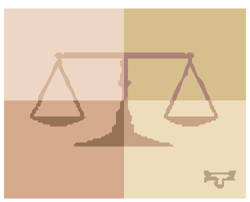
IT Infrastructure Systems Architecture

Server Platform Selection
- Workloads, Architecture, NFRs
why IT infrastructure matters



David G Heap
Executive IT Consultant
IBM Systems and Technology Group
david_heap@uk.ibm.com

Prologue...

- This is a continuing work in progress and based on sample findings from specific engagements in 2004. This includes a very large strategic project to replace a major global legacy application with WAS/DB services. All the examples are real but anonymous. Unless indicated otherwise, they are based on sample data collected by the author from a variety of major large enterprises during 2004

- The main objective of this presentation is to explain a tentative **approach** to IT systems infrastructure (in a data centre) by understanding a fairly well-defined major service / application / workload, creating a baseline scenario, evaluating various server platform alternatives against the baseline, and recommending specific platform reference architectures / solutions

- This rapidly evolving area is "IT Infrastructure Systems Architecture"

- This short presentation is a very rapid gallop through the approach, with many backup charts (not presented today) based on real engagements, factual examples and case studies

Agenda

Background and Method Description

- Business components (BCM), IT infrastructure components
- Outline elements of a method and an iterative IT consulting approach
- Black box basics, Non-func. reqts, juke-box example, workload types

Case Study: Global Application - Strategic Server Platform Selection

- Major drivers; size, db & java pathlength, availability, growth, batch
- Baseline data centre scenario, technical options, sample platform solutions
- Preliminary filtering using selection criteria, scoring, weighting
- Scaling and positioning of the 'final candidates'
- DB engine, Java engine, application architecture, COBOL transaction engine
- Sample recommendations - standards, initial platform positioning
- Next steps...

Many Backup Charts

- *Many useful real examples, facts and thought-provoking case studies*
- *DB and J2EE standards and Java performance engineering*

A **Business Component** is a part of an enterprise that has the potential capability to operate semi-independently, as a separate company, or as part of another company

A Business Component

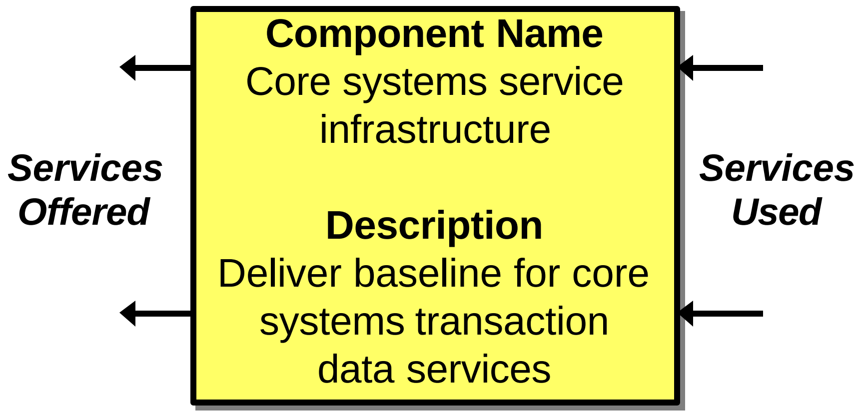
- is a 'black box'. Users & customers don't need to see the business activities that are inside it
- has discrete boundaries, defined by the services that it uses as inputs and offers as outputs
- can (mostly) be neatly separated at reasonably clean, logical 'cleave points'
- includes the resources; $$, people, technology and know-how necessary to deliver value
- has attributes and targets, such as cost, revenue, importance to the business, etc.
- is usually inter-connected to other business components to form more complex business services and processes

Business Services

- are the goods or services that a business component offers to other business components and/or external parties

So, what about an IT Component Based **IT Infrastructure** Model?

An IT service component is a group of cohesive IT Infrastructure services supported by appropriate information applications and data services, IT management processes, an IT organisational structure and IT targets and performance measures.



surely, all these are the main components of an IT Infrastructure architecture...

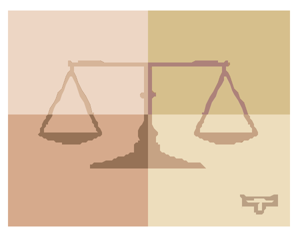
Simplify the IT operating model:

- develop an IT application/workload model with a small number of individual major components (say 20-50 in a large enterprise)
- identify which are core and non-core IT infrastructure components/services
- understand the true IT component/service costs and benefits and where the balance lies
- develop an appropriate operational IT strategy to support the corporate business strategy:
  - centralise or distribute IT services?
  - re-engineer, automate IT services?
  - **select technology platforms/standards**
  - inhouse, outsource, co-source?
  - onshore, offshore?
- drive the IT transformation by a core set of identified high priority projects & initiatives

Break the IT infrastructure into logical bite-sized chunks. Develop and deploy the appropriate IT strategy to support the overall corporate IT aims

IT Infrastructure - Platform Selection - Outline of Approach

- 1. Select a service/component/workload/application (eg email, core application)
- 2. Identify the component's key characteristics (eg Domino constraints)
- 3. Agree a baseline scenario (eg Domino campus server for 40,000 users)
- 4. Develop an application service questionnaire, interview service delivery managers to document current service functional & non-functional characteristics
- 5. Build a baseline service scenario for a multi-year (eg 3-5 year) comparison
  - understand future functional and non-functional requirements
  - develop realistic platform solution configuration options (eg i, z, p, Blades, storage etc)
  - establish and validate initial platform sizings
  - use standard costs for each configuration (hardware, mtce, software, people)
- 6. Validate filtering criteria, scoring questionnaire, do scoring interviews
- 7. Develop and validate weightings, do 'what-if' sensitivity analysis
- 8. Do detailed "Top 3" solution evaluation - constraints, trade-offs, business cases
- 9. Create strategic IT server infrastructure architectural roadmap
- 10. Recommend specific reference server architectures, with a rationale for each
- 11. Identify next/immediate actions



baseline scenario

eg: campus, 40,000 users

characteristics

volumes, growth
statefulness
availability
security, volatility
cost and value per tran.

workload

eg: email

platform solutions

eg: p-AIX, i-OS/400, z-OS
Blade-Windows, Blade-Linux

each platform solution must be technically credible and have an outline configuration with a real-world cost

'rules of thumb'

country specific (eg prices)
system configurations
performance metrics
utilisation metrics
staff productivity metrics

filtering criteria

eg: availability,
performance, IT process...

decision tradeoffs

$$costs, $$ benefits, $$risks

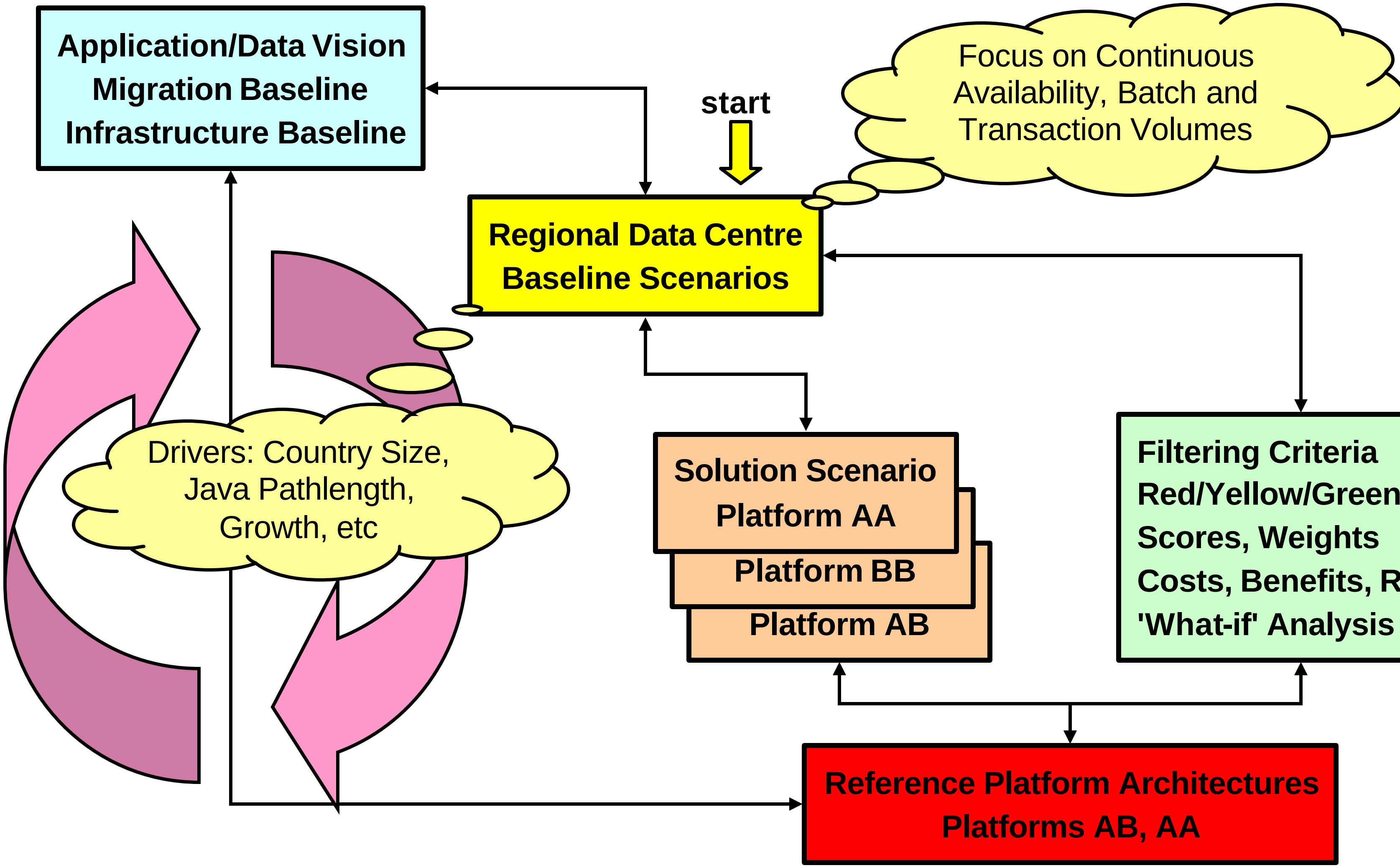
filtering/scoring
total cost model

ranked solutions

eg: platform A then C, with a rationale

rationale, ‘clip’ levels, 'what-if' and other major decision considerations

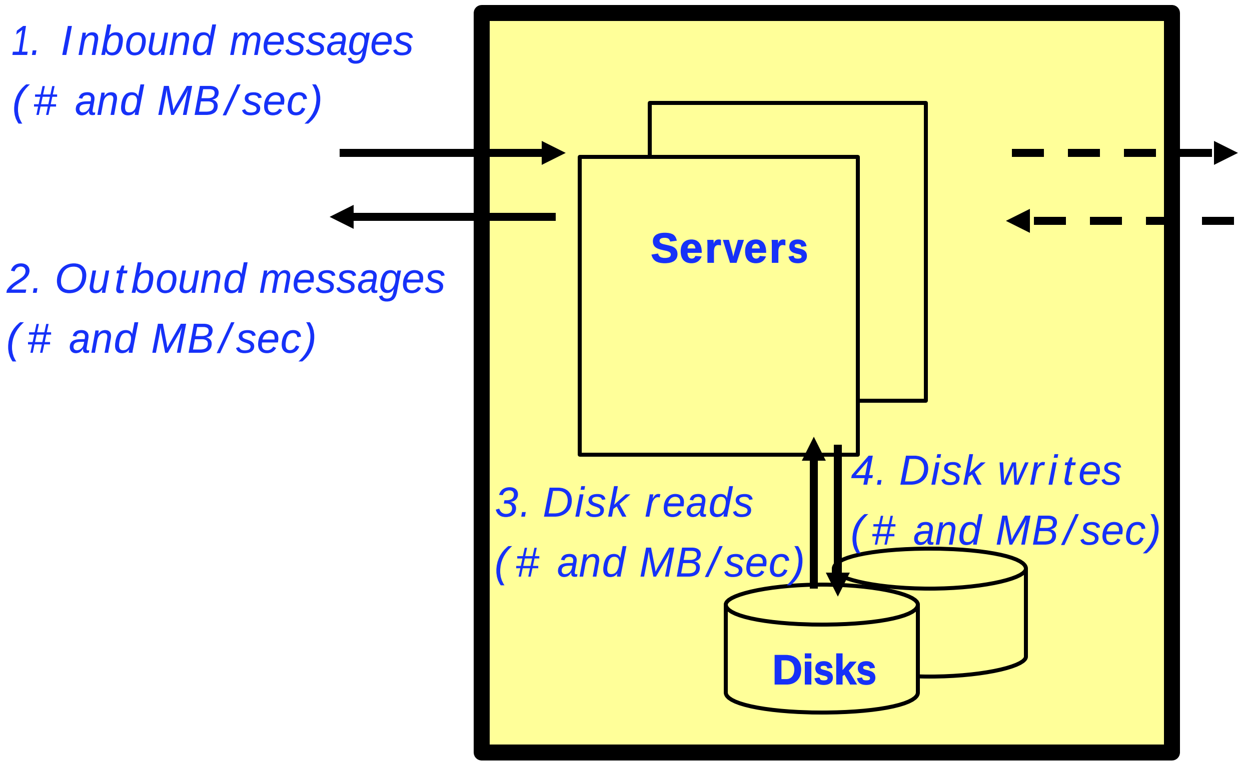
First Iteration of Method - Data Centre Baseline(s)



