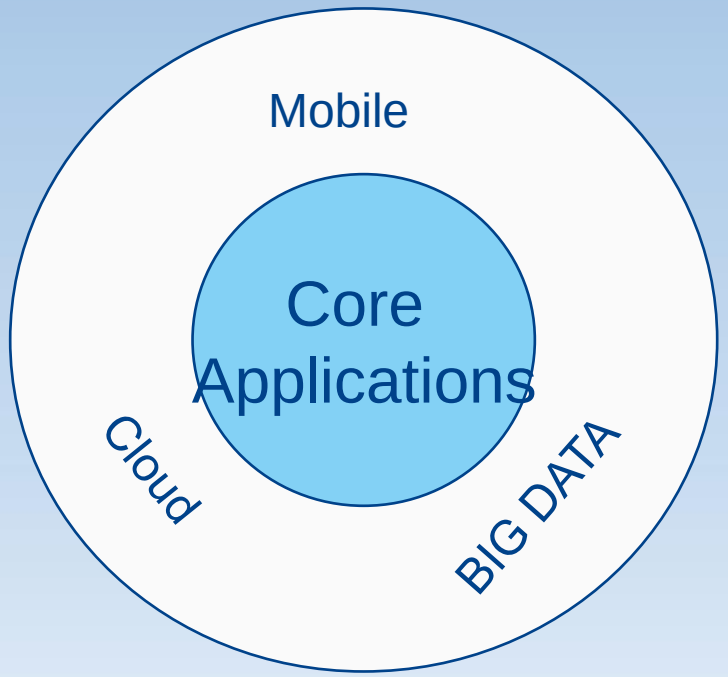
Session B12: Fiducia Driving Efficiencies with IMS

Pascal Meyer, Karlsruhe 03/2015

Objectives

- Motivation to bring Java on the Mainframe
- The benefit of JZOS and IMS-BMPs in Batch
- The benefits of Java in IMS MPRs for Online Processing
- Impact of Java on the development and production environments
- Next step



Fiducia: Driving Efficiencies with IMS

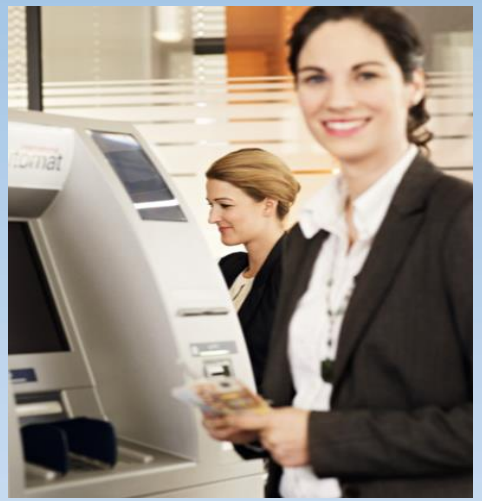
1	**Fiducia overview**
2	Business needs, Added Value of Java/Z
3	Java Batch Experience (JZOS and BMPs)
4	Java-IMS: Experience
5	Impact of resource consumption
6	Technology outlook (Fiducia / IBM)

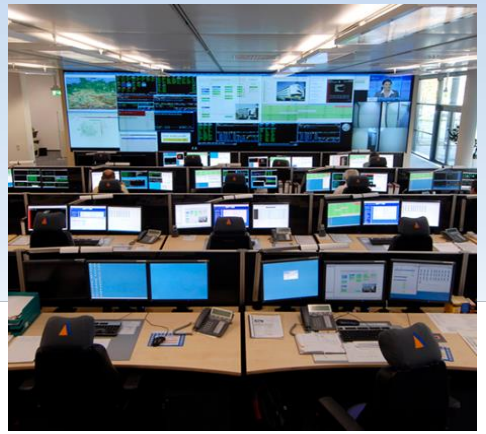
Fiducia IT AG - Overview

- One of the leading IT service providers in Germany
  - provide and operate a functional rich integrated core banking system
  - over 730 client banks
- Core competencies
  - developing and implementing integrated IT solutions
  - operating the compute center for applications requiring a high level of security and availability

Fiducia IT AG: full service provider for our Client Banks

- Full service provider for the Banks
- About 10,550 Points of sales
  - 98,800 PC workstations in banks
  - 23,000 self-service terminals, including
    - 11,300 ATMs
    - 9,100 bank statement printers
    - 2,300 service terminals
  - 6,500 other servers





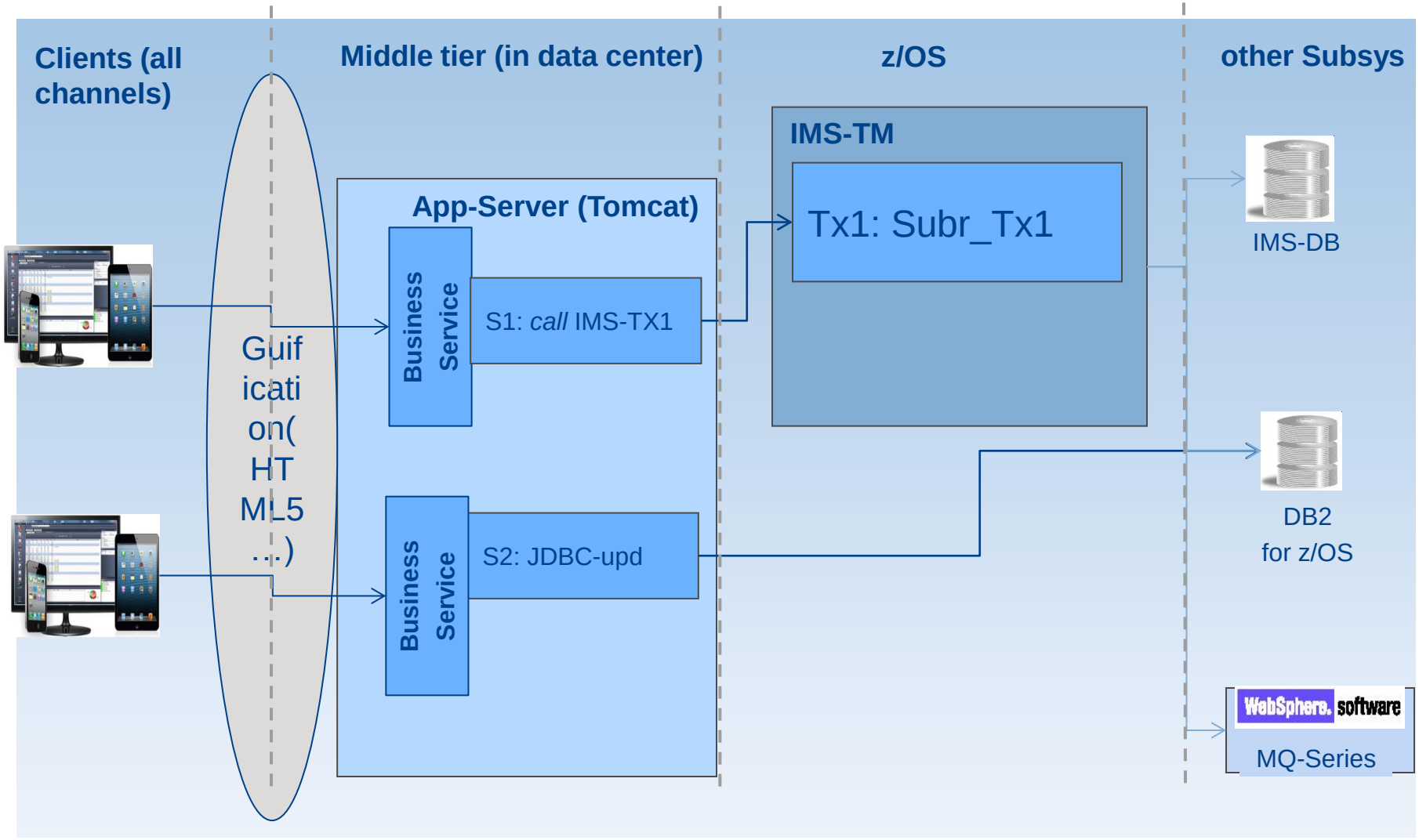
Fiducia IT AG - Numbers

- High available and secure compute centers
  - thousands virtual unix/windows Servers
  - 5 z/Series EC12
  - 2 productive Sysplexes with 10 IMS/DB2/MQ Subsystems serves the core banking
- Management of more than 67 million accounts at cooperative and private banks
- Processing volume
  - 20bn IMS transactions / year
  - In Peaks (IMS)
    - over 110ml Tx/day
    - over 4200 Tx/s

