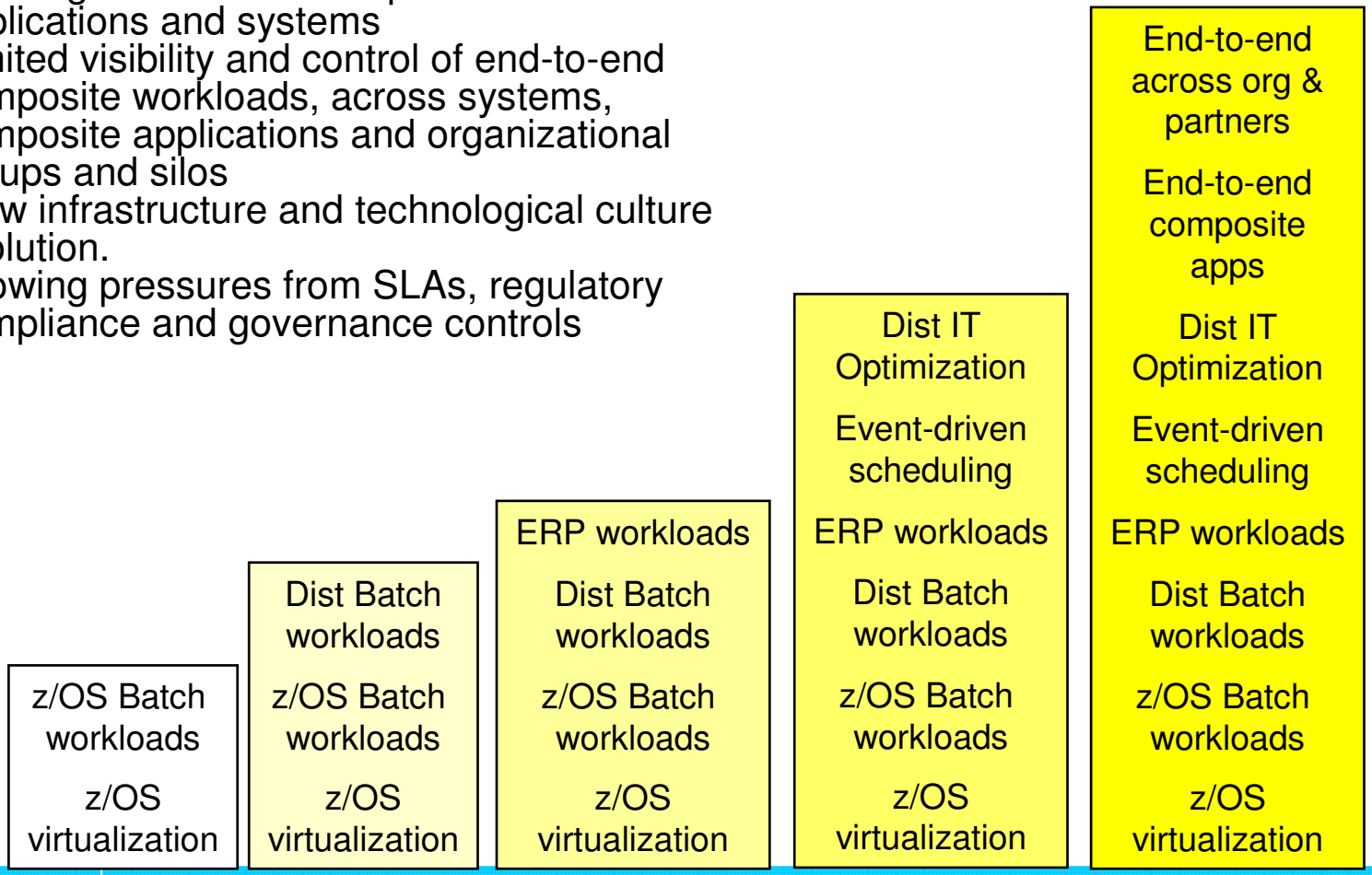
Workload Automation

# Scenarios on Dynamic Workload Execution Environments

Paolo Deidda, IBM Italia S.p.A.
Tivoli Architect, Rome Lab
Internet: paolo_deidda@it.ibm.com
Phone: 39-06-5966.2301, Mobile: 39-335-7454677

# Challenges in managing enterprise workloads are mounting

- Increased demand for integrated, optimized workload execution.
- Growing workload interdependencies across applications and systems
- Limited visibility and control of end-to-end composite workloads, across systems, composite applications and organizational groups and silos
- Slow infrastructure and technological culture evolution.
- Growing pressures from SLAs, regulatory compliance and governance controls



# Evolving Job Scheduling Requirements

Tivoli Dynamic Workload Broker main focus areas

- Automa based p – workl
- Orchestration of work flows
- Centralized management
- Optimize use of existing resources
- Provide increased availability, reduce errors
- Shorten batch windows

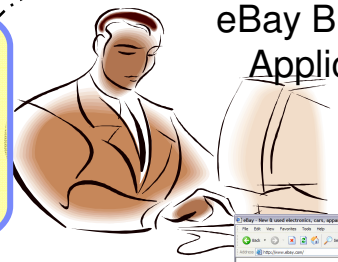
- Platform and application agnostic scheduling
- Single consolidated view of all scheduling points
- Event based scheduling
- Use of virtualization and grid computing technology
  - Manage prioritization and resource selection for jobs
  - Manage execution of parallel job steps across distributed resources
  - Identify and allocate resources to meet quality of service goals

