SUN Center of Excelence, AGH-UST

Configuration Management of Massively Scalable Systems

Marcin Jarząb, Krzysztof Zieliński, Jacek Kosiński
SUN Center of Excelence
Department of Computer Science, AGH-UST

Massively Scalable System - Architecture

***Massively Scalable* systems consists of many replicated nodes interconnected with high speed communication subsystem.**

The *distributed* techniques allow such systems to be scaled up:

- *Vertically* – adding more physical resources such as processors, memory, disks etc.
- *Horizontally* – incremental approach i.e. adding additional commodity hardware as needed

The number of computational elements in Massively Scalable systems can easily reach several hundred of processors!

The Goal

- Present the complexity and functionality of configuration and management of *Massively Scalable* systems
- Present the contemporary technology for the configuration management of *Massively Scalable* computer systems
- Problem has been put in the context of modern hardware and OS resource virtualization techniques

Massively Scalable System - Use Cases

The *Use Cases* for *Massively Scalable* systems are driven by the *non-functional* requirements of many applications deployed in *Data Centers*:

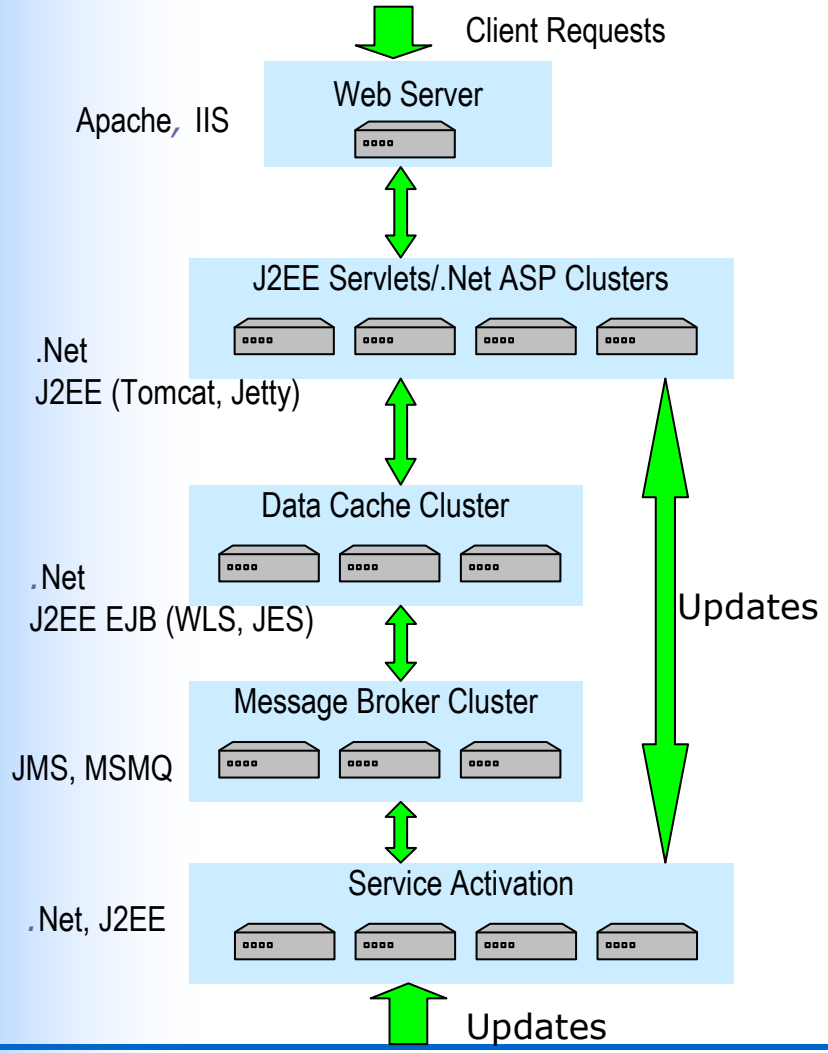
- *Internet Portals* offering access to wide spectrum of applications e.g. *CRM, ERP;*
- Often they follow *Service Oriented Architecture (SOA)* exploiting *Java 2 Enterprise Edition* (J2EE) and *Microsoft .NET* platforms using *Connectors* or *WebServices* for integration with *Legacy* applications.

