Accelerate to Green IT - A practical guide to application migration and re-hosting

Explore methodology, best practices, and lessons learned by the IBM Project Big Green team during application migrations

Joydipto Banerjee
Debasis Choudhuri
LK Swift

July 16, 2012

This guide has been developed based on implementation experience in moving application workloads from a distributed environment, such as AIX® workload on Power®, or pSeries®, RS/6000® hardware, Solaris workload on Sun hardware or Linux® workload on x86 hardware (that is, IBM eServer® to IBM System z® primarily IBM System z9® or z10® models).

Introduction

Many large application development and maintenance accounts, thinking of migrating core applications and databases to a new environment, are at a loss as to where to start, how to plan and implement the migration, and how to avoid pitfalls in the process. A lack of awareness of standard methodologies or guidelines adds difficulty in the formation of an estimate for quickly and effectively migrating applications from one platform to another.

This article explores the highly successful IBM Project Big Green where the objective was to consolidate approximately 3900 IBM internal servers onto about 30 System z Linux environments. The objective of this article is to introduce the overall approach followed, share the best practices and tools, and provide the initial pointers to server consolidation and virtualization space.

Although the article will focus on like-to-like migrations from one UNIX® platform to another, it will be equally helpful in other migration scenarios. It is targeted to Migration Engineers, Migration Architects, and Technical Team Leads, and can serve as a reference in any migration engagement at all skill levels.

Overview of the migration process

Let’s first understand the terminology workload: a workload is an application or set of applications running on an Operating System (OS) in either a virtualized or non-virtualized environment. A workload consists of an OS running on hardware, middleware running on the OS layer, and a
set or similar group of applications running on the middleware system. Examples of database workload might be:

- DB2® or Oracle workloads
- Web application workload, such as Java™ applications, WebSphere® applications, Weblogic applications, or others
- Front-end workload as static images or pages
- Middle tier application workloads such as WebSphere MQ, Message Broker, web service, and more

Migrating various UNIX workloads such as AIX, Solaris, or x/Linux onto z/Linux (or any platform), might not be technically challenging. Remember that such an engagement can become complex due to a lack of experience in assessment and proper planning. A methodical guideline, along with a properly-phased approach, solidifies the transformation process. Figure 1 captures the overall phases of a typical migration cycle:

**Figure 1. Migration overview**

Any migration engagement can be broadly categorized:

- The **Discover** phase involves discovery of your server inventory and application dependencies
- The **Map** phase involves creating the migration request and target topology
- The **Provision, Migrate, and Configure** phases involve building the target environment, application deployment, and porting
- The **Test** phase tests the applications in the new environment after your migration and then goes live.

**Detailed migration phases:**

1. Identification / inventoring
2. Server / application qualification
3. Planning and design
4. Server / application migration
5. Post production

**Figure 2** below provides migration flow showing specific sub-categories. A typical migration engagement starts with **Server Identification and Inventorying** – a process which scans in-scope servers and identifies potential migration candidates. This potential candidate list is further refined
in the following step which is *Server / Application Qualification*. After a detailed feasibility study, the final migration candidates are chosen and logically grouped to form *waves*. This final list of qualified servers/applications is then taken to the next phase, known as *Planning and Design* where you do the detailed target topology and migration planning. With the detailed design finalized, you now enter into the implementation phase called *Server / Application Migration* where you build the target environment, migrate the applications are migrated, and thoroughly test them. Once you complete the migration, the new servers go live and the final *Post Production* phase follows where you decommission old servers.

**Figure 2. Detailed migration phases**

We will now delve into each of the phases and activities to understand the steps involved.

**Identification and inventorying**

Your first step is to identify which servers to migrate. You'll also inventory both servers and the software on each server.

**Server identification**

In the migration engagement, identifying the right set of assets, i.e., servers for the migration, is important. Defining a server in-scope or out-of-scope according to the agreed workload management is done by the project Architects and transformation program office. During the inventory verification process, one of the first tasks is to take stock of the existing workload (if Intel-based, mainframe, or other platform.). IBM Tivoli® Application Dependency Discovery Manager (TADDM) is a useful product to gain an understanding of the dependencies between servers, applications, network devices, software, configuration files, operating systems and other IT infrastructure components.
You can use the sample workload distribution model in Table 1 for initial selection criteria for the target environment. The actual utilization of source server in terms of CPU, RAM, Network I/O, and the percentage considered for a candidate to be chosen is hidden in this representation. This model, however, can be used with given agreeable utilization metrics to decide if a server is a candidate for migration or out-of-scope.

