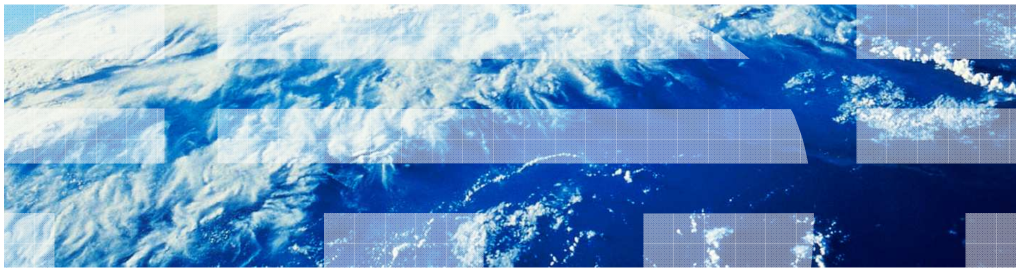
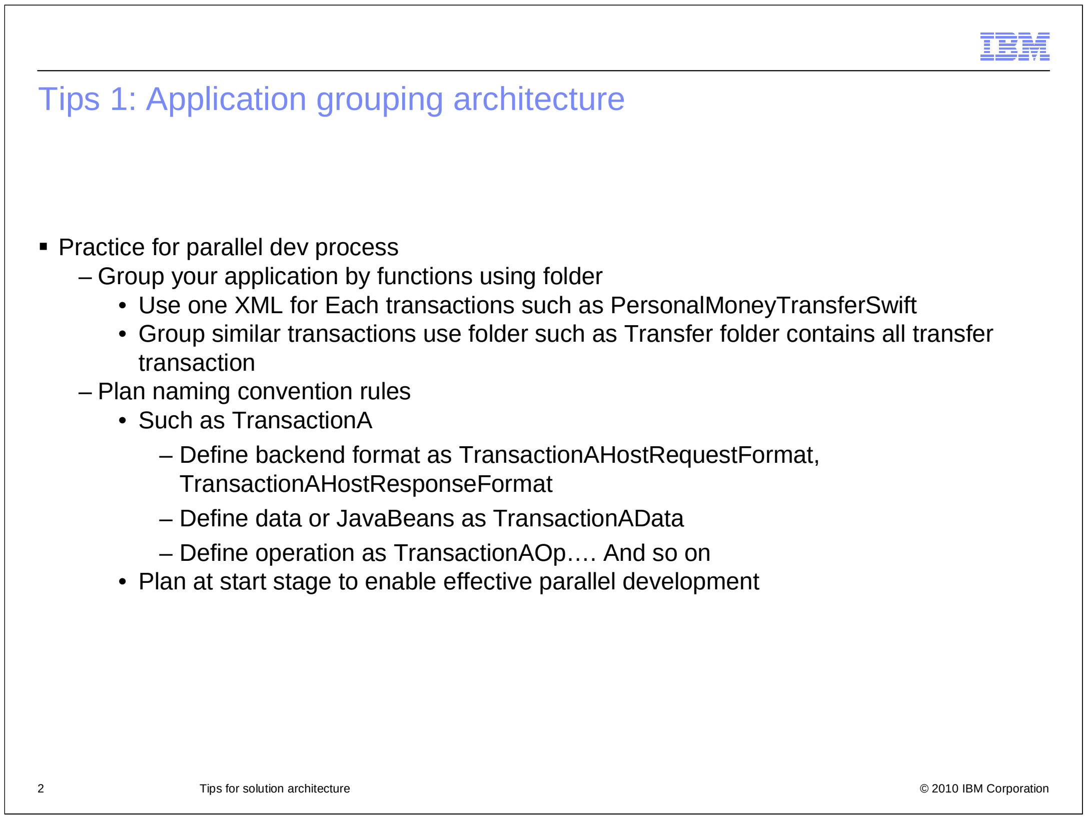
WebSphere Multichannel Bank Transformation Toolkit

Tips for solution architecture



WebSphere software

This presentation is to show some BTT tips for solution architecture.



