Workload Placement
and other Aspects of Analytics, Design, and Planning
Content

- Introduction
  - Sign-up
  - Link to previous lectures
  - Overview

- What - Constraints

- How - Migration Techniques

- When - Wave planning

- Where - Placement algorithms

- Mapping to Cloud Images

- Projects

- Reading list
Gain vs. Pain of Transformation

As you saw in Enterprise IT, real world (Lecture 2a)

We started here with discovery (Lecture 3)

Migration Cost

Steady state cost benefit

Pain: We want this to be the least!

Gain: We want this to be the most!

E.g., a cloud as in Lecture 2

Source System Cost for steady state

Target System Cost for steady state

Design & Approach
Planning & Implementation
Realization

Improved HW/SW Utilization
Reduced Power Consumption
Improved Operational Processes

Cost
Improved HW/SW Utilization

We started here with discovery (Lecture 3)
Abstract Architecture for Enterprise IT Transformation

Today we are here

Customer and infrastructure data collection
Analytics, Design, Planning
Procurement and physical setup
Migrate and modify
Test and remediate
Synchronize and cutover

Non-customer sources

Consolidated data repositories

Migration Analytics

Project workflow and status visualization

Today we are here
Detailed Architecture for Enterprise IT Transformation

We’ll talk most about this part

Customer and infrastructure data collection (manual and automated)
- Infrastructure data collection
- Organizational data (owners ...)
- Customer goals and plans

Analytics, Design, Planning
- Target options
- Technical compatibility
- Business constraints
- Placement optimization
- Wave planning
- ROI analysis

Procurement and physical setup
- Hardware and software ordering
- Setup of physical infrastructure if new

Migrate, modify
- Core migration methods:
  - Lift-and-shift
  - Copy
  - Provision and backup-restore
  - Apply changes from design

Test and remediate
- Unit tests
- Comparison tests
- User acceptance tests
  If fails, back to “modify” or initial state

Synchronize and cutover
- Educate
- Resynchronize test system with source
- Switch operation over
- Registrations etc.

Non-customer sources
- External product descriptions
- Benchmarks

Consolidated data repositories
- Customer data
- Historic data
- General technical and financial data
- Models

Migration Analytics
- Operational Improvement

Project workflow and status visualization
- Benchmark assessment
- Macro design
- Micro design
- Contracts
- Possibly retry

Possibly several optimization phases

For large migrations in waves = groupings migrated at different times. Possibly even micro-design and procurement in these waves
### Overview of Analytics, Design and Planning

- All the data used here are in the consolidated data repositories, or must be estimated based on those data.

#### Analytics

**Benchmark assessments**
- Estimating gains and cost with few customer data, more historic data

**Technical compatibility**
- Depending on consolidation layer: SW-OS, OS-VM or OS-HW, cloud image options, software upgrade and change options (possibly with prices)
- Simple size constraints
- Migration costs for different scenarios

**Business constraint analysis**
- Workload-level performance (rough – no benchmarks for complex changes)
- Availability, backup, disaster recovery
- Security and compliance
- Other mgmt processes: Maintenance windows, change mgmt, ....

**Optimization (ultimately for cost)**
- Using resource utilization factors, for target systems from customer goals.
- Placement: Multi-dimensional bin packing: multiple utilization dimensions, and the technical and business constraints. Goal variable: typically cost, using the financial data. Might also be power or floor space.
- Rarer: storage and network usage optimization

**Wave planning**
- Groupings of servers or business application for joint migration

---

**Procurement and physical setup**

---

**Consolidated data repositories**

---

**Analytics, design and planning workflow**

- Lots of reports and visualizations, as this is not fully automated yet
- ROI analysis may also include soft cost (licenses, server management ...).
- What-if-scenarios, storing important intermediate results, signoffs
WHAT - CONSTRAINTS
Technical Constraints

- **If new hardware:**
  - Is there binary compatibility with the old hardware, i.e., everything can move unchanged?
  - If not, can one at least keep the same OS (operating system)?

- **If virtualization:**
  - Can the current OS be virtualized?
  - Can all software on it be virtualized?

- **If OS changes:**
  - Does the off-the-shelf software also run on the newer / other OS?
  - Is there custom code that needs changes? (E.g., different system calls, different Java versions.)