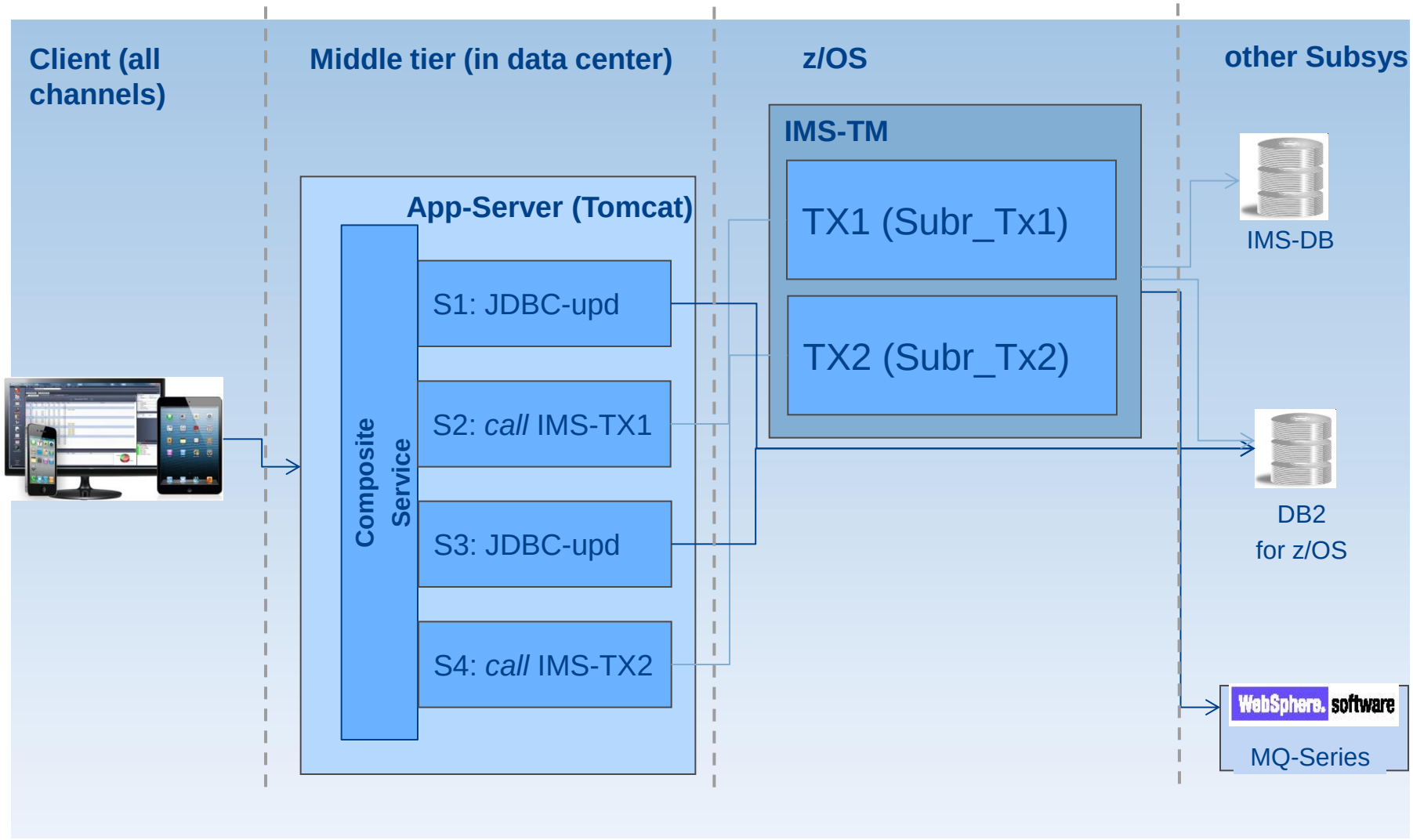
Fiducia: Driving Efficiencies with IMS

1	Fiducia overview
2	**Business needs, Added Value of Java/Z**
3	Java Batch Experience (JZOS and BMPs)
4	Java-IMS: Experience
5	Impact of resource consumption
6	Technology outlook (Fiducia / IBM)

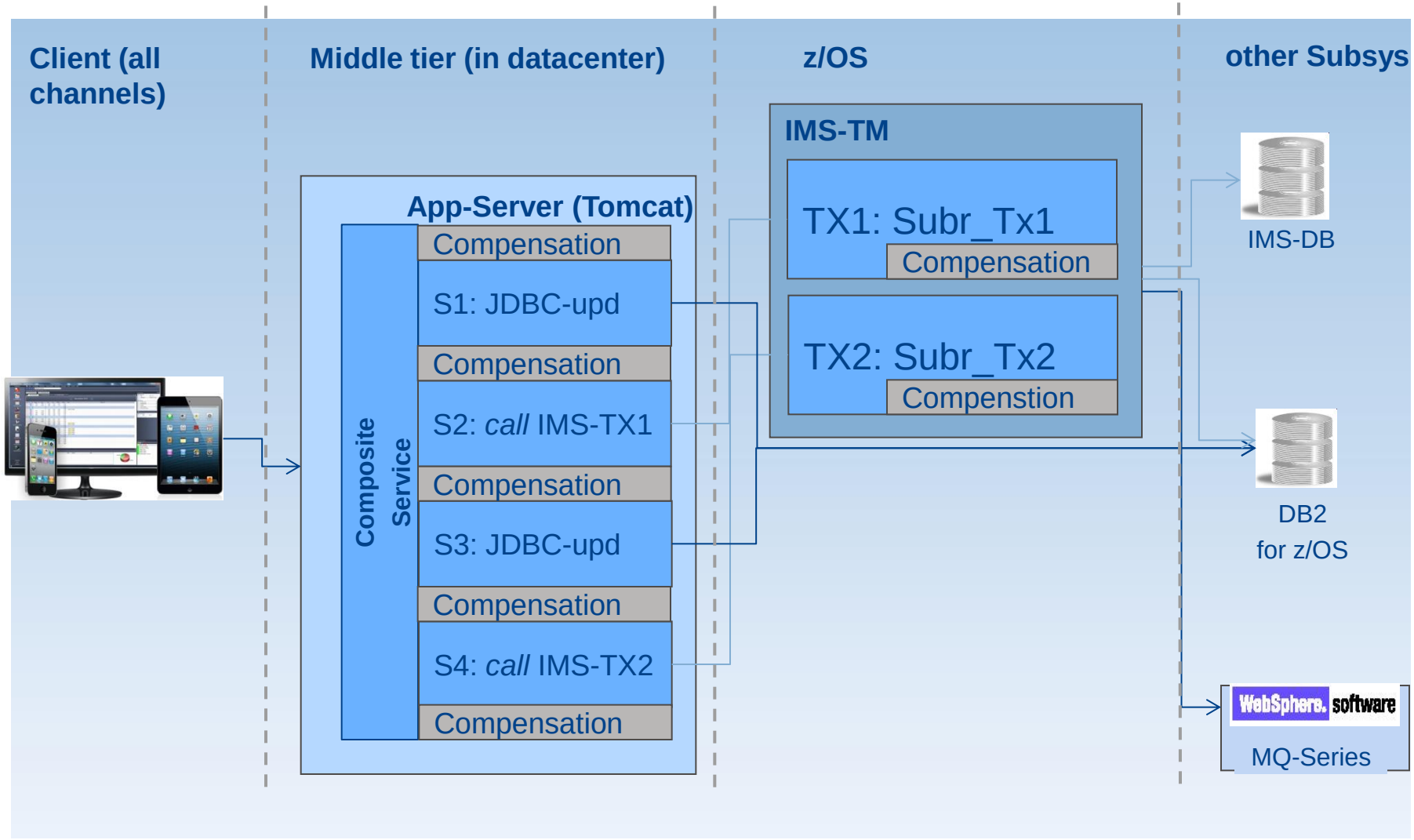
The standard ways to implement business transactions @Fiducia



The representation of complex business transactions spread across platforms without TX-Manager can become very complex

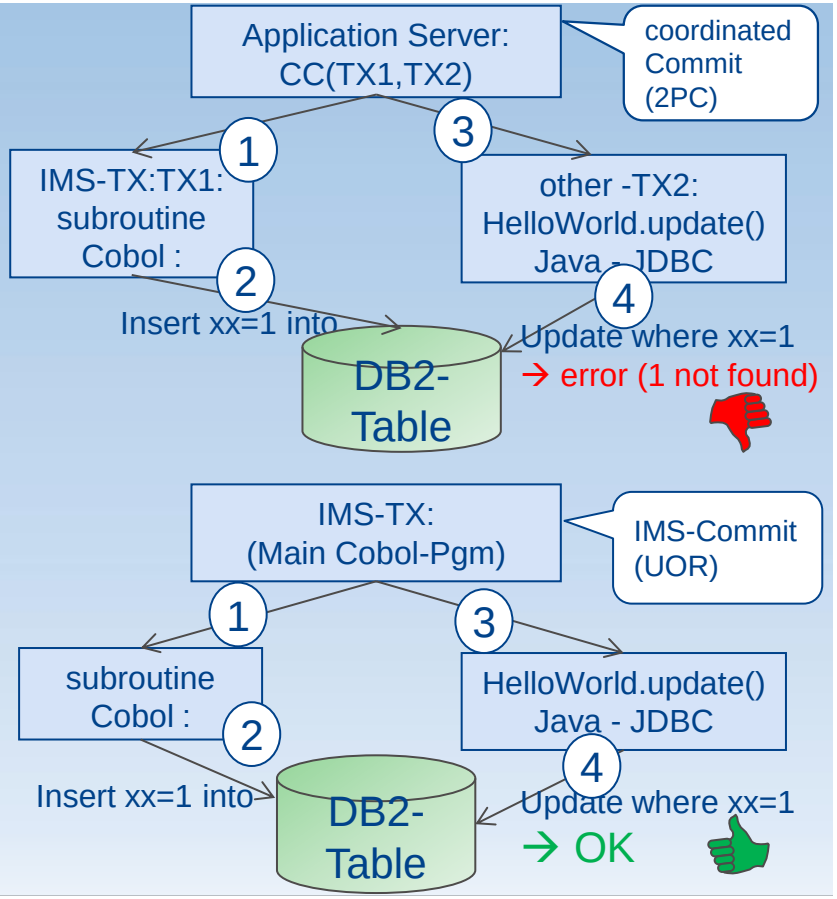


Complex business transactions spread across platforms without TX-Manager requires compensation



