SOAP, WSDL (don’t get religious)**
- **Express Data/Content Efficiently**
  - **XML and/or JSON (don’t get locked in)**



- **Eliminate Overhead**
  - **There’s usually plenty if you look closely**
- **Exploit CICS-based Scripting**
  - **Ideal way to rapidly build and deploy robust CICS service**

Dynamic Scripting with JavaScript

- **By 2004 it was clear that our customers needed something to orchestrate CICS transactions into high performance services**
  - **It had to be suitable for automating “micro flows”**
  - **It had to scale and perform at extreme levels**
- **We settled on server side JavaScript**
  - **Industry standard JavaScript engine (c/c++)**
  - **Open source code base**
  - **Ported to run on System z and inside CICS**
- **Significant advantages**
  - **Full fledged robust programming language**
  - **Object oriented and easily extended**
  - **Can interact with any CICS resource**
  - **Client side web developers can become server side developers with ease**
  - **Millions of code examples on Internet**
- **Native support for JSON and other web-centric service and data architectures**

CICS TS and HostBridge

- **CICS TS and HostBridge share common design objectives…**
  - **Improve Operational Efficiency**
    - Greater capacity
    - Managed operations
    - Increased availability
    - Deeper insight
  - **Enhance Service Agility**
    - First-class applications
    - First-class platforms
    - Modern interfaces
    - Foundational enhancements
- **Next up: Case studies of recent (and unique) projects to illustrate these two aspects**

Operational Efficiency

- **Lowered cost of ownership**
- **Greater capacity**
- **Increased availability**
- **Managed operations**

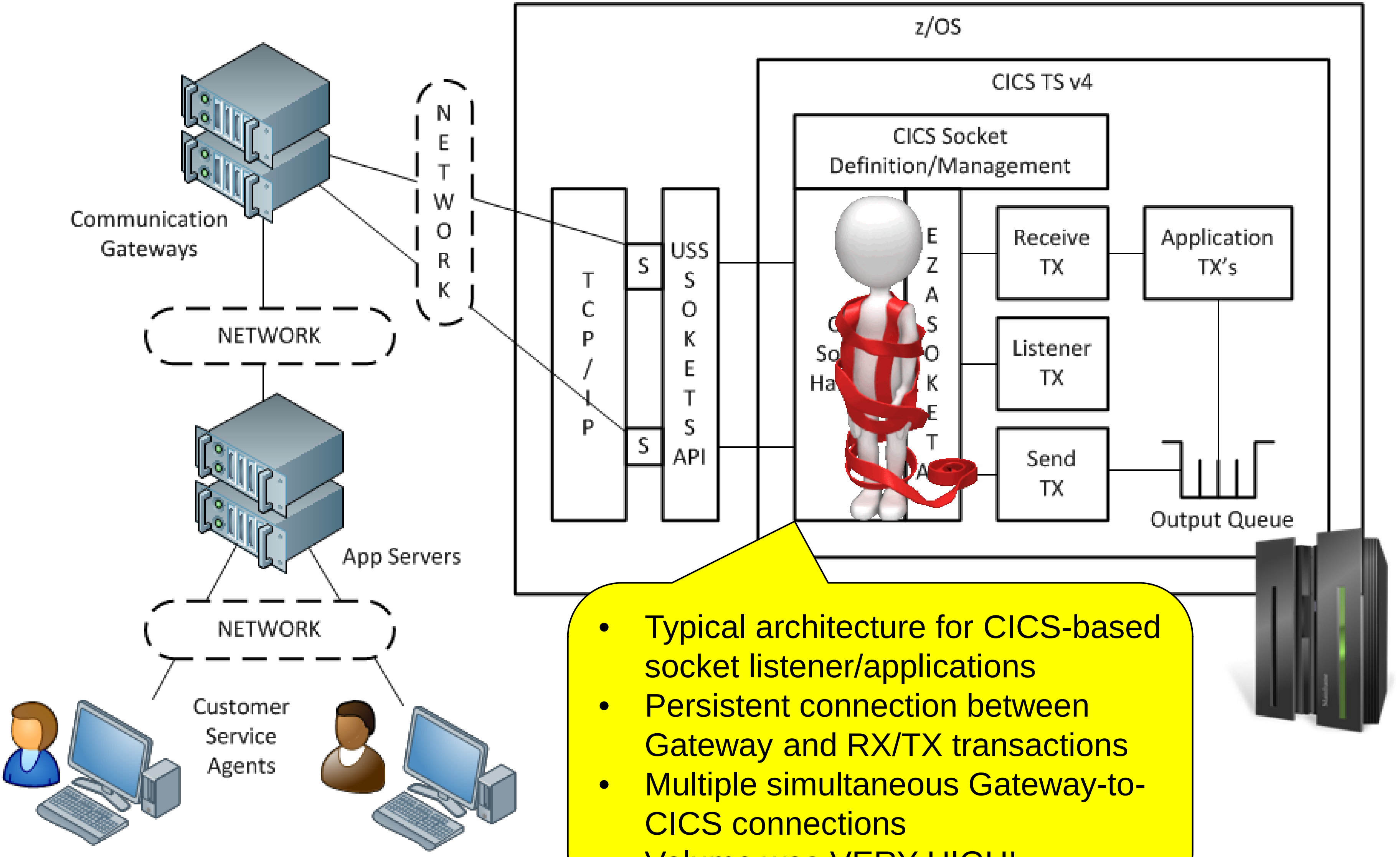
A Tale of Two Customers

- **Customer A**
  - **Industry: Telecommunications (US)**
  - **Very high daily/consistent transaction volume**
  - **Long-standing investment in COBOL-based socket apps**
- **Customer B**
  - **Industry: Financial Services (International)**
  - **Very high transaction volume on one day each month (and in compressed time period)**
  - **Long-standing investment in PL/I-based socket apps**

Common Objectives

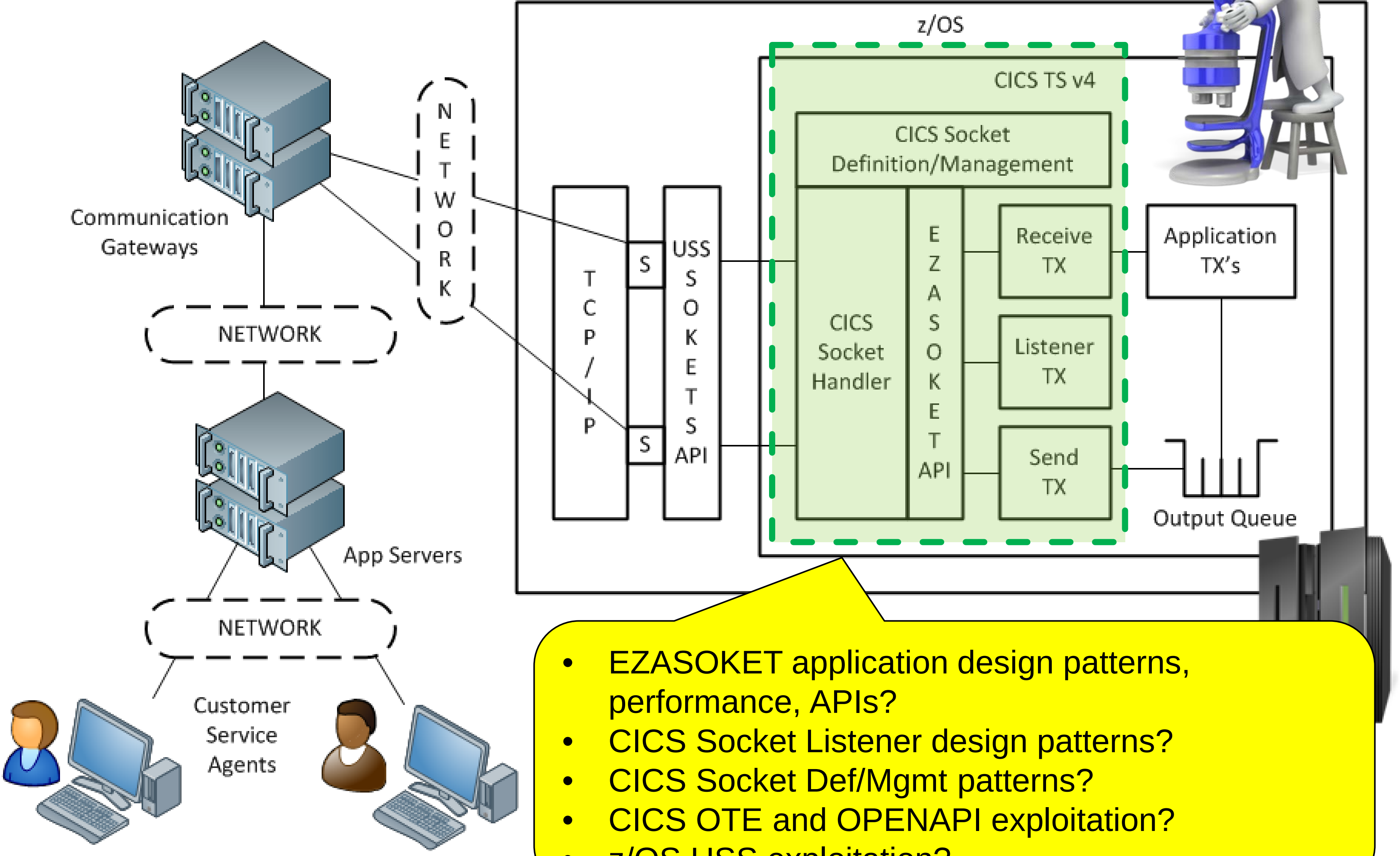
- Both customers had common objectives
- Business Objectives
  - Respond to competitive pressures in their industry
  - Lower incremental cost of high-volume CICS application processing (i.e., marginal value > marginal cost)
  - Move new/additional workload to System z and reinforce CICS TS as the most cost effective platform for their business
- Technical Objective (at least their hope)
  - Streamline System z and CICS integration paths
  - Reduce the CPU burn (GP) associated with socket applications and infrastructure
  - “Make the plumbing less expensive”