- Process driven
- Dynamic topology
- Service oriented architecture integration
- Adapters based on standards - web services

# An SLA Driven Business Scenario: Configuration Change

Jobs are to be scheduled on the available resources through the scheduler system

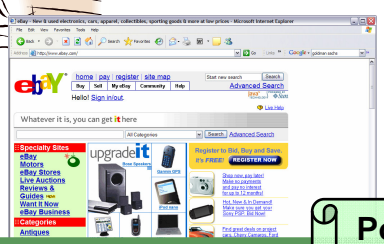
eBay Business Application



**SLA**
- Response Time < 10s
- Availability 99.99%
- All items prices updated by 8 AM

User

Browse the Web Application



**Policy:** Job3 requires Appl A

Tivoli Dynamic Workload Broker

Job1 → Job2 → Job3

**Static Way**
- Target resources for the jobs must be set in advance
- Topology must be defined manually

**Dynamic Way**
- Resources automatically discovered.
- Targets are assigned dynamically depending on the resource requirements

Computer Systems, Applications and Middleware Pool

Configuration Change

# An SLA Driven Business Scenario: Failover

If a resource goes down or there is a network failure or a immediate maintenance the SLA defined availability must continue to be matched

eBay Business Application

**SLA**
- Response Time < 10s
- Availability 99.99%
- All items prices updated by 8 AM

Browse the Web Application

User

**Policy:** Use backup server

Tivoli Dynamic Workload Broker

Job1 → Job2 → Job3

**Static Way**
- Running or planned jobs fail. Administrator must:
  - Find available and appropriate resources
  - Re-configure all jobs
  - Re-submit them

**Dynamic Way**
- Adapt the job targets depending on their availability.

Server Failure

Computer Systems, Applications and Middleware Pool

# An SLA Driven Business Scenario: Peak Times

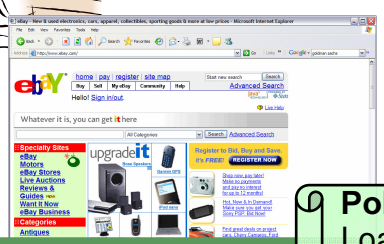
The Response Time SLA target must continue to be met also during peak days such as Christmas week, Thanks Giving, etc.

eBay Business Application

**SLA**
- Response Time < 10s
- Availability 99.99%
- All items prices updated by 8 AM

User

Browse the Web Application



**Policy:** Loadbalance Jobs on server pool

Tivoli Dynamic Workload Broker

Job1 → Job2 → Job3 → Job4 → Job4 → Job5

**Static way**
- During peak times some resource may get overloaded
- If new resources are provisioned the jobs must be reconfigured in order to leverage new possibilities.

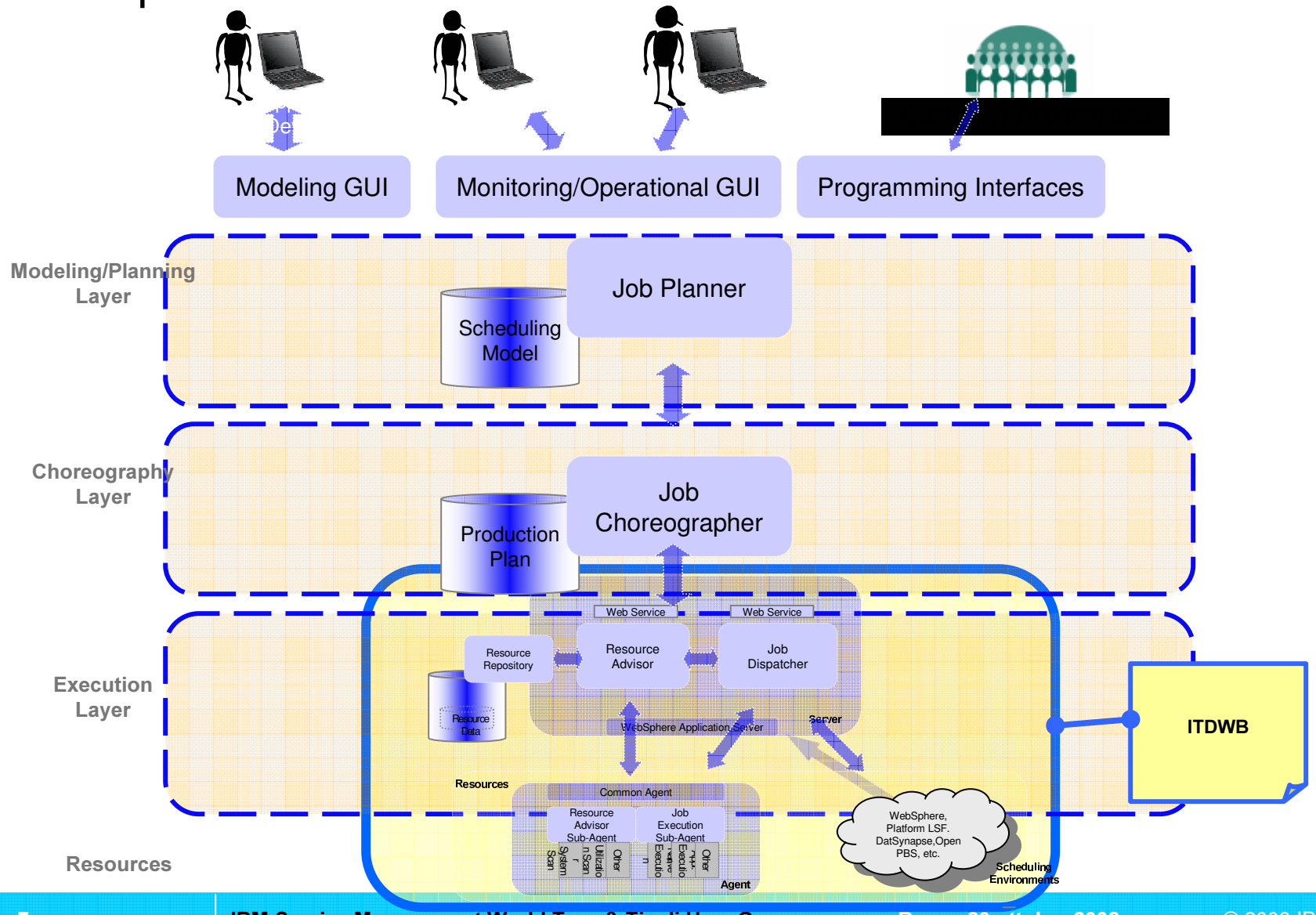
**TDWS**
- Load balancing provides load distribution across resource pool.
- Resource allocation provides load distribution over time
- New resources just provisioned are immediately discovered
- Jobs automatically run on those resources