Initial focus on baseline data centre scenario then future/growth options

Systems Black Box - A Logical Building Block for Analysis

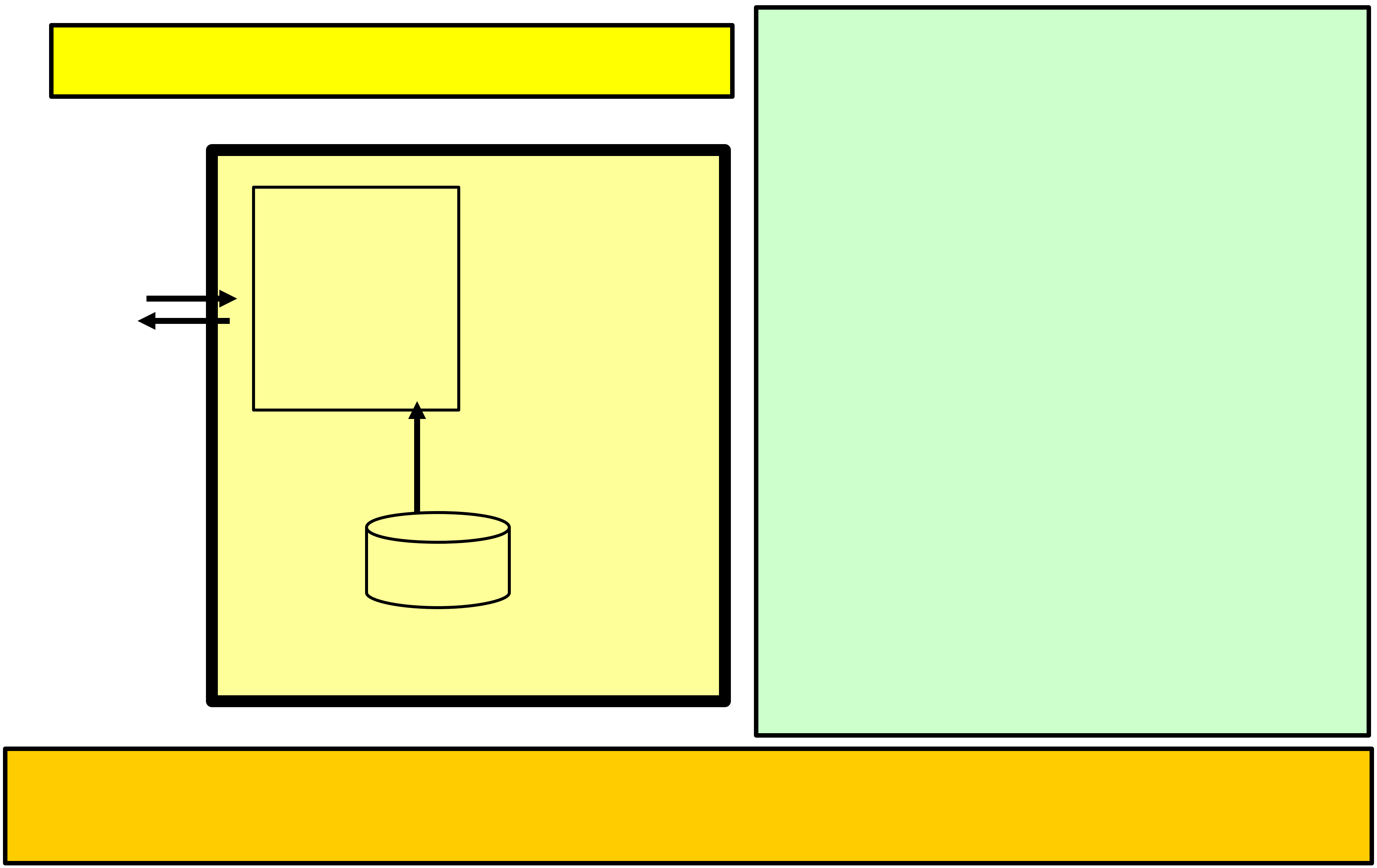
Primary IT Service - eg Core Application



Sample Non-Functional Characteristics

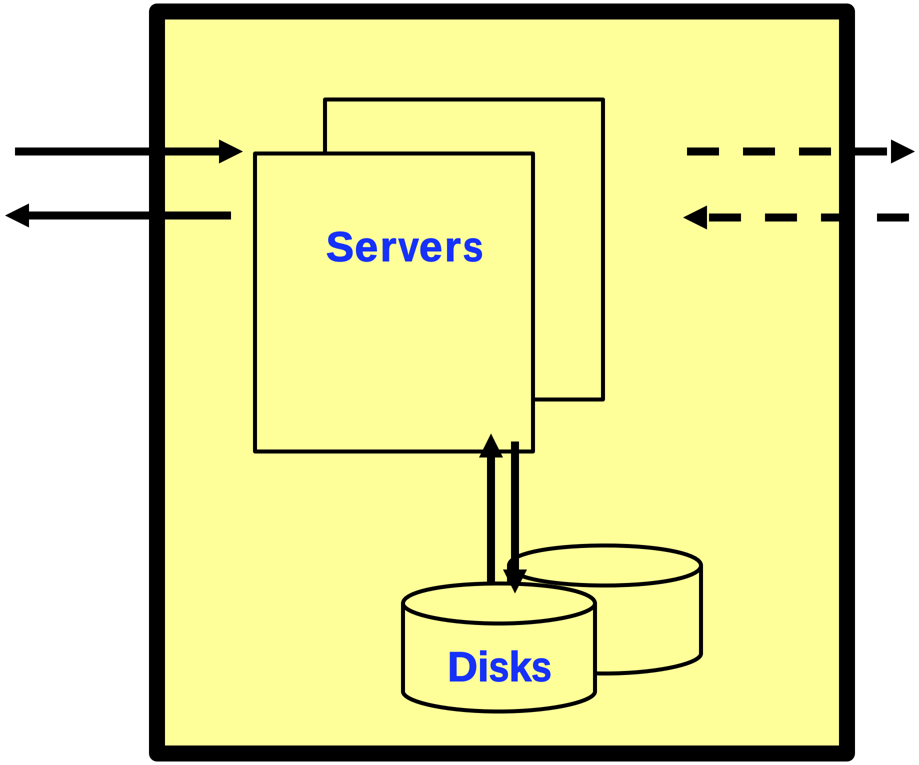
- Required Volumes & Growth
- Response Time
- Pathlength/unit of work
- Statefulness
- Data Integrity
- Scalability
- Service Quality
  - 'Normal' Availability
  - Scheduled Maintenance
  - Site Contingency
- Security
- Cost & Value/unit of work
- Code Volatility

Characteristics vary considerably by IT workload and computing style. For example, web serving, high performance computing, OLTP, audio streaming, collaborative



Systems Black Box Example - Personal Audio Juke Box

Systems Black Box: Examples of Major IT Services & Workloads



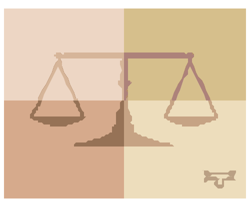
Primary Service/Workload

- SAP App & DB servers
- Core Systems OLTP
- High-Perf Computing
- HTTP Web Content
- eMail - Exchange
- Collaborative Workflow
- Data Warehouse & Marts
- Video or Audio Streaming
- Systems Management

Black boxes can be 'nested', for example, by separating out the application and database server characteristics. So, hybrid, multi-tier solution options can be considered

IT Infrastructure Systems Architecture

IT Infrastructure Architecture Study
(Objectives, Current Issues, Logical Topology, IT Drivers)



Example: Data Centre IT Infrastructure Study

- **Objective:** a short project to identify ~3 server infrastructure architectural options for a major new global WAS/DB OLTP application in a 'primary-backup' local pair of geo-regional data centres

- **Output:** an outline systems infrastructure of a geo-regional data centre running multiple country instances of a major new global WAS/DB OLTP application on a set of (front), middle-tier and backend servers.
  - include representative software stacks on each server (eg operating system, database software, transaction handling and j2ee middleware.)
  - considerations include country size (transactions & accounts), transaction characteristics, batch, coexistence, deployment options, costs, risks, skills

- **Platform Scenarios:** Several baseline components considered, in appropriate hardware & software combinations, for the application server and database server. Including, for example:
  - server platforms: z990-zOS, z990-Linux, i5/p5-AIX, HS40blades-Linux, storage,...
  - database software: DB2 UDB on LUW; DB2 UDB on zOS, Oracle 9i RAC...
  - transactional middleware: WAS, MQ, (CICS)...

Example: Major Issues with the Current Legacy Application

- Time & Cost to Deploy a New Release - the legacy core applications are very tightly coupled. Very significant central and regional testing is required plus local tailoring before full deployment. Rollout of a major release can take 2-3 years across many separate instances worldwide

- Application Architecture - new (modular) applications are being developed which have removed the need for many older (current) application functions and components. So, which functions will the next generation core application contain? Common services? Coexistence during rollout?

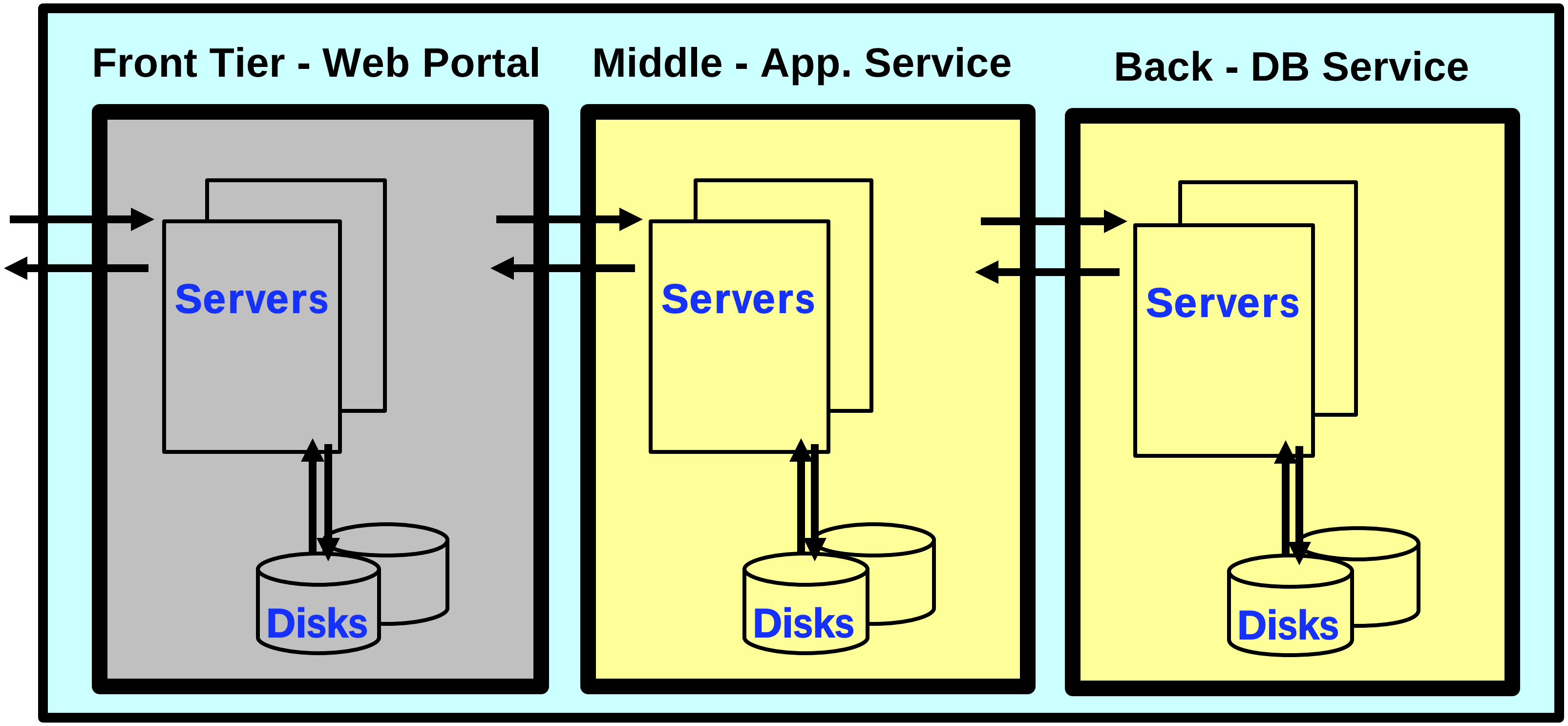
- Scale - there are often current design limitations. In many countries one of the biggest constraints is batch, especially month-end. Operational 'housekeeping' is a non-trivial task