- **If software changes:**
  - Can customer data be transferred without much change?
  - Are there tools for this data transformation? (E.g., there are tools to move data between different types of databases)

- **If migration to cloud:**
  - Are there images of the needed sizes in the cloud?
  - With the desired operating systems?
  - And, if it’s a PaaS cloud (platform-as-a-service), with the desired software on them?

Instead of just “yes / no”, the answers may be upgrade / change costs. Needs large knowledge bases or good experts
Business Constraints

- **Performance constraints for business applications?**
  - May influence whether we can move something to a different data center, or a different network structure. I’m not aware of standard tools to evaluate this like the placement of individual machines.

- **Availability, backups, disaster recovery?**
  - If we only migrate to a different data center or newer hardware, this may not be a problem, i.e., one just keeps the current software doing it
  - With virtualization, don’t move a server and its standby to the same physical server
  - In a cloud, does the user interface allows us to do guarantee this?
  - What repair times etc. does a cloud provider offer if hardware or hypervisor fails?
  - Does a cloud help with backup etc. and we might not need the old software for it?

- **Security and compliance:**
  - Certain applications and data may not be allowed to be on the same server or even in the same network segment with others
  - Some applications may not be allowed to be in a cloud managed by another party
  - One may need encryption if there is new wide-area data transfer, or new external storage

- **Other mgmt processes:**
  - **Maintenance windows:** Is the way a cloud provider does patching, hardware upgrades etc. suitable? (“This application will not be reachable from Saturday 9pm to Sunday 6am”)
  - **Changes** in the infrastructure may need to be logged and auditable (for compliance), which is harder in a cloud.
HOW - TECHNIQUES
How to Decide Which Migration Type to Use?

1. **Discover/Analyze**
   - MAP
   - Provision
   - Migrate

2. **Fit for Scenario**
   - Analysis directs approach

3. **Workload/Application Migration**
   - Image Migration
     - App
     - OS
     - x series
   - Image Fix-up
     - Cloud
     - XPZ

4. **Tooling**
   - Image copy
   - Fix-up

5. **Test**
   - user
   - Source
   - Target
Classifying Workloads

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Migration Strategy</th>
<th>OS upgrade</th>
<th>MW upgrade</th>
<th>Application upgrade</th>
<th>OS/MW Post-configuration</th>
<th>Systems management Post-Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 (live virtualization and migration)</td>
<td>“classical” p2v, migrate over to target and cut-over instantaneously post-migration</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>S2 (migration of stand-alone apps into test env)</td>
<td>Stand-alone app hosted on single machine on source is migrated into a virtualized test env on target</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>Yes (IP, hostname, filesystem)</td>
<td>Yes</td>
</tr>
<tr>
<td>S3 (distributed apps)</td>
<td>Distributed app hosted on multiple machines on source is migrated into a virtualized env on target</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>Yes (IP, hostname, UID, filesystem, dependencies)</td>
<td>Yes</td>
</tr>
<tr>
<td>S4</td>
<td>Like s3 but MW upgrade. Virtualize into target test env and run MW upgrades post-migration.</td>
<td>No</td>
<td>yes</td>
<td>Yes/no (depending on MW version)</td>
<td>Yes (IP, hostname, filesystem) followed by application specific re-configuration (Data sources, Queues, etc.)</td>
<td>Yes</td>
</tr>
<tr>
<td>S5</td>
<td>Like S2 but requires application upgrade. Migrate and upgrade installation post-migration</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>Yes (IP, hostname, filesystem)</td>
<td>Yes</td>
</tr>
<tr>
<td>S6</td>
<td>Like S1, S2, S3 but with OS/MW version upgrade. Rebuild system on virtualized target environment on new OS/MW version</td>
<td>yes</td>
<td>yes/no</td>
<td>yes/no</td>
<td>Yes (everything needs to be reconfigured after re-building on target)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Migration Complexity

- **Super Complex**
  - C/C++

- **Sweet Spot**
  - Scripts
  - J2EE
  - MQ
  - DB2

- **Easy Schmeezy**
  - WPS
  - WAS

- **‘Flips’**
  - IHS

- **Any**

*Industry specific and legacy enterprise applications most likely to be complex to super-complex*