Resource Overloaded

Compu… Applications and Middleware Pool

Resource Provisioned

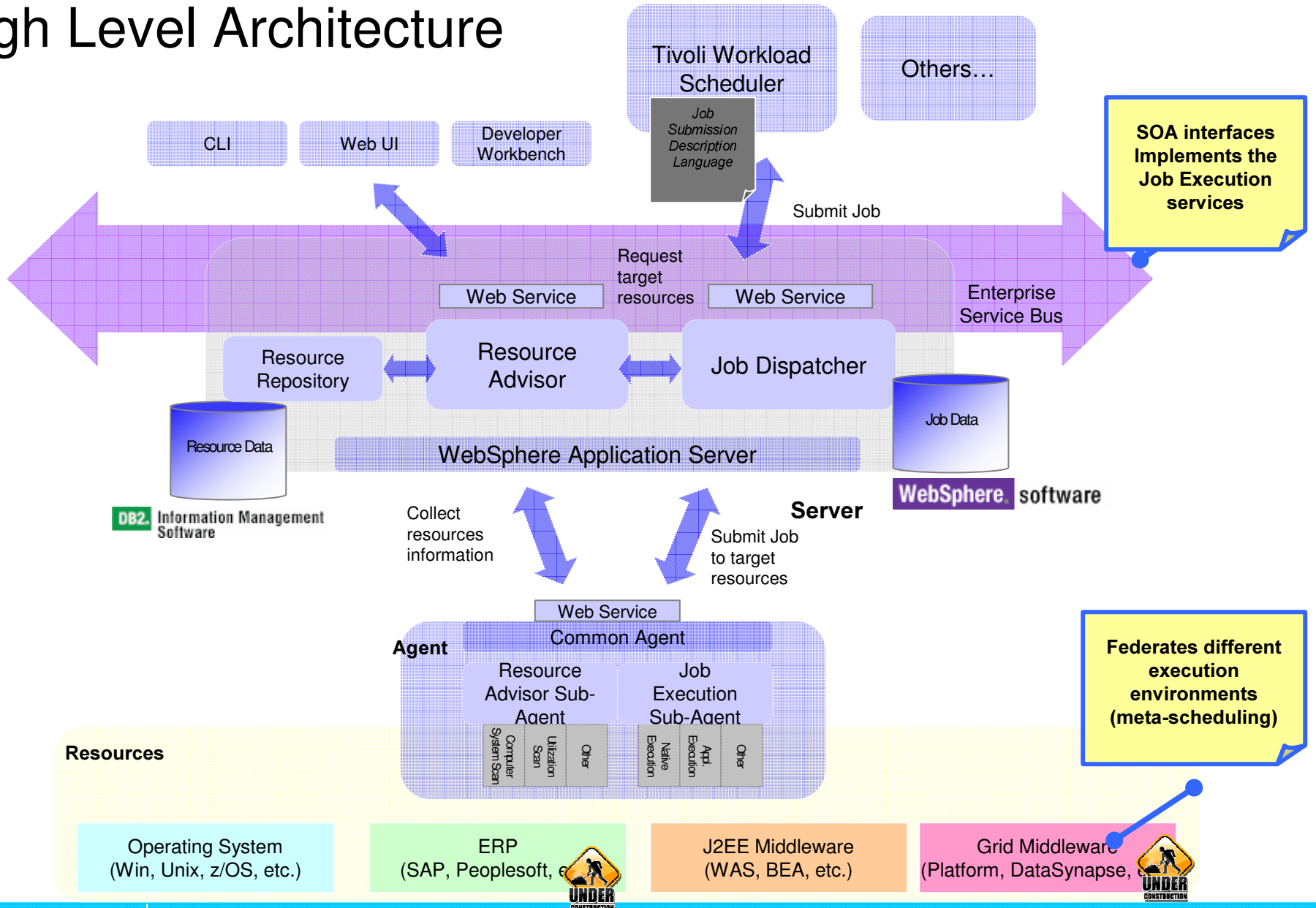
# Positioning: Tivoli Workload Automation Layered Conceptual Architecture