- Platform - many legacy applications were designed and first implemented 20 years ago. Need to select an open strategic platform and to balance a variety of selection/decision criteria such as cost, QoS, risk and skills

What is the most appropriate future IT server infrastructure?
What about performance, cost, migration strategy, coexistence, risk, skills...

*this is an example for a specific customer application, and will vary by customer and application

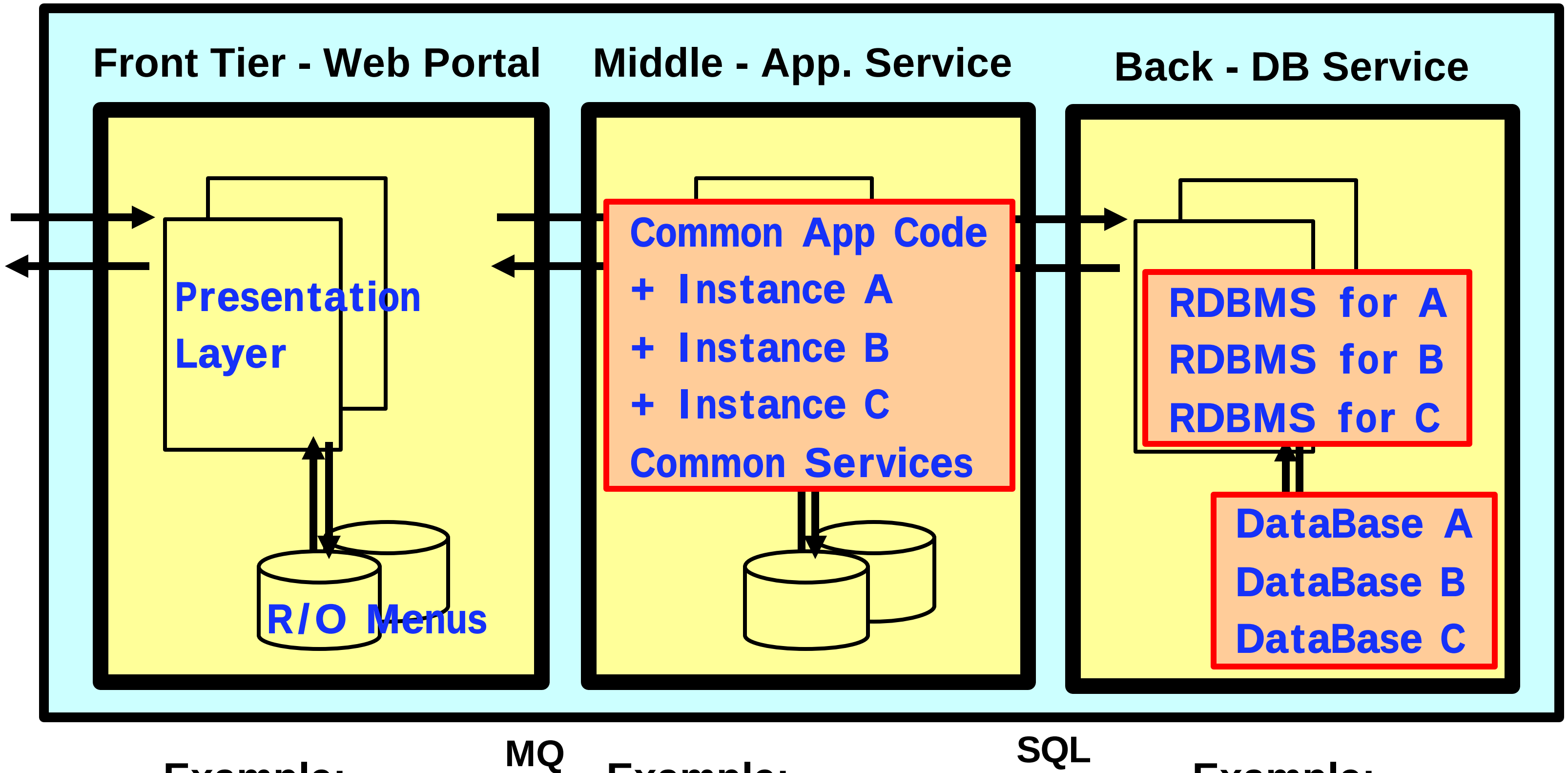
Logical Systems Components - An OLTP Application/Service



Example:	Example:	Example:
WAS, UDB (RO)	WAS	DB2
HS40 Blades-Linux	p570-AIX HADR	z990-zOS

Each box may have different non-functional requirements - especially topology, statefulness, availability, cost/tran, security, data integrity. Each must consider backup, failover etc and might be implemented as virtual servers on one physical server

Baseline Application Design - Logical Topology



Example:
WAS, UDB (RO)
Blades-Linux

Example:
WAS Business Logic
p570-AIX HADR

Example:
DB2 Master Data
z990-zOS

The design includes middle tier common core code with instance-specific application tailoring and separate DBs. Multiple application instances might be deployed into a single system image, but need to remain separable for possible future separate deployment

*this is an example for a specific customer application, and will vary by customer and application

Many Inter-Related Topics & Drivers

Sample A lication XYZ - Major Op ions and Decisions

'non-platform' ers

Reduce Ma ainties
drives -> r , PoC
drives -> asure t actions
drives - roduction a scale
drives - ge with IBM technology vision

Exploit hanging Platform Technologies
drives - need for portability of DB
drives - eed for portability of J2EE/WAS

Reduce ity of Server Management
drives -> standard components
drives -> a
drives -> co tions

- drivers

Jav lity
drives -> perf
drives -> baseline ho
drives -> focus on 'Top 10 ns
drives -> consider COBOL/C erim

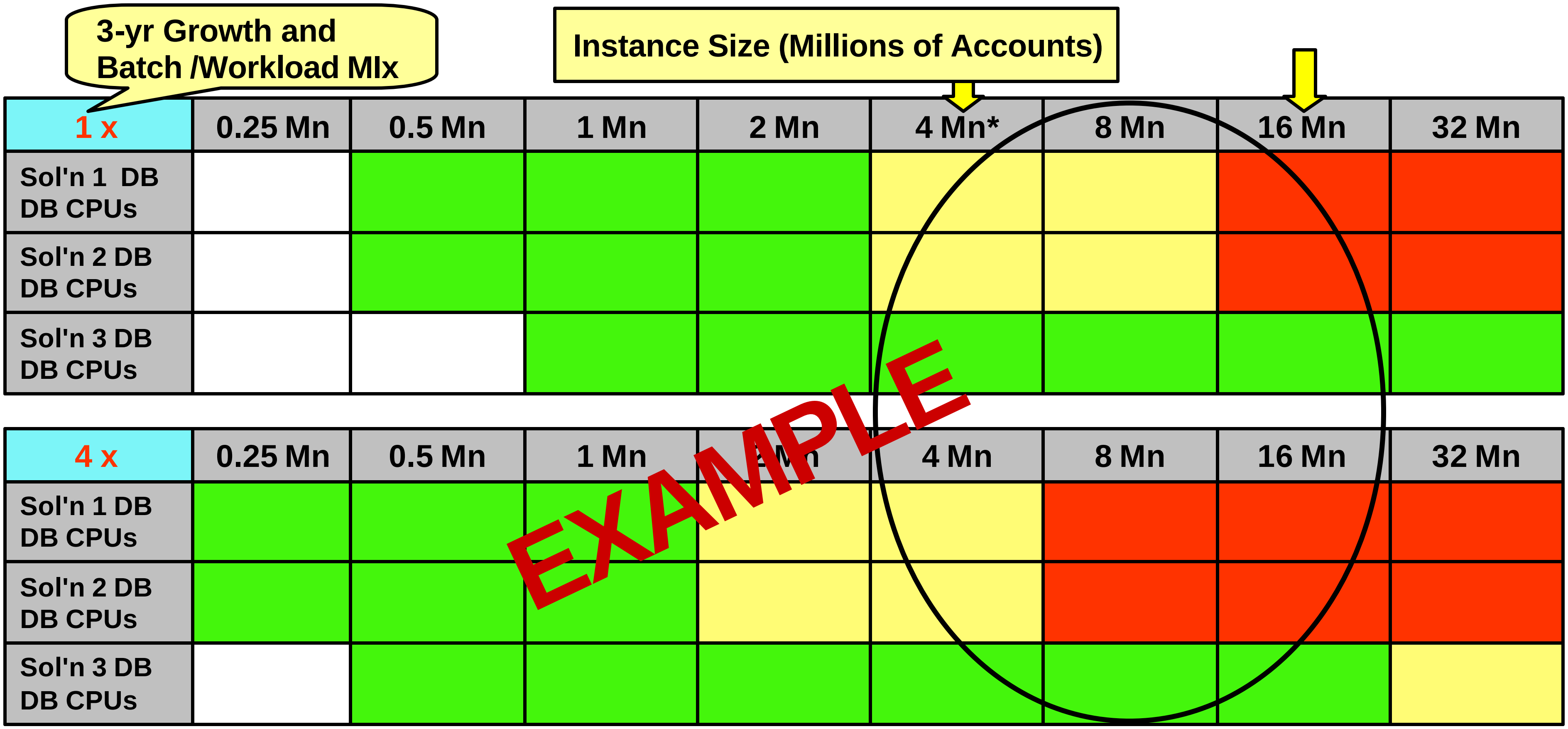
Business Need for Continuous Availability
drives -> business value assessme t
drives -> baseline vertical DB des
drives -> significant LOB busin ecision

Future Bat Proces ements
drives -> n flow)
drives - impact
drives structure

A coherent s e recommendations. A perspective on risk mitigation, skills, ion delivery speed, servi elivery cost, people productivity and the business value of continu s availability

This educational piece is intended for your use in selling. It is NOT a deliverable for your customers

Database Engine: Production Platform by Instance Size, Growth & Batch



*240 OLTP tps

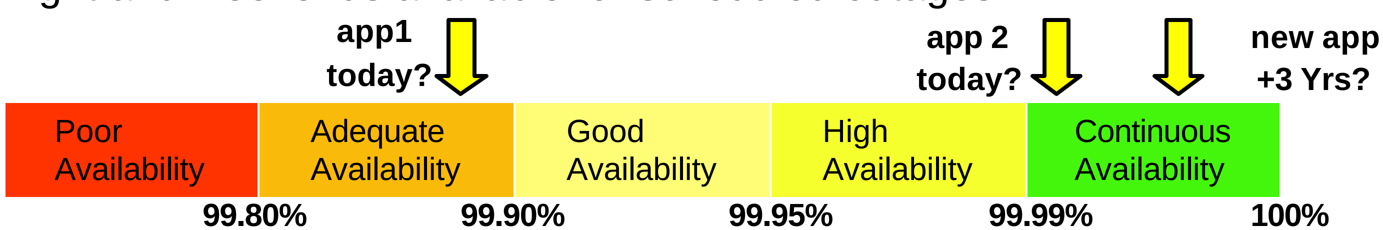
This is production only, today. It must be uplifted to include failover, dev & DR.
The baseline data centre cases (4Mn, 16 Mn) indicate Sol'n 3 as most likely solution
Adding growth and batch/mixed workload highlight current Sol'n 3 strengths