*Frequency distribution for characterization of workloads expected to be industry specific*
### Replatform Migration

- Production application migrated from Sun Solaris 2.5.1 to IBM AIX 5.3. LHS effort required 4 FTEs, RHS effort required 11 FTEs partially due to 270KLOC C++ code

<table>
<thead>
<tr>
<th>Layer</th>
<th>From Product &amp; Version</th>
<th>To Product &amp; Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Solaris 2.5.1</td>
<td>AIX 5.3</td>
</tr>
<tr>
<td>Database</td>
<td>Oracle 7.3.3</td>
<td>Oracle 10g (10.2.1.0)</td>
</tr>
<tr>
<td>Application (Code Compiler)</td>
<td>CenterLine C++ 4.2 (non-standard ANSI C++)</td>
<td>VisualAge XLC 8.0 (standard ANSI C++)</td>
</tr>
<tr>
<td>Queue Manager</td>
<td>WebSphere MQ Series 5.0</td>
<td>WebSphere MQ Series 6.0.1</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Veritas Volume Mgr 2.6</td>
<td>Veritas NetBackup 5.1</td>
</tr>
<tr>
<td>Scripting Language</td>
<td>Perl 5</td>
<td>AIX Perl Runtime 5.8, Live Extension 2.1</td>
</tr>
<tr>
<td>Web Server</td>
<td>Netscape Enterprise 2</td>
<td>WebSphere Application Server (WAS) Network Deployment 6.0.2</td>
</tr>
<tr>
<td>Application (Code)</td>
<td>Java JDK 1.1.6</td>
<td>Java 1.4</td>
</tr>
<tr>
<td>Middleware</td>
<td>BEA MessageQ 3.2B</td>
<td>BEA MessageQ 5.0</td>
</tr>
<tr>
<td>Middleware</td>
<td>IDI ObjectQ 4.1.1</td>
<td>IDI ObjectQ 6.0.1*</td>
</tr>
<tr>
<td>Middleware</td>
<td>Rouge Wave 7.86</td>
<td>Rouge Wave SourePro 9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>From Product &amp; Version</th>
<th>To Product &amp; Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Apache Xerces XML Parser 1.6</td>
<td>Xerces XML Parser for C++ + 2.7</td>
</tr>
<tr>
<td>Source Code Mgt</td>
<td>Sablme</td>
<td>SCME</td>
</tr>
<tr>
<td>CORBA support</td>
<td>Iona Orbix 3.0.1</td>
<td>Iona Orbix 3.3.9</td>
</tr>
<tr>
<td>CORBA support</td>
<td>Iona OrbixWeb 3.0</td>
<td>Iona Orbix 3.3.7</td>
</tr>
<tr>
<td>Build</td>
<td>Lucent nmake 3.1.2</td>
<td>Lucent nmake 3.8</td>
</tr>
<tr>
<td>Balancing &amp; Failover</td>
<td>No Balancing or Failover</td>
<td>Veritas Cluster Server 5.0</td>
</tr>
<tr>
<td>Data Abstraction (Database &amp; Application)</td>
<td>Progress Persistence 3.0</td>
<td>Progress DataXtend 9.1*</td>
</tr>
<tr>
<td>Code (Library)</td>
<td>EMA Library</td>
<td>EMA Library (ported to AIX)</td>
</tr>
<tr>
<td>Code (Library)</td>
<td>UTI with GENI</td>
<td>UTI with GENI (ported to AIX)</td>
</tr>
<tr>
<td>Data</td>
<td>Teradata Tools/Utils 4.1.1 (required by UTI GENI)</td>
<td>Teradata Tools/Utils 7.1 (required by UTI GENI)</td>
</tr>
<tr>
<td>Environment</td>
<td>MO</td>
<td>ITO</td>
</tr>
</tbody>
</table>

* Progress certified DataXtend 9.1 for XLC 8.0 during project

**Key:**

- **Straightforward**: actual work matched expectation
- **Somewhat more difficult and time-consuming than expected**
- **Significantly more difficult and time-consuming than expected**
WHEN - WAVE PLANNING
Wave Planning

