



Project OpenSolaris™ Dynamic Service Containers featuring Nimsoft SLM

Session: S311526

Robert Holt

Customer Architect, Sun Microsystems Inc.

Jason Carolan

Distinguished Engineer, Sun Microsystems Inc.

Dan Birck

SE Director, Nimsoft, Inc.

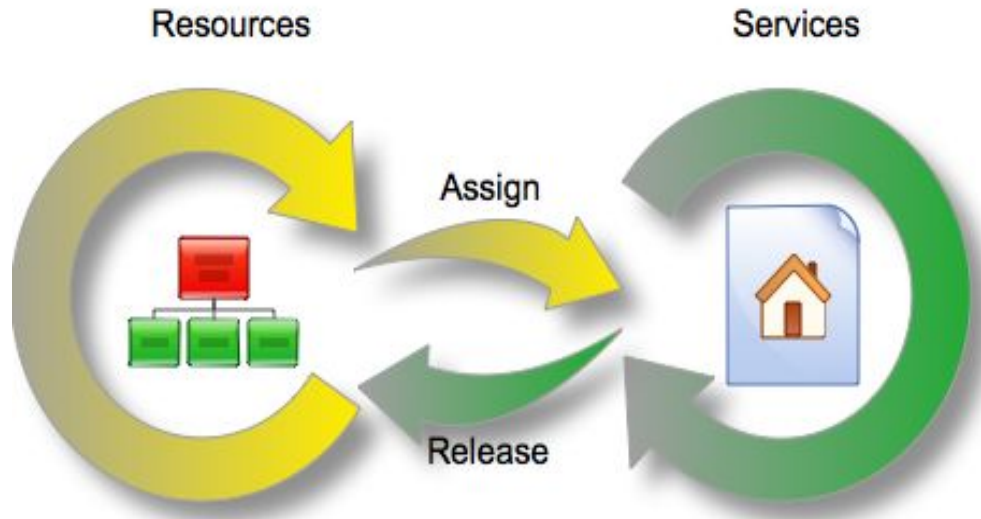
Mark Webster

Account Exec, Nimsoft, Inc.

Agenda

- >What is DSC?
- >Motivation and Background
 - Demo #1
- >DSC Internals
 - Demo #2
- >Monitoring Services and the Cloud
 - Nimsoft Demo #3
- >Futures & Call for Help
- >Q&A

Shifting Dynamics in Systems Management



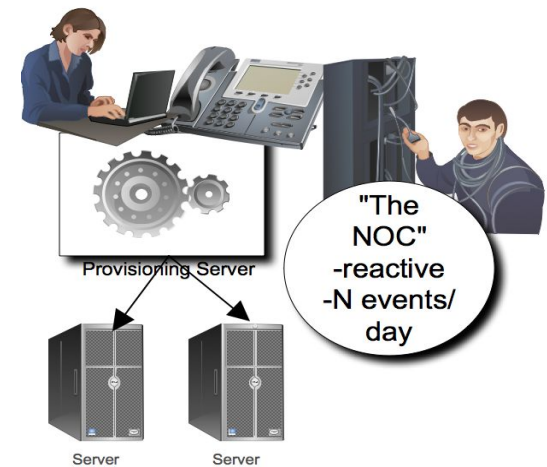
What is Project Dynamic Service Containers?

- An Open Source and an OpenSolaris Project
 - (<http://dsc.kenai.com>)
- Built using OpenSolaris, MySQL, BASH, PHP, etc.
- Yeah so...
 - A set of software to manage scalable application deployment and service level management leveraging virtualized environments
 - “if you can package it, we can manage it”
 - Continuous automated deployment of payloads across nodes in a highly scalable & decentralized model
 - Leverage network content load balancing, service level monitoring, etc. to allow dynamic scaling
 - Deploy 1000s of workloads in minutes across 10s of nodes

Background / Challenge

➤ Conventional grid software and provisioning software such as Sun Grid Engine, BOINC, N1SPS, Tivoli... are