'white' is underutilised, 'green' is comfort zone, 'yellow' is current leading-edge, 'red' is unknown, 'black' is unattainable

Major Design Option: Continuous Application Availability

- e2e Design for Continuous Availability {eg zSysplex}
  - **application** design point - no single point of failure, one logical copy of data
  - usually no sysplex wide outage. but a 'perfect storm' can cause an unscheduled long outage every 2-3 years (eg cable, major database corruption, etc)
  - 'normal' incident impact (eg LPAR loss) - some in-flight transaction impact, majority continue unaffected, new transactions re-routed, in-flight trans rolled back
- e2e Design for '24x7' High Availability (Fast Restart) {eg pAIX HADR}
  - **application** design point - expect occasional failure but provide fast recovery

    eg 20-30 seconds incident, maybe 1 unscheduled incident per month, with best practices, - but there will also be 'perfect storm' unscheduled outages
  - 'normal' incident impact (eg LPAR loss) - all in-flight transactions impacted, roll-back, and queue all new until recovered
- Design for '18x5' High Availability {eg some current major applications}
  - **application** design point - minimal outages during office hours, rapid recovery
  - overnight and weekends available for scheduled outages



Drivers: Growth & Batch in Geo-Regional Data Centres

- Each pair of data centres assumed to have 'continental geo' scope
  - based on current deployment of systems/data centres
  - EMEA: ~4 timezones, AP: ~8 timezones, AG: ~5 timezones

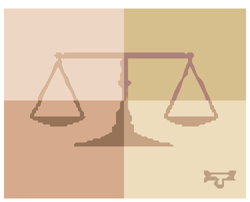
- This has very significant implications for batch and mixed workloads
  - different absolute time of day to close a country's end-of-day position
  - shorter critical batch window, increased requirement for concurrent batch & online work. with most significant impact at month-end (30-50% increase)
  - the new application design has different transaction and workload profiles. The increasing use of 'out of hours' channels (eg internet) accelerates the move towards processing mixed workloads
  - likely organic 30% annual growth rate in online and batch workload. 5 years growth drives a scalability requirement for >4x current capacity

Likely need for >4x Growth Capability beyond Today's Configuration

Example - Baseline Regional Scenario and Sample Solutions

(sample data centre scenario, technical component options, sample platform solution scenarios)

note: these are examples of a process. it is the process that is important, not the specifics of these examples



Sample Description of the NFRs of a Baseline Scenario

1. A pair of Geo-Regional Data Centres - up to 10 miles apart - Primary site for production. Secondary site for DR and development

2. Total of 4 Mn customer accounts. Country A - 2 Mn accounts; Country B - 1 Mn accounts; Image C - aggregation of 10 smaller countries - 1 Mn accounts

3. Total of 4TB for prod database (2 + 1 + 1). Total actual disk = 16TB (4 x 4TB)

4. 12 million simple OLTP Java trans/day (peak of 240 tps) at 2.5 x COBOL pathlength. Plus 250,000 (2%) complex 'choreographed' Java trans/day at 10 x Java utility transaction pathlength (an additional load of 20% on top of the simple OLTP Java transaction load). 30% annual growth in total transactions

EXAMPLE

5. Options: Continuous Availability **or** High Availability (Max. of 12 unscheduled disruptive incidents = **total** annual outage of 120 minutes. Max. of 12 scheduled one hour outages per year. Failover to disaster site in <4 hours with no data loss)

6. Batch operations. 10,000 jobs per night. 6 hour (9pm-3am) critical batch. 50% month-end uplift. 95% of systems batch resources used by a small number (5%) of long-running overnight jobs

Worldwide total: 12 million accounts, 36 million trans/day, 12TB database size

note: these are examples of a process. it is the process that is important, not the specifics of these examples

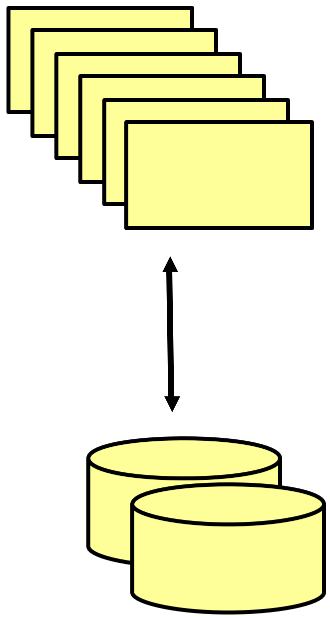
Server Technical Components - Sample Options

Middle Tier - Application Server Options

Txns	**MQ, WAS**	**MQ, WAS**	**MQ, WAS**
Opsys	**z-OS or z-Linux**	**p-AIX**	**x-Linux**
H/ware	**z990 zAAP**	**p570**	**HS40 Blades**

Back Tier - Database Server Options

DB s/w	**z-DB2**	**p-UDB**	**p-Oracle**	**x-UDB**
Opsys	**z-OS**	**p-AIX**	**p-AIX**	**x-Linux**
H/ware	**z990**	**p570**	**p570**	**x445**



Application Servers
Typically horizontally scalable

Database Servers
Typically vertically scalable
but also horizontal partitioned
options

Plus hybrid solutions - eg zOS-DB2 & pAIX-WAS by country size
Plus application code base path-length - J2EE vs. COBOL
Plus transaction type (simple, utility, complex) plus batch plus storage

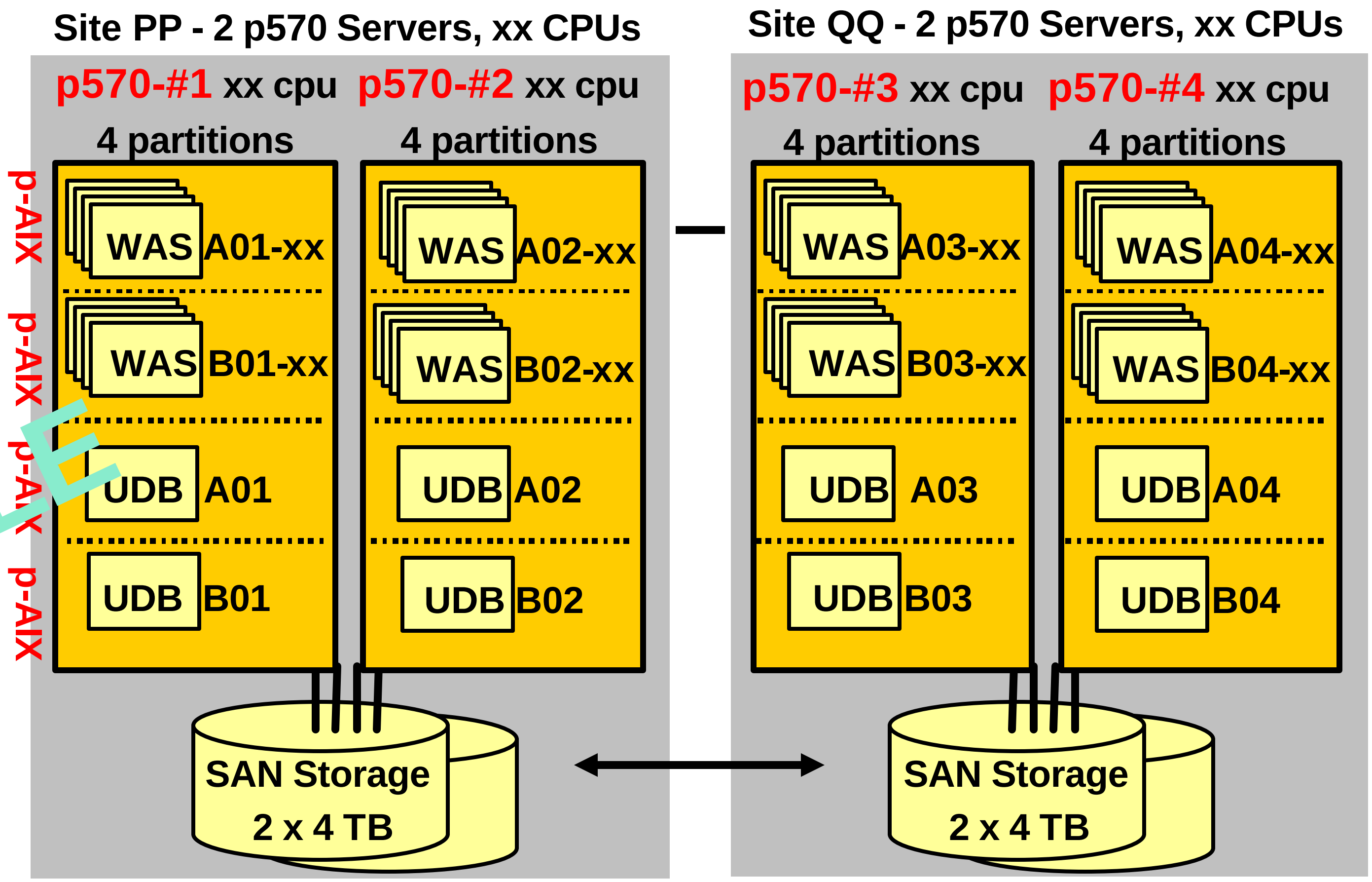
Sample Solution: p570; Java, WAS on AIX; DB2-UDB on AIX

2 Countries A & B
4 Primary LPARs
2 + 1 million accounts
6 + 3 million trans/day

rPerf	Mid	Back	Total
A			
B			
Totals			

AIX v5.2
UDB v8.1
WAS v5.1
Java JDBC

EXAMPLE

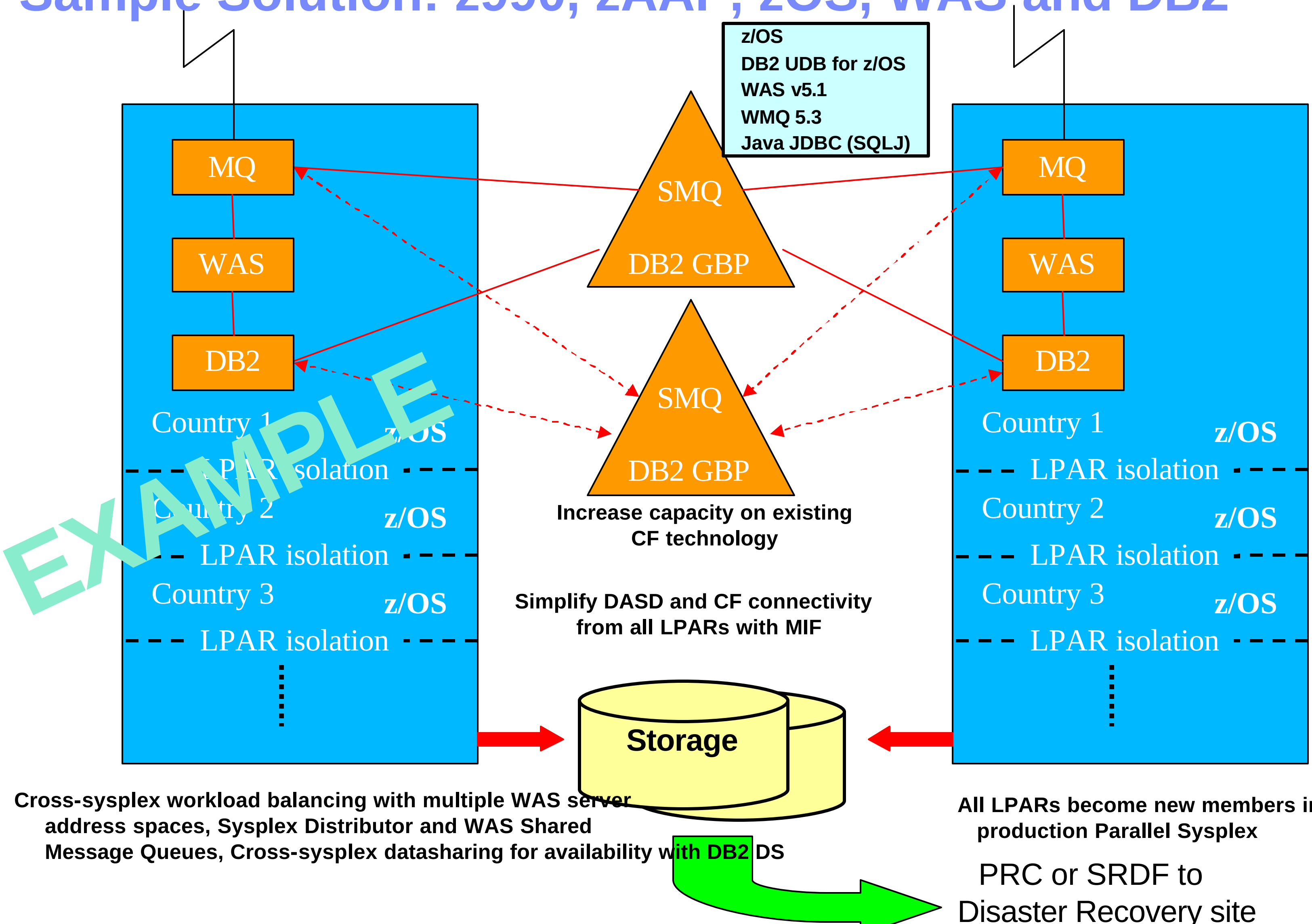


Key assumptions
Off-platform pathlength
Java <-> SQL pathlength

this is a representative, realistic comparative scenario. .. but subject to very many caveats, detailed assumptions etc

note: these are examples of a process. it is the process that is important, not the specifics of these examples