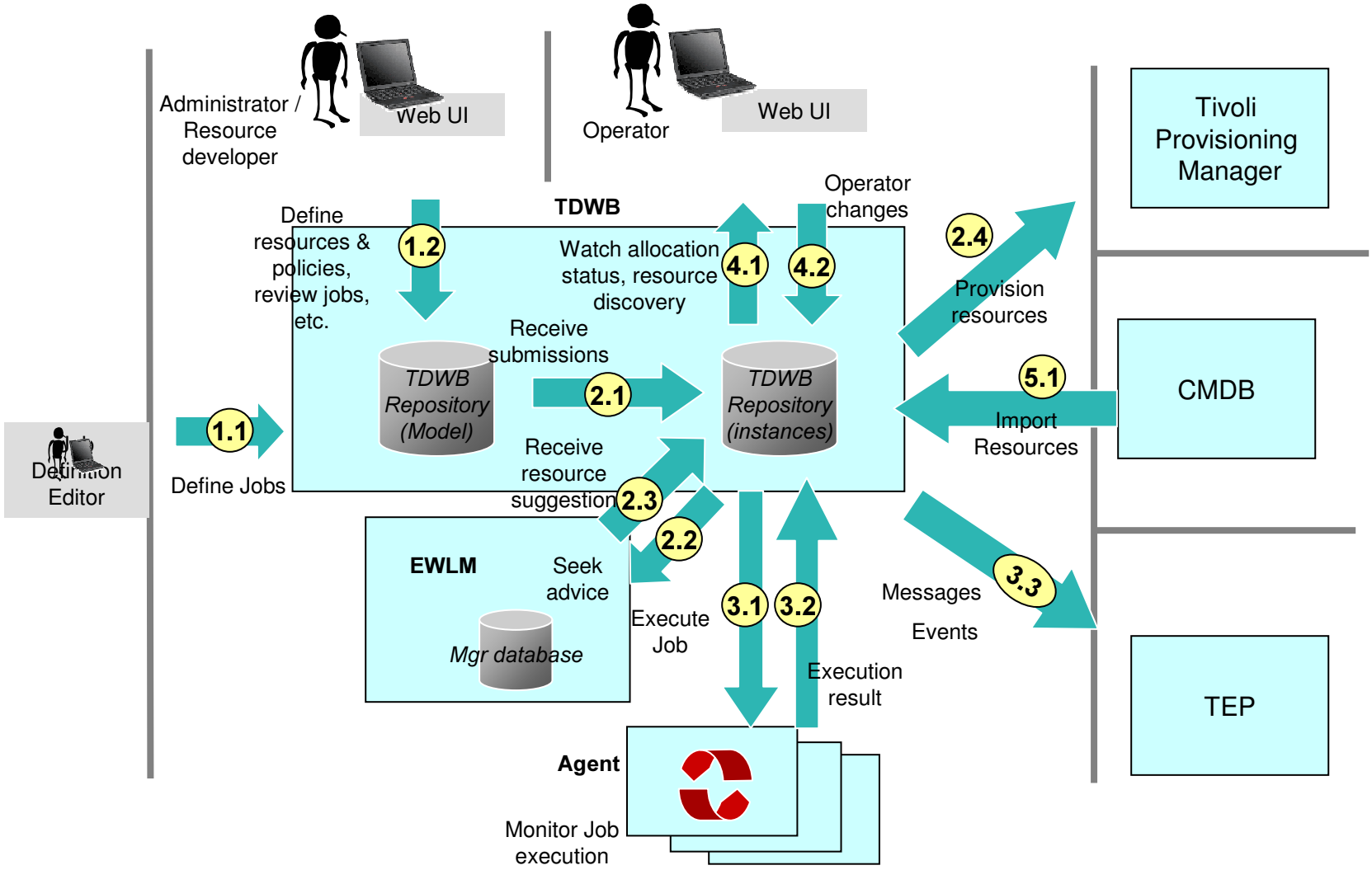
# IBM Tivoli Dynamic Workload Broker (Highlights)

- Synergy with TWS or Stand-alone
  - Add-on product on top of TWS 8.3 or 8.2.1 (either distributed or z/OS)
  - Stand-alone product to enable dynamic scheduling on SOA
- Resource-aware Scheduling
  - Automated discovery and updates of scheduling environment resources (Computers, Operating Systems, File Systems, Networks)
  - Support for Logical Resources to map special features of workstations and grouping
  - Matching of Job resource requirements with best available resources
  - Control of concurrent utilization of consumable resources
  - Assignment of resources with the objective to optimize the resource utilization
  - Support for variable substitution during job submission
- Web Console for administering the infrastructure and agent management
- Graphical editor (Job Brokering Definition Console, JDBC) to define jobs in JSDL
  - A job can also embed a script or windows batch
  - JSDL to become a standard
- Different types of jobs
  - Executables, Scripts (embeddable in JSDL), J2EE (EJB, JMS)
- Platforms
  - Linux (on System-x, System-p and System-z), AIX, Windows
  - HP and Solaris coming in 2007