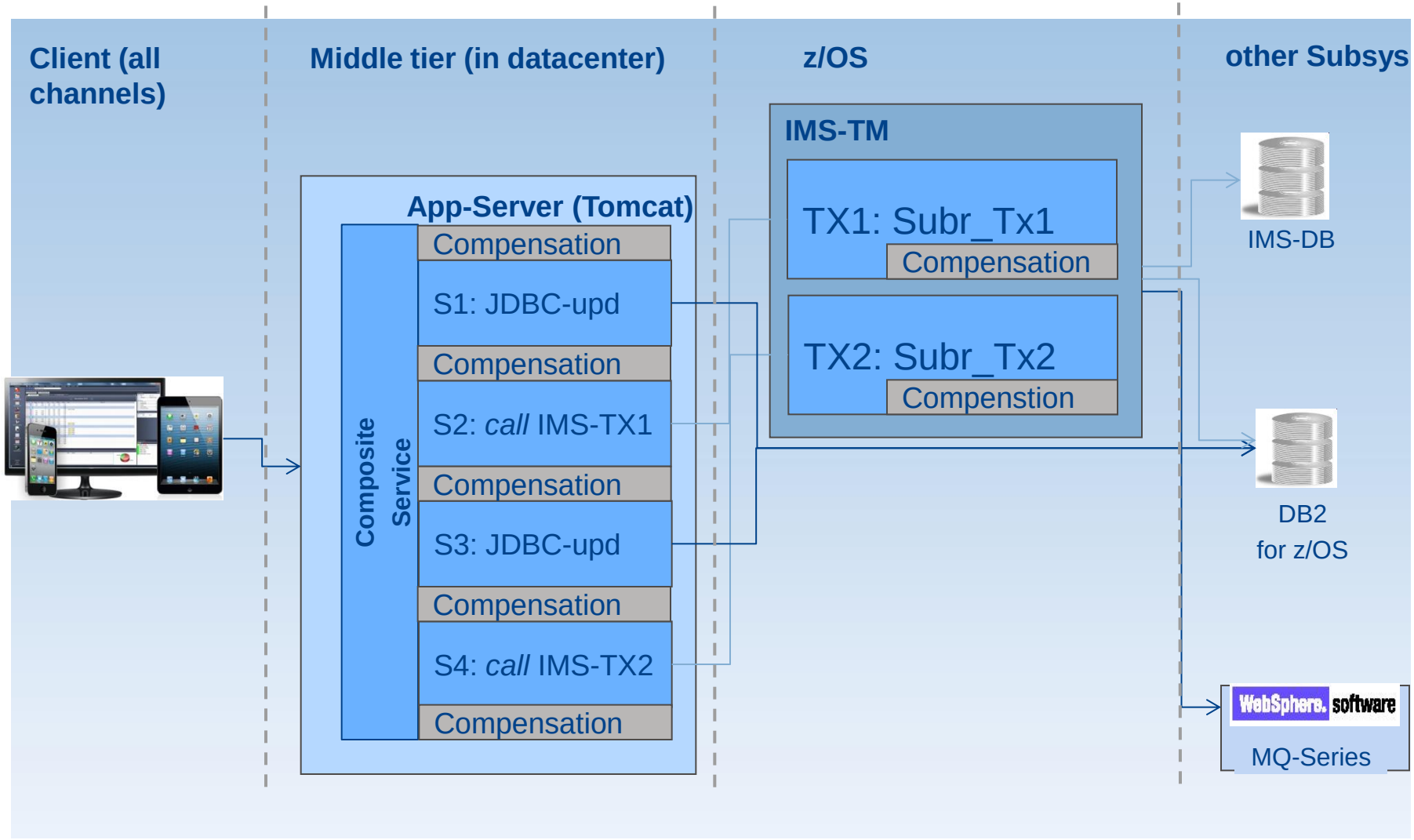
The breakthrough: since 2011 the support for Cobol-Java mix is available for MPRs and BMPs

- Application servers (Websphere …):
  - coordinated commit over several Txs, including IMS-TX is possible
  - but during execution the coordinated TXs don‘t know anything from each other
- IMS guarantees a single UOW over Cobol-Java-Mix in an MPR (Message Processing Region)
  - One ESAF- (External Subsystem Attach Facility) connection for both Cobol static SQL and JDBC based database accesses

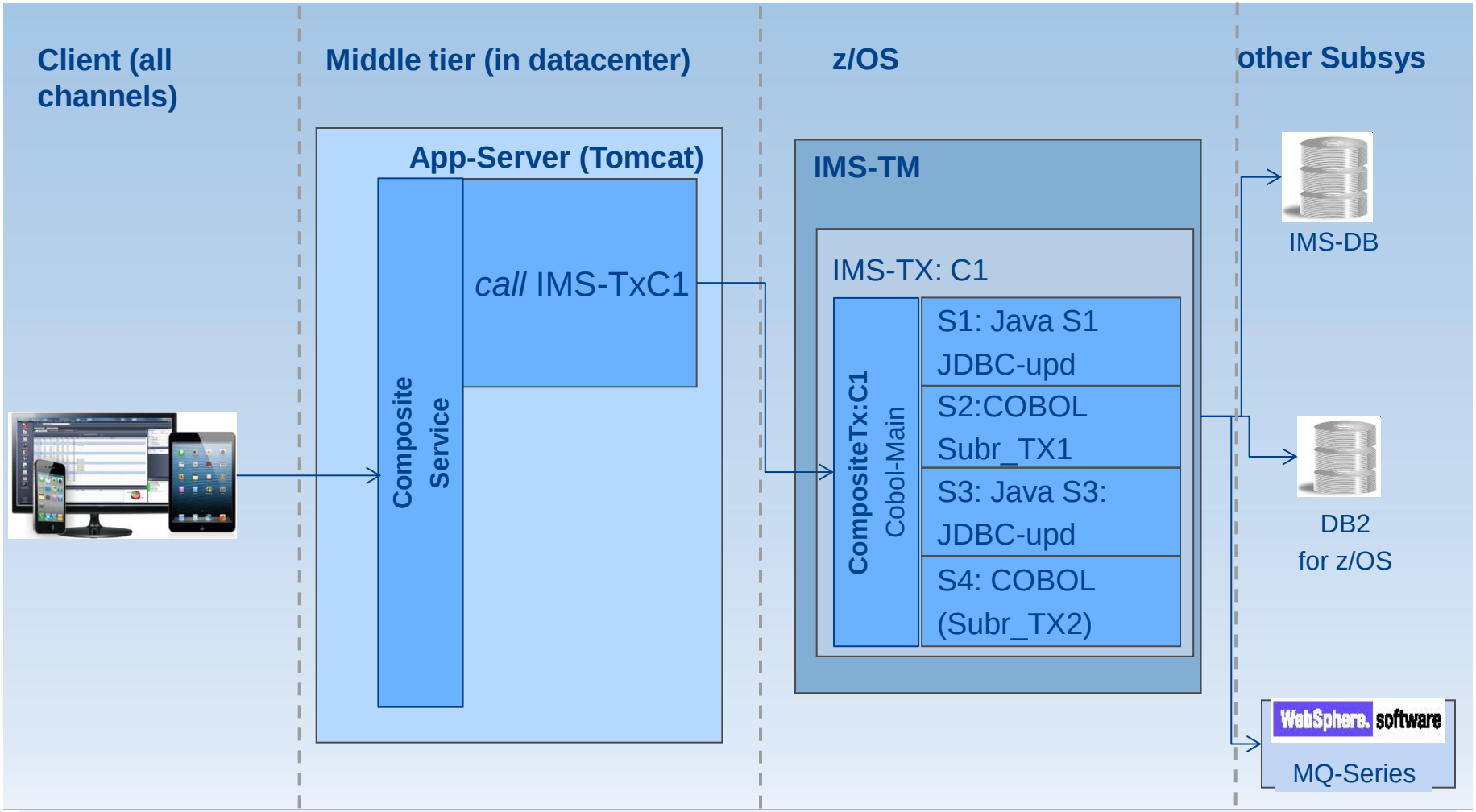


Mixing Java just like another langage with COBOL inside an MPR becomes reality

Complex business transactions spread across platforms without TX-Manager requires compensation