Initial Conditions



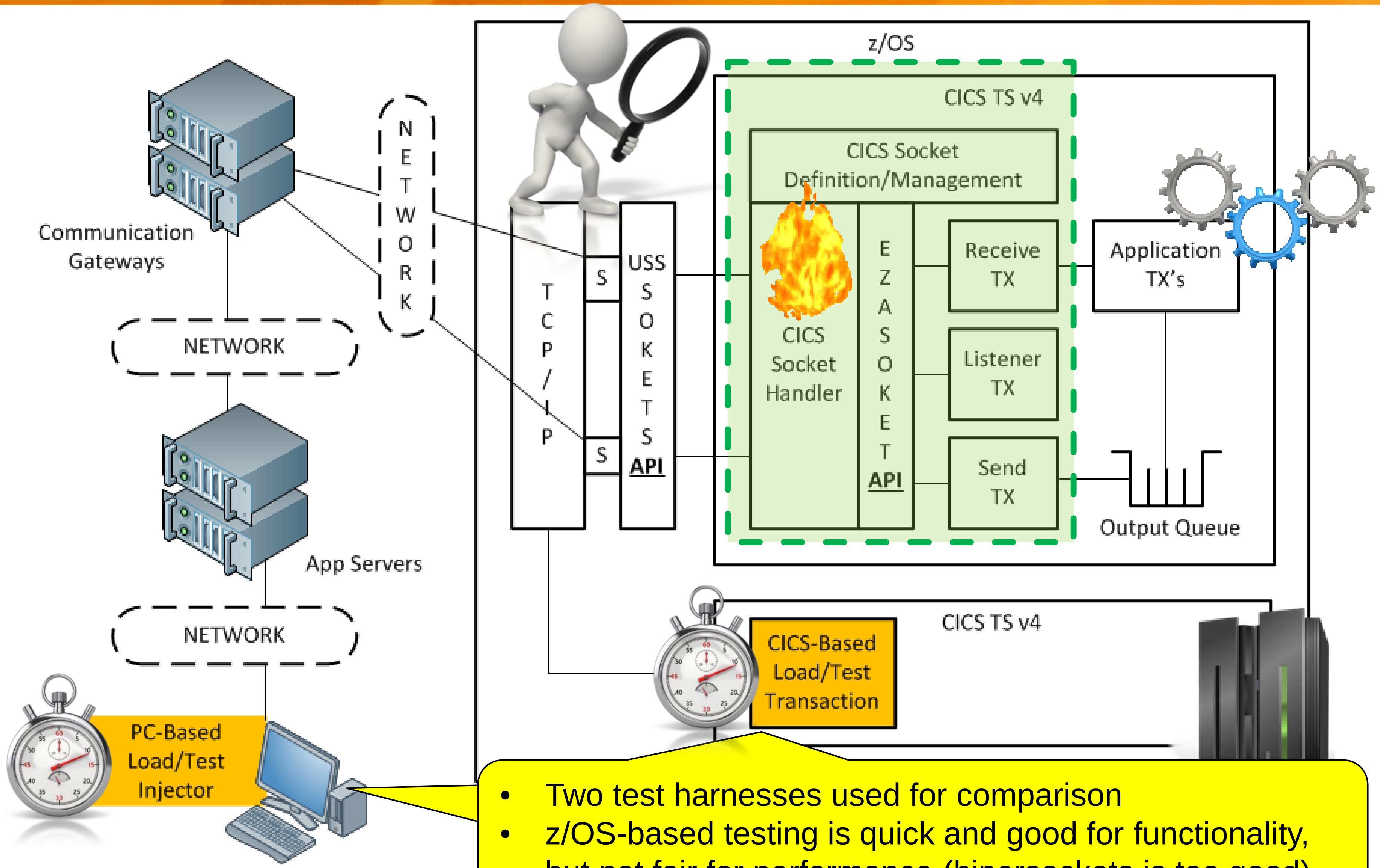
- Typical architecture for CICS-based socket listener/applications
- Persistent connection between Gateway and RX/TX transactions
- Multiple simultaneous Gateway-to-CICS connections
- Volume was VERY HIGH!

So We Examined This …



- EZASOKET application design patterns, performance, APIs?
- CICS Socket Listener design patterns?
- CICS Socket Def/Mgmt patterns?
- CICS OTE and OPENAPI exploitation?
- z/OS USS exploitation?

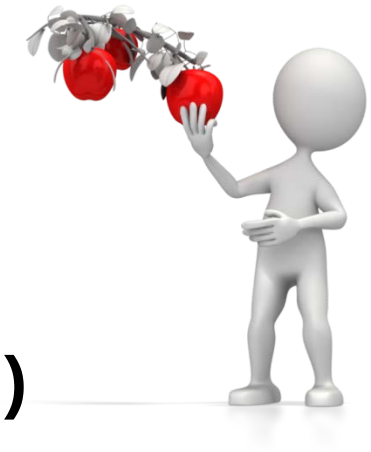
By Doing a Lot of This …



- Two test harnesses used for comparison
- z/OS-based testing is quick and good for functionality, but not fair for performance (hipersockets is too good)

Where the Data Led Us

- **Under volume testing, the CPU burn associated with the CICS Sockets Support was measurable and linear (confirmed customer’s theory)**
- **I won’t characterize it as “high” or “low” because the only thing that mattered was whether it could be lower (or not so linear)**
- **Thus, we began to:**
  - **Isolate various components and their impact**
  - **Consider how to provide alternative functionality (but complimentary to CICS TS)**
- **Low hanging fruit seemed to be CICS Socket Handler (via EZASOKET API)**



“CICS Socket Support”