- Typically you can’t move everything at once, e.g.:
  - Too many servers to transport at once
  - Too much disturbance in the enterprise
  - Too few experts to perform certain types of changes at once (say from 1 database type to another)
  - Different maintenance windows for different business application

- So you need a plan for what to move when, and details such as personnel, notifications, etc.
  - Largely by business applications, e.g.,
    - “travel reimbursement” and “awards” on weekend 1
    - “payroll” on weekend 2
    - ...
  - But business applications may have common servers or software components, so maybe while we migrate “employee directory” many other applications go down
    - May be acceptable
    - Or needs special solutions with replication, so that “employee directory” keeps running essentially all the time.
Simple View of Wave Planning

- In reality, not all dependencies are equally important, so plain min-cut is not a final solution.
- Can help with initial representation, though, in particular if mapping of business-applications to servers is not given from the customer.
WHERE - PLACEMENT
ALGORITHMS
Running Example, Basic Case

- P2V transformation (= physical to virtual), all on the same hardware type

Planned virtual machines, exactly the former content of 6 physical machines with CPU utilization

Physical machines. We’ll fill them up to 70% CPU utilization

How one later really produces the VMs is the “image migration” part of the next 2 lectures.

How many servers do we need? Which VM goes where?
Placement as Bin Packing

- This is called a “bin packing” problem in Computer Science
- It is NP-complete. This means there is no known polynomial-time solution, and finding one would be a real breakthrough
- Hence one uses heuristics if one has large problem instances. (We might have hundreds or even thousands of VMs). We will look at some basic ones.
- “First fit” (FF) is the simplest, essentially without intelligence: Take the VMs in the given order and put each on the first server where it fits.

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM₁</td>
<td>35</td>
<td>35</td>
<td>VM₂</td>
<td>40</td>
<td>40</td>
<td>VM₆</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>VM₃</td>
<td>15</td>
<td>50</td>
<td>VM₅</td>
<td>25</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM₄</td>
<td>15</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Recall: We only fill a server up to 70%
- That’s why VM₂ does not fit on P₁ with VM₁.
Placement as Bin Packing, More Heuristics

- **“Best fit” (BF):** Take the VMs in the given order and put each on the server where it fits best, i.e., where it leaves the smallest rest

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th></th>
<th>P2</th>
<th></th>
<th>P3</th>
<th></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>CPU</td>
<td>Sum</td>
<td>VM</td>
<td>CPU</td>
<td>Sum</td>
<td>VM</td>
<td>CPU</td>
</tr>
<tr>
<td>VM₁</td>
<td>35</td>
<td>35</td>
<td>VM₂</td>
<td>40</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM₅</td>
<td>25</td>
<td>60</td>
<td>VM₃</td>
<td>15</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM₆</td>
<td>10</td>
<td>70</td>
<td>VM₄</td>
<td>15</td>
<td>70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- In this case Best-Fit was really better than First-Fit, but not always. E.g., if VM₂ has 15% and has VM₃ has 40%, we get 3 servers again
Placement as Bin Packing, More Heuristics

- **“First fit decreasing”** (FFD): Order the VMs by size, largest first. Then put each on the first server where it fits.

| P1 VM | CPU | Sum | P2 VM | CPU | Sum | P3 VM | CPU | Sum | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VM2</td>
<td>40</td>
<td>40</td>
<td>VM1</td>
<td>35</td>
<td>35</td>
<td>VM6</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>VM6</td>
<td>25</td>
<td>65</td>
<td>VM3</td>
<td>15</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM4</td>
<td>15</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- So in this case FFD was not better than FF

- **“Best fit decreasing”** (BFD): Order the VMs by size, largest first. Then put each on the server where it fits best, i.e., where it leaves the smallest rest.
Formalization of the Optimization Problem

Inputs:
- Let $VM_1, ..., VM_n$ be the virtual machines, with CPU utilization $CPU_1, ..., CPU_n$.
- Let $P_1, ..., P_m$ be the physical servers.