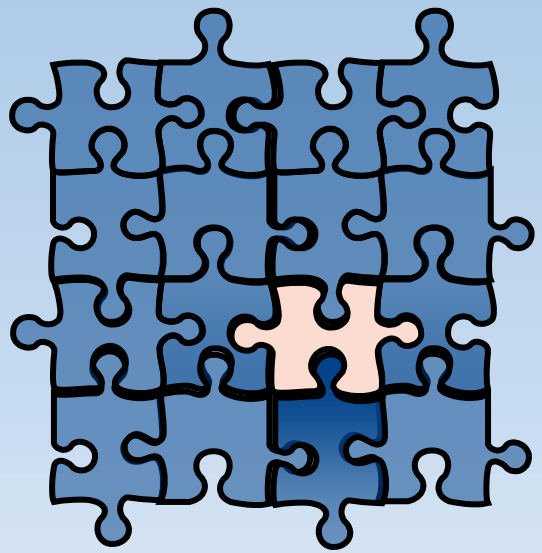


Java inside IMS MPRs can help reduce the complexity of such business cases



Investing in Java on z/OS allows us to stay fit for the future

- Permanently expand the existing Application set
  – Reuse of existing services
- implement new functions in java and use it everywhere
- use 3rd party software inside IMS
  – ZIP, Crypto,..
- Option to Use technologies and languages where skill is widely available and new generation is willing to use



Option to “evolve over time“ from a mainly COBOL based System to a mainly Java based system

Fiducia: Driving Efficiencies with IMS

1	Fiducia overview
2	Business needs, Added Value of Java/Z
3	**Java Batch Experience (JZOS and BMPs)**
4	Java-IMS: Experience
5	Impact of resource consumption
6	Technology outlook (Fiducia / IBM)

Batch Evolution: current situation

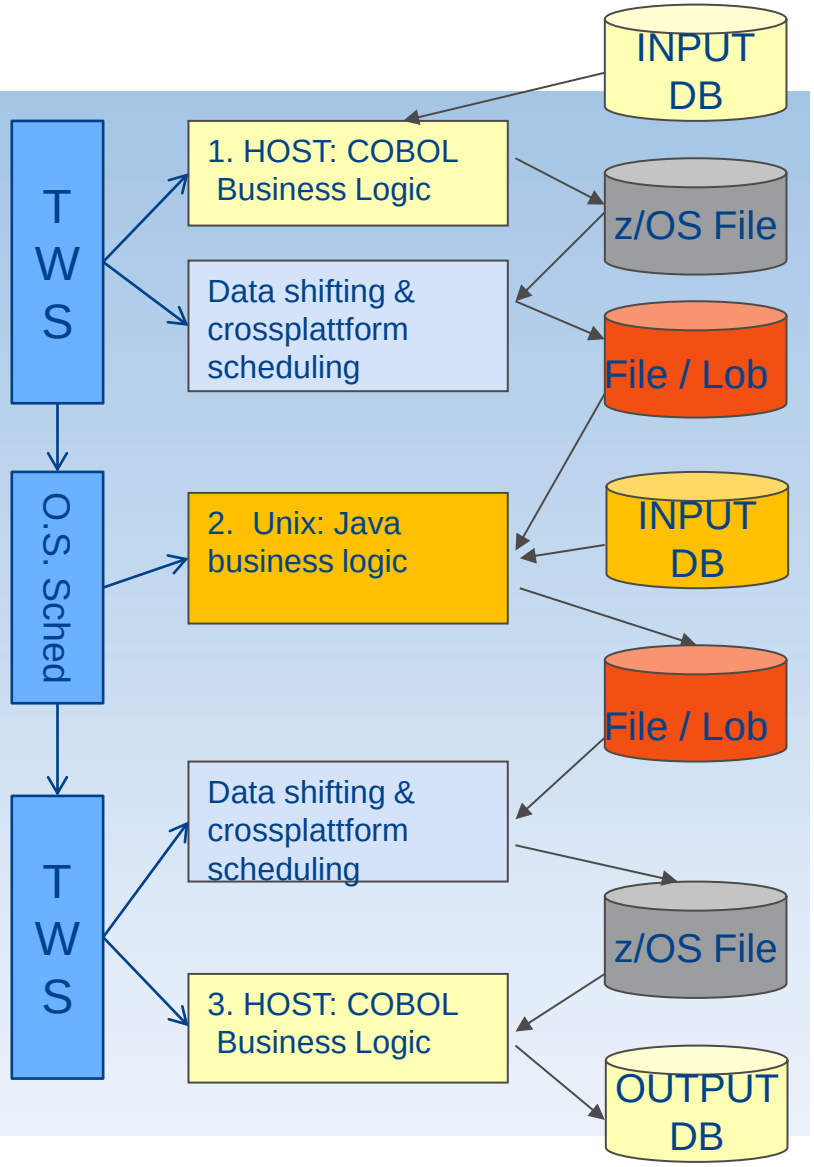
- Mainframe COBOL…
  - Core Applications
    - 776.000 Jobs/day
    - 80% over night

- Open Systems : Java
  - Core Application
    - 1.634.930 Jobs
  - BI
    - 3.785.542 Jobs
  - other
    - 223.000 Jobs

Currently more jobs on distributed systems than on the mainframe

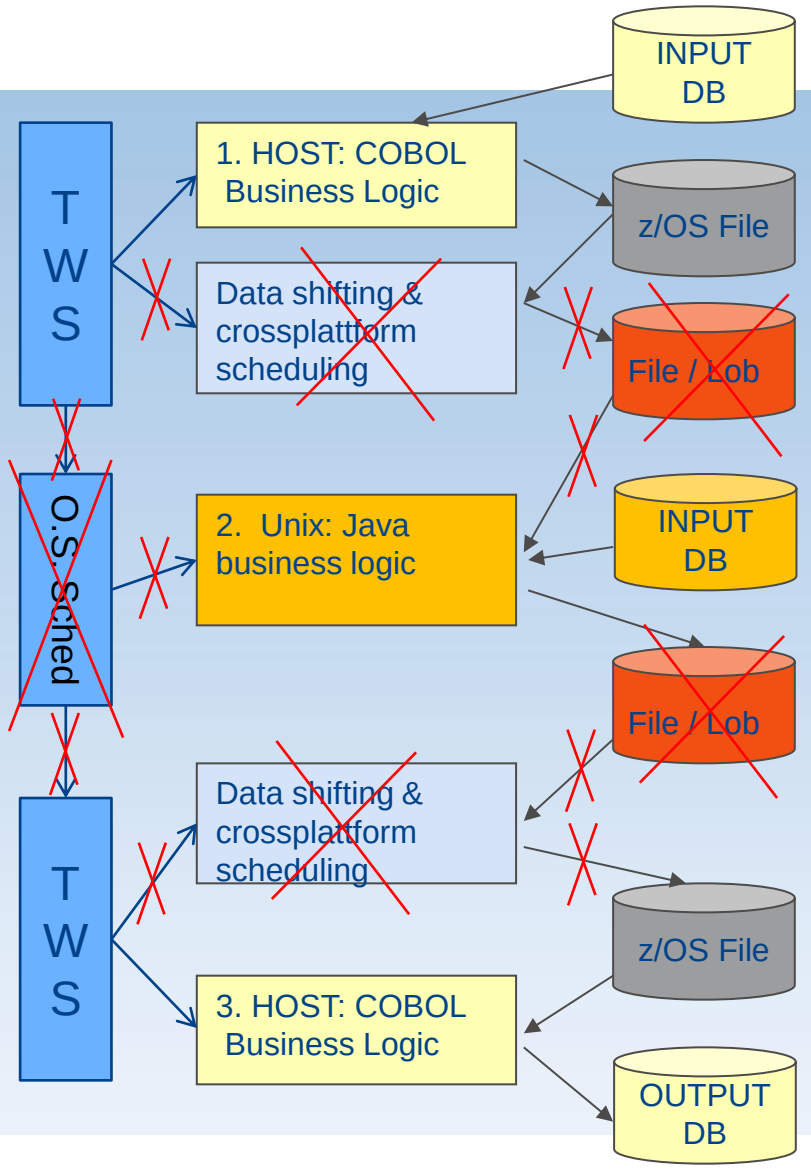
Batch Evolution: the results of “evolution”

- What happened typically in large companies
  - 1980 – today: batches on MVS, z/OS in HL language
  - since 199x also batches on Unix, windows… in HL lang.
  - since 200x batches in Java
- Some of these jobs spread over platforms have dependencies
- probably different Schedulers for different platforms
- coordinated Scheduling over all platforms required
- Data exchange over platforms required and or data sharing expected