**Table 1. Sample workload distribution model**

<table>
<thead>
<tr>
<th>Affinity to:</th>
<th>Server characteristics:</th>
<th>Platform characteristics:</th>
</tr>
</thead>
</table>
| Mainframe    | • Lower sustained CPU peaks and average memory needs  
• High and/or transactional I/O  
• Proximity to other data in mainframe | • Software available on mainframe only  
• Other components of application on mainframe |
| Intel        | • Already virtualized on Intel  
• Small counts of isolated images | • Software available on Intel only  
• All Linux needs not met by Linux on mainframe |
| AIX/UNIX     | • High sustained CPU peaks on AIX/UNIX only  
• Software available on AIX/UNIX only | • Already virtualized on AIX/UNIX |

**Server and application qualification**

Further refine your list of potential candidates for migration.

**Transformation planning and feasibility study**

1. **Define the server complexity to estimate the migration effort**
   
   Once a server or a set of servers is identified as a potential candidate to migrate onto a virtualized target environment, the next important step is to categorize the server as simple, medium, complex, or very complex depending the various parameters of server hardware and software study. Figure 3 provides an example of the selection criteria:

**Figure 3. Migration complexity based on server type**

- **Simple** servers host only one application or one piece of an application under a single instance of an OS, such as Wintel-based single or dual-core CPU server hosting a single web application running in a WebSphere environment.

- **Medium** servers can host two or three separate applications, but do not have multiple virtual machines (VMs) defined, and still run under an instance of an OS, such as servers that run multiple instances of an application as WebSphere Application Server (WAS), DB2, IBM Http Server (IHS) front-end under the same OS, typically found in development and test environments.

- **Complex** servers are often servers with multiple CPUs which might have separate logical partitions (LPARs) defined. Each LPAR has its own copy of an OS or multiple VMs with separate copies of an OS and hosting multiple, or unrelated applications sharing the same resources of the system (such as network I/O). An example might be a System p® with multiple LPARs running separate AIX, p-Linux OS, or other OS and VMs, and running many different applications, sharing the same network I/O.
Very complex servers typically have multiple CPUs which can have separate LPARs, each with its own copy of an OS or multiple VMs with separate copies of an OS and which hosts multiple, unrelated applications sharing some system resource (such as network I/O), and which is clustered with other separate servers through hardware or software load-sharing or fail-over. Examples might be multiple LPARs running a separate OS of p-Linux or AIX hosted DB2 database with HACMP cluster.

2. **Define the application complexity to estimate the migration effort of the applications**

Server complexity definition may not derive the full understanding of the migration effort as a server may be simple, but the product, technology, or application running can be complex. Complexity categorization from an application perspective is equally important in understanding the migration. Figure 4 indicates the complexity range for applications from simple to very complex.

**Figure 4. Migration complexity based on application type**

**Simple applications**
- Databases if they include:
  - Smaller databases
  - Intra-Data Center (DC) migration
  - Servers with single instance implementation
  - Server with up to two application owners
  - Databases with no native language code to avoid code remediation
  - Applications with weekend outage windows available to them
- WAS/Java applications if they include:
  - Smaller JVM size or single JVM implementation
  - WAS AS-IS move, e.g., WAS 6.1.x to 6.1.x, WAS 5.1 to 6.0.x, 6.1 to 7.0.x (no API change)
  - App server with up to two application owners only
  - Application with no native language code (e.g., C/C++) on them to avoid code remediation
- IHS if it is:
  - Mostly static pages like HTML, images, JavaScript with few CGI scripts, Perl modules, or direct calls to database and dependency with hard-coded IP
- Domino® if it is:
  - A self-contained application within an .NSF database having no interaction with outside data sources or scripts

**Medium applications** are always on a case-by-case basis as evaluation between simple and complex depends on volume, user base, architecture, middleware products, and a combination of all these. For example, WebSphere Commerce (WCS) migration from WCS 6.x to 6.x without any custom JSP or custom module is a medium migration, but the moment the volume of the custom JSP or program modules increases, and versions upgraded
from 5.5/5.6 to 6.x, it tends to move towards complex or very complex depending on effort estimation analysis.

• Other examples of medium complexity migrations include:
  • Simple J2EE application migration which needs code re-work, but is just an API change from 1.4.2 to 1.5 (JRE version)
  • Type 2 to type 4 driver change
  • Domino applications using database or Java connections to external data sources [including the use of Lotus Enterprise Integrator® (LEI)]
  • Custom applications developed using tools available to the target environment (little porting required)

Complex applications
• Database with partitioned databases (DB2 with DPF) or larger size (Cross DC migration). Likely indicators include:
  • 1TB+ of storage attached to the server
  • Databases needing 365x24x7 support with small maintenance windows
  • Databases currently implementing Disaster Recovery (DR)
  • Data warehousing databases in need for high CPU/IO resources
  • Servers with many instances, such as three or more instances on the box
• WAS/Java applications - WAS version 4.0 or 5.0 to 6.0/6.1/7.0 (because of architecture change), WCS 5.5/5.6 to 6.x with minimum customization, portal migration with PDM or WCM with no customization
• IHS - a mix of static and dynamic content, large application back-end dependency with complex rewrite rules and complex back-end calls and many dependencies of CGI/Perl script with directory or external Perl module dependencies, CGIs coded with poor coding standards (requiring re-write)
• Domino® - third-party application code or extenders, Domino elements used within a portal, use of low-level Domino APIs or OS APIs, moving a medium complexity Domino application from Windows® to Linux (or Linux on System z).