The activity of such systems requires efficient mechanisms for:

- *On-demand resource allocation and accounting of resource consumption;*
- *Isolation of computations of independent groups of users.*

SUN Center of Excelence, AGH-UST

Massively Scalable System - Deployment Strategies



Uses special purpose plugins for load-balancing. Large volumes of client HTTP requests are routed through the Web server to the Servlets or ASP cluster underneath to fulfil those requests.

Encapsulates all presentation logic required to services the clients access the system, accesses the shared information, constructs the response which is delivered to the Web Server. Used Patterns: ***Transactional Data Caching***, ***Cache routing***, ***Application Server pooling.***

The Cache is replicated over the Application Server instances and kept in sync through replication or lazy propagation techniques. ***Data Partitioning*** *which sets data into segments, based on partitioning rules (****Cache Routing****).*

Infrastructure for managing the propagation of events from the back-end operational systems to the data cluster.

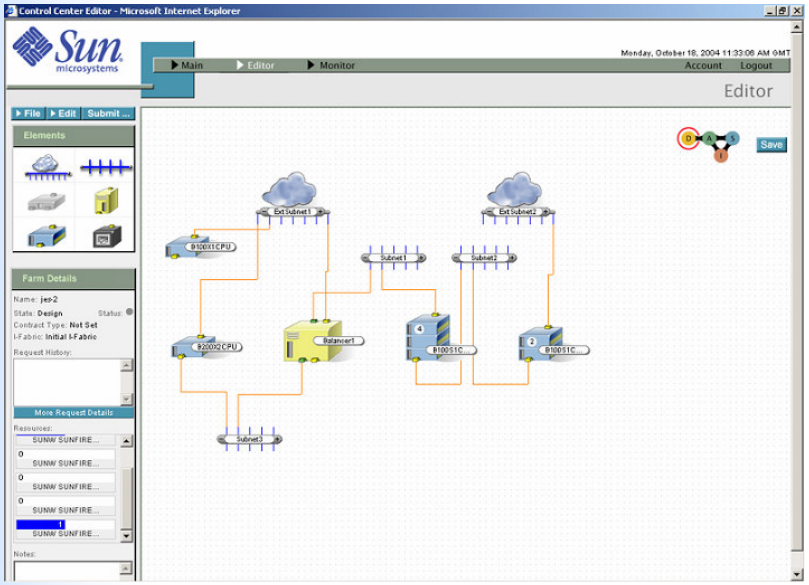
The back-end EIS systems that own the information.

Enabling Technologies, Solutions

- ***Virtualization*** - form of resource management, which manipulates devices, storage address spaces, and data objects
- ***Provisioning*** - addresses the tasks needed to deploy and activate a service which means making hardware and software available on an as-needed basis

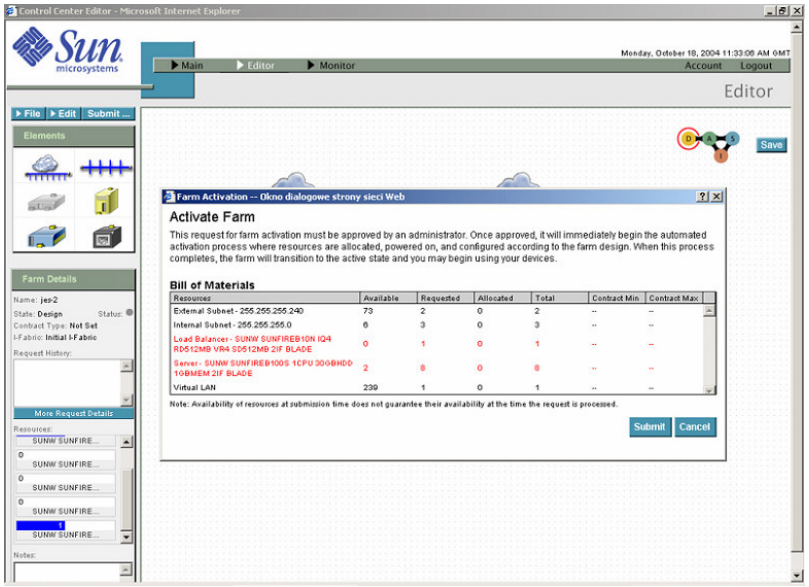
Problem	**Solution**
Virtualization of Hardware Resources	Blades technology with SUN's *N1 Provisioning Server*, *N1 Grid System* or *IBM Tivoli* platforms
Virtualization of Operating System Resources	*Solaris 10 Containers, IBM Logical Partitions, HP Virtual Partitions, Linux (Xen, VLinux)*
Application *Provisioning*, *Deployment* , *Configuration*, *Version Control*, *Logging and Reporting*	*N1 Provisioning System* (Linux, Solaris, Windows, AiX), *Linux LCFG, IBM Tivoli*