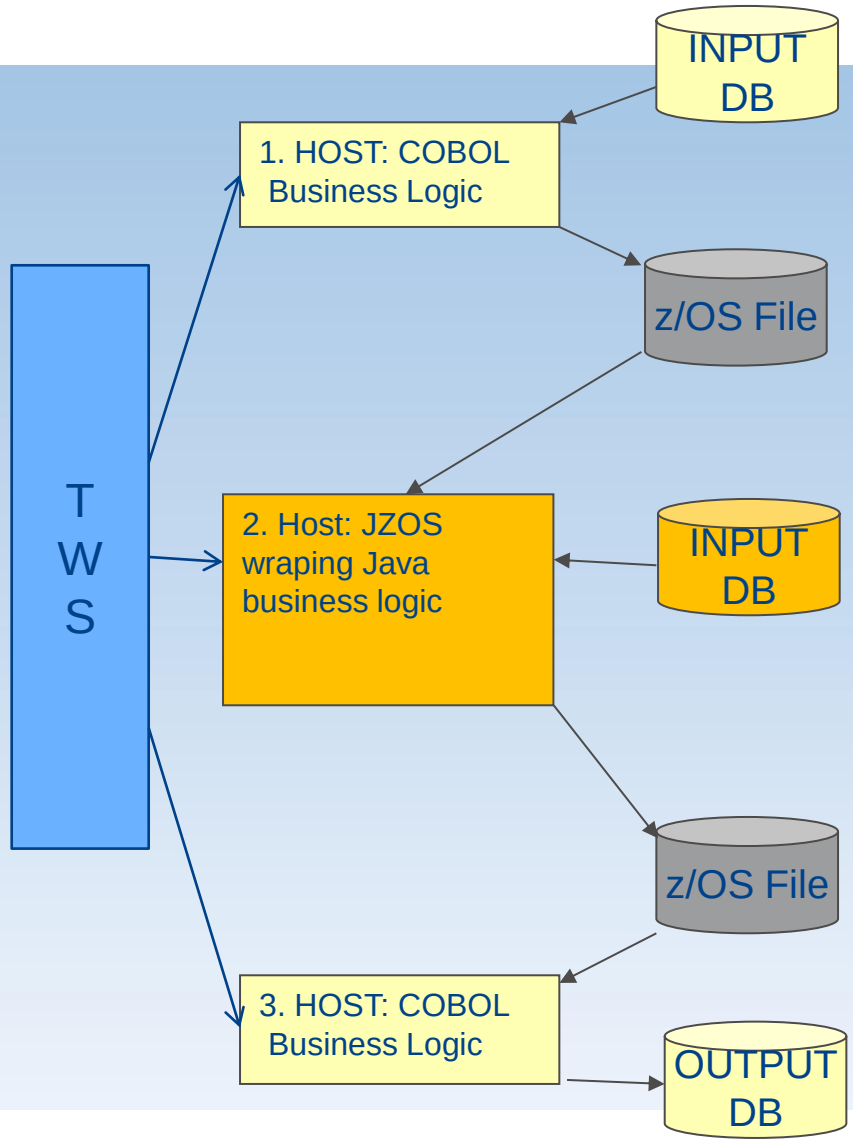
Why porting Java Batches to the Mainframe

- Reduce complexity by executing the Java Jobs under JZOS instead of Unix & Host
  - only one scheduler, no need to couple the schedulers
- Improve performance
  - no need to shift data over platforms, no format conversion…
  - Java application now is “near” to its data, no need for remote access and network traffic
  - Java batches eventually embedded in Container (Tomcat, Websphere…)
  - Higher throughput
- Improve efficiency of operating controls
  - everything is under single control (only Mainframe operating, no unix operation)
  - easier to configure (only TWS people are involved)
  - more effective in case errors (one person is responsible)



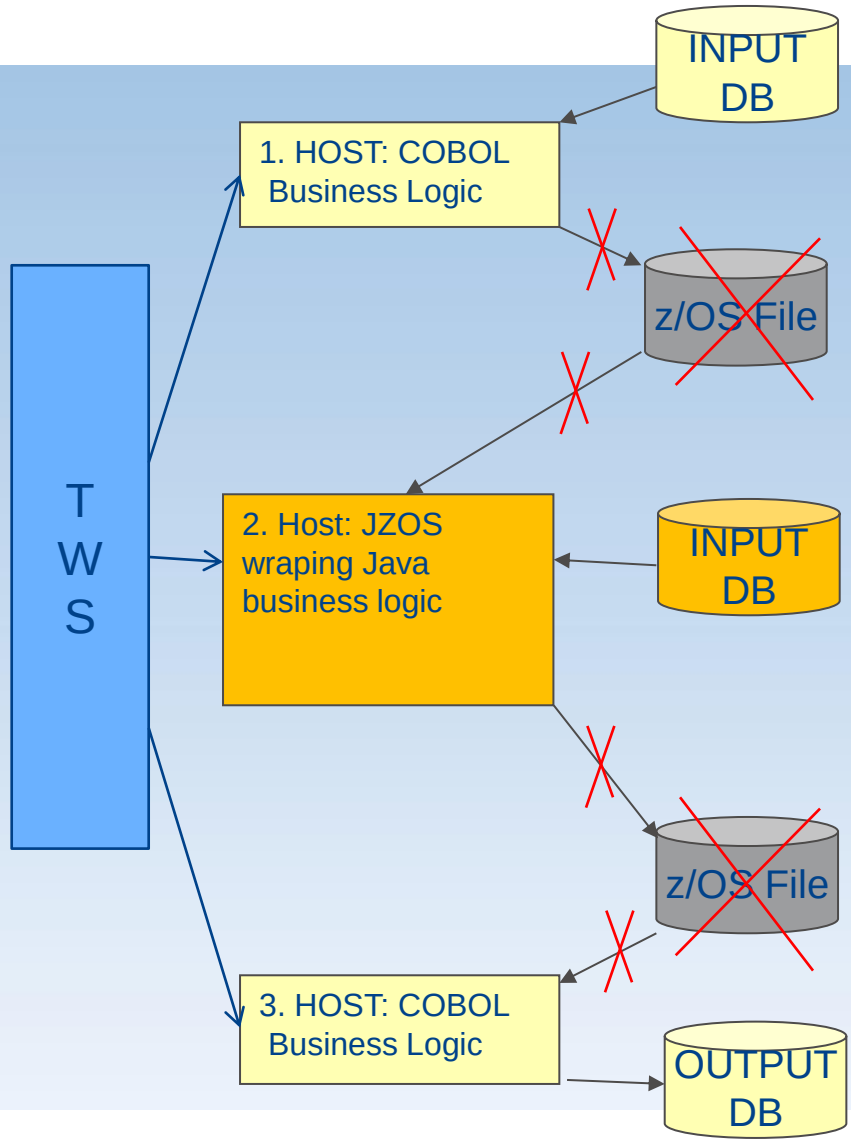
Why porting Java Batches to the Mainframe

- Reduce complexity by executing the Java Jobs under JZOS instead of Unix & Host
  - only one scheduler, no need to couple the schedulers
- Improve performance
  - no need to shift data over platforms, no format conversion…
  - Java application now is “near” to its data, no need for remote access and network traffic
  - Java batches eventually embedded in Container (Tomcat, Websphere…)
  - Higher throughput
- Improve efficiency of operating controls
  - everything is under single control (only Mainframe operating, no unix operation)
  - easier to configure (only TWS people are involved)
  - more effective in case errors (one person is responsible)



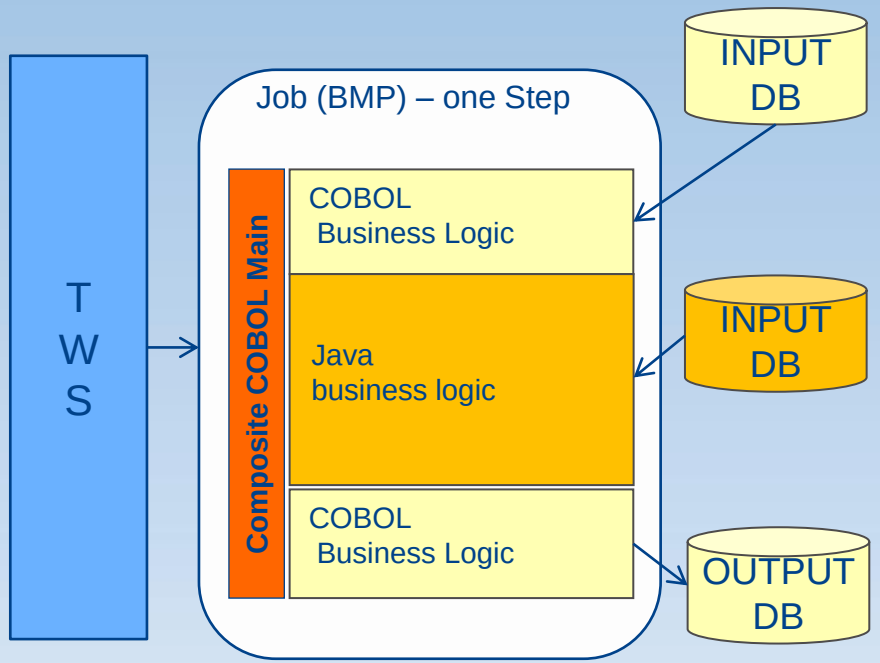
Combining High Level Language like COBOL and Java brings maximum efficiency