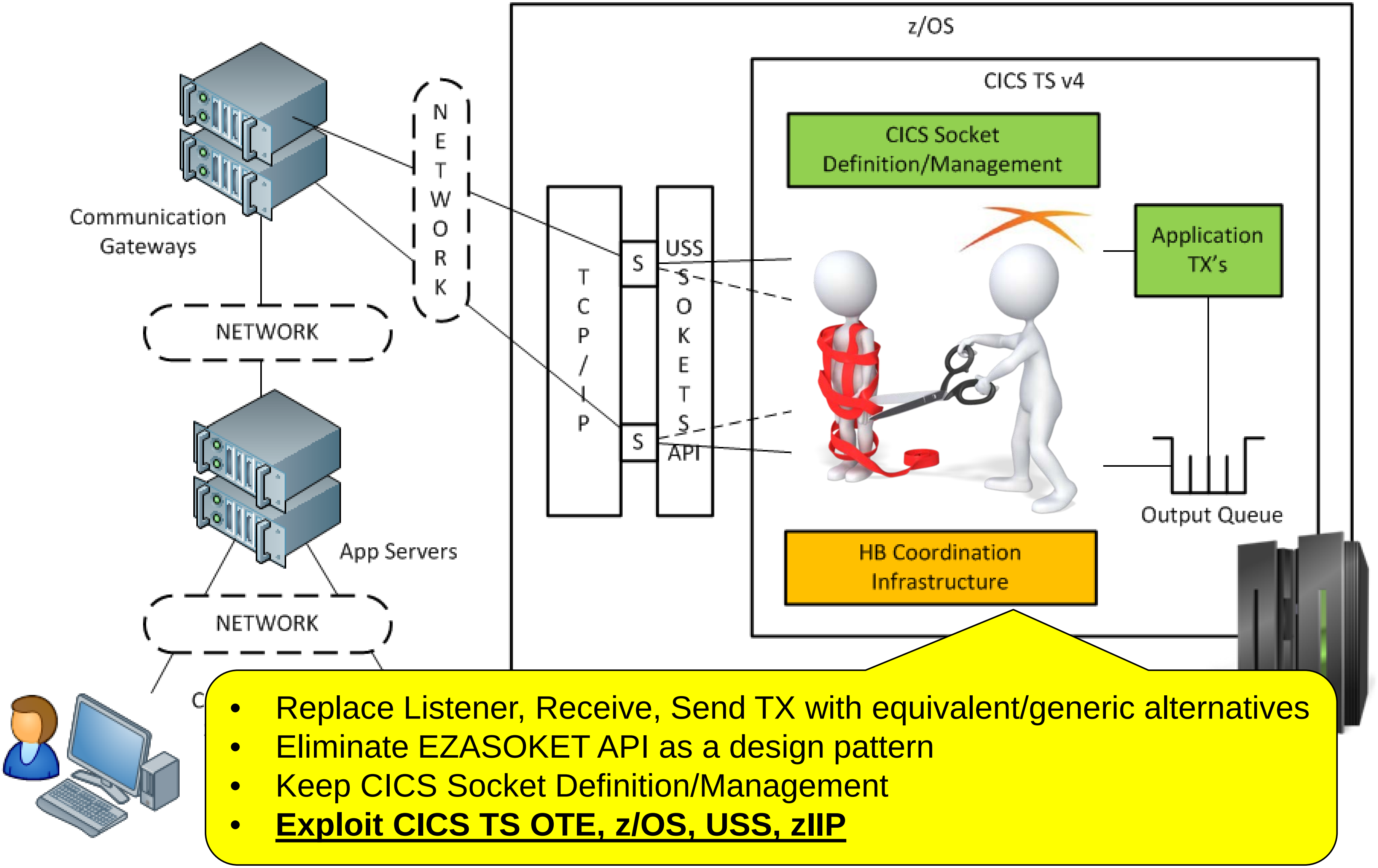
- **Provided as part of z/OS Communications Server**
- **What it includes:**
  - **Socket APIs (aka, EZASOKET or EZACICSO)**
  - **Listeners: standard and enhanced (i.e., CSKL)**
  - **Definition and management components (e.g., EZAO)**

Thus, I’m NOT referring to CICS TS features which use the CICS Sockets Domain.

- **A well-documented workhorse, but…**
- **It’s been around a long time (circa 1992)**
- **Older than CICS OTE**
  - **Thus… much of it’s original architecture**
- **Reengineered to support OTE**
  - **But… the general approach of the original architecture persisted**

- **However… much has changed in zOS and CICS TS!**

The Solution



- Replace Listener, Receive, Send TX with equivalent/generic alternatives
- Eliminate EZASOKET API as a design pattern
- Keep CICS Socket Definition/Management
- **Exploit CICS TS OTE, z/OS, USS, zIIP**

Solution Assessment

- **Excellent…**
  - **GP CPU burn associated with Socket I/O went <u>way down </u>(40-45%)**
  - **All components use native sockets**
  - **Transparent to the customer's applications**
  - **CICS Socket definition/management leveraged**
    - **EZAO still used to Configure, Start, or Stop Listeners**

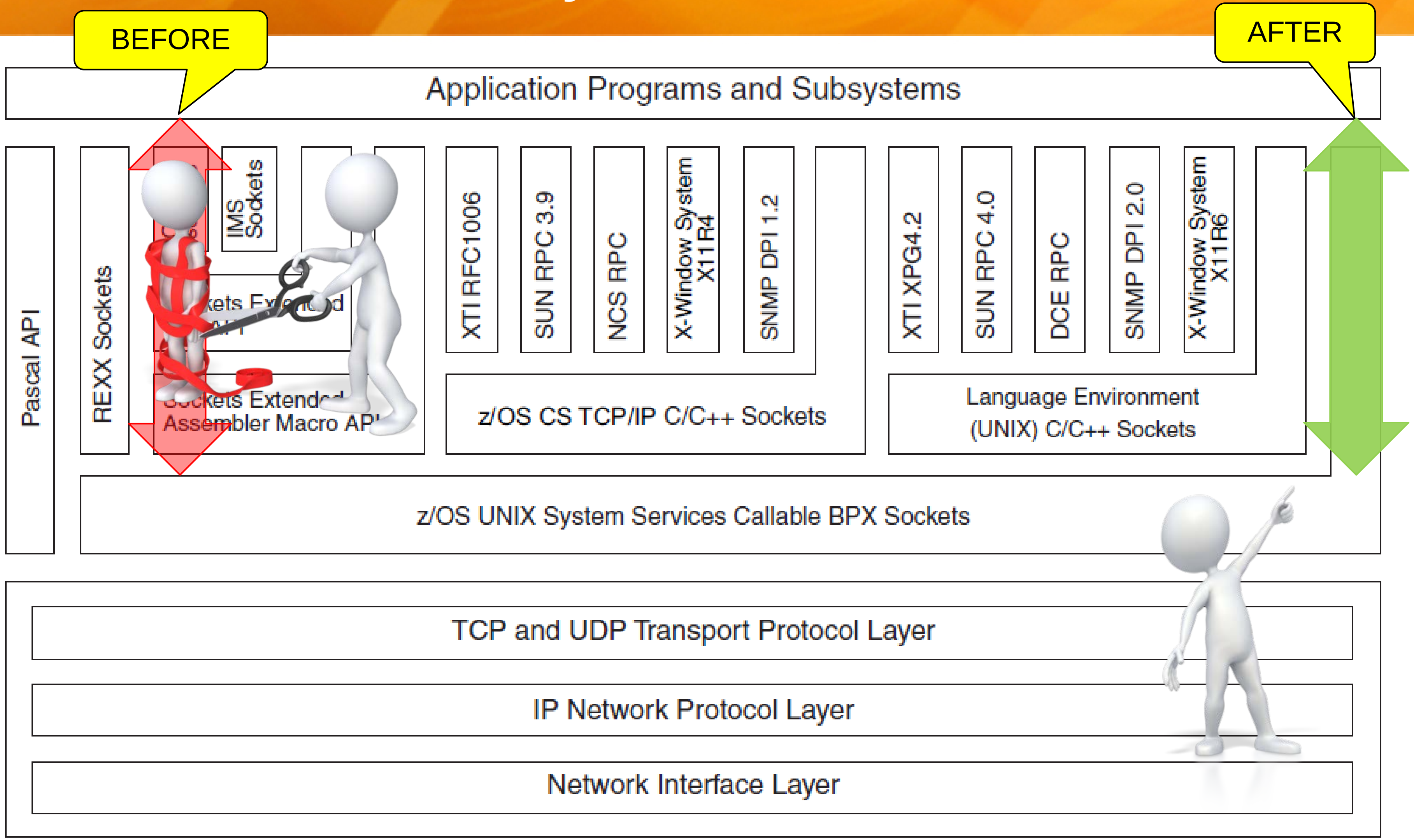
- **zIIP enablement potential maximized**
  - **HostBridge Socket Support code is zIIP enabled**
  - **Customer application code <u>not</u> zIIP enabled (per IBM-ISV T&C's)**
  - **Minimal task switching**

Value Proposition

- **What mattered most to one customer was processing new workload efficiently during their peak 4 hour period**
- **Assume:**
  - **5 million TX in peak 4 hour period**
  - **100% processed via HB Listener TX**
  - **20% processed via HB Worker TX**