Sample Solution: z990, zAAP, zOS, WAS and DB2



this is a representative, realistic comparative scenario. .. but subject to very many caveats, detailed assumptions etc

note: these are examples of a process. it is the process that is important, not the specifics of these examples

Initially 9 Server Scenarios - for Multiple Instances

Item	Scenario
ppu	pAIX-WAS + pAIX-UDB-HADR
ppo	pAIX-WAS + pAIX-Oracle 9i RAC
xxu	xLinux-WAS + xLinux-UDB-HADR
xxo	xLinux-WAS + xLinux-Oracle 9i RAC
xpu	xLinux-WAS + pAIX-UDB-HADR
z1d	zOS-WAS&DB2 (single image)
zzd	zOS-WAS + zOS-DB2
zlzd	zLinux-WAS + zOS-DB2
pzd	pAIX-WAS + zOS-DB2 (hybrid)

First 9 scenarios for 1 instance with 4 Mn accounts (12 Mn trans/day)
Next 9 x 8 scenarios for 250K, 500K, 1, 2, 8, 16, 32, 64 Mn accounts
Identify significant technology constraints at various scales

Decision Modelling Engine for Multiple Scenarios

Major Health Warning ... Do NOT Try this at Home

- The following two charts are very useful in the right context
- They are not generic marketing charts but are real examples for a specific workload for a specific customer at a specific point in time
- Their value is in triggering the right topics for discussion within the various IBM hardware, software and industry sector communities and identifying the key customer server platform decision criteria for filtering out some of the many, initial possible solutions
- There are known weaknesses (but do you have a better practical filter?) This approach is based on a scoring questionnaire, an expert peer group, lots of discussion, etc. However, they are NOT absolute numbers that can be used generically, just relative initial positionings

Example: Platform Strengths/Weaknesses - Database Service

Weight	Criterion/Sub-criterion	xLinux Oracle 9i RAC	xLinux UDB HADR	pAIX Oracle 9i RAC	pAIX UDB HADR	zOS DB2
5%	**Market Share & Strategic Direction**					
100%	Market Share, Strategic Direction					
35%	**Resilience & Security**					
55%	Stability & Resilience					
15%	Close to Continuous Availability					
10%	Production Batch					
15%	Security of Production Service					
5%	Contingency for Site Catastrophe					
35%	**Performance at Scale**					
60%	Performance at Scale					
20%	Granularity of Additional Capacity					
20%	Workload Management					
25%	**IT Process & Support**					
50%	In-house IT Skills/Processes					
20%	Systems Management Tools					
20%	Vendor Product Support					
10%	Third-party Software & Tools					
100%	RANKING	2	5	2	2	1
	Weighted Scores (Rounded)	725	650	725	725	950

Database Service - Scoring and Weighting
Questionnaire, Interviews, Mediated Process
Based on Today's Actual Production Experience

Platform Scenarios (zOS DB2, etc)

Selection Criteria (eg: Scale, Security)

Platform Scores (0-10) (<5 red, >8 green)

Criteria Weightings (eg 25% for Process)

Max. of 1000 Points (eg 35 for production batch)

Rounded Scores (to nearest 25)

Platform Ranking (1 to 5)

Platform Selection - Database Service Scenarios

Weight	Criterion/Sub-criterion	xLinux Oracle 9i RAC	xLinux UDB HADR	pAIX Oracle 9i RAC	pAIX UDB HADR	zOS DB2
5%	**Market Share & Strategic Direction**					
100%	Market Share, Strategic Direction					
35%	**Resilience & Security**					
55%	Stability & Resilience					
15%	Close to Continuous Availability					
10%	Production Batch					
15%	Security of Production Service					
5%	Contingency for Site Catastrophe					
35%	**Performance at Scale**					
60%	Performance at Scale					
20%	Granularity of Additional Capacity					
20%	Workload Management					
25%	**IT Process & Support**					
50%	In-house IT Skills/Processes					
20%	Systems Management Tools					
20%	Vendor Product Support					
10%	Third-party Software & Tools					
100%	**RANKING**	**2**	**5**	**2**	**2**	**1**
	Weighted Scores (Rounded)	**725**	**650**	**725**	**725**	**950**

zOS-DB2 has very significant strengths, especially its continuous availability design point, proven scale & batch

Oracle RAC is a software cluster implementation with implementations in ~1-2 Mn account range. UDB HADR is new, with comparable scale, and has a 'fast failover' design point

z-OS DB2

++ Resilience, Availability, Batch, Security, Contingency, Scale, WLM, Skills, Tools...

p-AIX Oracle 9i RAC

\+ Availability, Performance

\- Batch, Software Clusters

p-AIX UDB HADR

\+ Fast Failover, Performance

Continuous Ops, Batch, SMP Scale limit

x-Linux Oracle 9i RAC

\+ Granularity

-- Batch, Scale

x-Linux UDB HADR

\+ Granularity

-- Batch, Scale

9 Middle/Back-end Server OLTP Scenarios reduced to 4

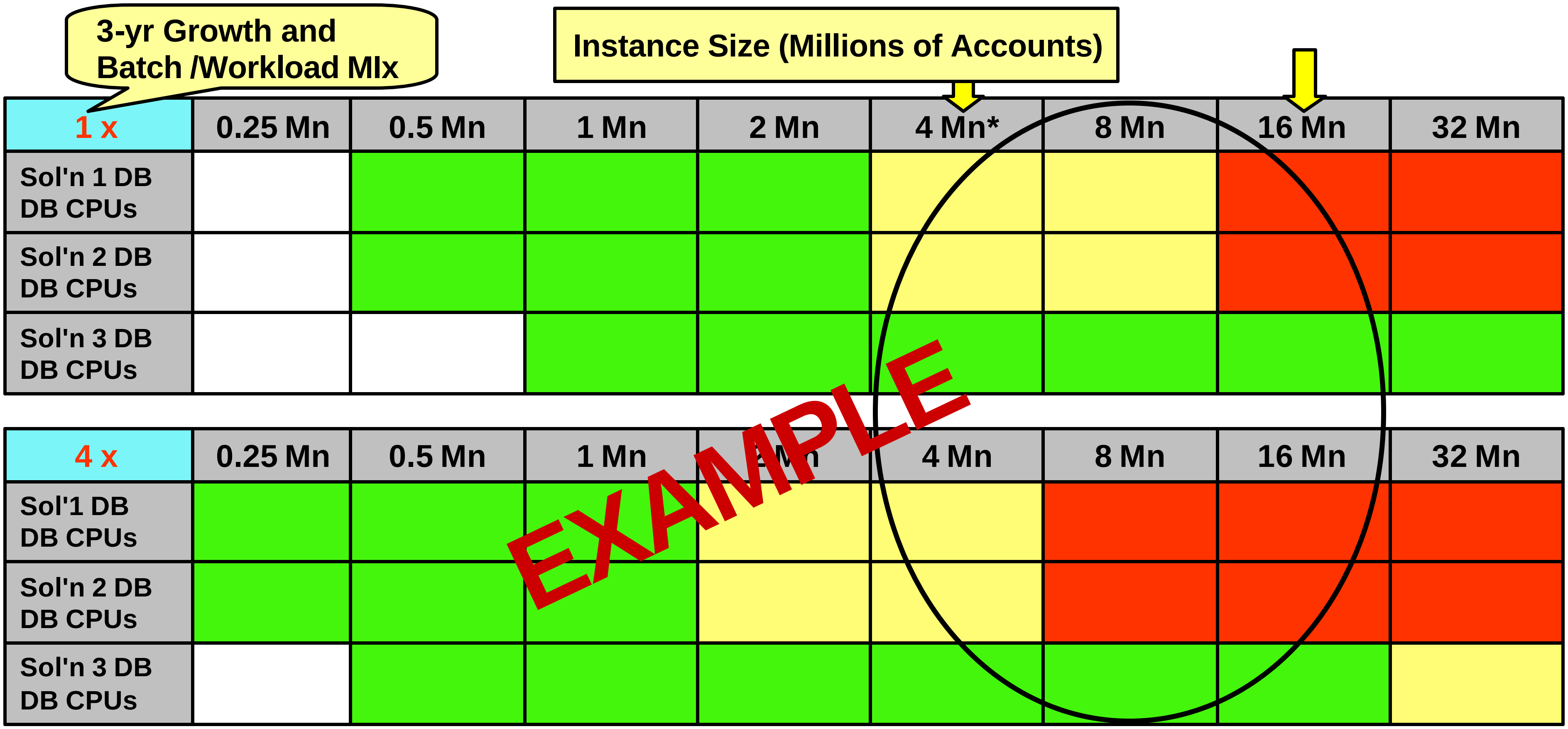
Item	Scenario	
ppu	pAIX-WAS + pAIX-UDB-HADR	Y
ppo	pAIX-WAS + pAIX-Oracle 9i RAC	Y
xxu	xLinux-WAS + xLinux-UDB-HADR	N
xxo	xLinux-WAS + xLinux-Oracle 9i RAC	N
xpu	xLinux-WAS + pAIX-UDB-HADR	N
z1d	zOS-WAS&DB2 (single image)	Y
zzd	zOS-WAS + zOS-DB2	N
zlzd	zLinux-WAS + zOS-DB2	N
pzd	pAIX-WAS + zOS-DB2 (hybrid)	Y

- 3 Linux Scenarios dropped
  - DB not yet proven at scale
    - >1 Mn accts/1TB/100 tps for UDB or Oracle 9i RAC
  - ISV application not currently supported by Linux*
- 2 zSeries Scenarios dropped
  - zLinux WAS dropped (zOS single image or z&p hybrid solutions better)
  - zOS-WAS+zOS-DB2
    - split platform overhead
    - load balancing

Overall project technology risk already significant (Large Application, J2EE/WAS)
3 Linux and Intel server platform options removed because current lack of
a: proven DB Scale on Linux-Intel, b. ISV application support, c. proven batch