Tips 1: Application grouping architecture

- Practice for parallel dev process
  - Group your application by functions using folder
    - Use one XML for Each transactions such as PersonalMoneyTransferSwift
    - Group similar transactions use folder such as Transfer folder contains all transfer transaction
  - Plan naming convention rules
    - Such as TransactionA
      - Define backend format as TransactionAHostRequestFormat, TransactionAHostResponseFormat
      - Define data or JavaBeans as TransactionAData
      - Define operation as TransactionAOp…. And so on
    - Plan at start stage to enable effective parallel development

2 Tips for solution architecture © 2010 IBM Corporation

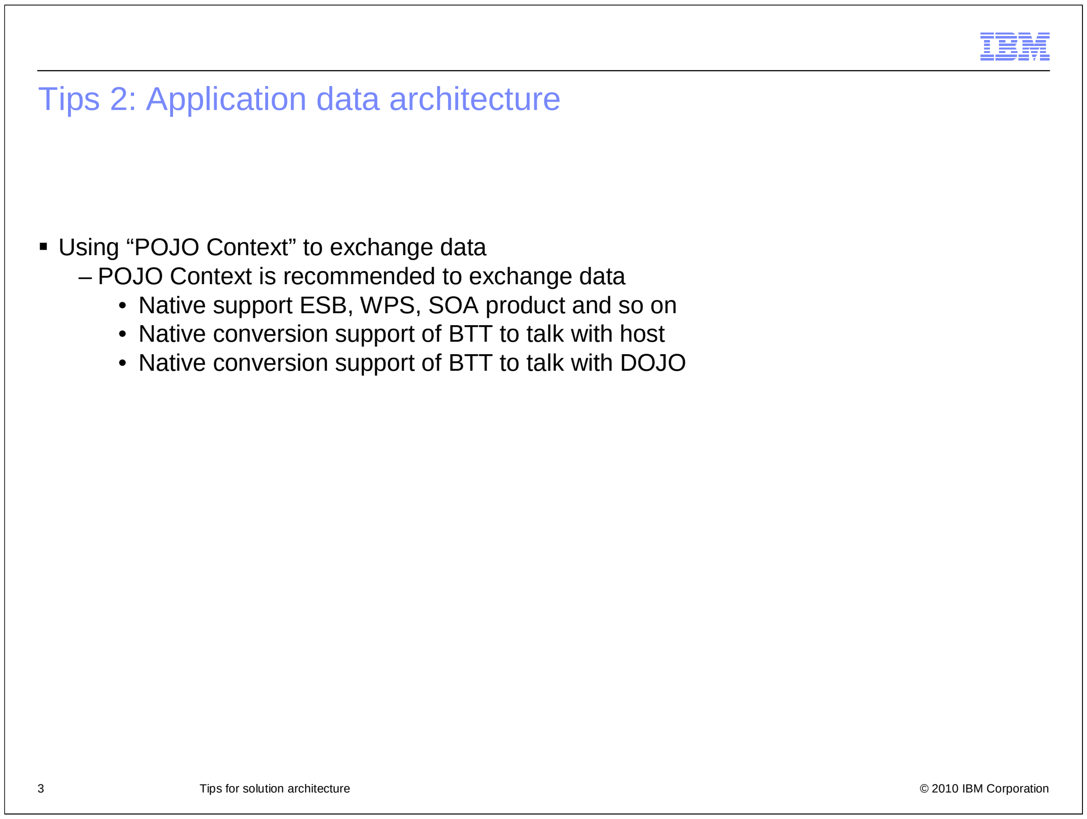
Everyone’s brain is limited and cannot recognize to many rules, so keep the naming conversion as simple as possible.

Self-define your transactions and group them by folder

The size and manageability of BTT XML files has to be controlled; if all the XML code resides in a few large XML file, it is impossible to manage for team development.

That is why it is recommended to use BTT XML mixed modularity. The operation and process instances can be placed each in a separate XML file, and the global, shared definitions (such as the higher-level contexts and all their data and services) are placed in the root XML files.

In typical banking industry, the transaction number is very large, such as 1000 or bigger. It is better to organize transactions by folder, for example Payment folder contains payment transactions and Credit Card contains credit card transaction.