5,000,000	Peak 4 hour transaction volume
20%	% of TX processed via HB Worker
1,000,000	TX processed via HB Worker
80%	% of TX processed via Std. Worker
4,000,000	TX processed via Std. Worker
903	Est. GP CPU Reduction for HB Worker (seconds)
807	Est. GP CPU Reduction for Std. Worker (seconds)
1,710	Total Est. GP CPU Seconds Reduced
28.49	Total Est. GP CPU Minutes Reduced during Peak Period

Pathway - Old vs. New

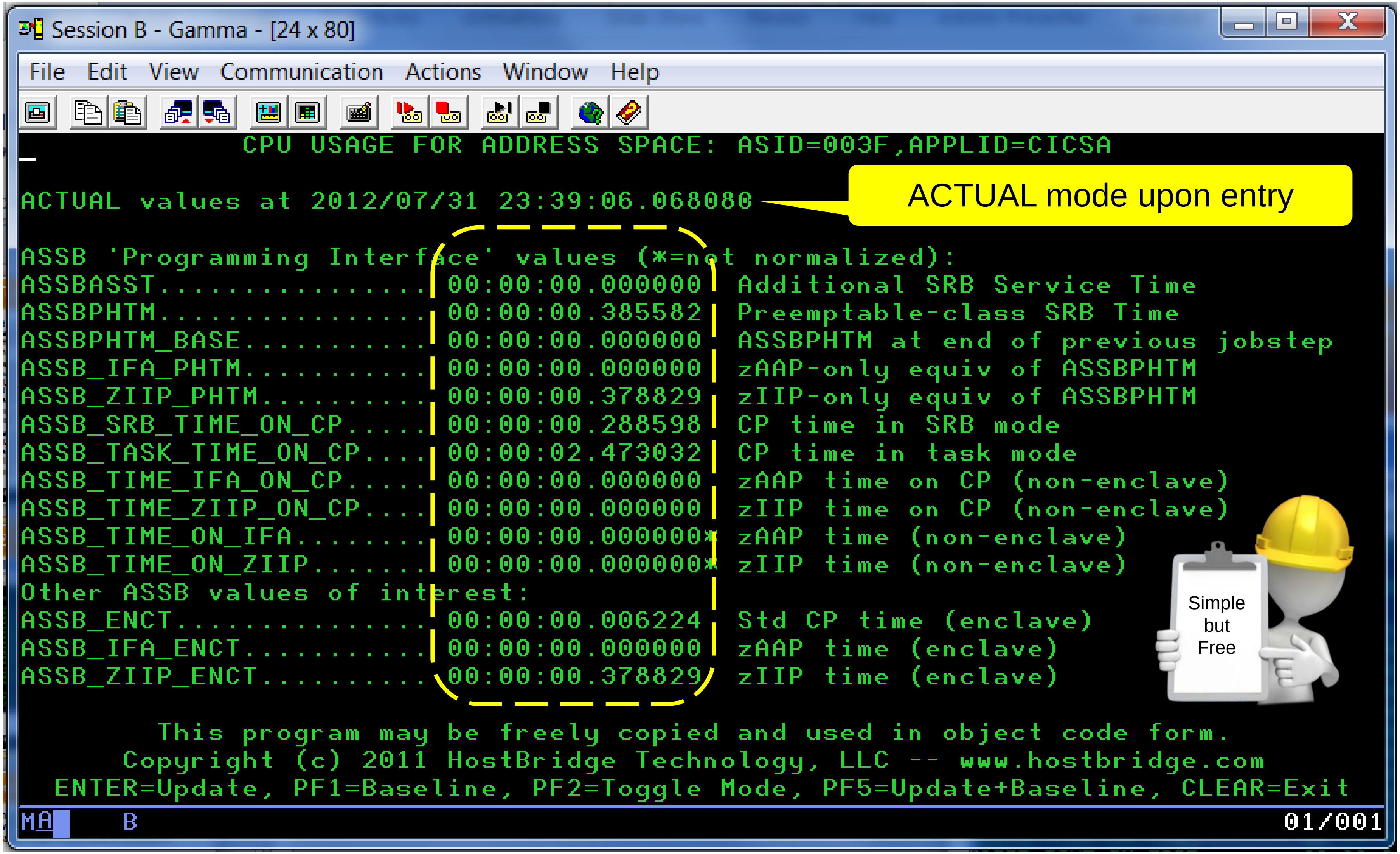

BEFORE
AFTER
Application Programs and Subsystems
Pascal API
REXX Sockets
IMS Sockets
kets Extended API
Sockets Extended Assembler Macro API
XTI RFC1006
SUN RPC 3.9
NCS RPC
X-Window System X11R4
SNMP DPI 1.2
z/OS CS TCP/IP C/C++ Sockets
XTI XPG4.2
SUN RPC 4.0
DCE RPC
SNMP DPI 2.0
X-Window System X11R6
Language Environment (UNIX) C/C++ Sockets
z/OS UNIX System Services Callable BPX Sockets
TCP and UDP Transport Protocol Layer
IP Network Protocol Layer
Network Interface Layer

z/OS Communications Server, IP Sockets Application Programming Interface Guide and Reference

Tooling Developed

- It’s difficult to get a snapshot of a CICS region’s total resource consumption that is:
  - high-resolution (microseconds)
  - low-overhead
  - Immediate
  - Includes zIIP and zAAP
- Ended up developing a simple transaction to display MVS ASSB timers (HBZT)
- Allowed us to:
  - drive testing fast
  - quickly assess results from all angles
- Special thanks
  - Larry Lawler (UNICOM)
  - Ed Jaffe (Phoenix Software)
- It’s free - send me an email

CPU Measurement (HBZT)



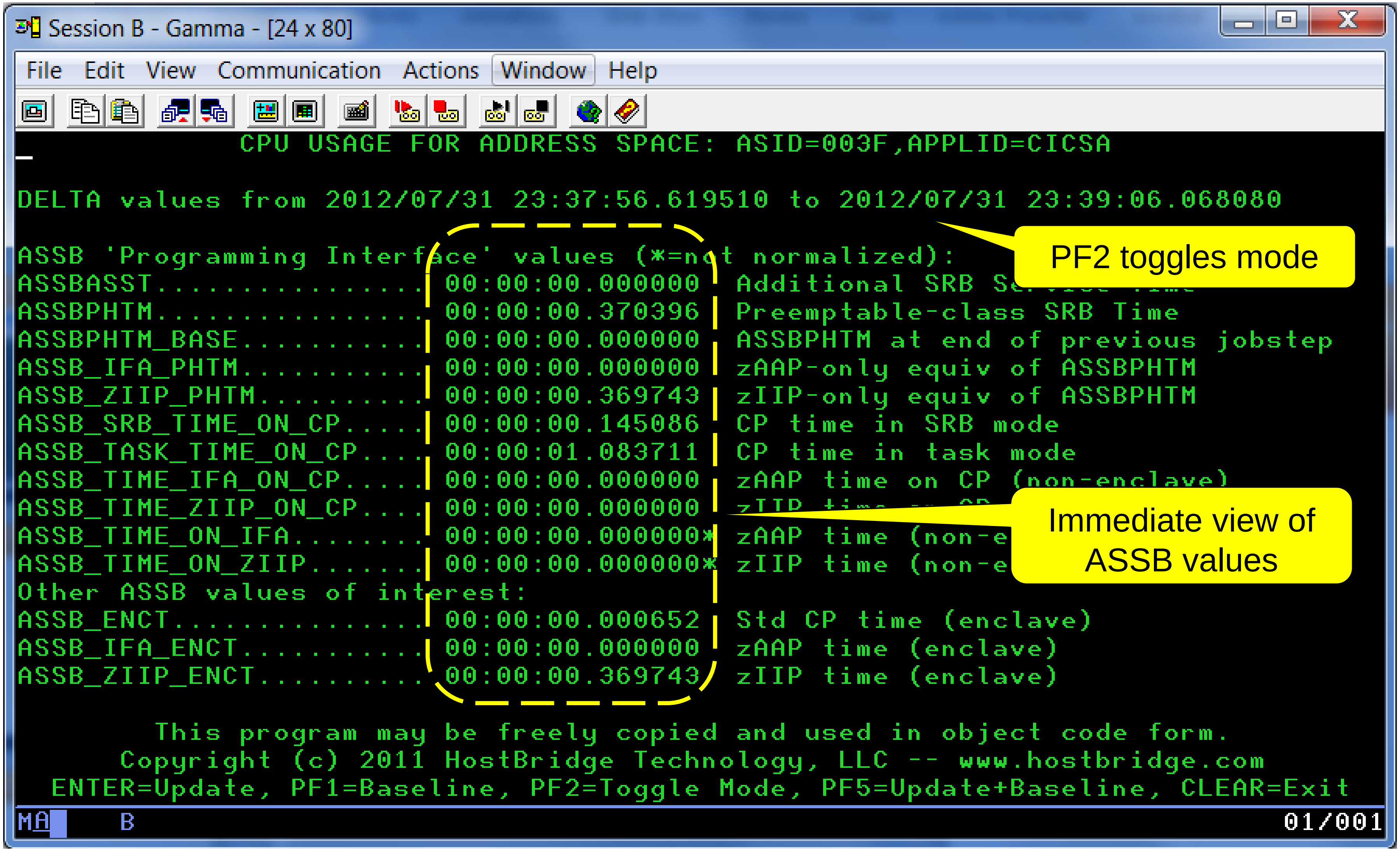
Session B - Gamma - [24 x 80]
File Edit View Communication Actions Window Help

```
             CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA
_

ACTUAL values at 2012/07/31 23:39:06.068080

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................ 00:00:00.000000  Additional SRB Service Time
ASSBPHTM................ 00:00:00.385582  Preemptable-class SRB Time
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM.......... 00:00:00.378829  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:00:00.288598  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 00:00:02.473032  CP time in task mode
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.000000  zIIP time on CP (non-enclave)
ASSB_TIME_ON_IFA........ 00:00:00.000000* zAAP time (non-enclave)
ASSB_TIME_ON_ZIIP....... 00:00:00.000000* zIIP time (non-enclave)
Other ASSB values of interest:
ASSB_ENCT............... 00:00:00.006224  Std CP time (enclave)
ASSB_IFA_ENCT........... 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT.......... 00:00:00.378829  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA      B                                                              01/001
```