N1 Provisioning Server



Each N1 resource is located in an I-Fabric (Infrastructure Fabric) which combines storage, networking and computing resources into a contiguous infrastructure that users can deploy and manage to meet changing requirements.

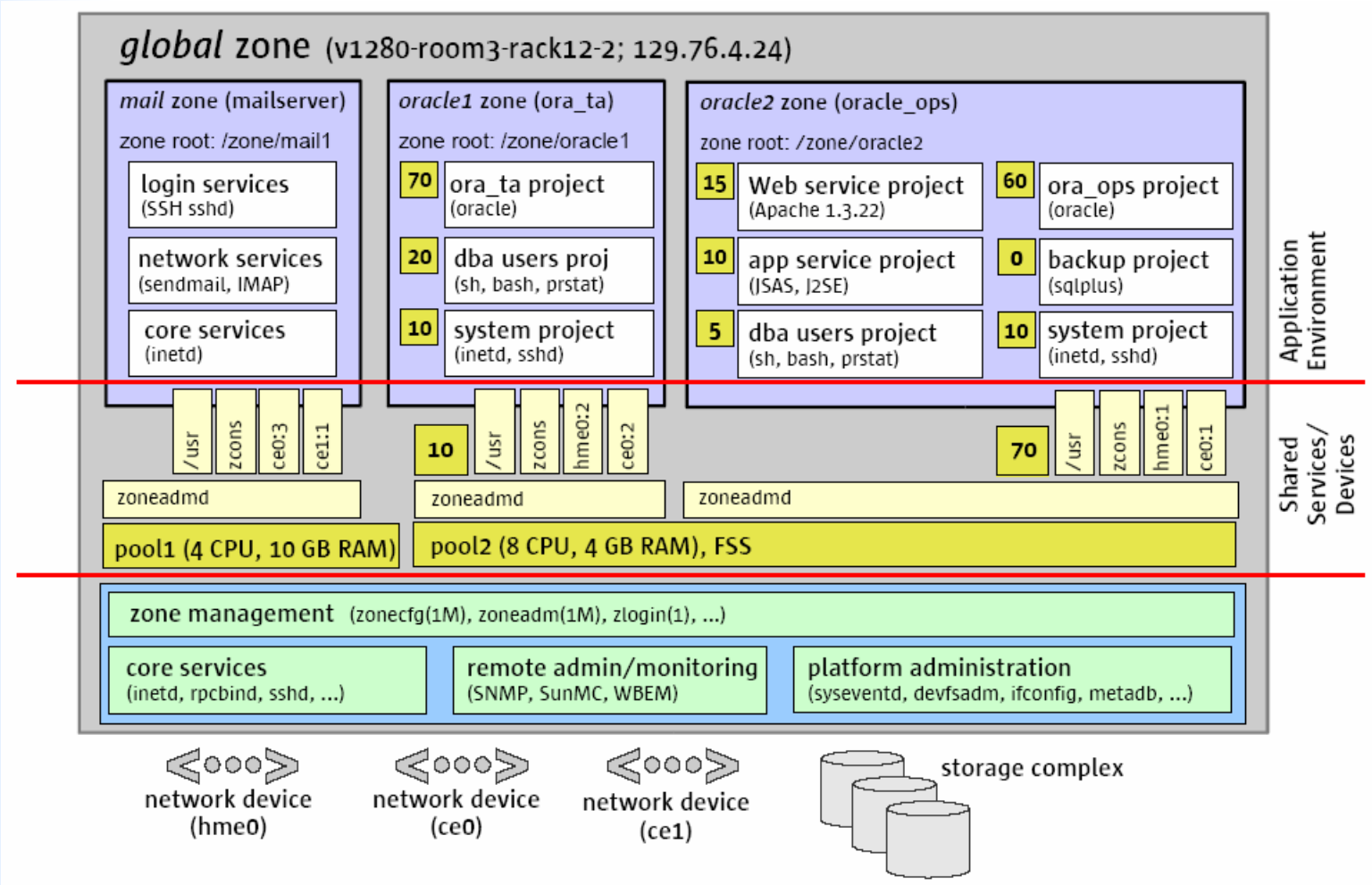
After the new Farm is configured, the next step is the Activation Farm. This step is performed by the system administrator.