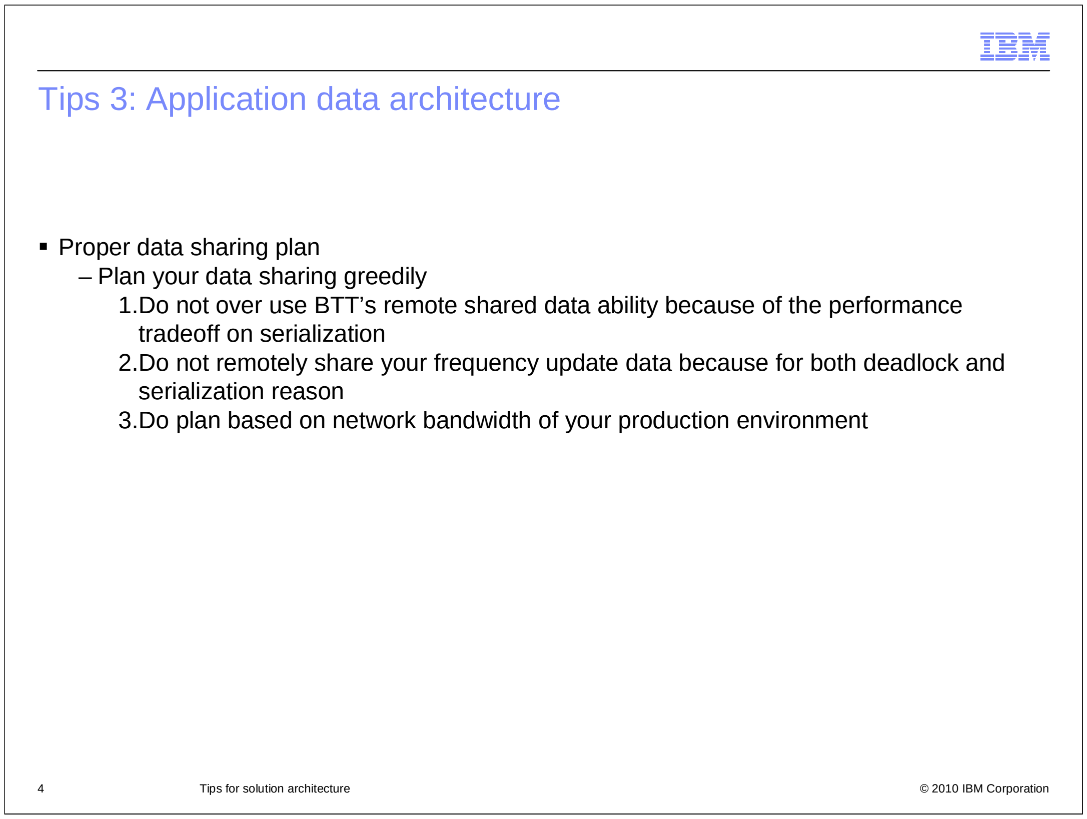
Tips 2: Application data architecture

- Using “POJO Context” to exchange data
  - POJO Context is recommended to exchange data
    - Native support ESB, WPS, SOA product and so on
    - Native conversion support of BTT to talk with host
    - Native conversion support of BTT to talk with DOJO

3 Tips for solution architecture © 2010 IBM Corporation

Use POJO Context when integration with ESB

BTT Context direct support POJO with the same API, do not redefine it use XMLs which will waste development effort and cause difficulties on maintenance.



Tips 3: Application data architecture

- Proper data sharing plan
  - Plan your data sharing greedily
    1. Do not over use BTT’s remote shared data ability because of the performance tradeoff on serialization
    2. Do not remotely share your frequency update data because for both deadlock and serialization reason
    3. Do plan based on network bandwidth of your production environment