ACTUAL mode upon entry

Simple
but
Free

CPU Measurement (HBZT)

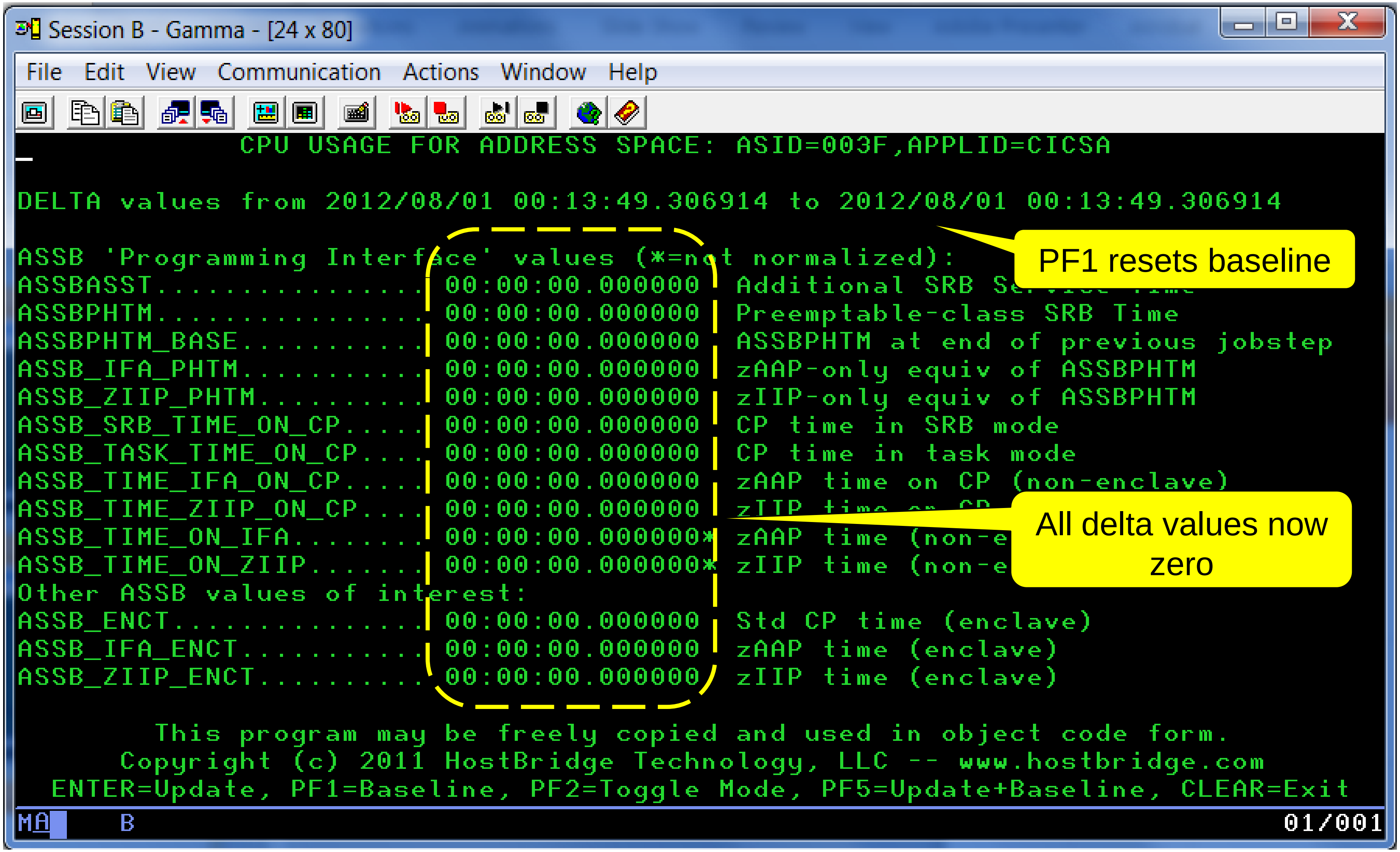


```
             CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA
_

DELTA values from 2012/07/31 23:37:56.619510 to 2012/07/31 23:39:06.068080

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................. 00:00:00.000000  Additional SRB Se
ASSBPHTM................. 00:00:00.370396  Preemptable-class SRB Time
ASSBPHTM_BASE............ 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM............ 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM........... 00:00:00.369743  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP...... 00:00:00.145086  CP time in SRB mode
ASSB_TASK_TIME_ON_CP..... 00:00:01.083711  CP time in task mode
ASSB_TIME_IFA_ON_CP...... 00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP..... 00:00:00.000000  zIIP
ASSB_TIME_ON_IFA......... 00:00:00.000000* zAAP time (non-e
ASSB_TIME_ON_ZIIP........ 00:00:00.000000* zIIP time (non-e
Other ASSB values of interest:
ASSB_ENCT................ 00:00:00.000652  Std CP time (enclave)
ASSB_IFA_ENCT............ 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT........... 00:00:00.369743  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA      B                                                              01/001
```