*as at ddmmyyyy - see www.isvxxxx.com

Database Engine: Production Platform by Instance Size, Growth & Batch

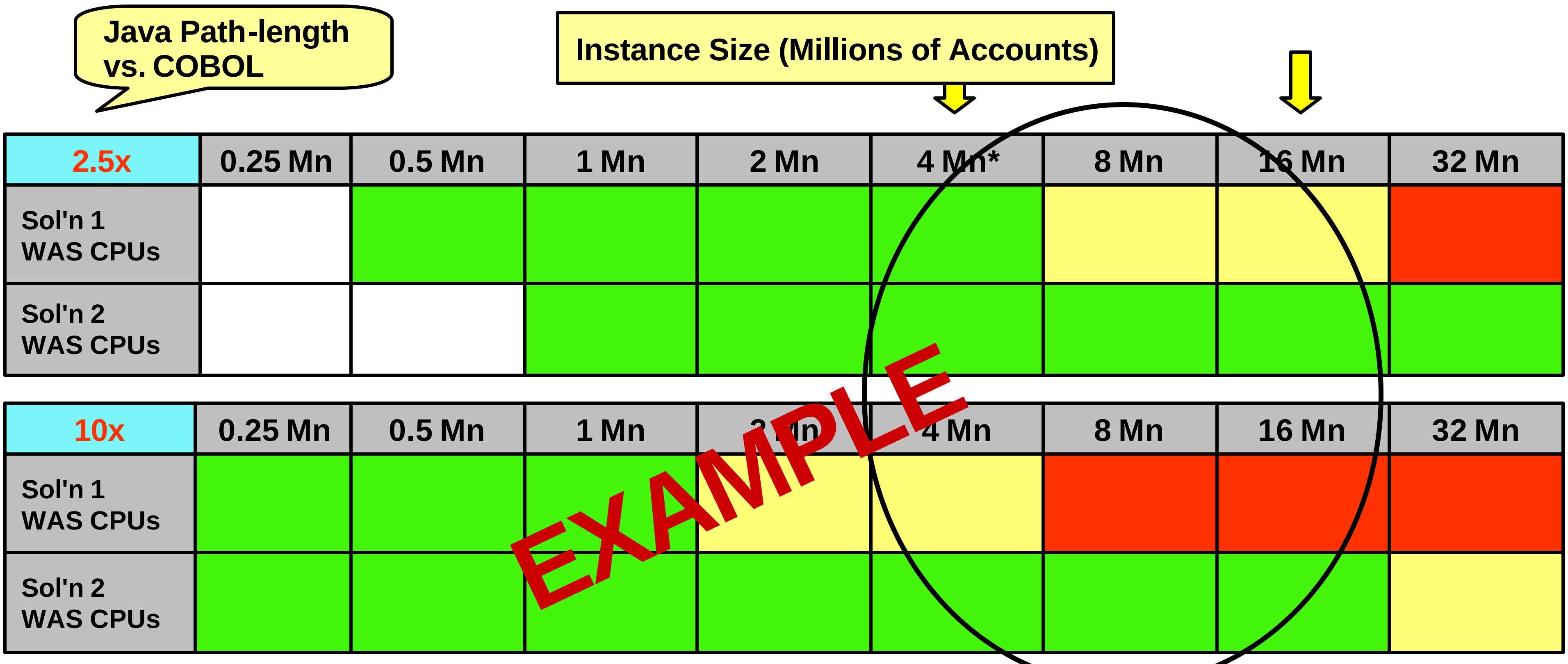


***240 OLTP tps**

This is production only, today. Must be uplifted to include failover, dev & DR.
The baseline data centre cases (4Mn, 16 Mn) indicate Sol'n 3 as most likely solution
Adding growth, batch/mixed workload highlight current Sol'n 3 strengths

'white' is underutilised, 'green' is comfort zone, 'yellow' is current leading-edge, 'red' is unknown, 'black' is unattainable

Java Engine: Production Platform by Instance Size & Java Path-Length

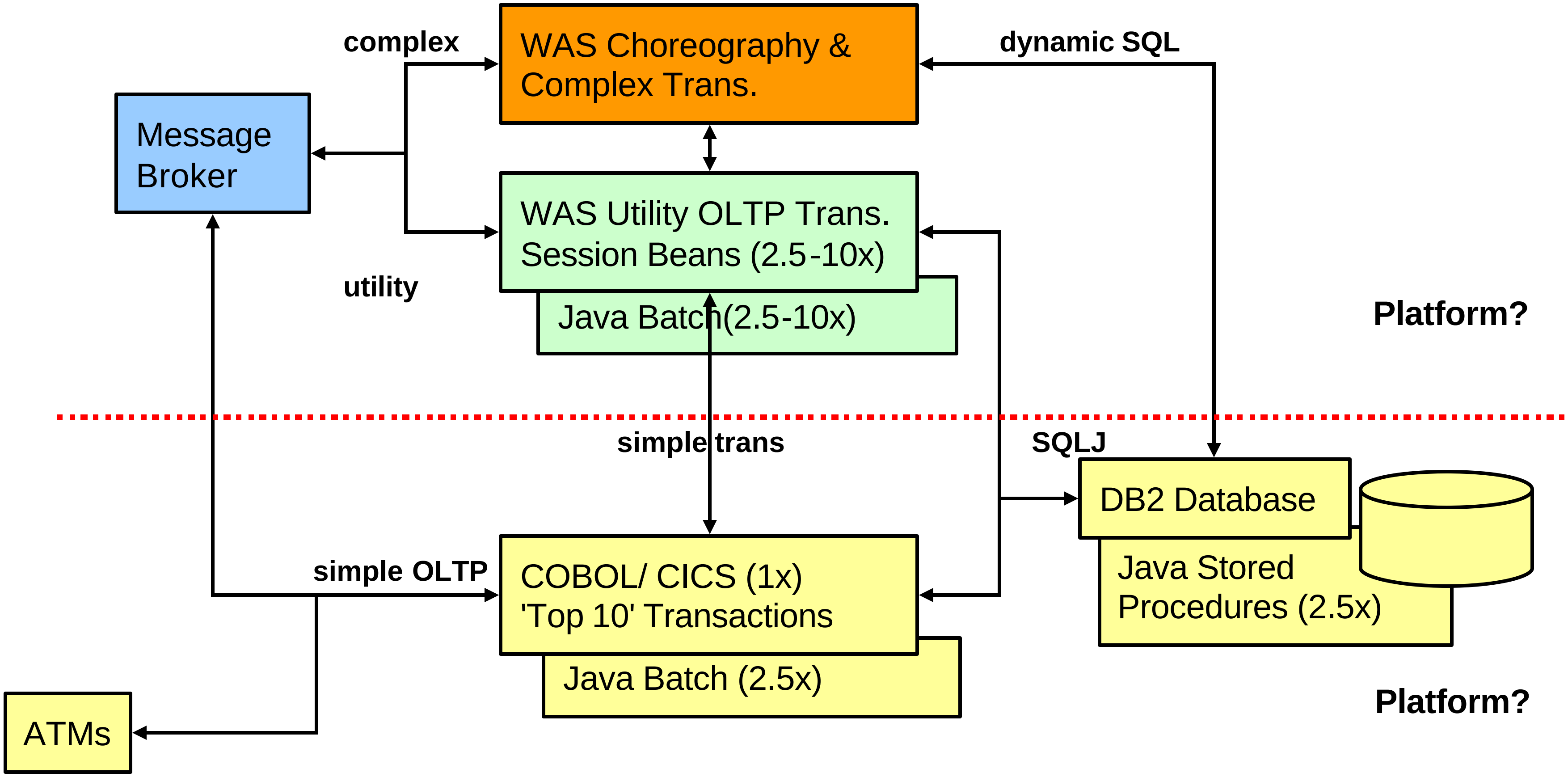


*240 OLTP tps

This is production only, today. Must be uplifted to include failover, dev & DR
Horizontally scalable, but very sensitive to actual Java pathlength achievement
Limited measurement data today on any OLTP WAS middle-tier >100tps
Must also add 30% annual transaction growth and choreography

'white' is underutilised, 'green' is comfort zone, 'yellow' is current leading-edge, 'red' is unknown, 'black' is unattainable

Back to Infrastructure Architecture... the separation of complex Java transactions from high-volume COBOL transactions



Strong case for retaining 'Top 10' simple high volume transactions in COBOL CICS

Example: 'Non-Server-Specific' Recommendations

- Isolate Application from Platform through Architectural & Design Standards
  - Common SQL standards for cross-platform portability (Unicode, SQLJ,...)
  - Strong J2EE design standards enforced by framework approach (esp XML, EJB)
- Establish a Systems Performance Engineering Culture and Approach
  - Establish **application** and systems design point for continuous availability and largest required scale
  - Set initial pathlength targets for simple high vol, medium and very complex transactions
  - Define baseline night and daytime critical batch scenarios. Validate scaling limits with reference platforms
  - Define baseline coexistence/phased strategic global migration plan
- Use "Proof-of-Concept" to establish core performance metrics and baseline
  - Specific pathlength measurements for simple OLTP and complex transactions
  - Validate baseline batch scenario
  - Build initial draft of platform reference architectures

Establish a Performance Engineering Centre of Excellence to analyse stress test data (online and batch) and build reference architectures

Example: Strategic Server Platform Recommendations **Today**

- Define Three Server Platform Reference Architectures
  - 1: zOS-WAS&DB2   2: pAIX-WAS + zOS-DB2   3: pAIX-WAS + pAIX-UDB
- Preliminary Recommendations - Database Server Platform Deployment
  - Database engine - use **zDB2** for all data centres with a service design point requiring continuous availability - includes all major geo-regional data centre instances
  - use UDB LUW for development and unit test, and for those instances where a 'fast failover' approach is an appropriate level of availability
  - use DB2 family, and SQL standards, as the baseline for portability between platforms
- Preliminary Recommendations - J2EE/WAS Server Platform Deployment
  - J2EE engine - **pWAS or zWAS** middle-tier for utility transactions and choreography
  - based on a high-volume Java transaction 'engineered down' to 2.5x COBOL pathlength
- Evaluate, now, a COBOL/CICS transaction engine for the 'Top 10' high volume trans
  - interim solution to mitigate Java pathlength issues if significant for utility transactions
  - migrate later to Java/J2EE when more mature and if justified from a business perspective

Significant trade-offs required between development/deployment speed, Java/COBOL skills, availability, technology risk and total service delivery cost

Next Step - Assess Viable Server Options at +2, +5 years

	"Pioneer"	"Safe Java"	"Pragmatic"
Simple	BI-WAS	p-WAS	z-CICS
Utility	BI-WAS	p-WAS	z-WAS
Complex	BI-WAS	p-WAS	z-WAS
Batch	BI-Grid	z-DB2	z-OS
DB	BI-Oracle	z-DB2	z-DB2
Risk	Higher	Medium	Lower
Cost (TCA)	Lower	Medium	Higher

Total Project
100 M$, 2-3 years develop
10-15 years deployment
Business Case A - TTM, New Product

High Overall Project Risk
Major new application
Significant business change
J2EE technology base
Business Case B - Dev. productivity
Lack of Java skills
✍ High risk application
✍ Java framework to enforce standards

So, Minimise Server Platform Risk
Business Case C - Platform Selection
Tradeoff technology risk vs. platform cost
At +2 and +5 years

And Ensure Future App & DB Portability
DB and J2EE Applications

Nets out to ~3 major server options at +2 Years - How do you choose between them?