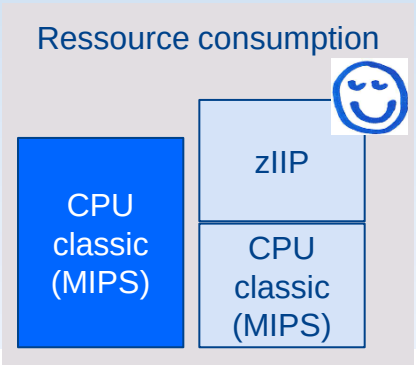
- Reduce again complexity by executing one program combining COBOL and Java in one Step as BMP instead 3 steps (one COBOL, one Java one COBOL)
  - take benefit of the overall UOW of IMS over COBOL and Java
- Improve performance
  - no need to persist data between steps
  - Data stay in memory
  - just overhead of marshalling (ebcdic / UTF16)
  - Higher throughput due to minimization of IOs (the best IO is no IO)
- Bring maximum benefit with data intensive Workload
- Use of application patterns specifically designed for high throughput
  - use the typical pattern of COBOL batch coding as base and adapt those for Java
  - careful/restrictive use of Hybernate and Spring

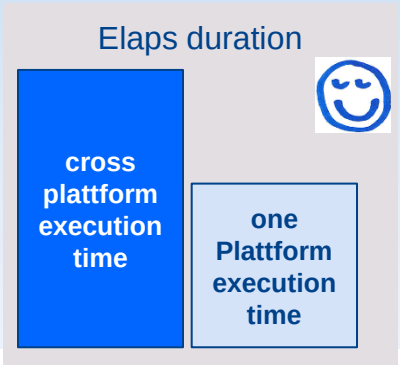


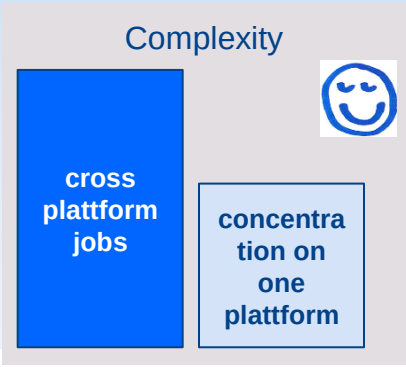
Combining High Level Language like COBOL and Java brings maximum efficiency (IMS-BMP)

- Reduce again complexity by executing one program combining COBOL and Java in one Step as BMP instead 3 steps (one COBOL, one Java, one COBOL)
  - take benefit of the overall UOW of IMS over COBOL and Java including IMS-DB and DB2
- Bring maximum benefit with data intensive Workload
- Use of application patterns specifically designed for high throughput
  - use the typical pattern of COBOL batch coding as base and adapt those for Java
  - careful/restrictive use of Hybernate and Spring



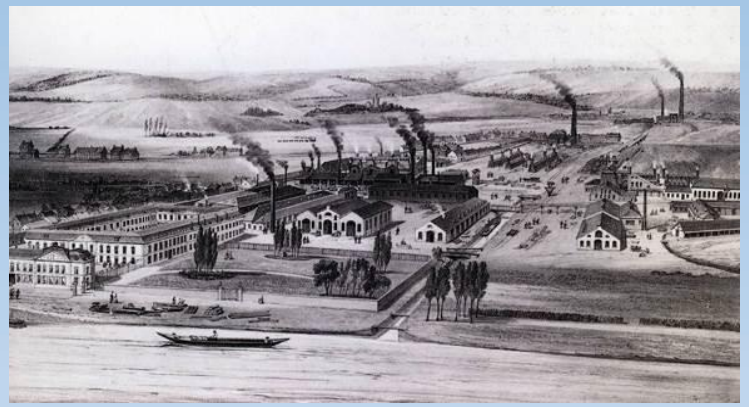






Use Cases / Experience

- JZOS in use for productive purposes since 2008
- JZOS is used for sensible applications since 2011
- JZOS and BMPs in use for mission critical applications since 2013
  - huge amounts of Data is manipulated
  - throughput is very important
  - strict deadline for the business
- Over time other eligible applications will be deployed on the Mainframe by using JZOS or BMPs application
  - new Applications
  - existing Application moved from distributed systems



➢ **Optimize throughput, reduce elapse time, reduce overall resource consumption**

Fiducia: Driving Efficiencies with IMS

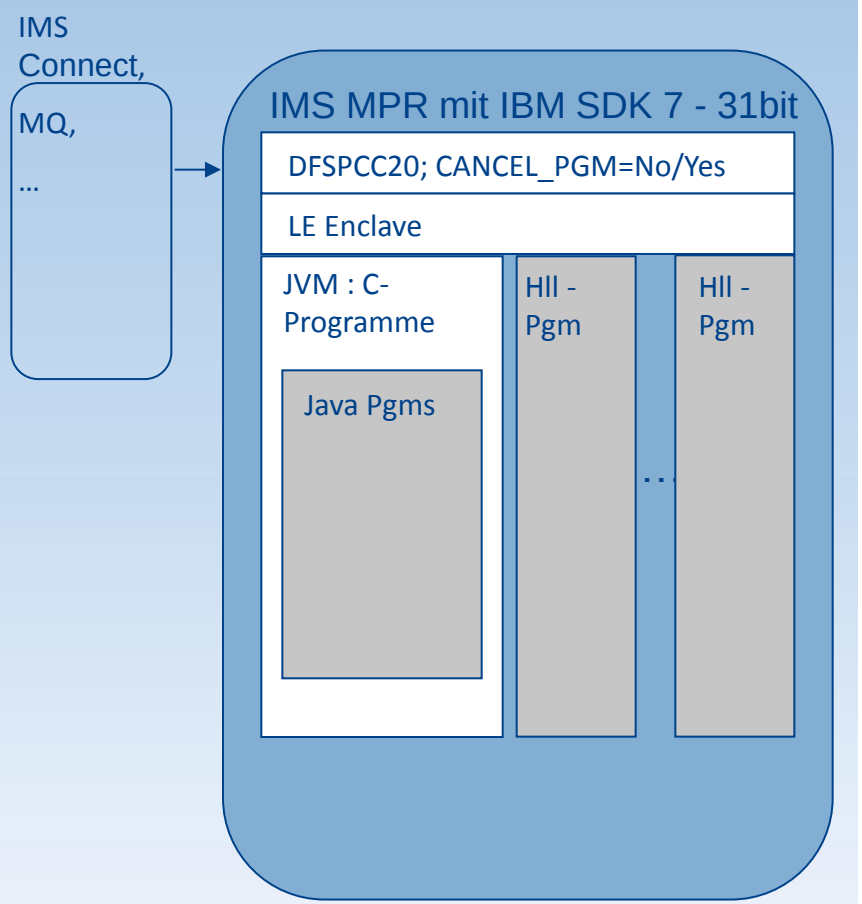
1	Fiducia overview
2	Business needs, Added Value of Java/Z
3	Java Batch Experience (JZOS and BMPs)
4	**Java-IMS: Experience**
5	Impact of resource consumption
6	Technology outlook (Fiducia / IBM)

Technology: the potential of IMS-Java

- In MPRs: the main program is COBOL
- Delivered Java functionality
  - Support of Java SE (31 bit) which covers the scope of building “Business code“
  - JVM is resident in the MPR for its hole life
- Inter Language Communication (ILC) between COBOL and Java allows: COBOL calls Java and Java calls COBOL (cascading)
  - Java and COBOL runs within the same transaction (UOW) with Database support for:
    - DB2 from COBOL (static SQL) and Java (JDBC Type-2)
    - IMS-DB from COBOL and Java
    - MQ access from COBOL and Java (available im MQ V8)

IMS-Java: the JVM is persistent in the IMS-MPRs with 2 major processing paradigms

- 2 different runtime CFGs of the MPRs
  - High optimized behavior (option CANCEL_PGM=NO)
  - Standard behavior (option CANCEL_PGM=YES)
- Activating the JVM requires specific configuration
  - POSIX(ON)
  - XPLINK(ON)



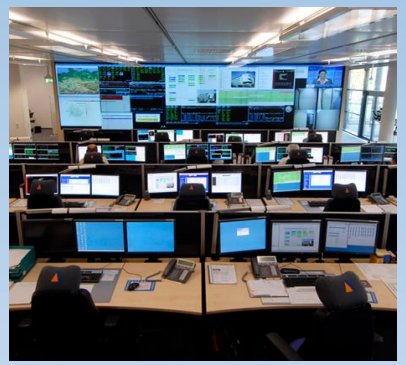
Enable your production environment: How to run your existing code in Java enabled MPRs

- All programs need to be compiled with Enterprise COBOL
- All modules must be amode31
- Deploy your high performance transactions to CANCEL_PGM=NO MPRs
  - application behavior should stay unchanged in terms of response time and resource consumption
- Deploy your standard performance transactions to CANCEL_PGM=YES MPRs
  - the LE-enclave will no longer be rebuilt after each TX, so memory is not cleaned after TX-End. The use of storage(00) might be applied
  - test carefully your programs before moving in production
  - application behavior should stay unchanged in terms of response time and may change in terms of resource consumption depending on TX complexity
- Memory:
  - Region Size: 0,8 - 1,5 TB
  - le settings: Hihg Heap values (120MB, 32MB)
  - JVM Footprint: 80MB

Increase the region size!

Enable your production environment: are you ready?