PF2 toggles mode

Immediate view of ASSB values

CPU Measurement

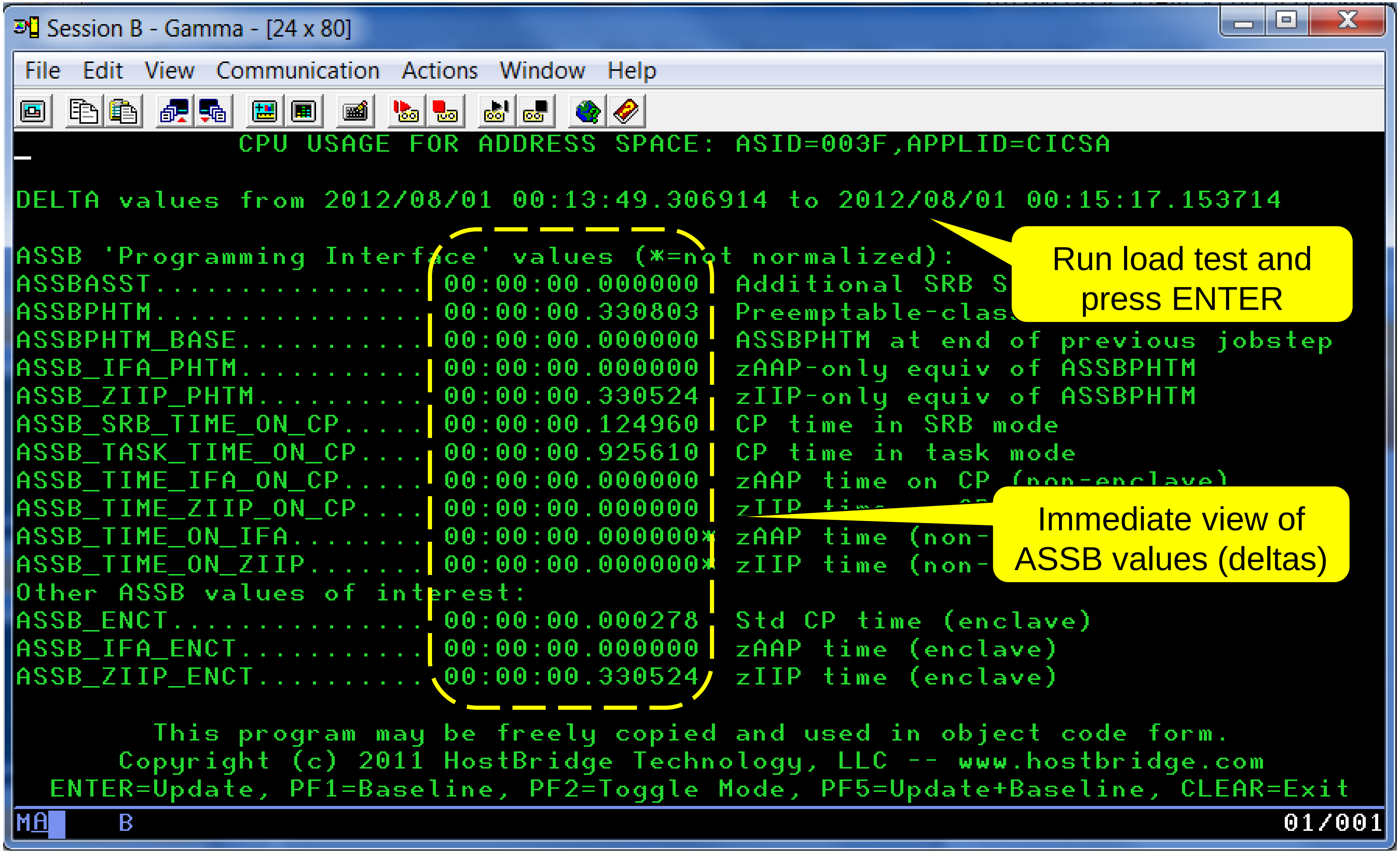

Session B - Gamma - [24 x 80]
File Edit View Communication Actions Window Help

```
             CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA
_

DELTA values from 2012/08/01 00:13:49.306914 to 2012/08/01 00:13:49.306914

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST.................  00:00:00.000000  Additional SRB Se
ASSBPHTM.................  00:00:00.000000  Preemptable-class SRB Time
ASSBPHTM_BASE............  00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM............  00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM...........  00:00:00.000000  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP......  00:00:00.000000  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.....  00:00:00.000000  CP time in task mode
ASSB_TIME_IFA_ON_CP......  00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP.....  00:00:00.000000  zIIP time on CP
ASSB_TIME_ON_IFA.........  00:00:00.000000* zAAP time (non-e
ASSB_TIME_ON_ZIIP........  00:00:00.000000* zIIP time (non-e
Other ASSB values of interest:
ASSB_ENCT................  00:00:00.000000  Std CP time (enclave)
ASSB_IFA_ENCT............  00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT...........  00:00:00.000000  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA     B                                                                01/001
```

PF1 resets baseline

All delta values now zero

CPU Measurement

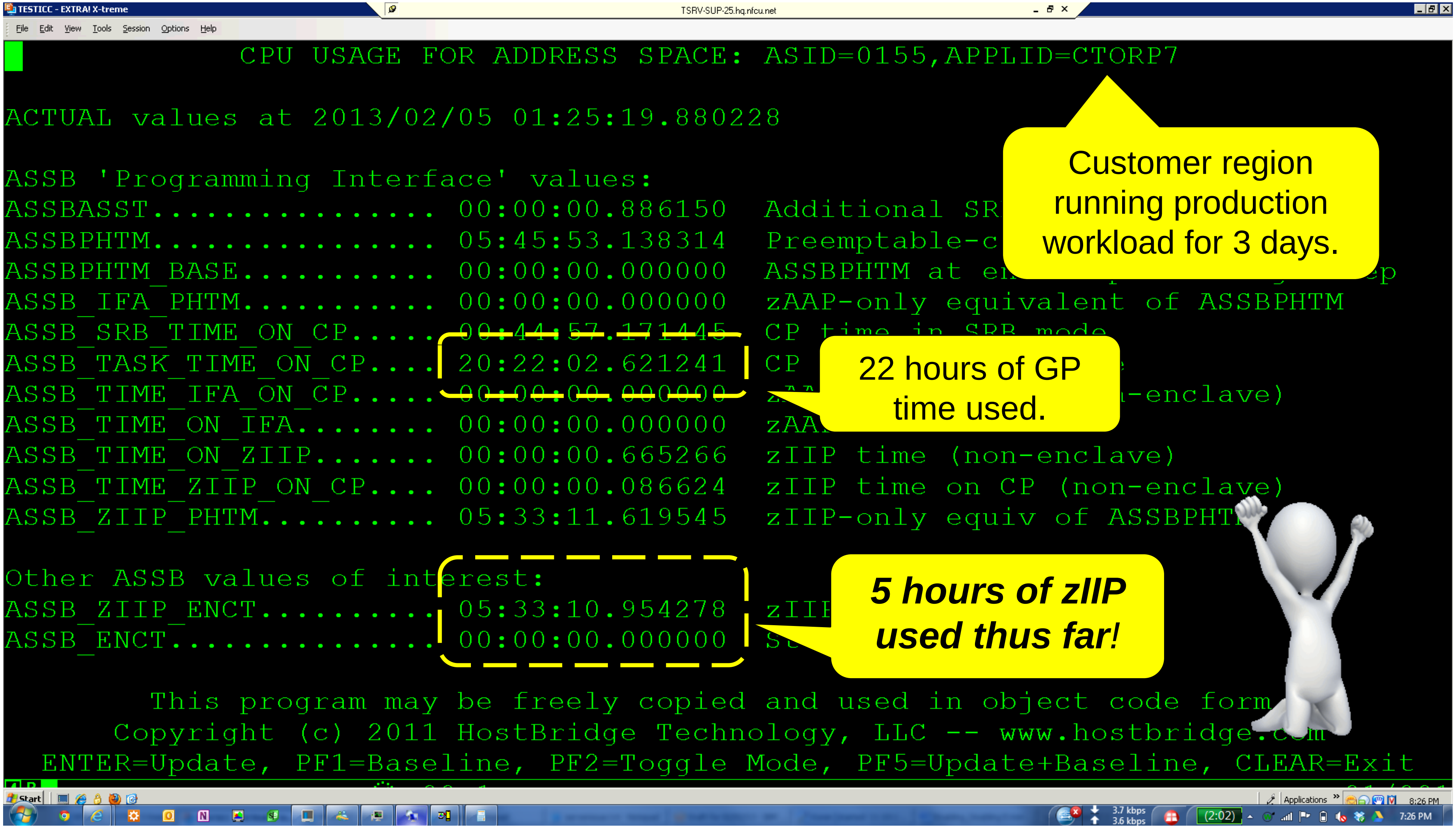


```
              CPU USAGE FOR ADDRESS SPACE: ASID=003F,APPLID=CICSA

DELTA values from 2012/08/01 00:13:49.306914 to 2012/08/01 00:15:17.153714

ASSB 'Programming Interface' values (*=not normalized):
ASSBASST................ 00:00:00.000000  Additional SRB S
ASSBPHTM................ 00:00:00.330803  Preemptable-clas
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at end of previous jobstep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equiv of ASSBPHTM
ASSB_ZIIP_PHTM.......... 00:00:00.330524  zIIP-only equiv of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:00:00.124960  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 00:00:00.925610  CP time in task mode
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAAP time on CP (non-enclave)
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.000000  zIIP
ASSB_TIME_ON_IFA........ 00:00:00.000000* zAAP time (non-
ASSB_TIME_ON_ZIIP....... 00:00:00.000000* zIIP time (non-
Other ASSB values of interest:
ASSB_ENCT............... 00:00:00.000278  Std CP time (enclave)
ASSB_IFA_ENCT........... 00:00:00.000000  zAAP time (enclave)
ASSB_ZIIP_ENCT.......... 00:00:00.330524  zIIP time (enclave)

        This program may be freely copied and used in object code form.
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
MA     B                                                               01/001
```