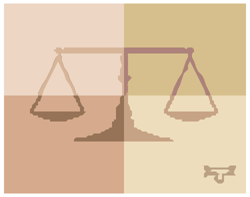
In Conclusion - Server Platform Choices

- Workload type is the starting point for doing realistic platform selection
  - many different workload types, computing styles and non-functional requirements
- Black-box approach is a useful construct for "IT Infrastructure Architecture"
  - useful links with CBM, SOA, Architectural thinking, etc
  - common terminology, useful for articulating volumes and NFRs
- Overall approach works - for email and mainline OLTP - but not easy to use
  - raises many of the 'right' questions. a forum for a very useful systems dialogue
  - key linkages of baseline scenario, selection criteria and application architecture
  - still many rough edges, need for documentation, etc
  - strong catalyst for useful future white papers, points of view, tooling, facts etc

- WAS projects need a very strong performance engineering control loop
- Get DB common platform standards agreed - DB2 Family - Unicode, etc

- Business case must carefully balance technology risk, speed to deploy, availability, service delivery cost, skills etc against business value in major strategic core application projects

Backup Charts



Non-Functional Requirements (Front, Middle, Back Tiers)

- Transaction Volumes and Expected Annual Growth
- Simple, Medium and Complex Transaction Pathlength
- Response Time (eg <0.5 seconds/simple online transaction)
- Complex Batch Job Handling (including housekeeping, bulk printing, end-of-day/month processing - both overnight and trickle daytime)
- QoS - Availability and Recovery
  - 24 x 7 x 365 service hours - scheduled and unscheduled outages
  - 24x7 continuous availability vs. 24x7 high availability vs.'18x6' availability
  - availability/unscheduled outages - #incidents, outage duration, impact
  - recovery from single-site catastrophe
- Security
- Statefulness

- Volatility (Number of Change Packages per Year)

- Target Cost per Std. Transaction (Simple, Medium, Complex)
- Target Pathlength per Std. Transaction (Simple, Medium, Complex)

- Interfaces and gateways to other major applications and services

Key Technical IT Risks - Summary of Categories

- Scaling Risk (Performance, Resilience and Cost)
  - Java maturity, performance, resilience (pragmatic design guidelines)
  - Very large scale (eg >4 million accounts require high-end scale)
  - Multi-tier architecture (complexity, bottlenecks, resilience)
  - Trade-offs vs. platform costs - hardware and software (opsys, db, was)
- Design Risks (Application Architectural Options)
  - Application/functional split - how split key components, what priorities
- Deployment Risk
  - many very small instances (consolidate application images)
  - co-existence with current legacy application for several years
  - production availability (ie full stress testing before live production)
  - batch mechanics (for, say 15 countries in different time-zones)
- Business Case Support
  - what $$$ business benefits - mature & stable core transactions
  - IT drivers for change - lack of agility
  - investment required: new application function, cost of deployment, new server hardware/software

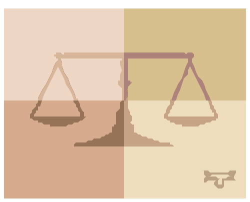
For each major platform solution scenario (ie 'Top 4')

- Chart of +/- considerations for various whole scenarios
  - examples: z-Linux-WAS + z-OS-DB2, or p-AIX-WAS + p-AIX-Oracle

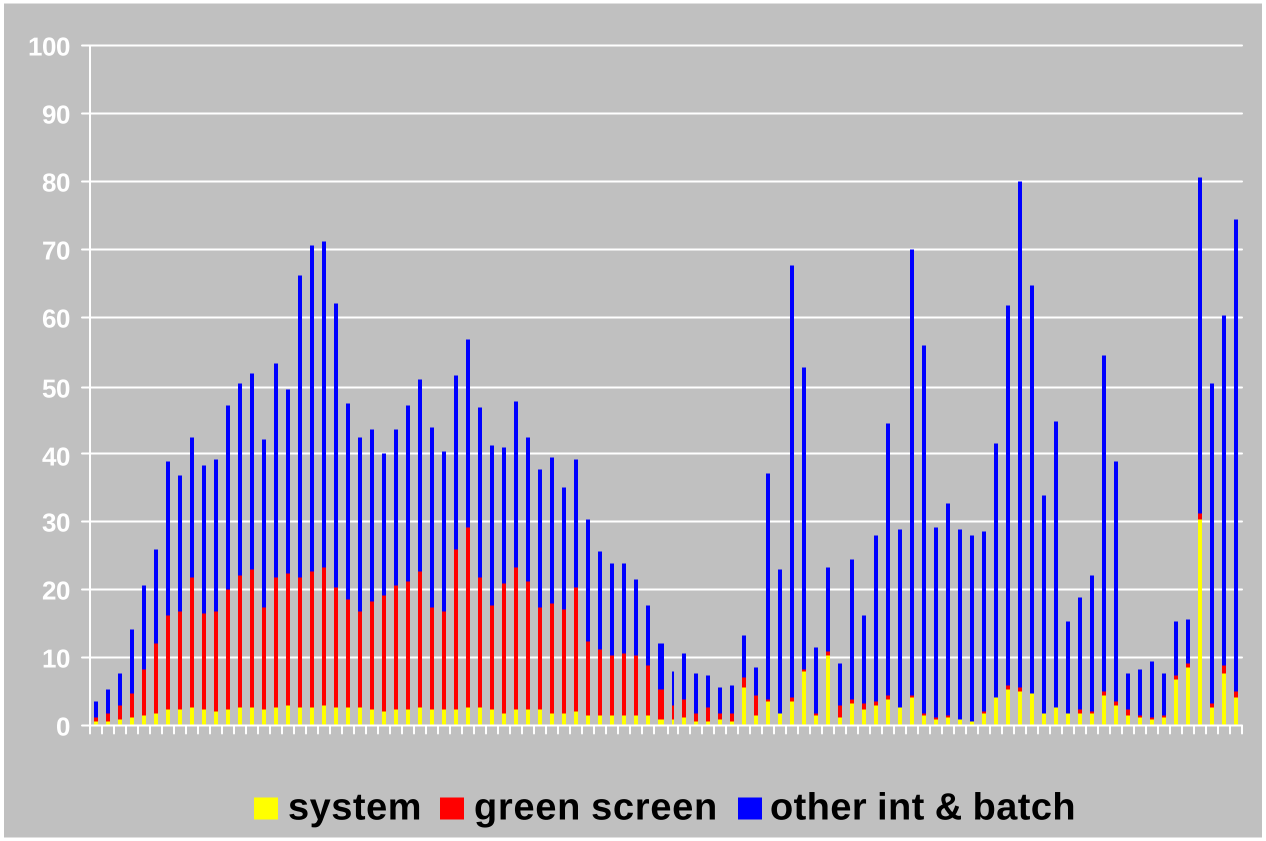
- Objective: give a sound technical assessment of each major solution

- Includes
  - batch (backup/reorg, overnight, day-end/month-end close)
  - Java pathlength for suggested approach, simple, medium, complex trans.
  - known production instances, likely scaling limits, low-end solutions
  - continuous availability limits (eg JVM, DB backup/reorg, versioning)
  - production technical risk
  - co-existence and migration strategy
  - access to other off-platform services
  - approx. 5-year cost (high, medium, low, but not quantified in detail)

Beyond Backup
Real Examples & "Thought-Provokers"



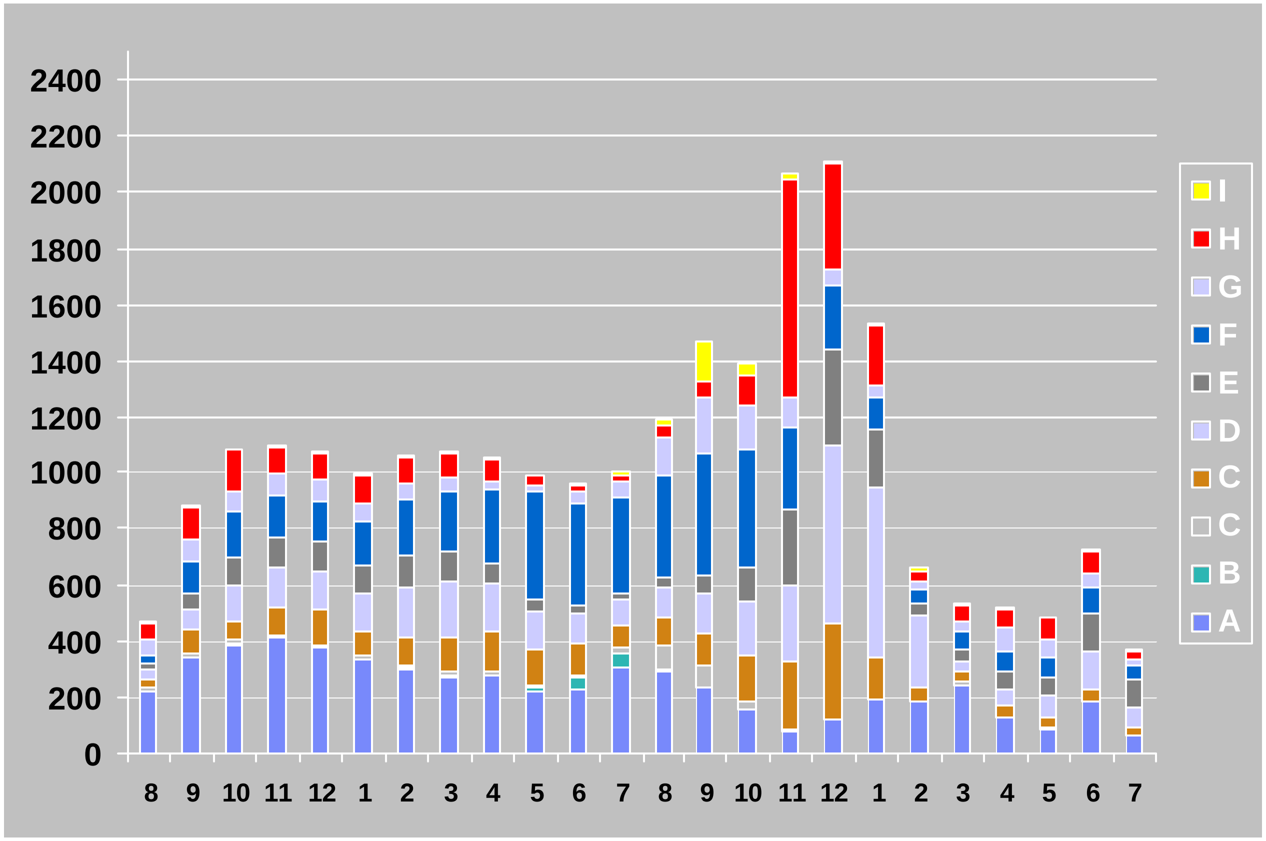
Example A: Country BBB LPAR: 24-hour CPU Utilisation



Avg. 24-hours 35.2%; Avg. Daytime* 38.9%; Peak*² 70.2%

xx-yy Oct 2004: 8:15am-8:00am system time. Country BBB has one Time Zone, at -2 hr system clock time
Measurements every 15 mins. Production instance located at data centre ZZ on server XX 4way LPAR
***Daytime utilisation averaged over 12 hours, 7am to 7pm system clock time. *² at 95th percentile of measurements**

Example B: 24-hour Multi-Country CPU Load Profile for August



Country 'A' consumes 23% of total. Note month-end peak 11pm -1am

Total weekday CPU load for August 2004 by hour of the day from 8am to 8am. Multiple Time Zones
10 Production country instances located at ZZ data centre on server xxxx 8way LPAR
***CPU load averaged for each hour for each country, accumulated for month of August**

Example C: Core Application - A CICS/DB2 Sysplex

zSeries #1
~2200 Mips
peak load

LPAR01
TORs/MORs
AORs 33%

zSeries #2
~2200 Mips
peak load

LPAR02
TORs/MORs
AORs 33%

zSeries #3
~2200 Mips
peak load

LPAR03
TORs/MORs
AORs 33%

Approx 60 million lines of COBOL code
Approx 400 M$ investment (10 years)
~12 million customer accounts
~8 million customers

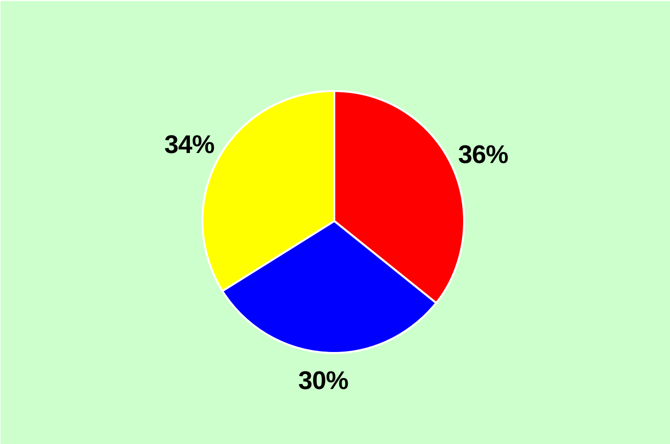
Sysplex handles core systems
~65 million CICS trans per day
~1700 CICS trans per sec at peak
Subsecond response time to network

3 LPARs on 3 separate zSeries machines
~60% avg. utilisation (peak day 8am-6pm)
~6600 Mips peak workday load
Load split ~40% CICS, ~60% DB2
Disk storage 12 TB x 4 (master + 3 copies)

1 planned outage/year for network changes
No Sysplex outage in last 18 months
DB2 instance, ~1 or 2 outages per year
MVS instance, ~1 or 2 outages per year
DR - <4 hours to recover with no data loss

Example D: Mainframe Cost Profile Evolution in 5 Years

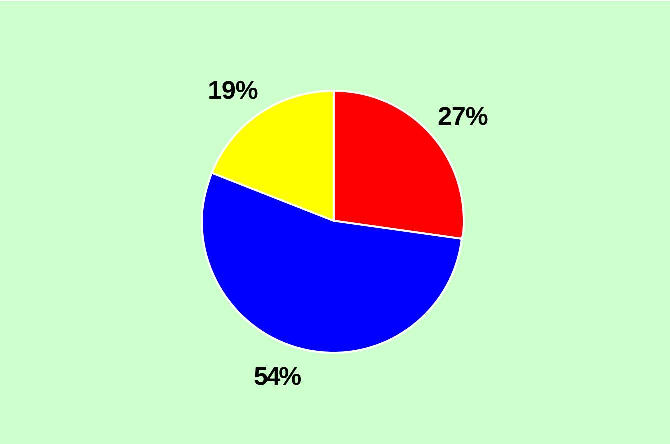
1999 - 54 M$



24 Million CICS trans/day
(~6 Billion CICS trans/year)

0.90 cents per CICS tran.