# High Level Architecture



# Tivoli Dynamic Workload Broker - Scheduling Lifecycle



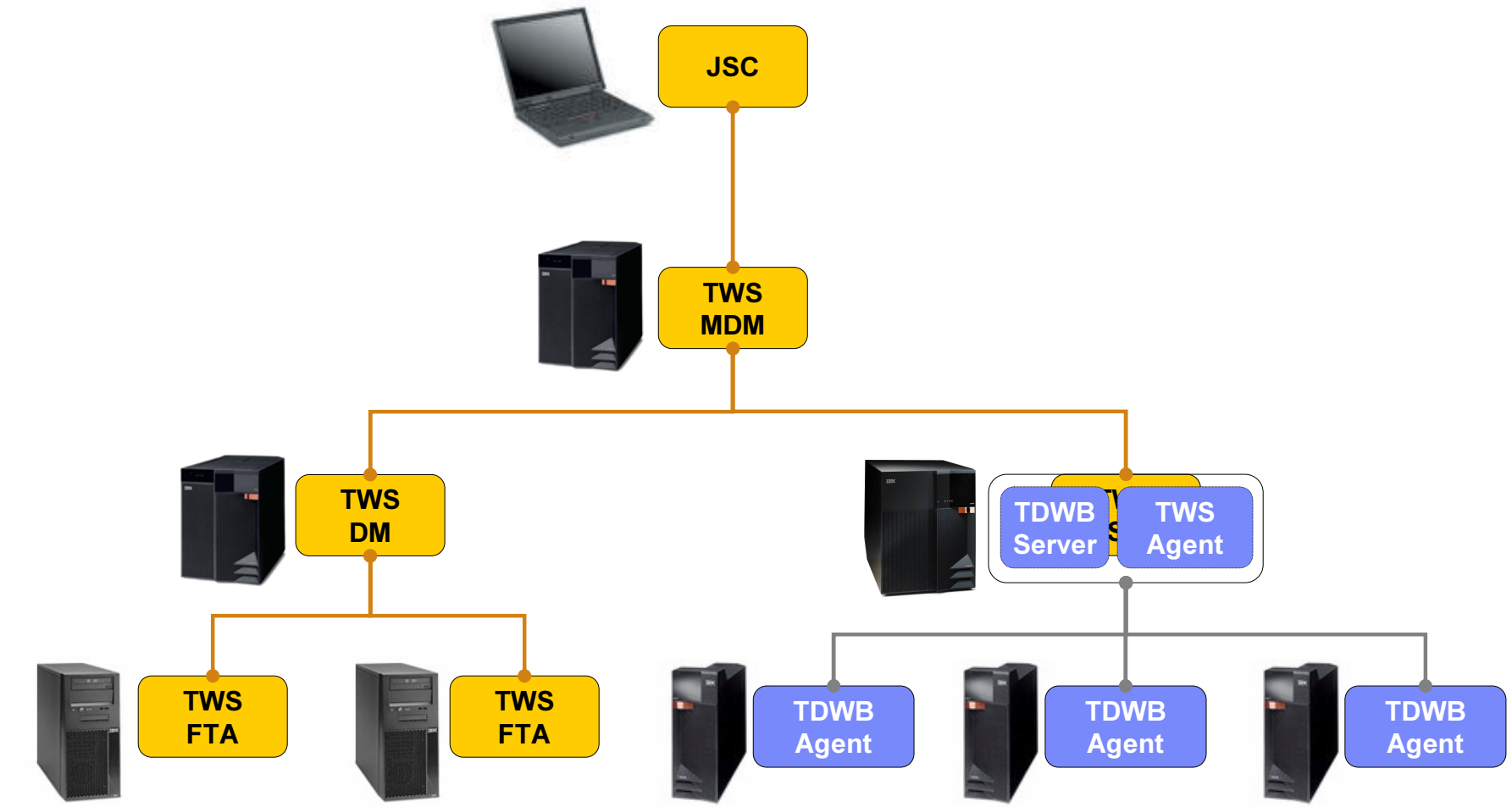
# Tivoli Dynamic Workload Broker – Job Definition

- **The** Job Submission Description Language **(JSDL) is a language for describing the job requirements for submission to resources.**
- **The JSDL language contains a vocabulary and normative** XML schema **that facilitate the expression of those requirements as a set of XML elements.**
- **To create and edit JSDL files, you can use the** Job Brokering Definition Console **(JDBC).**
- **The JSDL files are saved in the** Job Repository **as job definitions and become available for submission. JSDL files adhere to the XML syntax and semantics as defined in the JSDL schema**

```
<jobDefinition>
  <annotation> .. .</annotation>?
  <category> .. .</category>*
  <variables> … </variables>?
  <application> ... </application>
  <resources> … </resources>?
  <relatedResources> ...
  </relatedResources>*
  <optimization> ... </optimization>?
  <scheduling> ... </scheduling>?
</jobDefinition>
```