Variables to determine:
- Let $A_{ij} = \begin{cases} 1 & \text{if } VM_i \text{ on } P_j \\
0 & \text{else} \end{cases}$ (“A” for “assignment”)

Constraints:
- Each VM goes on exactly one physical server:
  \[ \forall i: \exists! j: A_{ij} = 1 \quad \text{or, equivalently,} \quad \forall i: \sum_j A_{ij} = 1. \]
- Each physical server should not be more than 70% utilized
  \[ \forall j: \sum_i A_{ij} \cdot CPU_i \leq 70. \]

Goal variable to optimize: Number $k$ of physical servers used, i.e., with at least 1 VM:
\[ k = |\{ j \mid \sum_i A_{ij} \geq 1 \}|. \]
Solving with General Solvers

Claim: This can be written as an integer programming problem, i.e., linear equations and inequalities only, and solved with general solvers, at least for small problem instances.

Rewriting:

- The only equation that is not linear is the definition of the goal variable, $k = |\{ j \mid \sum_i A_{ij} \geq 1 \}|$.
- We introduce intermediate variables $used_j \in \{0, 1\}$ denoting whether server $P_j$ is used by at least one VM.
- Then the goal equation becomes linear: $k = \sum_j used_j$.
- We can define $used_j$ from the initial variables to be determined ($A_{ij}$) as follows:
  - $\forall j: used_j \leq 1$ (* Thus it’s 0 or 1 *)
  - and $\forall j, i: used_j \geq A_{ij}$ (* Thus if $A_{ij} = 1$, then also $used_j \geq 1$ and thus = 1 *)
  - and $\forall j: used_j \leq \sum_i A_{ij}$ (* Thus if all $A_{ij} = 0$, then also $used_j = 0$ *)

For how many VMs will a solver solve this in reasonable time?
More Parameters than just 1 CPU Utilization

- Only looking at 1 CPU utilization parameter was a good start to understand how one solves such problems.
- In practice we need to look at more:
  - Also memory, disk IO bandwidth, network bandwidth, and possibly disk space if that is local on the servers (in contrast to network file systems or storage-area networks)
  - All these parameters vary over time. Do we take averages, peaks, time series?
Multi-Dimensional Optimization

- Let’s extend our example by memory, i.e., a second dimension.

<table>
<thead>
<tr>
<th></th>
<th>VM₁</th>
<th>VM₂</th>
<th>VM₃</th>
<th>VM₄</th>
<th>VM₅</th>
<th>VM₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (%)</td>
<td>35</td>
<td>40</td>
<td>15</td>
<td>15</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Let the physical servers have 4 GB each

- How do we extend the heuristics?
  - First-fit: Still put each VM on the first server where it fits (now both CPU and memory)
  - Best-fit: Put each VM on the server where it fits best – what is “best” now?
    - VM₁, VM₂ clear. Let’s look at 2 alternatives (blue, purple) for VM₃.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th></th>
<th>P2</th>
<th></th>
<th>P3</th>
<th></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>CPU</td>
<td>Sum</td>
<td>Mem</td>
<td>Sum</td>
<td>VM</td>
<td>CPU</td>
<td>Sum</td>
</tr>
<tr>
<td>VM₁</td>
<td>35</td>
<td>35</td>
<td>2</td>
<td>2</td>
<td>VM₂</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>VM₃</td>
<td>15</td>
<td>50</td>
<td>2</td>
<td>4</td>
<td>VM₃</td>
<td>15</td>
<td>55</td>
</tr>
</tbody>
</table>

- On P₁ it leaves a smaller (no) memory rest
- On P₂ it leaves a smaller CPU rest

- We need a metrics over the several dimensions.
- Typical: normalize the dimensions (e.g., everything in percent or as a fraction of 1) and take the sum.
Multi-Dimensional Optimization - Normalized

- Let’s normalize to fractions of the maxima, i.e., 70% for CPU and 4 GB for memory.