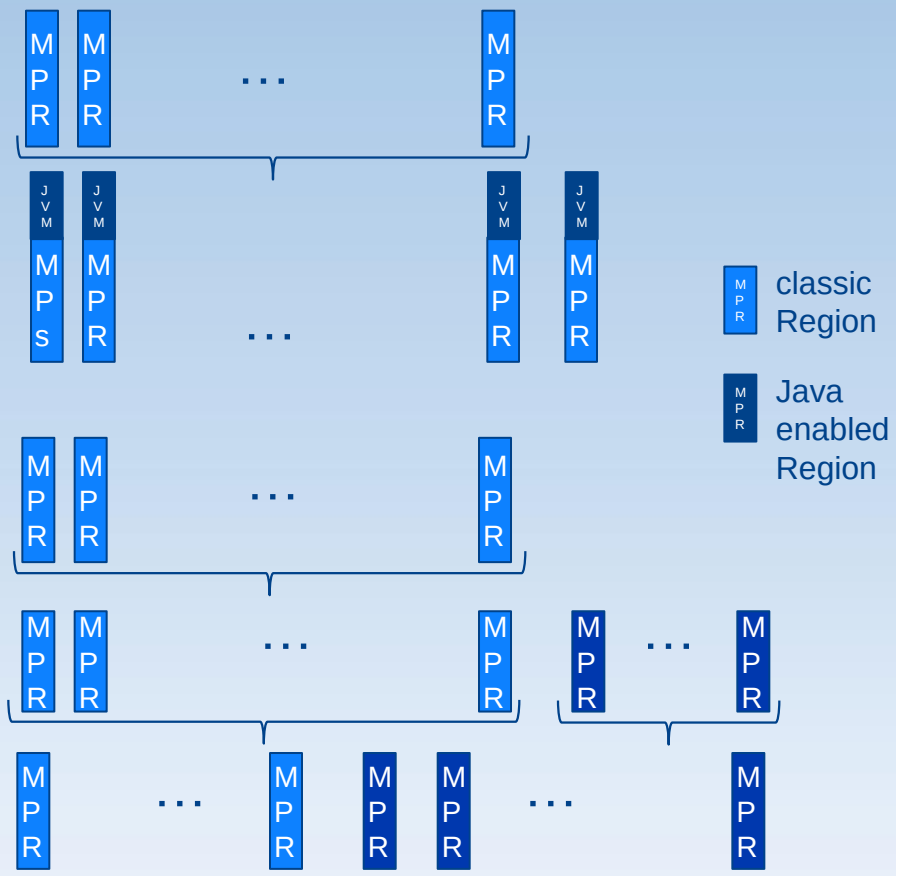
- Three major considerations
  - Provide education for the IT Operational team to have them understand how the new technology works
  - Enhance monitoring, accounting and error Handling to take care of the infrastructural changes
    - Additional data need to be captured and observed (JVM, inside o JVM, JDBC…)
    - new artefacts need to be monitored: USS, zFS…
    - new kinds of errors need to be captured
  - Take benefit of existing Knowledge in your company by mixing the Mainframe and Open-Systems people



➢ **Introduce hybrid teams, teams with dual skill**

IMS-MPRs: Strategies to introduce Java: Horizontal or Vertical

- HORIZONTAL: means first enable all regions and all application to process the transactions in Java enabled regions
  - ➢ java functions consumed by everyone can be implemented
- VERTICAL: build functional blocks used in dedicated applications
  - – introduce Java without need to enable “all” transactions and “all” regions.
  - ➢ Current strategy based on acquired experience: vertical approach.



Java/Cobol Mix: How to activate your applications: classpath issues

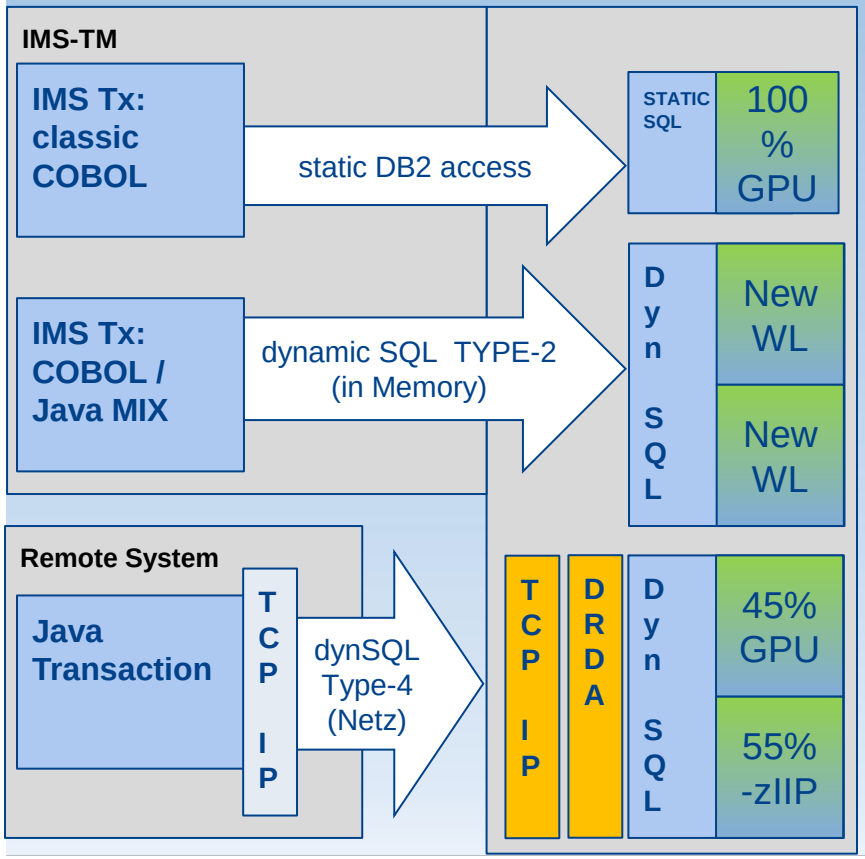
- JAVA and COBOL must be activated at the save time.
- No particular issues as long as the Classpath is composed of a few jarfiles,
- Never the less loading a Classpath is a cost intensive issue
- So the classic classpath mechanism is not realy adapted for short running processes (JZOS or BMP), especialy when the classpath contains many jars
  - starting a JVM with about 1000 Jars in the Classpath cost about 1 Min elaps.
- We opted for a self defined classloading mechanism
- Fiducias solution: boost up the classloading phase:
  - Full content of classpath is defined at time of deployment (opening each jarfile and building a list of all packages)
  - when a job is started, our own Classloader uses this List.
  - Large Classpathes can now be handled in an efficient way

Avoid large classpathes when possible, if not build your own alternative

Fiducia: Driving Efficiencies with IMS

1	Fiducia overview
2	Business needs, Added Value of Java/Z
3	Java Batch Experience (JZOS and BMPs)
4	Java-IMS: Experience
5	**Impact of resource consumption**
6	Technology outlook (Fiducia / IBM)

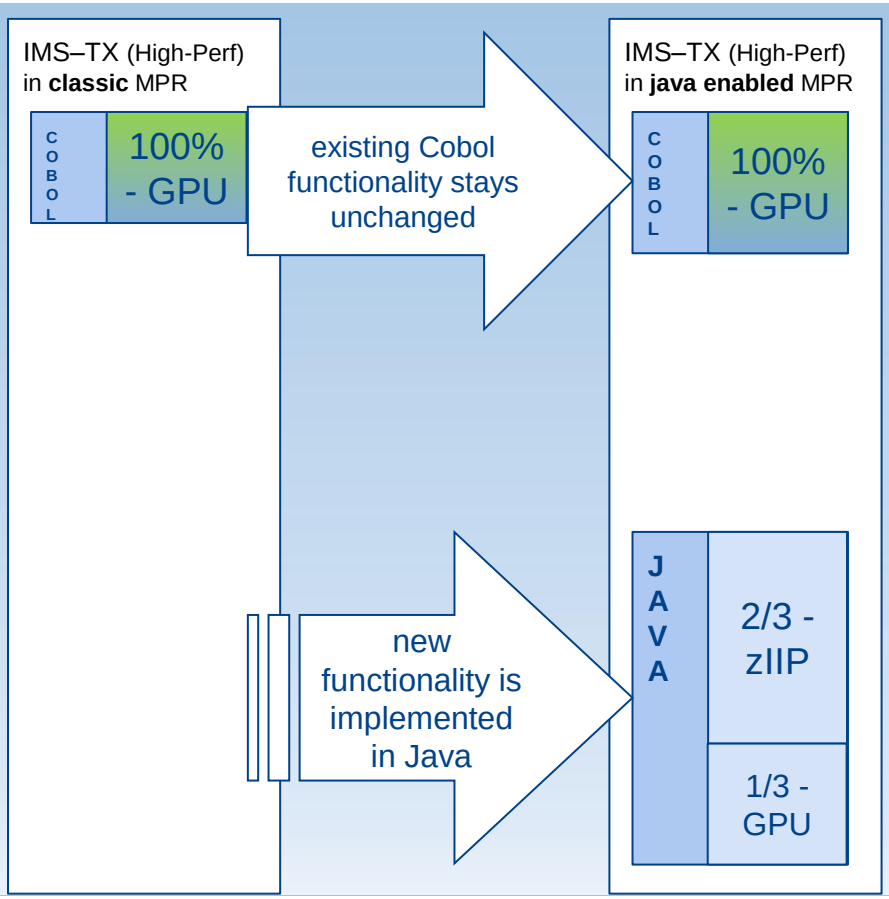
Resource consumption evolution when using dynamic SQL via JDBC instead of static SQL