In general, you can consider a migration to be complex when an application currently implementing DR, third-party application code, custom code with thousands of modules requiring porting, but using the same development environment, custom code requiring change of development environment (such as Visual Age to GNU tool suite)

Very Complex
• Databases - DB2 databases moving from AIX to zOS. This sort of migration requires major re-work for filesystem change, DPF with data volume over 1TB, cross database migration, such as ORACLE-to-DB2, Informix-to-DB2, migration for unsupported DB2 extender on zLinux.
• WebSphere - application currently running on an old version of WebSphere like WAS 3.5 or 4.0, and needs significant amounts of code rework so that the application code can be deployed in WAS 7.0, large volume of workflow customization in WebSphere Process Server using WebSphere Integration Developer (WID).
In general, a complex application can have a custom program developed in a language which does not have support in different OSs, a code-rewrite is needed in a separate language, applications requiring the use of multiple customized API programs, or custom applications requiring the use of APIs or libraries specific to the current environment.

**Wave planning: Migration in a group**

In Figure 5, the actual migration complexity level is determined by taking into account the complexity of both the application and servers.

**Figure 5. Blend the server complexity and application complexity**

The server/application migration typically happens through a wave (group) approach determined during server migration planning. After you identify the servers/applications that are qualified in the screening phase and assign migrations in a wave with high-level projected timeline. After the wave project kick-off, the server and application complexity will derive a migration timeline in terms of total complexity in the migration, associated with hardware/server and application binaries/data.

This article does not cover other wave planning processes such as application sequencing, application priority, financial year-end, or corporate freeze as it is too specific to each project.

Once you form a wave, the wave project manager prepares the project plan for the migration engagement. The project manager will consult with the client team to get an estimate of the effort required for system test and user acceptance test. From project planning perspective, the different phases are as follows (and shown in Table 2):

1. **Solution Initiation and Planning** - Performs the feasibility study and take critical technical decisions. Finalize the target environment.
2. **Execution and Control** - Create a detailed plan to execute and control the migration of each qualified application.
3. **Go-live and Close** - Finally, take the new environment live and follow project closure activities.

**Table 2. Wave planning phases**

<table>
<thead>
<tr>
<th>Solution Initiation &amp; Planning</th>
<th>Execution &amp; Control</th>
<th>Go-live &amp; Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Review and confirm materials. Consult asset principals.</td>
<td>• Complete detailed planning, execution &amp; control of the technical migration of each qualified application.</td>
<td>• Plan for cutover and post-implementation, closing activities.</td>
</tr>
</tbody>
</table>
Planning and design

As part of the planning and design phase, you and the customer will summarize the server and application behavior for the migration. Based on the assessment, you design a technical solution.

Application assessment

Application Assessment with the customer happens through a questionnaire survey followed by meeting with key stakeholders and technical crew members of the application team. Prepare a work-product of Application Assessment Questionnaire (AAQ) to capture the server behavior and application behavior of the scope servers/application to be migrated.

Some key attributes in an AAQ for user data capture are:

Server behavior:

- Server name (FQDN)
- Cluster (yes/no)
- Cluster server name
- Environment running (production/staging/testing/development)
- Server location (city/data center/hosting environment)
- Server type (web/application/database/hybrid)
- Network zone (internal/external/DMZ)
- Server IP address
- Hardware manufacturer
- Model type
- Number of processor
- Memory information
- Storage information
- Physical/virtual
- Server utilization
- Average utilization
- Peak utilization
- Peak timing
- Server configuration history

Application behavior:

- What is the name of the application running on the server?
- Provide a brief description of the application and its business function. Include what it does and the overall operation.
- Is this application a component or part of a larger, enterprise application group?
• If Yes, will this application and the other Enterprise Application Group (EAG) component applications need interlocking (for example, the application or other EAG component applications have dependencies or functions that are tightly coupled and must be treated as a unit for any project actions)?
• Was this application developed in-house or was it purchased off-the-shelf from an Independent Software Vendor? Chose one:
  • Custom / homegrown
  • COTS- no modifications
  • COTS- minor modifications
  • COTS- major modifications
• What is the principal application software vendor and release or version of the software used?
• What is the current platform this application runs on? (Windows, Linux, AIX, Solaris, Other)
• Is this the preferred platform or is another platform being considered or desired?
• Who is the Account Focal, DPE (Delivery Project Executive), or assigned delegate on all sign-offs (such as technical documents or UAT) for this application?
• Are any drawings or documents available that can aid in the overall understanding of how the application functions?
• Are any major changes, upgrades, or critical projects planned for the application or its host servers?
• Classify this application. Choose one:
  • Application standalone
  • Application & database
  • Infrastructure / utility
  • Web standalone
  • Web application & database
• Does this application use any common (shared) services? If Yes, please elaborate (for example: firewall, proxies and redirects, authentication, Lotus Notes® replication, MQ Series, web authentication). Usually a web-facing application can indicate that common services are involved.

More critical information would be needed to capture the network information and application details, as well as its future strategy and growth. Additionally, you might need separate questionnaires to capture details of the specific software, such as database(DB2), middleware (WebSphere Application Server), and messaging (WebSphere MQ), deployed for a particular application.

Technical solution design

Technical solution design is one of the most critical phases of the transformation management program as the input of inventory verification, server and application behavior, and the outcome of customer meetings are fed into the technical solution design.

The key activities of solution design captured in Technical Solution Design (TSD), an Excel-based work product, are:

• High level summary of the solution.
• Record of assumptions specific to the TSD and identified risks.
• Description of the solution including architectural decisions and application impacts.
• A table with all source server information.
• An application to server mapping table will contain an entry for each application and each server on which it executes. It will be a many-to-many relationship.
• A table with all target server information.
• An illustration and description of target environment.
• A solution description, consisting of architecture decisions and alternate considerations.
• Specific assumptions and risks.

Illustrations of certain architectural decision scenarios

While doing the technical solution design, key architectural decisions are required to be taken with respect to the target environment, target platform, target topology, and application compatibility with target environment. The three examples are:

• Example 1: product compatibility in Linux virtualized environment
• Example 2: Linux environment portability factor
• Example 3: Operational aspect in Linux environment Data Center

Example 1: Product compatibility in a Linux virtualized environment

• Subject area: Solution architecture on compatibility in Linux virtualized platform
• Issue or problem statement: The scope of the migration is to build a Linux stack which will be cloned into three customer environment such as, but not limited to:
  • Development
  • Test
  • Production

The J2EE application will be deployed in these three environments one after the other.
• Assumptions: J2EE container is WebSphere Application Server (WAS.)
• Alternatives:
  1. First, build the environment for Development with OS, WAS middleware, and then apply cloning technology for creating Test and Production environments
  2. Create the Development environment with OS, WAS Hypervisor Edition on Linux middleware, and then apply cloning technology to Test and Production
• Decision: Go with alternative 2.
• Justification: If you build Linux OS with WAS Standard or Enterprise edition, references of hostname and the IP address of the current environment will become integrated and tightly coupled during the installation. After building a new clone from the development Linux image, the WAS will not work in the new Linux-built image as it stores old hostname and IP reference in numerous places including cell-name and other spots in the WAS profile. Removing the profile and creating a new profile would cause additional effort for migration. To avoid this problem, choose WAS Hypervisor edition on Linux as it has no such tight coupling with the current environment. It can save you the manual effort of writing scripts to remove the dependency.
Example 2: Linux environment portability factor

- **Subject Area:** Example of solution architecture on platform selection.
- **Issue or Problem Statement:** Migrating the application and back-end database servers in AIX versus Linux on System z. All the server application and DB2 back-end components are currently running in non-virtualized AIX environment. The client wants to move the server to a virtualized environment, on either Linux or AIX, for the advantages of virtualization and a better computing platform from the scalability and operational cost perspectives.
- **Assumption:** The Linux virtualization platform and the AIX virtualization platform are available. The Linux pricing model is economically more attractive than AIX virtualization.
- **Alternatives:**
  1. Build all in-server components in AIX, virtualized, as the source platform is AIX. This involves minimum migration-related risks.
  2. Build all in-server component in Linux, virtualized, as it's economically beneficial to client.
  3. Build back-end database in AIX and front-end component in AIX and Linux, virtualized.
- **Decision:** Go with alternative 3.
- **Justification:** The DB2 component is ideally suitable for Linux on System z because Mainframe architecture is better equipped to manage workload where high I/O operation is expected and DB2 database back-end by nature incurs higher I/Os than application server component. The client, however, also needed clustering in the back-end DB2 component, which required keeping DB2 in AIX/pSeries because:
  - HACMP was a matured clustering tool compared to newly built Tivoli Storage Automation (TSA) and DB2 HADR in zLinux. Moreover, TSA was not tested well on z test environment.
  - Sustained high peak utilization for the DB2 servers (over 75%) created a close affinity to pSeries® hardware. The WAS component, which had a low CPU-intensive workload, was considered for Linux on System z.