<table>
<thead>
<tr>
<th></th>
<th>VM1</th>
<th>VM2</th>
<th>VM3</th>
<th>VM4</th>
<th>VM5</th>
<th>VM6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (%)</td>
<td>0.5</td>
<td>0.57</td>
<td>0.21</td>
<td>0.21</td>
<td>0.36</td>
<td>0.14</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>0.5</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
<td>0.25</td>
<td>0.13</td>
</tr>
</tbody>
</table>

- And restart the best-fit algorithm

<table>
<thead>
<tr>
<th>P1</th>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
<th>Mem</th>
<th>Sum</th>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
<th>Mem</th>
<th>Sum</th>
<th>VM</th>
<th>CPU</th>
<th>Sum</th>
<th>Mem</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>VM1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>VM2</td>
<td>0.57</td>
<td>0.57</td>
<td>0.25</td>
<td>0.25</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>VM3</td>
<td>0.21</td>
<td>0.71</td>
<td>0.5</td>
<td>1</td>
<td>VM3</td>
<td>0.21</td>
<td>0.78</td>
<td>0.5</td>
<td>0.75</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- On P1 the sum of the rests is 0.29 + 0 = 0.29
- On P2 the sum of the rests is 0.22 + 0.25 = 0.47
  - So best-fit with the sum metrics puts VM3 on P1.
  - Just wondering: If we’d know that a certain dimension is our bottleneck, should we better do it with certain weighted sums? But at least if we buy the physical servers new, we can try to select them so that we roughly have the same normalized reserves in each dimension
If you want to test yourself at home

- Can you finish the example of the previous slide?
- Could you do it with 3 dimensions?
- Do you have ideas what First-Fit or Best-Fit “decreasing” would mean? (If not, reference [1])
- Can you extend the formulas of the optimization to the memory dimension?
CPU Time Series

- CPU utilization is not really just one value. E.g., on my laptop (with 2 CPUs):
Peaks and Averages

- If we take **peaks** when planning the placement, we are safe. But we might waste a lot of server capacity if there are only a few peaks, and the average is much lower.

- Depending on the workload type, we might take “**normalized peaks**”, i.e., average over somewhat longer periods. E.g., we may decide it doesn’t matter if for 5 seconds a VM can’t get all the CPU it would like.

  ![Normalized peak for this time period](image)

  – In the laptop picture this got us from a peak of about 63% (when I started a database) to about 48%

- Working with averages over long periods is risky – the application may react more slowly under its maximum load and that may not be acceptable
  – Depends very much on the application. E.g., delays for a web server may be much longer than for airplane control.
Time Series

- If we discover utilization over several weeks, and find patterns, we may match per time interval, using normalized peaks for each time interval.

*Normalized peak for these times, over all measured days*

- By overall normalized peaks, we could not put these two VMs on one server ($\approx 50 + 40 = 90\%$, too much)
- But per 3-hour period, we never reach over 60\% together, so it’s fine
- Why measure more than one day?  
  - Generally better statistics  
  - Weekend patterns  
  - Even better also month-end, quarter-end, year-end, but often one can’t wait so long
**Time Series for Live Migration**

- If the migration is to an environment (e.g., cloud) with **live migration** of VMs, we can sometimes use even fewer servers, by combining differently at different times.

<table>
<thead>
<tr>
<th>Time of day</th>
<th>VM₁</th>
<th>VM₂</th>
<th>VM₃</th>
<th>VM₄</th>
<th>P₁</th>
<th>P₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-4</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>60</td>
<td>VM₁, VM₄</td>
<td>VM₂, VM₃</td>
</tr>
<tr>
<td>4-8</td>
<td>40</td>
<td>5</td>
<td>10</td>
<td>60</td>
<td>VM₁, VM₂, VM₃</td>
<td>VM₄</td>
</tr>
<tr>
<td>8-12</td>
<td>50</td>
<td>40</td>
<td>5</td>
<td>10</td>
<td>VM₁, VM₃, VM₄</td>
<td>VM₂</td>
</tr>
<tr>
<td>12-16</td>
<td>60</td>
<td>20</td>
<td>40</td>
<td>5</td>
<td>VM₁, VM₄</td>
<td>VM₂, VM₃</td>
</tr>
<tr>
<td>16-20</td>
<td>20</td>
<td>60</td>
<td>50</td>
<td>10</td>
<td>VM₁, VM₃</td>
<td>VM₂, VM₄</td>
</tr>
<tr>
<td>20-0</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>50</td>
<td>VM₁, VM₂, VM₃</td>
<td>VM₄</td>
</tr>
</tbody>
</table>

**Could they be together all day?**

<table>
<thead>
<tr>
<th></th>
<th>VM₂</th>
<th>VM₃</th>
<th>VM₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM₁</td>
<td>Not 8-12</td>
<td>Not 12-16</td>
<td>Not 4-8</td>
</tr>
<tr>
<td>VM₂</td>
<td></td>
<td>Not 16-20</td>
<td>Not 20-0</td>
</tr>
<tr>
<td>VM₃</td>
<td></td>
<td></td>
<td>Not 0-4</td>
</tr>
</tbody>
</table>

**Here we have a separate optimization problem for each time period.**

- On the previous slide, each time period was like an additional dimension in 1 overall optimization.