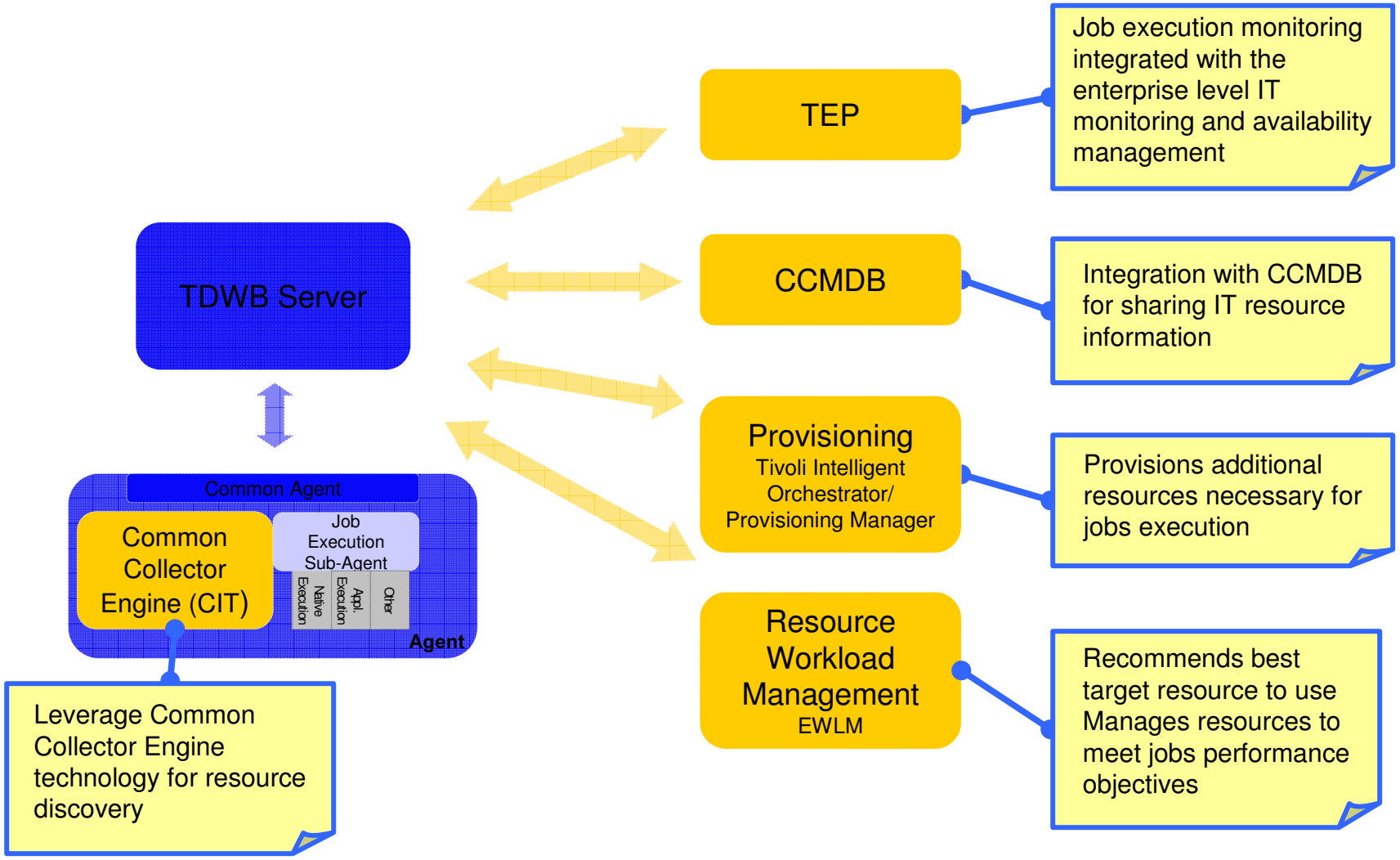

 