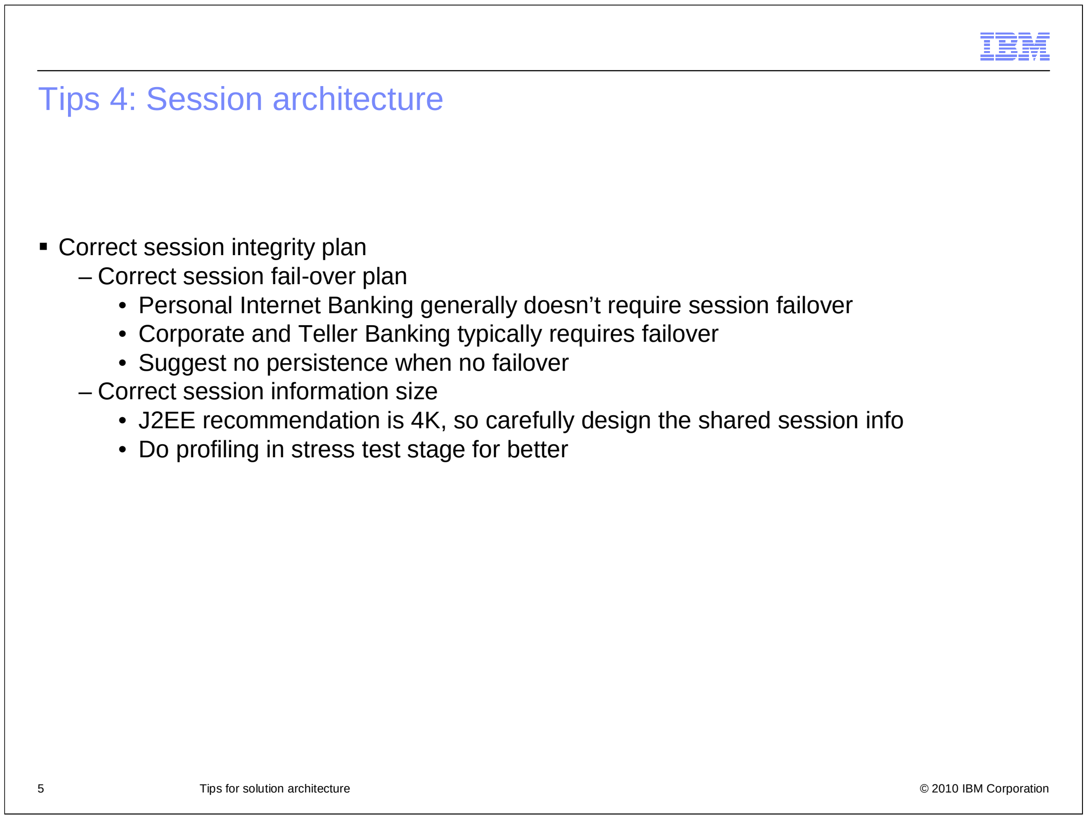
4 Tips for solution architecture © 2010 IBM Corporation

Do remote deployment correctly

Do not remotely deploy frequently changed data such as session context, this will greatly decrease the system performance

Do not remotely deploy large data because of the Java serial-deserial is very time consuming and synchronized, real case shows it downgrade the supported concurrent number to 10% or less.

Remotely deploy stand-alone services only when there no large volume data transactions.


Tips 4: Session architecture

- Correct session integrity plan
  - Correct session fail-over plan
    - Personal Internet Banking generally doesn’t require session failover
    - Corporate and Teller Banking typically requires failover
    - Suggest no persistence when no failover
  - Correct session information size
    - J2EE recommendation is 4K, so carefully design the shared session info
    - Do profiling in stress test stage for better

5 Tips for solution architecture © 2010 IBM Corporation

Session here stand for customer’s serial action such as query account, withdrawal, then do a transfer to CC and so on. Do session failover and persistence until required

Session persistence will notable downgrade system performance even you doesn’t persist it to DB

In real production case, the throughput decrease 20 percent or more.

Persistent session is not necessary for Internet Banking and typically required for Tellers, so the concurrent number for Tellers are much more smaller than Internet Banking

Session size profiling for performance gain:

Calculate the average session size of the application in order to ensure that the available hardware can allocate the planned number of users.

A profiling done with tools such as LoadRunner or Rational® Application Developer will help determine the exact size of a given session instance. Whereas, WebSphere® Application Server Performance Monitoring Infrastructure (PMI) can directly report the average HttpSession size in a server.