2004 - 75 M$



79 Million CICS trans/day
(~20 Billion CICS trans/year)

0.37 cents per CICS tran.

Reduction of about 60% in cost per CICS tran. in 5 years
26% annual tran. growth, 5% annual cost growth
Today, 55% of the IT budget is software

legend: red is hardware, yellow is people, blue is software

Example E: High Volume OLTP UDB Linux Cluster*

5 Application Servers (9 nodes)
initially 1 coordinator, 8 partitions

x360 4-way Xeon, 16 GB RAM
2 x GbE, 2 x dual FC (total 8Gbps)
Red Hat Linux 2.1
Veritas Cluster Server, Volume/File Manager
DB2 ESE v8, MQ
800 GB raw data on FAStT
total of 12TB FAStT (4TB raw data)

Database partitioned 8 ways by account #
Database includes events, positions, balances, journals, reference data, temporary tables, MQ message staging

24 Message Queues (3 per DB partition) achieved 1.6 million events per hour at 82% server CPU utilisation (10 million/day)

Existing DB2 application
Leading financial services enterprise
'Balances and Positions' database
Accounting for deposits and trades
Handles Postings and Queries

Proof of Concept Goal was to achieve 1.2 million postings per hour (>300/sec)

Use existing network, Veritas, Kerberos and link to mainframe infrastructure via Natural
8 week Proof of concept - 1H 2003
<5 days for installation & data migration

~10 million daily customer trades/events

Local failover < 1minute
DR failure to remote site - ~15 minutes

***High Volume OLTP Database Cluster with IBM DB2 for Linux - IBM SWG - Nov 2003:**
DB2 backoffice application. PoC of 5 x360 application servers running MQ, UDB, Linux, FAStT
http://ibm.com/db2/linux, select the tab "Papers" and look in the DB2 ICE and Linux Clusters section

Example F: Sabre - Airline Travel Shopping and Pricing Engine*

Current Mainframe System
10 million lines of assembler code
Used by airlines, agents, passengers
Ultimate OLTP system, uses TPF, high trans/sec
1,000 mips in 1995, 10,000 mips in 2001
very high and increasing 'look to book' ratio
adding a new rule could cost 1M$ in change costs ...

3 Major Subsystems:
Shoping & Pricing Engine, Bookings, Fulfilment

Issues: Cost/tran, Speed to change, Scalability

4-year 100M$ migration plan (>50% complete)
Move from TPF to UNIX & Linux, Java, Distributed Grid
Reduce cost/tran by >50%, Increase prod'y by 100%
Reduce time to make changes by 75%

Objective - Reduce Unit Cost by 50%
A simple query is already 80% cheaper

Sabre Overall (inc. Travelocity)
2B$ revenue, 200M$ R&D
35% w/w resv'n market share
80B$ value of products sold
90 airlines, 56,000 agent locations
15,000 trans/second (peak)
48 million total trans/day
79 mn fares, 6 mn schedules
1.2 mn fare changes/day
70GB database

Sabre ATSE - Simple North America Routes
12 million shopping & pricing requests per day
Coordinator & Rules Engine
16 x 2way Linux Intel (HP)
'Horizontal' Shopping Engine (12 Mn/day)
45 x 4w Linux Intel Itanium 32GB (HP rx5670)
MySQL replicas of master database
moving to flower and petal topology with Opteron
Availability & Pricing Engine
17 x 16w NonStop HP S86000, 4GB RAM

*** Computerworld - 31 May 2004**

Example G: zOS-DB2 compared with Oracle RAC*

	Oracle RAC	z-OS DB2	
Survey Base	198 Organisations 203 Clusters	168 Organisations 260 Clusters	Oracle RAC sample has many DW, other apps
OLTP applications	78 Clusters (typically single application)	260 Clusters (typically multiple applications	
Transaction Volumes (Production)	81% < 100K /day max. 600K[1]; avg. 138K	87% > 1 Million/day max. >45 Mn; avg 8.8 Mn	Sysplex- approx 10x daily volumes
Cluster Size (Prod & Planned)	81% 2node; 18% 3&4 node; 1% 6 node	36% 2 node; 37% 3&4node; 26% 5+ nodes	
Cluster Overhead (Locking, Coherency ..)	20% 2 node; 30% 4 node, 39% 6 node	11% 2node; 13% 4 node; 15% 8 node	RAC has approx double overhead with known bottlenecks
Production Tps (Peak & Sustained)	Peak ~400 tps Sustained ~100 tps	Peak & Sustained - 13 orgs > 1000 cplx tps	Sysplex - at least 10x sustained tps proven
12-month Availability (all outage types)	16% achieved 100% + 32% >99.90%	31% achieved 100% +31% >99.90%	
Recovery Time	Failover: 60-90 seconds Full recovery: 5-20 mins	Failover: 0-20 seconds Full recovery: 1-10 mins	

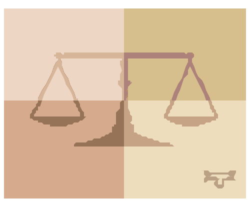
***Enterprise Database Cluster Solutions - ITG - Oct 2003:**

compares transaction processing workload on 78 Oracle RAC clusters and 260 zOS-DB2 sysplex clusters
Oracle includes planned and production sites. Sysplex sites are all production
[1]Planned systems - 15% > 1 million/day

IT Infrastructure Systems Architecture

DB and J2EE Standards

DB Portability: Unicode, SQLJ
J2EE/WAS Performance Engineering



WAS & DB2 - Session Beans, EJB, JDBC, SQLJ,...

- A very large number of Java calls to the DB are likely to use dynamic SQL. This is good for flexibility and occasional high value transactions, as it can automatically handle considerations of persistence and enable the Java programmer to focus on business logic

- SQLJ (static SQL) is precompiled and optimised for higher performance. Stable, highly tuned COBOL/CICS transactions and batch programs use SQLJ extensively.

- **EJB entity beans, XML, and container persistence** have all been used so the programmer does not need to write complex SQL. However, this can result in **very** substantial pathlength increases in the server side processing of core trans (eg 50-100x). This may be acceptable for very rarely executed transactions and low transaction rates, but is an extremely large overhead for mature 'core utility transactions' (eg 'get account balance') executed millions of times per day

- Using **servlets and stateless session beans**, the Java programmer can use static SQL (SQLJ) statements invoked via JDBC. Appropriate standards can get processor pathlength closer to typical core COBOL / CICS transaction pathlength (best current practice is around 2.5x COBOL, but can often be 10x or more)

DB Portability

Why?

- many current legacy DB systems are not portable; require major redesign, time impact
- so a major objective of a new application is to ensure future server platform portability
  - enable move to new platforms based on function/risk/cost tradeoff at future times
  - immediate use for development life cycle: unit test ? Int. Test ? Production

What Recommended Standards?

- Unicode (UTS-16)
- SQLJ to achieve baseline performance of high volume, utility transactions
- Use IBM SQL reference document for cross-platform development (778 pages)

Platform Benefits/Advantages

- avoids requirement for a one-time, 'big bang' server platform decision
- select the best production platform based on expected country/image size
- select the production platform based on tradeoffs at decision time
  - especially scale/volume of DB instance required (size in TB, trans/second etc)

DB Standards - Unicode and SQL

- Country instances using EBCDIC/zDB2 must be in separate DB2 subsystems or data sharing groups. There are also platform application differences which inhibit platform portability (eg LUW to zSeries)

- Solution Area: Specify Unicode for all new Databases
  - each country can be in a separate DB2 subsystem or data-sharing group or multiple countries can be in the same DB2 subsystem or data-sharing group
  - countries may also be isolated by schema, or share the same tables
  - SAP and Peoplesoft have adopted Unicode as their standard approach
  - specifically: store SQL column character string data as VARGRAPHIC (UTS-16)

- Benefit: Simplified development, test, deployment, maintenance
  - lower costs because of smaller number of DB images to setup, administer

    no application platform-specific embedded ASCII/EBCDIC code page changes,

    avoids zSeries ASCII/EBCDIC code page conversion
  - a very important component of database portability between platforms
  - Note: 2x disk space for single-byte countries (US/UK/Germany)... this can typically be compacted to ~1.5x by ESA compression on zSeries

Use of SQLJ for High Volume 'Utility' Transactions

- Problem: Extremely long pathlengths created by J2EE/EJB based applications with poor (or no) performance design criteria and standards

- Solution Area: Define SQL standards (esp. SQLJ) for high volume 'utility' Java transactions. Implement strong performance engineering approach for J2EE development

- Rapid adoption rate - most very large Enterprises use SQLJ today
  - Most use of SQLJ is access from WebSphere, WebLogic, iPlanet
  - All large-scale Java applications on zSeries use SQLJ
  - ANSI and ISO SQL Standard
- Major Benefits:
  - easier, standard coding across all platforms
  - high, consistent performance - lock down an optimised SQL access path
  - improved manageability - associate CPU, I/O, SQL, BP with a Java application
  - easier capacity planning and problem determination/isolation/resolution
  - fallback to yesterday's version (if a problem with new version)
  - better security (GRANT EXECUTE to specific end-user)

J2EE/WAS - Leading-edge Middle-tier Projects

Why?

- Current production and benchmark examples show extremely wide variations in transaction function and pathlength
- There are many current examples of WAS pathlength in the range of 50-300 million instructions caused by poor architectural design (excessive XML/XSL, EJB, ..) A strong performance engineering approach has reduced pathlength in many cases by 5-10x. Suggested target for 'simple' OLTP Java transactions is in range of 5-10 million

What Recommended Standards?

- Performance Engineering - process, architects, implementation standards
- Path-length budget of ~5-10 million for OLTP high volume (2.5 - 5x COBOL/CICS)
- SQL standards, esp. SQLJ optimisation for high volume simple 'utility' transactions
- Use PoCs to establish baseline Batch and SQLJ pathlengths and enforce via Java frameworks

J2EE/WAS Platform Selection Implications

- J2EE is currently relatively immature for high volume OLTP applications (>250tps)
- ensure horizontal scalability and minimal persistence overhead for OLTP
- select a Java production platform with high performance/good price-performance

Examples - J2EE & WAS Transaction Path-length

Situation	Initial Path-Length	Final Path-Length	Notes
AAA	120 MI	1.8 MI	CICS Backend, COOL:Gen based
BBB	60 MI	12 MI	Excessive XML/XSL, ~5 MI achievable
CCC	50 to 300 MI	10 to 25 MI	Many issues, excessive EJB usage, average 15 MI achievable
DDD	1200 MI	130 MI	Excessive XML/XSL, redesign required, bad performer on pSeries and Intel
EEE	12 MI	~5 MI	CICS Backend, simple non EJB Java
FFF	0.6 CPU Sec/TX	>0.1 CPU Sec/TX	Bugs in persistence layer
GGG	>10 CPU Sec/TX	<1 CPU Sec/TX	Persistence layer unrealistic, needs redesign
HHH	100 MI	<30 MI	Inefficient XML Security library, excessive EJB usage
JJJ	8 MI	~5 MI	Work underway, JSP and data mapping require tuning
KKK	>40 MI	10 MI	Inefficient JSP custom tags
LLL	>35 MI	Work not yet started	Excessive XML/XPath Usage <10 MI achievable

These are all examples of project 'firefighting' after designs were finished, the recommended approach is to include performance engineering from the start