Solaris 10 Containers

- Zones formerly N1 Grid containers splits machine into many different operating instances, having own IP addresses, booting capabilities, resources control and accounting, each retaining full security;
  - ***1 OS instance = 1 global and N local Zones***
  - Isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones
  - Processes that are assigned to different zones are only able to communicate through network API
  - **But Global Zone has „direct" access to local *Zones***
- ***Solaris Container*** = ***Zone*** + ***Resource Consumption***
  - *Dynamic Resource Pools;* used for dynamic adjusting machine resources
  - *Resource Controls;* per-Process resource limits extended to the ***Project*** and ***Tasks*** entities
    - ***Project*** provides a network-wide administrative identifier for related work
    - ***Task*** collects a group of processes into a manageable entity that represents a workload component
  - *CPU shares;* portion of the system's CPU resources that is allocated to a ***Project.*** CPU shares are never wasted!

Solaris 10 Containers cont.



Experiment with CPU Shares (1)

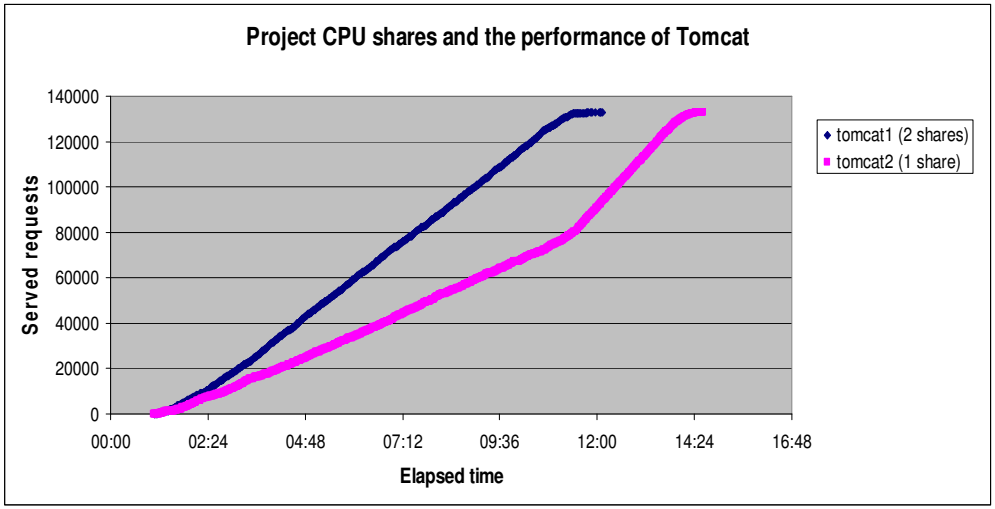
Target: Influence of number of assigned CPU shares to projects on the performance of Tomcat.