Manipulate session info is very expensive for application servers, so do planning correctly before design

Reduce Session information as small as possible when persistence is required

In BTT, session context and processor info need to stored in HttpSession

J2EE recommend less than 4k size in HttpSession. The larger, the slower.

Correct Session Information Planning

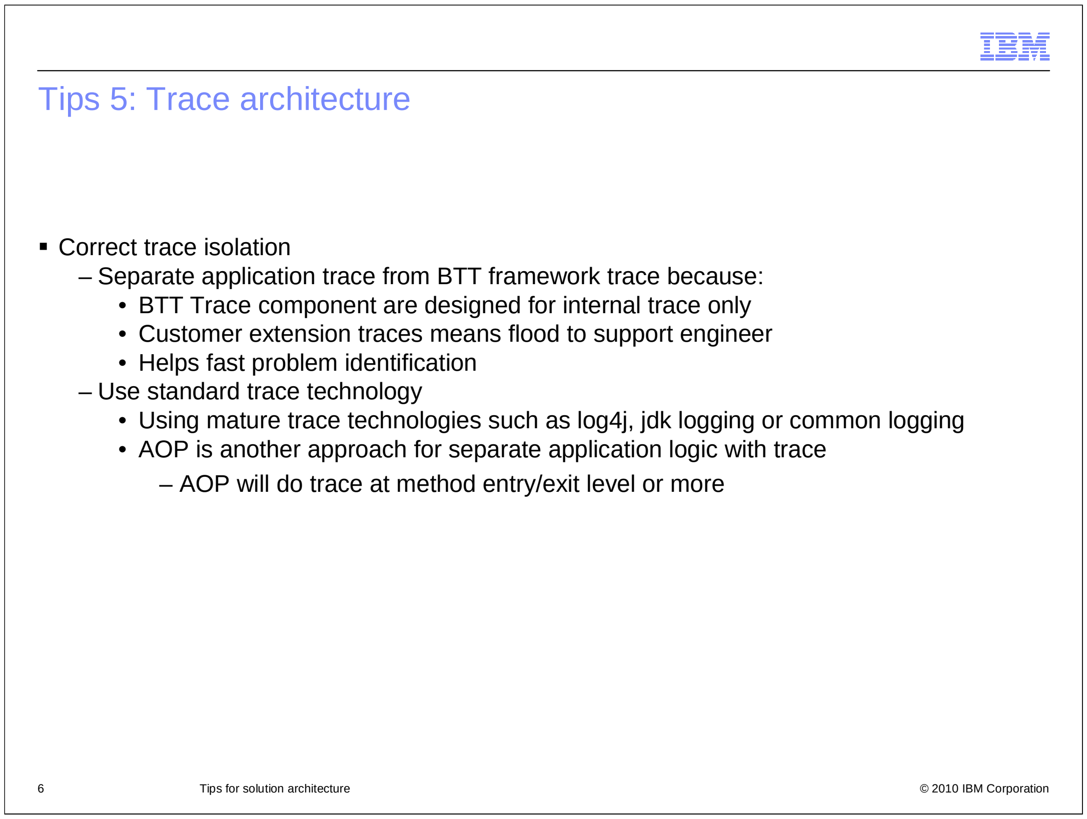
Determine the memory available to the application (JVM size) per server, Divide the total available memory by the planned number of users.

Keep failover in mind. In the case where one server in the cluster is down, the rest of the servers should still be able to allocate the extra resources. This means that you must account for one server less than the total number of servers available.

Ensure that all data defined in the session level is really necessary at that level:

The data is user-specific, and therefore cannot be placed on a higher-level, shared context.

The data is required along several user interactions, and therefore cannot be placed at a lower-level, operation context.



Tips 5: Trace architecture

- Correct trace isolation
  - – Separate application trace from BTT framework trace because:
    - BTT Trace component are designed for internal trace only
    - Customer extension traces means flood to support engineer
    - Helps fast problem identification
  - – Use standard trace technology
    - Using mature trace technologies such as log4j, jdk logging or common logging
    - AOP is another approach for separate application logic with trace
      - – AOP will do trace at method entry/exit level or more

6 Tips for solution architecture © 2010 IBM Corporation

I.Trace is firstly for people reading, so keep it simple and meaningful; do not make flood.