Run load test and press ENTER

Immediate view of ASSB values (deltas)

CPU Measurement



```
             CPU USAGE FOR ADDRESS SPACE: ASID=0155,APPLID=CTORP7

ACTUAL values at 2013/02/05 01:25:19.880228

ASSB 'Programming Interface' values:
ASSBASST................ 00:00:00.886150  Additional SR
ASSBPHTM................ 05:45:53.138314  Preemptable-c
ASSBPHTM_BASE........... 00:00:00.000000  ASSBPHTM at en                  ep
ASSB_IFA_PHTM........... 00:00:00.000000  zAAP-only equivalent of ASSBPHTM
ASSB_SRB_TIME_ON_CP..... 00:44:57.171445  CP time in SRB mode
ASSB_TASK_TIME_ON_CP.... 20:22:02.621241  CP                  e
ASSB_TIME_IFA_ON_CP..... 00:00:00.000000  zAA                n-enclave)
ASSB_TIME_ON_IFA........ 00:00:00.000000  zAA
ASSB_TIME_ON_ZIIP....... 00:00:00.665266  zIIP time (non-enclave)
ASSB_TIME_ZIIP_ON_CP.... 00:00:00.086624  zIIP time on CP (non-enclave)
ASSB_ZIIP_PHTM.......... 05:33:11.619545  zIIP-only equiv of ASSBPHT

Other ASSB values of interest:
ASSB_ZIIP_ENCT.......... 05:33:10.954278  zIII
ASSB_ENCT............... 00:00:00.000000  S

        This program may be freely copied and used in object code form
      Copyright (c) 2011 HostBridge Technology, LLC -- www.hostbridge.com
  ENTER=Update, PF1=Baseline, PF2=Toggle Mode, PF5=Update+Baseline, CLEAR=Exit
```

Customer region running production workload for 3 days.

22 hours of GP time used.

5 hours of zIIP used thus far!

HB Socket Support Summary

- **Operational efficiency is paramount to all System z customers**
- **The CICS TS Open Transaction Environment continues to evolve and creates new opportunities for customers and ISV's to extract savings**
- **The approach embodied by HostBridge Socket Support is only one example of what's possible**
  - **Applicable to any customer who uses CICS Socket Support**
  - **zIIP support can only be provided by a licensed ISV**
- **Bottom Line: There is no reason to devote high-value MIPS to integration or "plumbing"**
- **Oh… and the customers were very pleased**

Service Agility

- **Deliver First-class services from existing transactions, programs and data**
- **Express CICS services using modern interfaces**
- **Create foundational enhancements that allow rapid change and deployment**

Case Study (Very Fresh)

- **The Situation: Customer has a high-value COBOL batch subroutine that performs complex insurance claim reimbursement calculations**
- **Business Objective: Perform real-time claims processing via a web service**
- **An Option: Clone the program and make a CICS-specific version**
- **Reality Check: The business, financial and legal risk/cost of maintaining two code bases were big**
  - **If they were going to have two code bases, they might move one *off* System z**
- **The Idea: Execute the COBOL batch program as part of a CICS-based web service – *without changing it!***

“No Changes” ???

- **What does that really mean?**
  1. **No changes to the program object/load module – implies that the same load module must be used in batch and online.**
  2. **No changes to the program source code -- this allows for relinking the program with alternative I/O handlers**
  3. **No changes to the general program logic -- this would permit replacing the COBOL I/O verbs with either: (a) calls to I/O subroutines, or (b) EXEC CICS commands (assumes a second source base is OK)**

- **Customer wanted the first and could live with the second -- the third option was out of the question**

Subroutine Characteristics

- **26 passed parameters**
  - **Some are complex (e.g., record structures described by a COBOL copybook).**
  - **Whatever the interpretation of “no changes”, the structure of the subroutine parameters could not change**

- **References up to 46 different VSAM files**
  - **41 opened for input**
  - **5 opened for input-output**
  - **Only 1 actually written to**
- **Entry points for “open”, “process” and “close”**
  - **Extremely valuable!**
  - **Good application design makes integration easier**
- **All file handling performed via COBOL primitives**
  - **Thus, the subroutine CAN’T run on a CICS managed TCB**

Clear Points of Leverage

- **Leverage CICS Open Transaction Environment**
  - **OTE allows virtually any non-authorized code path to be executed within a CICS address space**
  - **When running on an Open TCB, there is nothing to prevent a program from directly accessing VSAM files (i.e., not use EXEC CICS commands) – but not from COBOL**
- **Leverage HostBridge Architectural Features**
  - **HB infrastructure can run inside a CICS address space, but not under a CICS managed TCB**
  - **Foundational to HB features such as zIIP enablement and Socket Support**
- **Leverage LE PIPI**
  - **PIPI is our friend**
  - **But only under a privately managed TCB, and with our own LE service routines**

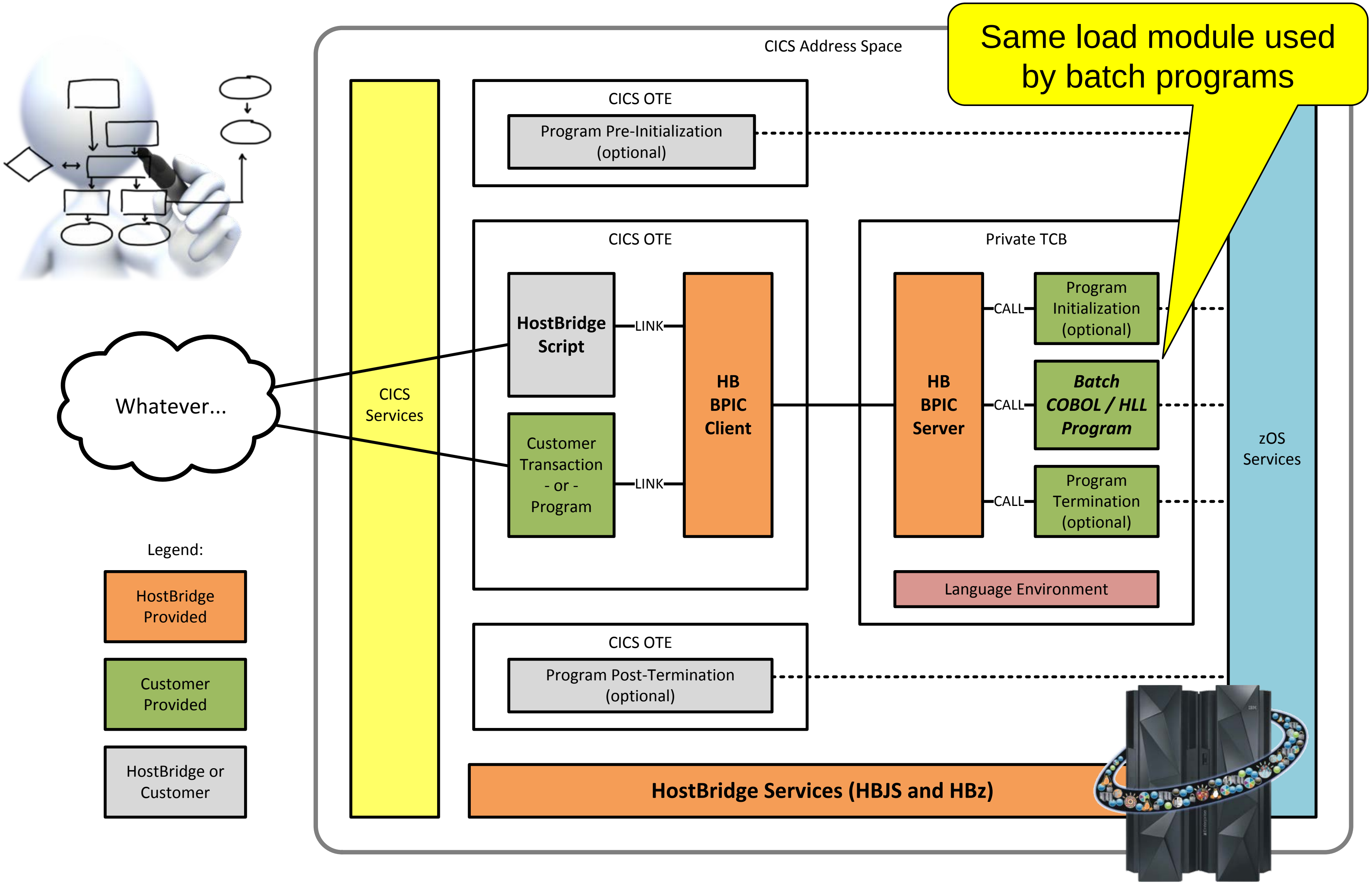
Web Service Externalization

- **How the subroutine was externalized as a service was influenced by a variety of factors**
  - **Request volume**
  - **Characteristics of the distributed application**
  - **The communication infrastructure between requestor and System z**
    - **Speed**
    - **Latency**
  - **Standards and preferences of the organization**
    - **“Formal SOA” (SOAP, WSDL)?**
    - **“Informal SOA” (HTTP with XML or JSON payloads)?**
  - **Complexity and size of the input/output parameters**