Configuration: 2 Tomcat instances started in projects *tomcat1* and *tomcat2*,
Load generated for each instance 75 VU,
Performed in two scenarios:

***(1) tomcat1* (2 CPU shares), *tomcat2* (1 CPU share)**

(2) during test duration switch CPU shares:
tomcat1 (from 2->1 CPU share) , *tomcat2* (from 1->2 CPU shares)

We are analyzing total SUM of requests served by each instance.



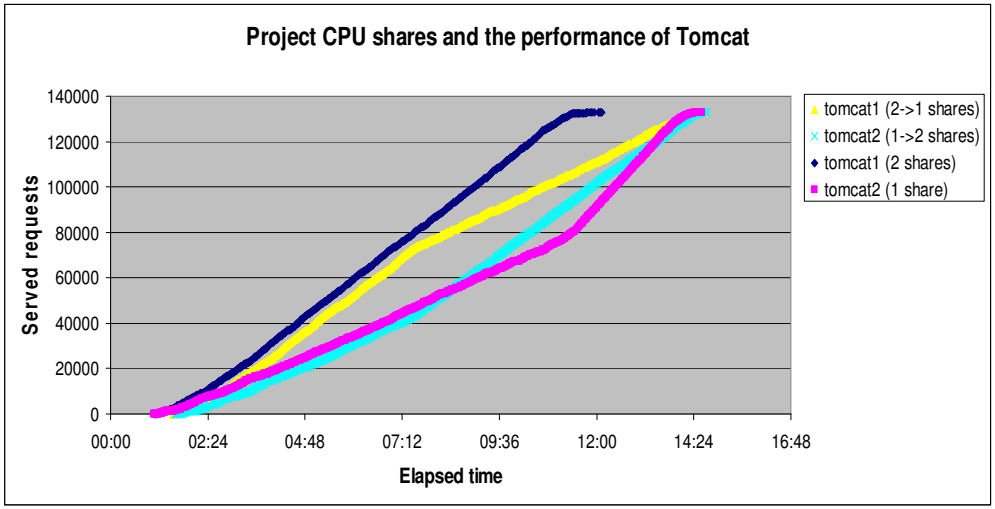
Experiment with CPU Shares (2)

Target: Influence of number of assigned CPU shares to projects on the performance of Tomcat.

Configuration: 2 Tomcat instances started in projects *tomcat1* and *tomcat2*,
Load generated for each instance 75 VU,
Performed in two scenarios:

(1) *tomcat1* (2 CPU shares), *tomcat2* (1 CPU share)
(2) during test duration switch CPU shares:
***tomcat1* (from 2->1 CPU share) , *tomcat2* (from 1->2 CPU shares)**

We are analyzing total SUM of requests served by each instance.

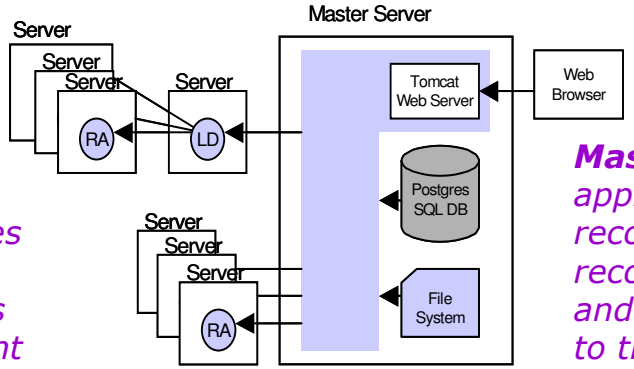


N1 Grid Service Provisioning System

- Platform that automates the deployment, configuration, and analysis of distributed applications;
- Enables organizations to manage applications as distinct units, rather than as large sets of installation files;
- Centralized control over deployments and configuration;
- Provides an object model to define a set of Components and Plans for system configuration, service provisioning, and application deployment automation;
- Management of *virtualized* Solaris OS resources (*Solaris 10 Containers*)