# TDWB and TWS together – Topology Example



# Tivoli Dynamic Workload Broker – Integrations

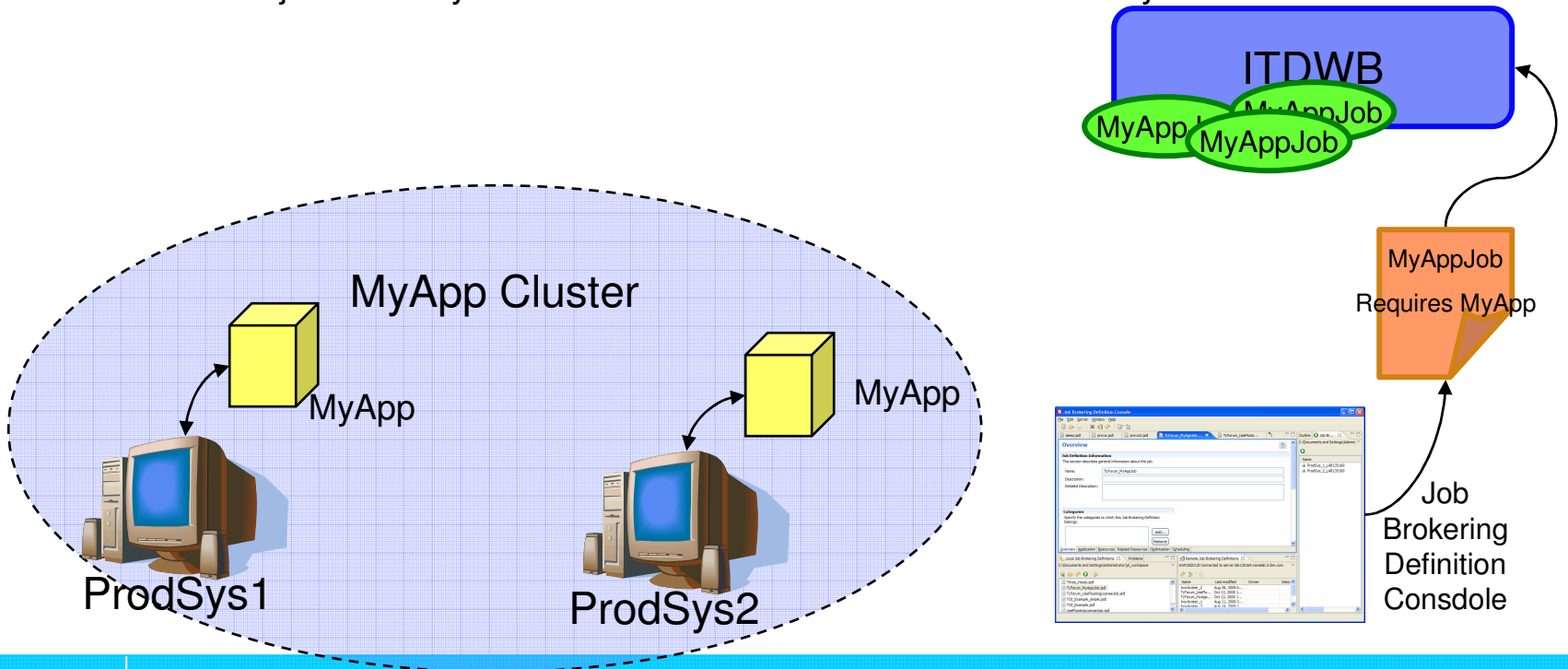


# Demo Scenario – Application High Availability

In this scenario an application (MyApp) is installed in two systems managed by a Cluster Manager. Only one MyApp instance is running at a time. Jobs need to go where that application is running. MyApp instance are modeled as TDWB logical resources associated to the two systems

- define job requirements using Tivoli Job Brokering Definition Console
- show how to associate logical resources representing a clustered application to different hosts
- show the job routing based on the logical resource availability. Set the Logical resource set alternatively to off-line to show that submitted jobs actually follow the available resource or wait until any resource is available