- Dynamic SQL is factor 2 to 4 more resource intensive als static SQL
- This is accepted with remote applications using DB2.
- Using java/Cobol Mix in IMS instead remote leads to lesser resource consumption in DB2
  - no Network traffic (TCP), no DRDA
- Ressources for remote initiiated SQLs (Type-4 / DRDA) are splitted in 45% GPU, 55% zIIP (Fiducia mesurements)
- Resources consumed in DB2 with JAVA/COBOL Mix may be elligible to new pricing models (new Workload)

From a DB2 Ressource consumptions perspective, hosting a jdbc application in IMS instead remote is more effective

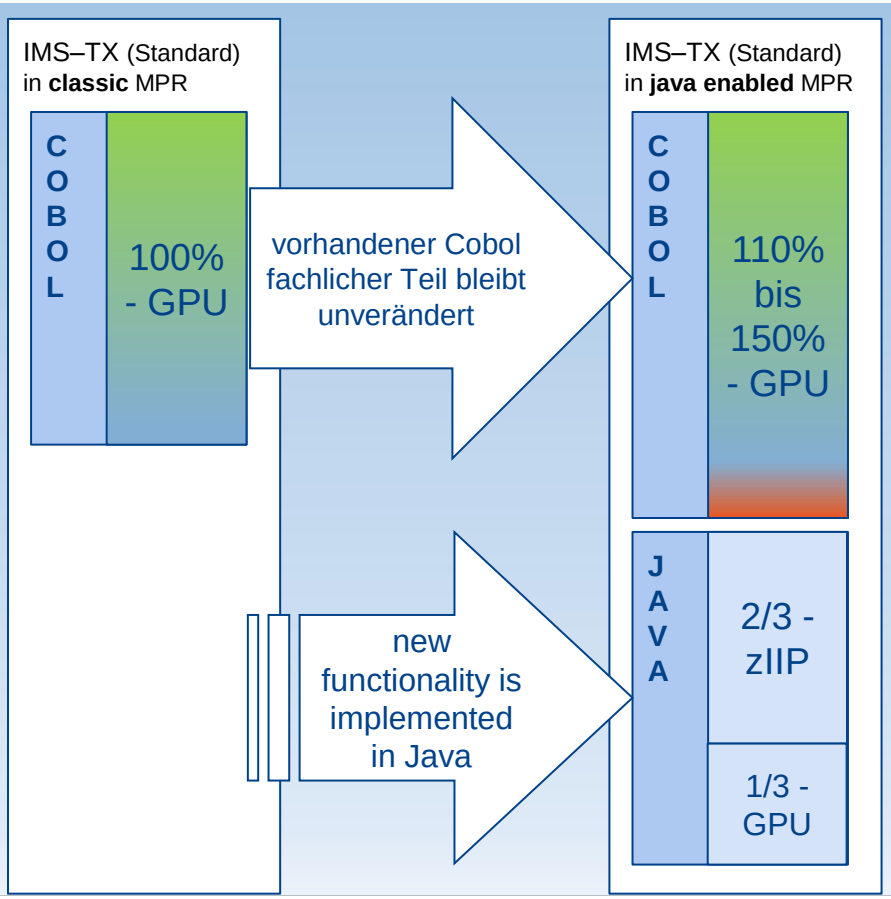
Evolution resource consumption and costs when processing a High Performance IMS-Tx extended with Java.



- Ressource consumption of existing COBOL stays unchanged in java enabled MPR
- Ressource consumption of a Java implementation is usualy splitted in 1/3 on GPU und 2/3 on zIIP (pessimistic view)
- Assuming Java is a factor 3 more ressource intensive than COBOL
  - the resource comsumption on GPU will be the same as a COBOL implementation
  - if not enougth zIIPs capacity available for the rest, it goes also on GPU

Introducing Java in High Performance Tx‘s has no impact on the resource consumption of the existing COBOL Code

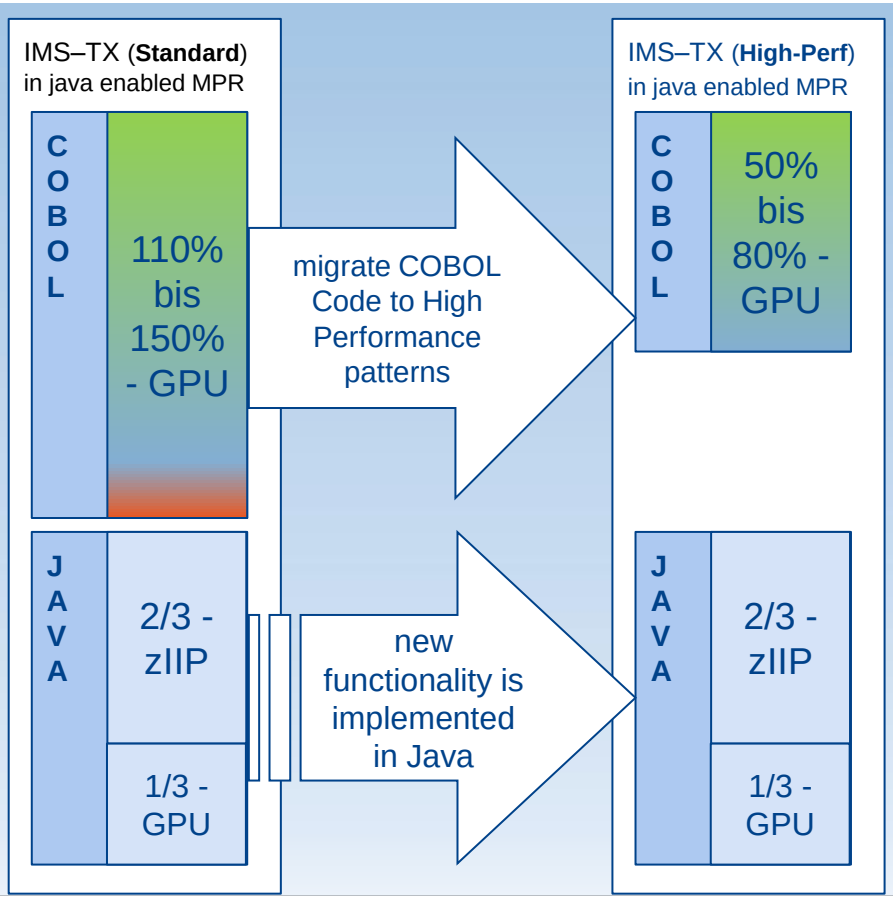
Evolution resource consumption and costs when processing a standard Tx extended with Java.



- Ressource consumption of existing COBOL grows in java enabled MPR
  - COBOL LOAD / DELETE instead of enclave Rebuild
  - STORAGE(00) to initialize memory
- Ressource consumption of a Java implementation is usualy splitted in 1/3 on GPU und 2/3 on zIIP (pessimistic view)
- Assuming Java is a factor 3 more ressource intensive than COBOL
  - if not enougth zIIPs capacity available for the rest, it goes also on GPU

Using Java in Standard Regions leads to GPU consumption increase.

Effects of migrating a standard Tx to a High Performance Tx.



- Migrate the existing COBOL Code to be elligible for processing in High Performance MPRs
  - Adapt the existing COBOL to be reentrant/reusable
- Resource consumption on GPU will shrink far below ots original consumption as Standard Tx
- Migrate only a tansactions if it will bring an ROI
  - have an impact on the Peak time (reduce costs $$$)
  - that impact severely your environment (eliminate „GroundLoad“).
  - Txse that are massively consumed

Examine each elligible TX, migrate only those that may generate an ROI

Fiducia: Driving Efficiencies with IMS

1	Fiducia overview
2	Business needs, Added Value of Java/Z
3	Java Batch Experience (JZOS and BMPs)
4	Java-IMS: Experience
5	Impact of resource consumption
6	**Technology outlook (Fiducia / IBM)**

Items on Fiducia's and IBM's roadmap:

- The Technology has reached its maturity
- IBM and Fiducia still work closely together to enhance this technology
  - current map: transactional JMS based MQ integration
    - delivery in these days
  - move forward in the delivery of valuable enhancements focusing on
    - response time
    - operational aspects

➢ **Maturity is reached, enhancements are ongoing**

“The Show is going on”

- one subsystem is widely enabled with prototypes
  - 118 from 280 Regions are Java enabled
  - 72% (1.222.000 peak hour) of the executed TXes. used the prototypes
- First „Low-Load App“ (build with Fallback) in Prod since Q3/2014
- Second App „middle Load“ (build without fallback) will go in Prod in Q3/2015
- Adaptation of our development and deployment landscape to take benefit of this new technology is nearly finished
  - patterns, lifecycle
  - Industrialization of the hole lifecycle

All prerequisites for an industrial use of Java on Z are done

Vielen Dank