Example 3: Operational aspect in Linux environment Data Center

- **Subject Area:** Example of Solution Architecture on Data Center Selection.
- **Topic:** Operation Model and hosting location. Build the infrastructure in the Poughkeepsie Data Center or the Boulder Data Center.
- **Issue or Problem Statement:** Migrating the server workload from a current IBM Data center located in Southbury, to new virtualization Linux environment in Poughkeepsie, NY or Boulder, CO.
- **Alternatives:**
  1. Poughkeepsie
  2. Boulder
- **Decision:** Go with alternative 1.
- **Justification:** The requirement of server capacity (high level) and the storage capacity versus availability of z9 and z10 as well as shared capacity in Poughkeepsie matched closely considering the future growth path.
• Close proximity of Southbury and Poughkeepsie helps ease data migration.
• Advantage of the same time zone of Southbury and Poughkeepsie minimizes the operational complexity.

A note on the capacity planning and server sizing:

Capacity management techniques are applied to determine optimal allocations or sizing for resources in a new consolidated server environment and to ensure performance for anticipated workloads. Appropriate sizing of the target server hardware is critical to ensure:

• Performance
• Future growth
• Consolidation ratio
• Financial benefits

Figure 6 shows the interrelationship of these factors to planning and sizing.

Figure 6. Factors for capacity planning and server sizing

The server sizing is based on the data collection performed during the assessment/inventory phase. As an example, you can calculate sizing on a Relative Performance value (rPerf) for each server after analyzing their current hardware model, performance metric of CPU, and number of cores and chips available in the existing hardware. Look for the following critical items:

• Server make
• Server model
• Number of CPUs
• CPU cores
• CPU speed
• CPU utilization
• Installed RAM
• Utilized RAM
• Storage allocated and used
The estimate is based directly on the peak CPU% of the other servers, adjusted for consideration of workload characteristics. The accuracy of a server consolidation sizing estimate is dependent on the input provided. The most common reason for inaccurate estimates is incorrect CPU% utilizations of the current servers. Each individual server's peak CPU utilization and the pattern of peak demand across all the servers that are used in the sizing are crucial to a good estimate. If peak loads are complimentary, that is they occur at different times, the server capacity requirement can be significantly less than if peaks are concurrent. Variations in the workload characteristics are also an important factor. Variations in workload characteristics can result in a 4x delta in the sizing result. Incorrect or inaccurate input makes the sizing results invalid. It is very important to check that the inputs used in the sizing accurately reflect the workloads and CPU% utilizations of the current servers.

It is also important that you gather the peak CPU% data correctly. They should represent the Average CPU% during a 15-30 minute interval of peak demand, not an instantaneous peak. If the customer has data on average CPU% utilization for an 8-hour shift or a day, a peak-to-average ratio may need to be applied to correctly reflect peak interval CPU% utilizations.

**Sizing parameters for the benchmark** include:

- Application programs
- Performance monitors
- Data files (datasets), and databases
- Scripts (user commands) or jobs
- Working set sizes
- Terminal simulation
- Size of user population
- Average think time and think time distribution
- Transaction rates
- Response time criteria
- Operational methodology

**Best practices** include:

- IBM uses information from the questionnaire as input to the sizing
- Sizing simulation converts customers' planned business volumes into potential workload
- CPU, memory, and disk requirements for the potential production
- Functional estimation to actual workload
- Understand sizing guidelines, methodologies, and tools
- Validate/differentiate sizing request between a sizing, a capacity planning, or a performance analysis exercise and the tools and methodology used in each unique scenario, as well as how it applies to this customer environment
- Use IBM Sizing Tools, where applicable, or ISV Sizing methodologies/tools, making sure the most current version of the sizing tool is used
- Understand how micro-partitioning affects the sizing and explain in output results
- Provide headroom and multiple sizing including a growth projection
- Tool accuracy - expected to be as good as +/- 30%
• Obtain a volumetric data point and make sure it's signed off, else it may trigger a domino effect and incorrect sizing
• Consider the impact of parallelism on batch processing

Sizing tool reference:

• **IBM proprietary sizing tool: VISIAN**
  VISIAN is an Excel-based IBM internal tool which captures source server technical configuration (such as number, type and speed of CPUs, memory), source server resource utilization (%CPU, %Memory, %NW, and more) and takes into account virtualization layer characteristics, limitations and overhead. VMware ESX, MSVS, Virtual Iron, and pSeries hypervisors are supported.
  VISIAN calculates the following:
  1. Number of required target servers
  2. Information about each target server
     a. Number of virtual machines
     b. List of source servers to virtualize in each target server
     c. Utilization of CPU, memory, network, disk space and disk I/O
  3. Physical space required (rack units)
  4. Hardware and virtualization software cost