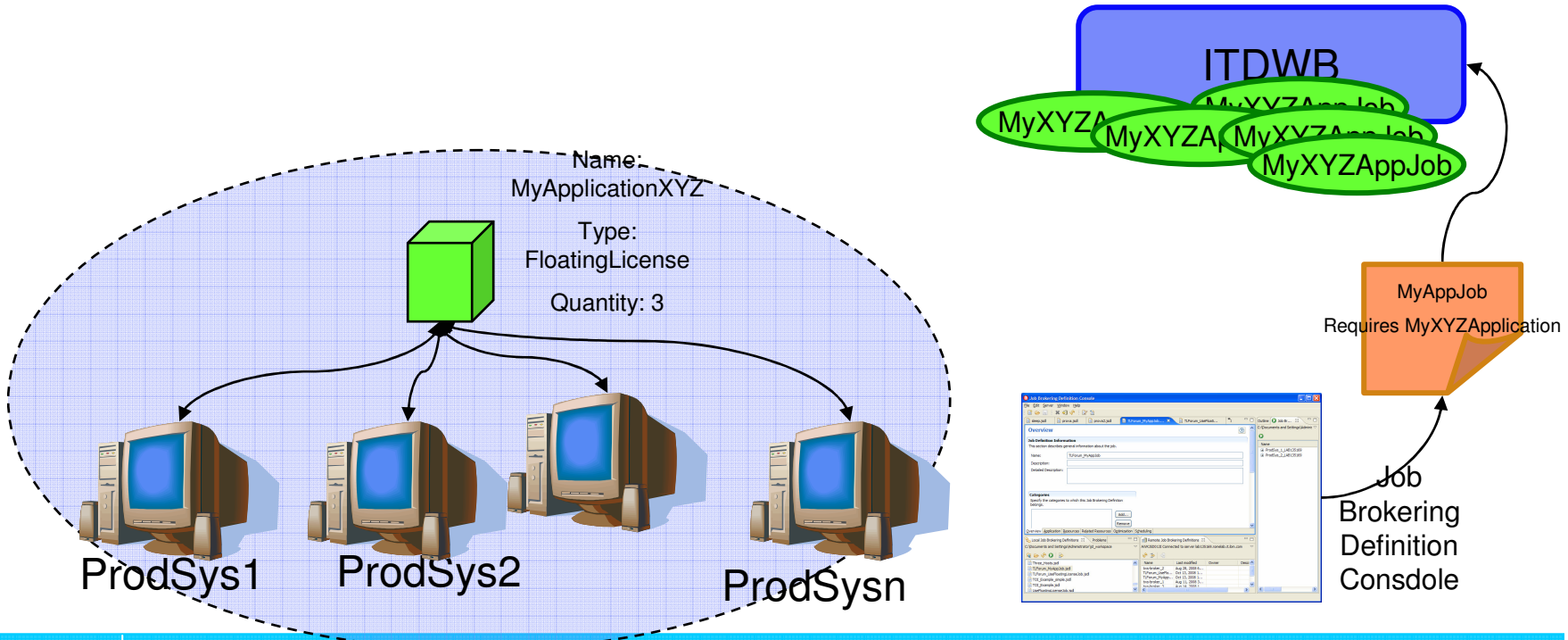


# Demo Scenario– Software License Control

This scenario shows how ITDWB handles jobs that require and consume resources. That could be the case of software licenses, that available in a given number, may be consumed by running jobs. Whenever jobs terminate they release the quantity they used so that other waiting jobs can be submitted.

- how to define job allocation using Tivoli Job Brokering Definition Console
- in particular how to share logical resources with different hosts
- how jobs get queued until other jobs terminate and release resources they are using

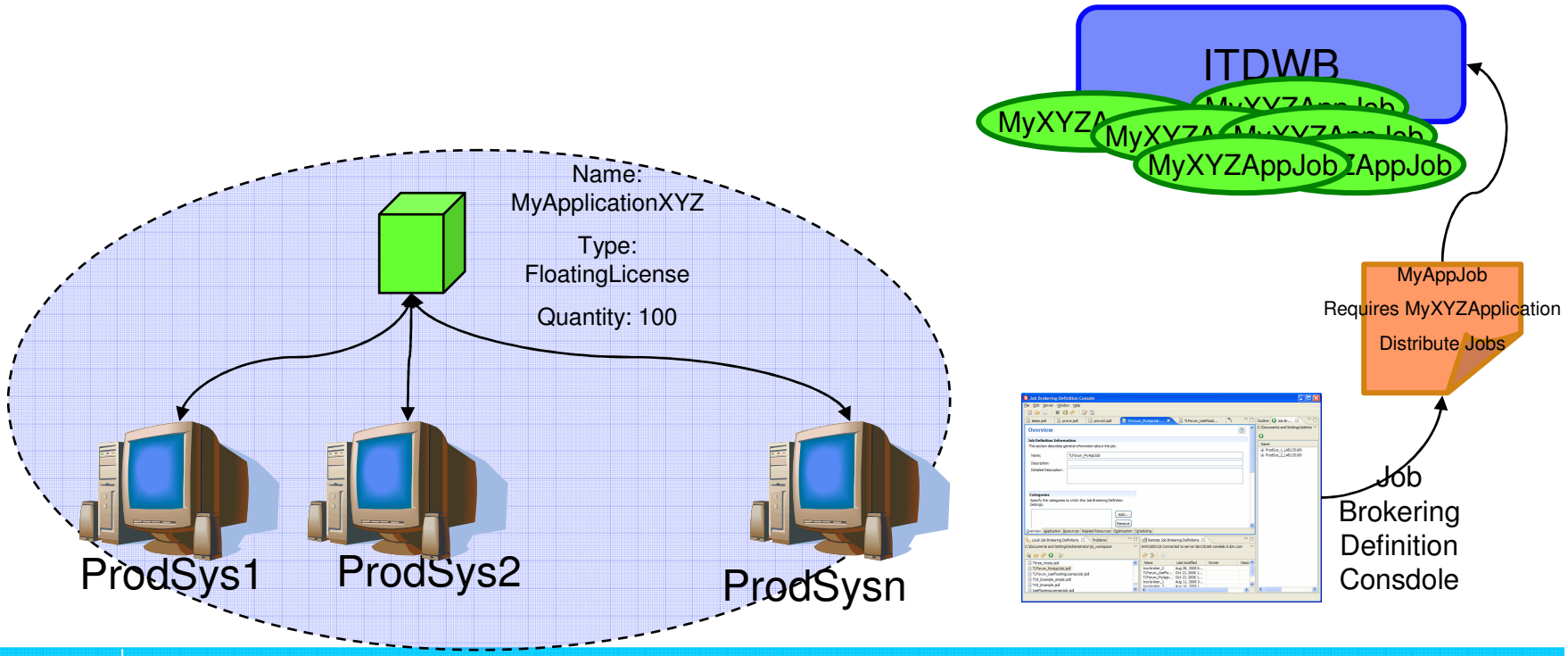


# Demo Scenario – Heavy processing batch application workload balancing

This scenario focuses on how jobs can get distributed accordingly to policies. Typical patterns for this scenarios are batch applications spawning many homogeneous parallel heavy processing jobs on a pool of homogeneous machines, for example to create reports, calculate interests, update prices, update lists, etc.

- how to define job optimization policy using Tivoli Job Brokering Definition Console
- how the load balancing based on different policies (driven by resource consumption or just round robin)



# Demo Scenario – Choreography of dynamic jobs

This scenario shows how ITDWB dynamic jobs can be part of a more complex batch application that choreographs many of them in end-to-end context. In this scenario ITWS plays the fundamental role to plan job submissions to ITDWB and choreograph their execution based on their results.

- how dynamic jobs can be used from TWS
- how to create a job stream using TDWB jobs