**So how many servers would we need without live migration?**
Before the Placement: Server Performance Factors

- Often one migrates from one server type to another (a newer server of the same series, or a very different server).
  - CPU utilization will change a lot, e.g., if the new server is much faster
  - Disk and memory needs stay about the same (assuming we don’t change the software and OS)
  - IO bandwidth may also change (e.g., due to different caches, or different communication protocols)

- Very roughly, one can compare CPU utilization by CPU speed

- But different processor and server architectures make a big difference. So there are benchmark tables (in particular from the IDEAS company) to make better estimates of performance on new hardware. This is called “relative performance models”.

MAPPING TO CLOUD IMAGES
Which Images Do We Map To in the Cloud?

Amazon Machine Images (AMIs)

An Amazon Machine Image (AMI) is a special type of pre-configured operating system and virtual application software which is used to create a virtual machine within the Amazon Elastic Compute Cloud (EC2). It serves as the basic unit of deployment for services delivered using EC2.

Read the Amazon EC2 Developer Guide for information on safely using shared AMIs.

Amazon Linux AMI

A supported and maintained Linux image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2).
Which Images Do We Map To in the Cloud?

- It is very difficult to select an image
  - Large number of choices (IBM RC2 roughly 8k images, Amazon EC2 almost 1200 public AMIs, VMware over 900 “solutions”)
  - Very little information about what is contained
    - Package versions?
    - Filesystems, users, groups?
    - Middleware?

- How to define “the best” image?
  - Technical match?
  - Cost to migrate?
  - Cost to run?
  - Performance?
Which Images Do We Map To in the Cloud?

- Example of the description within a VMware “solution”
  - What is missing?
  - Why is the missing information important?

Example of the description within a VMware “solution”

- What is missing?
- Why is the missing information important?

Instantiated on VMware Player and all contents of the virtual machine are zipped in one file. It needs to be extracted to the Virtual Machines directory and opened by using VMware Player.

Installed with default applications of CentOS.

- CPU's: 1
- Memory: 1G
- Disk: 80G

Authentication Credentials

- Username: user
- Password: user1234
- Root Password: user1234
Cloud Image Selection Simple Case

Get list of images with an array of MW installed, MW is (Name, Version) pair and array is sorted alphanumerically by Name

Output: cumulative cost

Get Target's array of MW

There is Next Image

Yes

Get an Image from the list

No

Image has next MW

Yes

All MW in the image have to be uninstalled. Add transition Costs for the uninstall of these MWs to current cumulative cost.

No

Target has MW

Yes

Image has MW

No

All MW in the Target have to be installed. Add transition Costs for the install of these MWs to current cumulative cost.
Cloud Image Selection Example

- Migration requires a server with:
  - An operating system which
    - Is 64bit
    - Has a commercial support contract
    - Is Unix-like (AIX > 6, RHEL > 5.5, SLES > 10)
  - A web server for simple static HTML pages
  - IBM DB2 Data Server 9.5 or above

Which image do we choose?

<table>
<thead>
<tr>
<th>Image 1</th>
<th>Windows Server 2008 R2 64bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Microsoft Internet Information Services</td>
</tr>
<tr>
<td></td>
<td>IBM DB2 Data Server 9.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image 2</th>
<th>Centos Linux 6.1 64bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IBM DB2 Data Server 9.7</td>
</tr>
<tr>
<td></td>
<td>Apache HTTP Server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image 3</th>
<th>Red Hat Enterprise Linux 5.6 64bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IBM DB2 Data Server 9.1</td>
</tr>
<tr>
<td></td>
<td>Apache HTTP Server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image 4</th>
<th>Red Hat Enterprise Linux 6.1 64bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IBM DB2 Data Client 9.7</td>
</tr>
</tbody>
</table>
Cloud Image Selection Example