Conceptual Alternatives

- **Host the subroutine inside a CICS region as a reusable service**
  - **Exploit CICS OTE and HB infrastructure**
  - **Run COBOL subroutine on private TCB managed by HB**
- **Host the subroutine outside of a CICS region**
  - **Create a "service address space"**
  - **All service consumers (CICS or batch) could access the same code**
  - **Whether this is ever a good idea depends on the volume of batch invocations**
- **Eliminate the batch job by using a CICS address space to do all the work**
  - **Sounds strange but feasible**
  - **Ruled out due to all sorts of CICS and Batch window management issues**

BPIC Solution Architecture



Same load module used by batch programs

CICS Address Space

CICS Services

CICS OTE

Program Pre-Initialization (optional)

CICS OTE

HostBridge Script

Customer Transaction - or - Program

LINK

LINK

HB BPIC Client

Private TCB

HB BPIC Server

CALL

CALL

CALL

Program Initialization (optional)

Batch COBOL / HLL Program

Program Termination (optional)

Language Environment

CICS OTE

Program Post-Termination (optional)

HostBridge Services (HBJS and HBz)

zOS Services

Whatever...

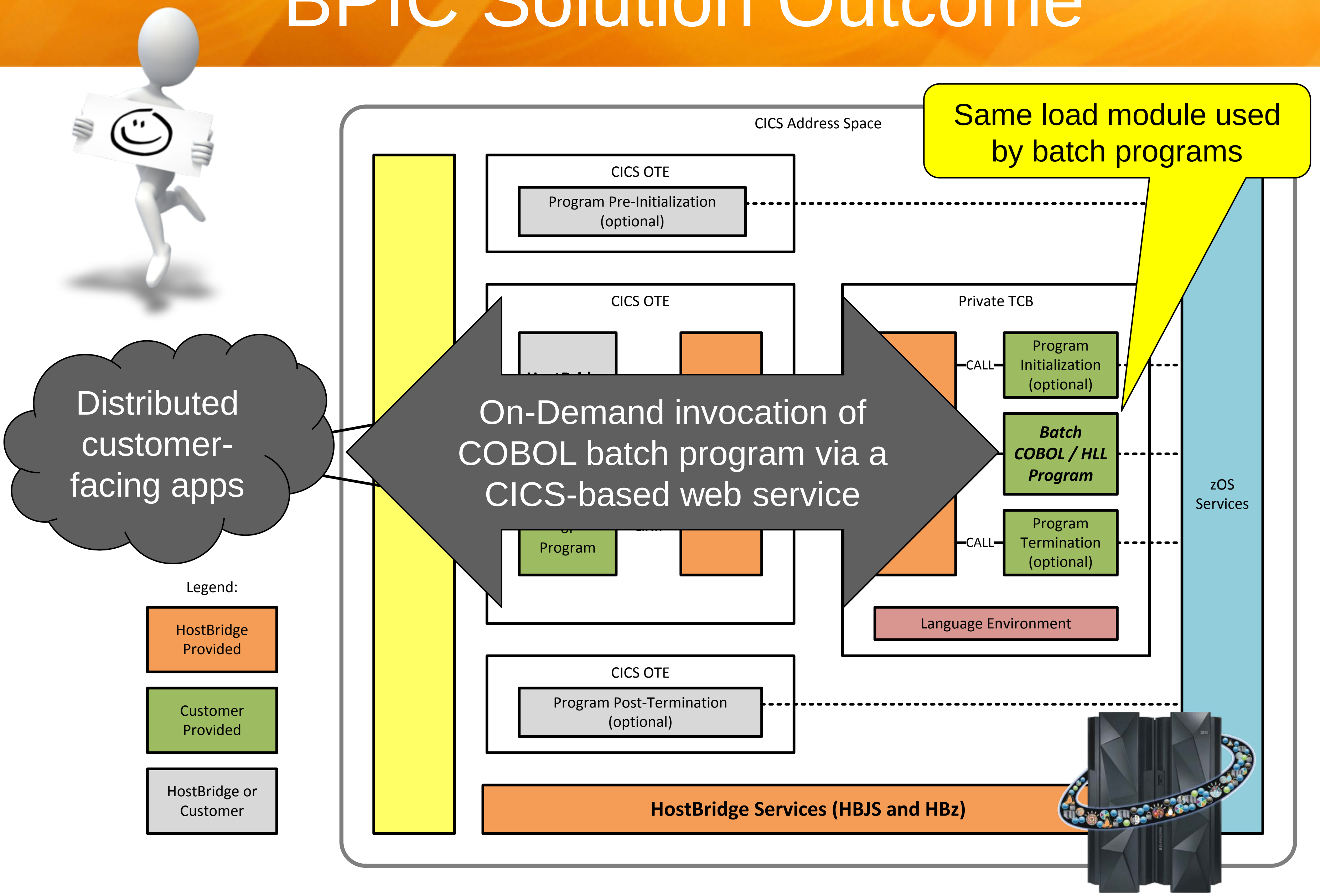
Legend:

HostBridge Provided

Customer Provided

HostBridge or Customer

BPIC Solution Outcome



Same load module used by batch programs

Distributed customer-facing apps

On-Demand invocation of COBOL batch program via a CICS-based web service

CICS Address Space
CICS OTE
Program Pre-Initialization (optional)
CICS OTE
Private TCB
CALL
Program Initialization (optional)
Batch COBOL / HLL Program
CALL
Program Termination (optional)
Program
Language Environment
zOS Services
CICS OTE
Program Post-Termination (optional)
HostBridge Services (HBJS and HBz)
Legend:
HostBridge Provided
Customer Provided
HostBridge or Customer

Solution Summary

- **The technology investments in HBz and HBSS laid a foundation for doing other creative things under the CICS OTE environment**
- **The complication in this case was that the customer wrote the target code… and it was in COBOL!**
- **But it turned out that…**
  - ***HB + CICS OTE + LE PIPI = SOLUTION***
- **Figuring out what to call it was challenging**
  - **Idea: "Batch Program Inside CICS"**
  - **Thus… "BPIC"**
- **Special thanks to the IBM LE, COBOL and CICS teams for their support**

Customer Business Drivers

- **Operational efficiency is paramount**
  - **Respond to competitive pressures in your industry**
  - **Lower incremental cost of high-volume application processing**
- **But so is service and data agility**
  - **Web, Mobile, SOA, Cloud, AJAX, Javascript, XML, JSON**
- **Solutions must be “both/and” not “either/or”**
  - **Extract the proven value of existing System z apps and data**
  - **Integrate them with the widest array of non-System z apps and interfaces**

    -- AND --
  - **Do it in a way that squeezes costs/MIPS out of every process**

Summary

- **HostBridge and CICS TS share common objectives**
  - **Operational Efficiency & Service Agility**
- **CICS TS continues to break new ground and HostBridge stays in step to exploit**
  - **HostBridge 6.62 is our corresponding support release for CICS TS 5.1**
- **In TS 5 we are excited about:**
  - **Continued OTE enhancements**
  - **Ongoing 64-bit storage enhancements**
  - **The WAS Liberty Profile**
- **FACTS…**
  - **Poor integration solutions have often given System z apps a bad rap**
  - **Customers who exploit modern CICS capabilities (and ISV integration products!) will be rewarded**

WANTED: Tales from the Trenches

- We KNOW you are doing cool stuff (we see it every month)
- We are looking for Tales from the Trenches to SHARE next time
- Practical stories about how you are:
  - Meeting business challenges
  - Overcoming technical hurdles
  - Transforming your CICS apps for the future
  - Leveraging new features of CICS
- The objective is to create an active feedback loop of user experiences within the CICS community
- We will help