• **Third-party popular sizing tools**
  A. VMware Capacity Planner
     Unlike other tools, VMware Capacity Planner is a hosted application service which works only for target VMware environment. It installs a number of components on the network that collects and manages data. The data is then sent back to VMware for analysis. A significant disadvantage of this is non-ownership of the software and the inability to use it for ongoing work. When the vendor analysis is complete, the client is usually presented with scenarios offering different configurations to achieve the virtualization goals. The Capacity Planner service is available from VMware channel partners, including consultants, hardware vendors, software vendors and other outlets.
  B. Novell PlateSpin PowerRecon
     Novell's PowerRecon tool integrates functions for remote data collection, workload analysis, and planning and scenario comparisons for server consolidation. It automatically analyzes the following dimensions of workload: CPU, disk, memory, and network.
  C. CiRBA
     CiRBA can develop rough estimates for hardware sizing as a starting point by analyzing CPU, memory, IO, overhead, and storage.
  D. VMware Guided Consolidation
     This built-in tool is a part of Virtual Infrastructure 3 (VI3) targeted at smaller IT environments.
     It performs analysis on a selected group of systems, gives advice on the best servers for virtualization, and can perform the Physical-to-Virtual (P2V) conversion.

Target topology for frame placement:
Another important aspect of migration, especially in a virtualized environment, is to design and make decisions on the target topology and distribution of the Virtual Machine (VM) guest into the right frame or physical container.

**Application stacking and dependency analysis:**

Capacity planning exercise and dependency consideration should be discussed and decided at the solution design phase. Consider a variety of deployment factors to arrive at the right application stacking. Some deployment factors are outlined here while doing the separation:

i. Software stack version - For example, WAS 6.0 applications versus WAS 7.0 applications. WAS support lifecycles differ and their maintenance or fix pack release frequency may not be same.

ii. Security - Group the applications needed for level 4 data security, SSO versus non-SSO authentication in a separate frame to maintain better separation of duties and isolation.

iii. Performance and throughput - Applications needing faster response SLA, applications requiring higher memory footprint to sustain the desired performance, such as JVM with a 2GB heap size might lead to a dedicated application server compared to a simple application requiring a 256 to 512 MB JVM heap size.

iv. Scalability - The shared applications which are scalable to upgrade, the applications which have plans to introduce web services in an upcoming release, would-be disaster recovery applications, and other categories.

v. Availability - Based on SLAs

vi. Disaster Recovery (DR) level - Group Tier 1 and Tier 2 applications to design an optimized shared DR infrastructure.

**Application-level analysis for frame placement in virtualized environment:**

The decision of VM guest placement in a virtualized environment is important from an application correlation perspective, for example to spread across VMs for higher SLA applications. You can identify the application functional requirement such as, but not limited to, data processing, higher I/O driven batch jobs, high volume transaction processing, web rendering along with peak load times over a period of a quarter or a year. Accordingly, you can then decide to place them in a right frame to distribute the workload of the whole frame.

You can also exceed 100% allocation of resources, that is **oversubscription**, a virtualization on-demand capacity planning, to address the actual physical capacity recommended as an upper threshold. This decision might be supported by the fact that not all VMs will run at peak at the same time, and therefore processor capacity will be available to account for over allocation. For example, a Linux image for batch-server kinds of workload can work cohesively with another Linux image of the transaction server, beyond the capacity available because the batch workload will be active at night when the transaction server is idle or semi-idle. The overall workload can thus be balanced well and meet oversubscription.

**Server and application migration**

Finally you are ready to work on migration of the server and applications.
IT environment build

Once the solution design is in place, it is time to work on the target environment build. A document commonly known as a Build Sheet is compiled containing details and specification for the would-be target images. By this time, sizing of the target hardware should be completed as well as the list of user requirements related to user IDs, file systems, and other items.

The actual IT environment build process might be automated using tools like the IBM Tivoli Provisioning Manager (TPM) or it might be manual. Depending upon the approach adopted, the build sheet might be Excel-based (for manual process) or a self-service, web-based interface portal pointing to the automated provisioning tool (for example TPM).