- Get into your project groups
- Work through an algorithm to find the lowest-cost image
- Requirements
  - 64 bit
  - Has a commercial support contract
  - AIX > 6, RHEL > 6, SLES > 10
  - A web server
  - IBM DB2 Data Server >= 9.5

<table>
<thead>
<tr>
<th></th>
<th>OS</th>
<th>Middleware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Remove</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Upgrade</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

*Note, for this scenario assume that an OS upgrade does not force middleware upgrades / installs.*

Image 1
- Windows Server 2008 R2 64bit
- Microsoft Internet Information Services
- IBM DB2 Data Server 9.5

Image 2
- Centos Linux 6.1 64bit (Open Source)
- IBM DB2 Data Server 9.7
- Apache HTTP Server

Image 3
- Red Hat Enterprise Linux 5.6 64bit
- IBM DB2 Data Server 9.1
- Apache HTTP Server

Image 4
- Red Hat Enterprise Linux 6.1 64bit
- IBM DB2 Data Client 9.7
Cloud Image Selection

- Build a catalog of configuration details for each Cloud
  - Initialize the cloud:
    - Instantiate cloud images and run discovery tools.
    - Store discovered information for a future use
  - Before migrating a system on the cloud validate this cloud’s data:
    - Images do not change information (unique ID)
    - New images get created or existing once get removed
- Take target requirements from a user, examine the catalog, and recommend the best machine images in a cloud to meet those requirements at the lowest “cost”.
  - When up to date information about cloud’s images exists, how do we find the best one?
  - What are my/customer’s policies?
  - What are my/customer requirements?
Reading List

- Background terminology: The first 1.5 pages each of


On the overall reading list but none of it occurred in this lecture:

Homework #1 – Deploy the Apache Geronimo PetStore demo

- You are a migration engineer and have been asked to migrate a customer’s J2EE application to the cloud.
  - This will be a workload migration so you need to build a new server in the cloud and install all of the software required to run their application.
  - According to the customer, the last known location of the application can be found here: http://www.ibm.com/developerworks/library/os-ag-petstore/

- Your assignment is to get this application running locally on your laptop or in a virtual machine (VM).
  - We suggest that you use VirtualBox and Vagrant as your target VM so that you don’t have to install the application natively on your laptop
  - To prepare an environment to get the application running, you will need to download and install various middleware like Java, a J2EE Container, Derby, etc. (be careful to get the correct versions)
  - Not all of the download links will work. This is where you need to be a resourceful software engineer to find the tools that you need. ;-)

- A future assignment will have you deploy this into a Platform As A Service cloud but for now the lesson is to learn how difficult it can be to manually recreate the environment to deploy existing applications, even for standards like J2EE.

- Good Luck! (Feel free to email the professors with questions. Even the best migration engineers need someone to bounce ideas off of. This should be a fun search, so don’t get frustrated. This is what “real world” application migrations are like… always lots of surprises!)
Just Enough Vagrant to be Dangerous

- Download and install VirtualBox from virtualbox.org
- Download and install Vagrant from vagrantup.com
- Open a Terminal Windows / Command Prompt and type:
  ```
  mkdir petshop
  cd petshop
  vagrant init precise32
  vagrant up
  vagrant ssh
  ```
  - This creates a project directory
  - This changes into that directory
  - This creates the Vagrantfile using Ubuntu 12 32-bit
  - This creates the Virtual Machine using VirtualBox
  - This ssh’s into the VM you just created

- You should now be in an Ubuntu 12.04 LTS (Precise 32-bit) virtual machine
- To get to your source project folder:
  ```
  cd /vagrant
  ```
  - This is the petshop folder on your laptop mapped inside the VM (how cool is that?)

- To exit the vm:
  ```
  exit
  ```
  - This gets you out of the VM and back to your laptop

- To shutdown the vm (after exit):
  ```
  vagrant halt
  ```
  - This powers off the VM when you are not using it

- To start the vm back up later and get into it:
  ```
  cd petshop
  vagrant up
  vagrant ssh
  ```
  - This changes into your petshop folder
  - This powers on the VM again
  - This ssh’s into the VM so you can continue working