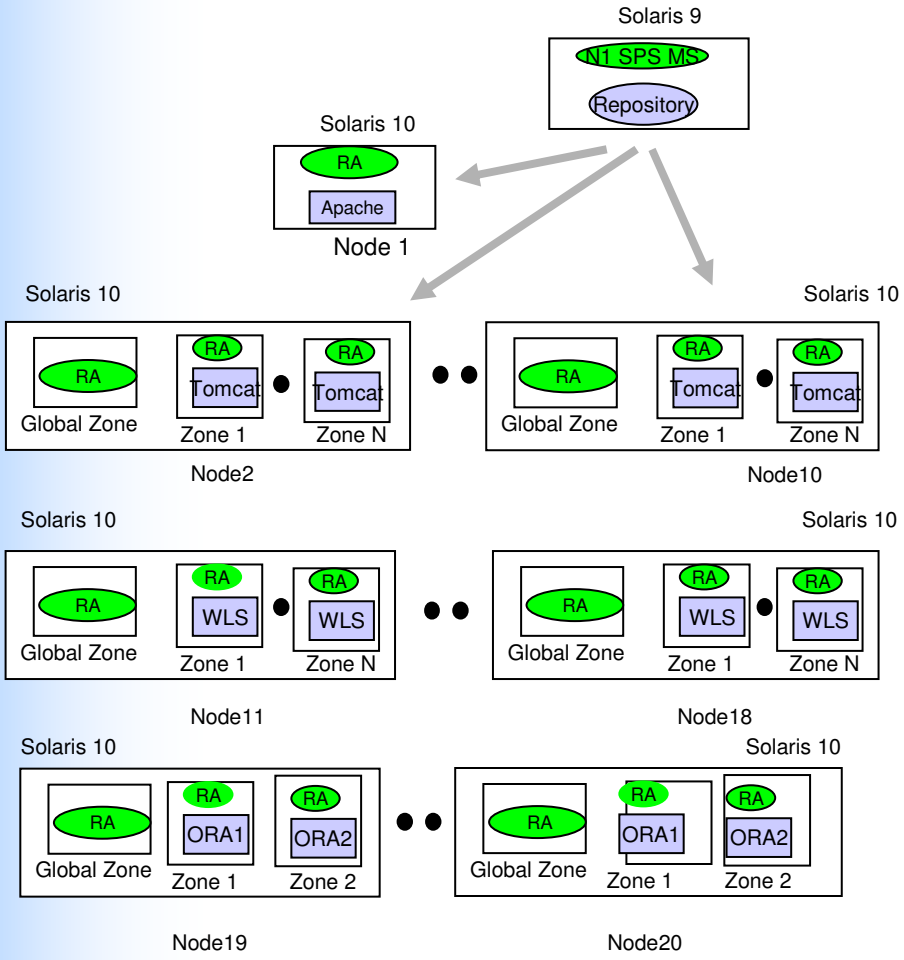
Local Distributor (LD) *is a forwarding gateway for commands to be executed by an RA.*

Remote Agent (RA) *resides on each server managed by an N1 SPS. The RA accesses the local system environment on the server.*



Master Server (MS) *is a central application that maintains user records, server history and state records, modeling definitions, and other global data relating to the application deployment process.*

Case study



1. Deployment of the farm using the *N1 Provisioning Server*, each image has already installed *N1 SPS RA* for remote management of software configuration at each node and *N1 SPS MS* for managing the whole deployment process.
2. Installation of Solaris 10 containers with N1 SPS, RA agent is also installed.
3. Installation of *Apache* with N1 SPS*; load-balancing* plug-in with defined list of Tomcat's clustered instances.
4. Installation of Tomcat clustered infrastructure; copying Tomcat distribution from N1 SPS repository with „variables substitution".
5. Installation of the WLS software on selected zones using silent-installation mode performed by N1 SPS.
6. Oracle clustered instances are installed manually!
7. Applications deployment and core services (JDBC multipools, JNDI) configuration are done via JMX infrastructure provided by the application servers.

Conclusions

- Configuration management of *Massively Scalable* systems involves running of applications over virtualized resources
- The process should integrate flexible control of virtualized resources with mechanisms for setting up applications, parameters configuration and support for easy replication of exiting installations over large numbers of nodes