Whichever approach you adopt, some of the basic details in the Build Sheet are:

- **Request group details**
  - Date created
  - Requester

- **Source servers summary**
  - Number of servers
  - Total CPUs
  - Total Memory

- **Target servers summary**
  - Number of servers
  - Total desired CPUs
  - Total desired memory
  - Total local disk size

- **Administration information**
  - Application name
  - Customer
  - Project manager

- **Host and network information**
  - Host
  - Host location
  - Host architecture
  - Primary IP address
  - Fully qualified domain name

- **Software components**
  - Operating system
  - Database
  - Middleware

- **Local filesystems**
  - Mount point type
  - Size (MB)
  - Owner
  - Group
  - Permission
Once the above request is reviewed by all the stakeholders, the migration team submits it to the server build team which prepares to hand over the images once they are ready. The migration team then begins the activities outlined in the migration plan.

**Application migration and unit tests**

Before the migration activities can begin, it is necessary to document all the steps involved in the process. This phase is called Migration Planning and requires preparation of a Migration Plan. The Migration Plan is a very detailed document that describes all the tasks to be performed in sequence by the migration team. It includes the name of the activity, its owner, start dates and expected duration. Each member of the migration team is expected to perform his own tasks as mentioned in the plan. The Migration Plan thus forms an excellent tracking document. A spreadsheet-based Migration Plan typically has the following sections:

- **Cover page:** Project name, document approvers, revision history
- **Servers:** Migrate server names
- **Pre-migration:** Tasks for each software applicable for the migration.
  - Verify the installed DB2 client/server
  - Obtain detailed list of table spaces from the source server
  - Prepare for the DB2 backup and restores
  - Create DB2 instance on the target
Some of the application-specific tasks included in this section are performing server readiness check, migrating application files systems and user home directories from the source to the target servers.
- **Migration:** Tasks relevant for each software. Environment setup and code remediation (setting user profiles, login shells, environment variables, correcting hardcoded paths in application scripts) are also done at this time. Examples for DB2 tasks include shutting down database servers in the source environment, initiating off-line database backups, and restoring database onto the target. Once the application is correctly installed and is operational in the new environment, conduct environment verification tests/unit tests.
- **Post-migration:** Perform cleanup tasks. Remove any custom script or temporary user IDs created during the migration.
- **Contact Details:** List the names of all the persons involved during the migration activity, along with their contact details.
- **Issues:** (optional) Document issues faced during the migration or any relevant comments.

**Server readiness check:** *(Applicable for both production and non-production environments)*

After delivery of the target images to the migration team, a series of checks are performed on the server images to validate that they conform to the requirements (mentioned in the Build Sheet).
This step is known as the Server Readiness Check and consists of UNIX commands to check the image parameters.

Example:

1. Have the volume groups, volumes, filesystems, and mount points been set up and configured as specified by the Build Sheet?
   
   Example:
   
   # lvs    or    # lvdisplay
   # vgs    or    # vgdisplay
   # cat fstab     ( to check mount points )

2. Have the filesystems been setup correctly as specified in the Build Sheet?
   
   #df –h  < filesystem>

The usual checks are categorized as **Users**, **System**, **Storage**, **Software Installed**, and **Any Special Instructions**. The Migration Engineer runs through each and approves or rejects them. If there are major discrepancies, the images are sent back to the infrastructure team for correction. Only after sign-off is the Application Migration begun.

Pre-migration tasks:

- **For non-production environments:**
  
  Before you shut down the source servers and applications, inform users about the outage. Each middleware specialist performs a set of tasks pertaining to the setup and configuration of the software like DB2, Lotus Domino, and WebSphere MQ. In parallel, the tasks of migrating the application binaries and file-systems from the source to the target servers are initiated. This is also when user home directories are copied from the source to the target environment.

  The commonly used methods of file transfer from the source to target are **tarring** and then using **ftp** mode to copy the files or using **rsync**.

- **For production environments**:
  
  In Production environment, the tasks relating to setting up and configuration of various software like DB2 or Lotus Domino are carried out as mentioned earlier. Instead of source environments, the application files and binaries are copied from the newly migrated development servers (after the servers go live). As in a non-production environment, the user home directories are copied from the respective production source servers.

Migration tasks (**For both Production and non-Production environments**):

This is the main phase in which the actual migration tasks as defined in the migration plan are carried out by the respective specialists, in the areas of various software like DB2, Lotus Domino, or WebSphere MQ.

- The Migration Engineer verifies that the application files systems and permissions on the target environment are set up as in the source servers. Some key activities during this time are:
  
  - setting user profiles, login shells
  - setting application environment variables
- correcting hardcoded paths in the application scripts
- Once the application is installed correctly in the new environment, do any application source code remediation as determined during the feasibility study phase and the results of the Unit Tests. The main reasons for code remediation are changes in:
  - Operating system
  - Compiler
  - Software versions
  - UNIX shell script
- Conduct a thorough Unit Test before you hand the new server is over to the application team for User Acceptance Tests.

**Note:** The effort and complexity of the code remediation and porting work is almost negligible in Production environment because most the work was already done in the Development servers.