1.For example, BTT internal trace uses #CHA, #HTML stands for special meanings

2.BTT Leveled support engineer need more negotiation effort for customer extend Trace Tag such as #BranchA, #TellerA

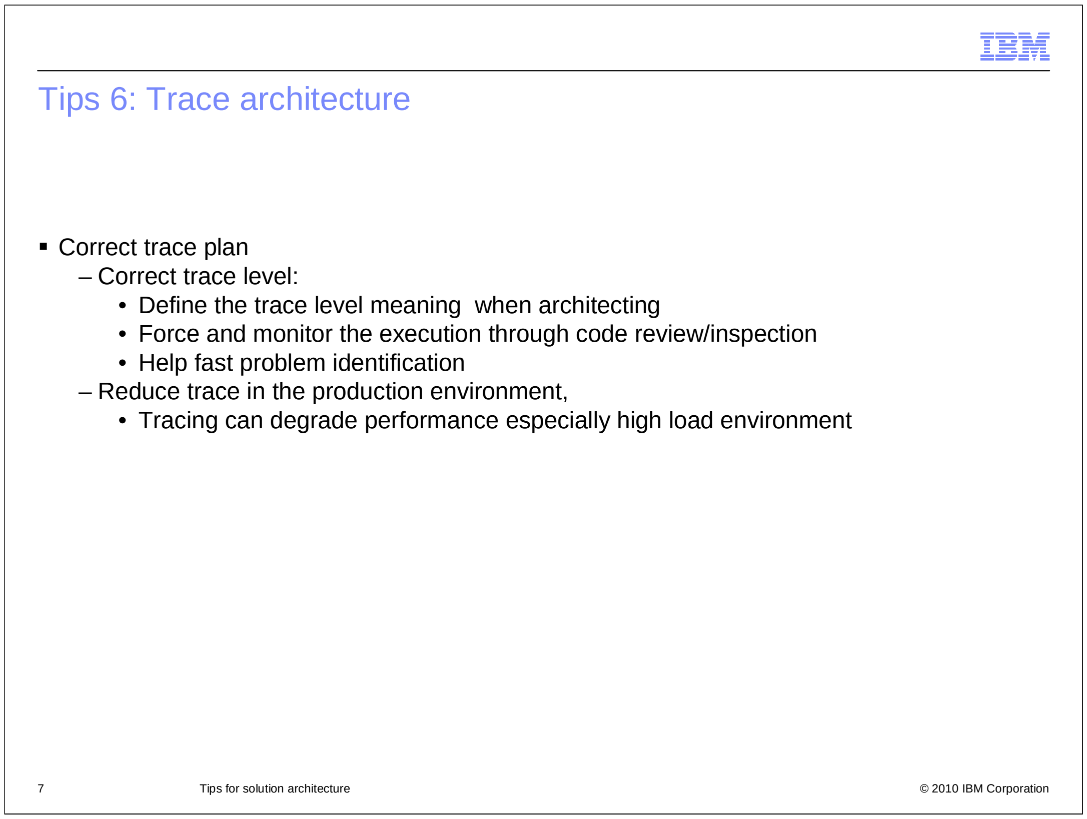
3.Separate trace/log can help customer fast find the position of BTT problems.

II.Standard technology of trace is mature these days

I.can reduce developer skills requirement

II.can reduce the cost for support

III.AOP can reduce trace development effort



Tips 6: Trace architecture

- Correct trace plan
  - Correct trace level:
    - Define the trace level meaning when architecting
    - Force and monitor the execution through code review/inspection
    - Help fast problem identification
  - Reduce trace in the production environment,
    - Tracing can degrade performance especially high load environment

7 Tips for solution architecture © 2010 IBM Corporation

I.Correct trace level

1.Improper level will increase the I/O dramatically

2.Improper level is not notable at development time, so need to plan it first

3.Review/Inspection before roll out

II.Reduce trace in production environment

I.Suggest use profiler tools to identify the percentage of trace

II.Put trace file on high performance storage such as RAID disks

Trademarks, disclaimer, and copyright information