**Systems integration testing and user acceptance testing**

Once the migration work is completed, the applications installed and ported to the new environment are handed over to the client team to verify and validate. During this phase the client's testing team conducts the application-level tests to check that business functionality does not break and performance level is satisfactory. The client team may consult with the migration team for certain issues or troubleshooting during this testing.

During the testing phase, the Wave Project Manager or assigned Assessment Engineer will have a coordination role, working with the client team to:

- Collect test progress and defect status from client's test team daily and will assist with pushing defect resolution
- Report test status to Management and Project Office on a weekly basis
- Assist with test dependencies, risks, and issues

The Wave Project Manager or the assigned Assessment Engineer will not, however, execute tests or validate results as this is generally the application test team's responsibility.

**Servers cutover to production**

*For non-Production environments:*

With the User Acceptance Tests complete, the client team signs off on the new servers. The post-migration tasks involve removing any custom scripts and files which were temporarily installed on the new server to aid in the migration.

*For Production environments:*

In the production environment, a completely new set of cutover activities now begins. This involves shutting down the production servers in the source environment and going live in the new environment. During this transition, the users are affected so this activity is usually done during the application maintenance window and mostly over weekends to minimize application downtime.
A spreadsheet-based cutover plan consists of the following sections:

- **Servers** lists the names of the production servers to be cutover.
- **Pre-cutover** lists tasks which include preparation to shut down the source servers, final verification of the installed software and application files in the production environment, database backup in source and restoration in target environment, submission of URL change requests, and other actions.
- **Cutover** lists the actual shutdown sequence of applications and batch jobs in the source environment, starting these in the target environment, implementing URL/DNS change requests, final tests, and notifying users about the availability of the new system.
- **Post-cutover** lists the wind-up of cutover activities. Tasks deal with coordinating changes with the upstream/downstream production environment, completing all the remaining documentation tasks, and monitoring the new environment until it is stabilized.
- **Rollback** addresses the eventuality of the new environment failing after it goes live with a provision to return to the earlier environment. The steps include initiating the back out process, starting up of the software and application in the old environment, running the batch jobs, and redirecting the URL/DNS to point back to the old systems.
- **Assumptions** lists any assumptions related to the cutover, such as:
  - All programs, scripts and tables are on target environment prior to data moves
  - Only application shutdown data is required to achieve production in target environment
  - All testing is complete to satisfy the relocation into target
- **Contact details** names all persons involved in this cutover, along with contact details.
- **Issues** is an optional documentation of issues faced during the cutover or any relevant comments.

Common to both Production and non-Production environments:

**Health Check**

The application team is responsible for performing a health check on the application to make sure everything is up to speed. The infrastructure team then performs a final health check on the images before go-live. This includes security patches, monitoring, and ensuring that backups are in place. Depending on how many images are involved, this can take two to four days, and needs to be planned accordingly. The infrastructure team will need to give the thumbs-up at the go-live meeting indicating that these items are complete.

**Post production**

After the servers are in production, the migration team takes two final steps to provide a warranty period and sunset the old servers.

**Post cutover warranty**

Once the target servers are in production, the migration team usually monitors the performance of the new environment and is on stand-by to resolve any issues. This generally lasts from ten days to two weeks and is referred as *warranty period*. During this period, the client has access to
migration team members for any clarification or troubleshooting. After the warranty period ends, the client team is responsible for all maintenance and server upkeep.

**Sunset and decommission or repurpose**

After both the new non-production and production servers become operational and complete a pre-defined number of days without any major issues or downtime, the old servers are sunset and either decommissioned or used for another purpose.

**Tracking**

During the entire migration process, it is essential to track the project phases and deliverable from end-to-end. For this reason, it is advisable to have a checklist, commonly known as a *Technical Dashboard Checklist*, an Excel-based artifact which contains the items in Figure 7.

**Figure 7. Technical dashboard checklist**

In the Technical Dashboard Checklist, the Item/Task column lists the activities or deliverables, namely the tasks in the migration, cutover, and other plans. It also lists the owners of each task, the target and completed dates, and the completion status. During the migration phase, as a best practice, the Migration Engineer updates this checklist at the end of each working day to reflect the current status of each task and deliverable. A color-coded scheme (green, yellow, and red) helps visualize the health of the project at any given time.

**Conclusion**

This article introduced the concept of application migration along with guidelines on how to plan, prepare, and finally implement the activities. You now have a fair idea of the entire migration phases, the major architectural decisions required, the work products to prepare, and know how to avoid some pitfalls in the process.
Related topics

• In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
• Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment.

© Copyright IBM Corporation 2012
Trademarks
(www.ibm.com/developerworks.ibm/trademarks/)