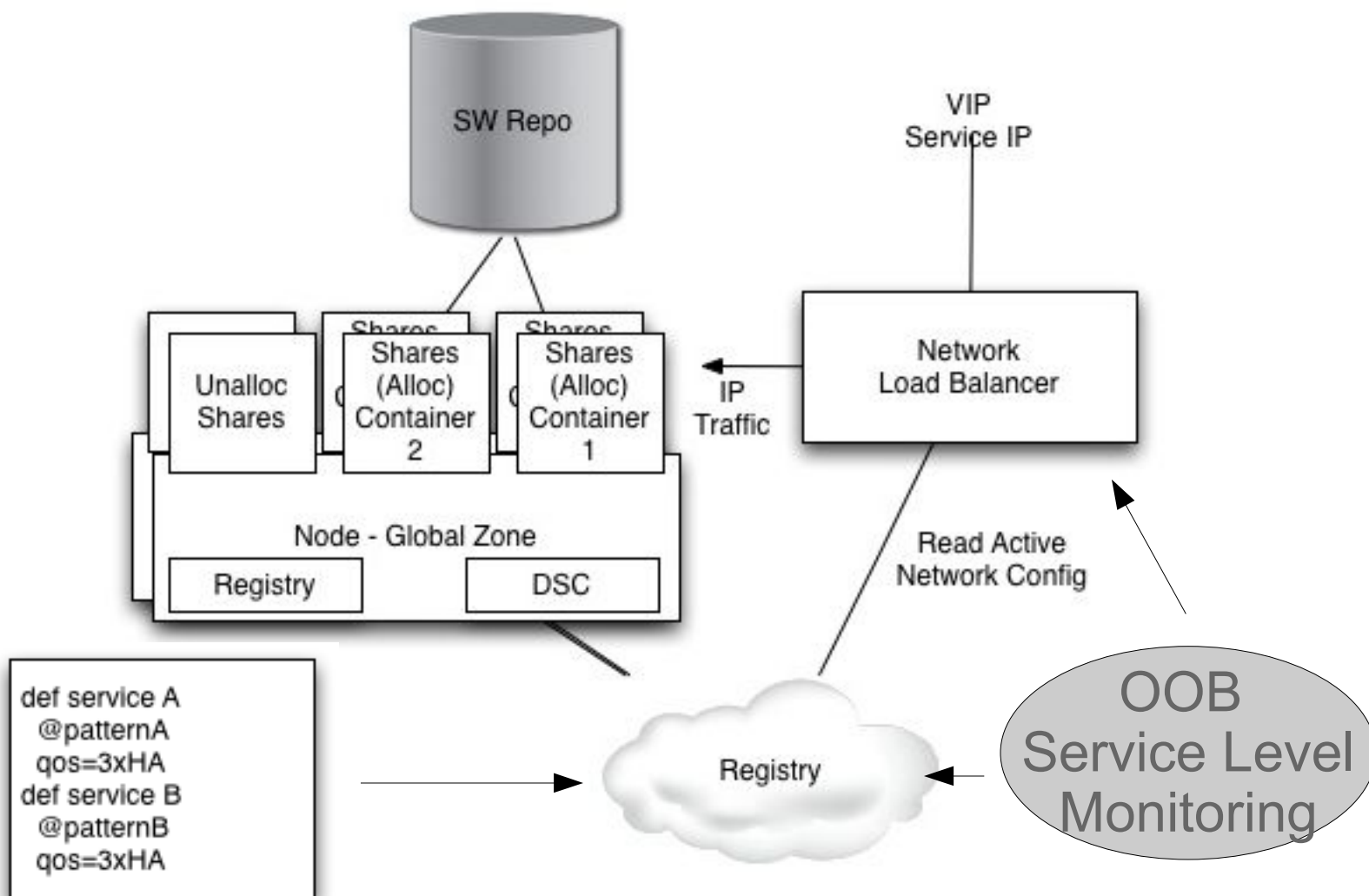
- Heavy weight, expensive, mostly static
- Service levels are an afterthought
- Centralized and complicated -- “PUSH”
- Exploit virtualization as a separate process
- Proprietary models and APIs that limit adoption



➤ How could we exploit off the shelf technologies to demonstrate a highly adaptive “Solaris” Cloud?

- Get the provisioning technologies “out of the way”
- Link the network as a key component for service delivery

Simple Topology



Example Use Cases

➤ Manage Tier 1 & 2 Services in Production

- Simplified deployment, integration with Solaris Immutable Service Containers (ISC)
- Auto-scale services via Zeus feedback, load balancing

➤ Create simple inexpensive “developers” cloud

- Create, destroy development environments quickly
- Copy environments into test, prod

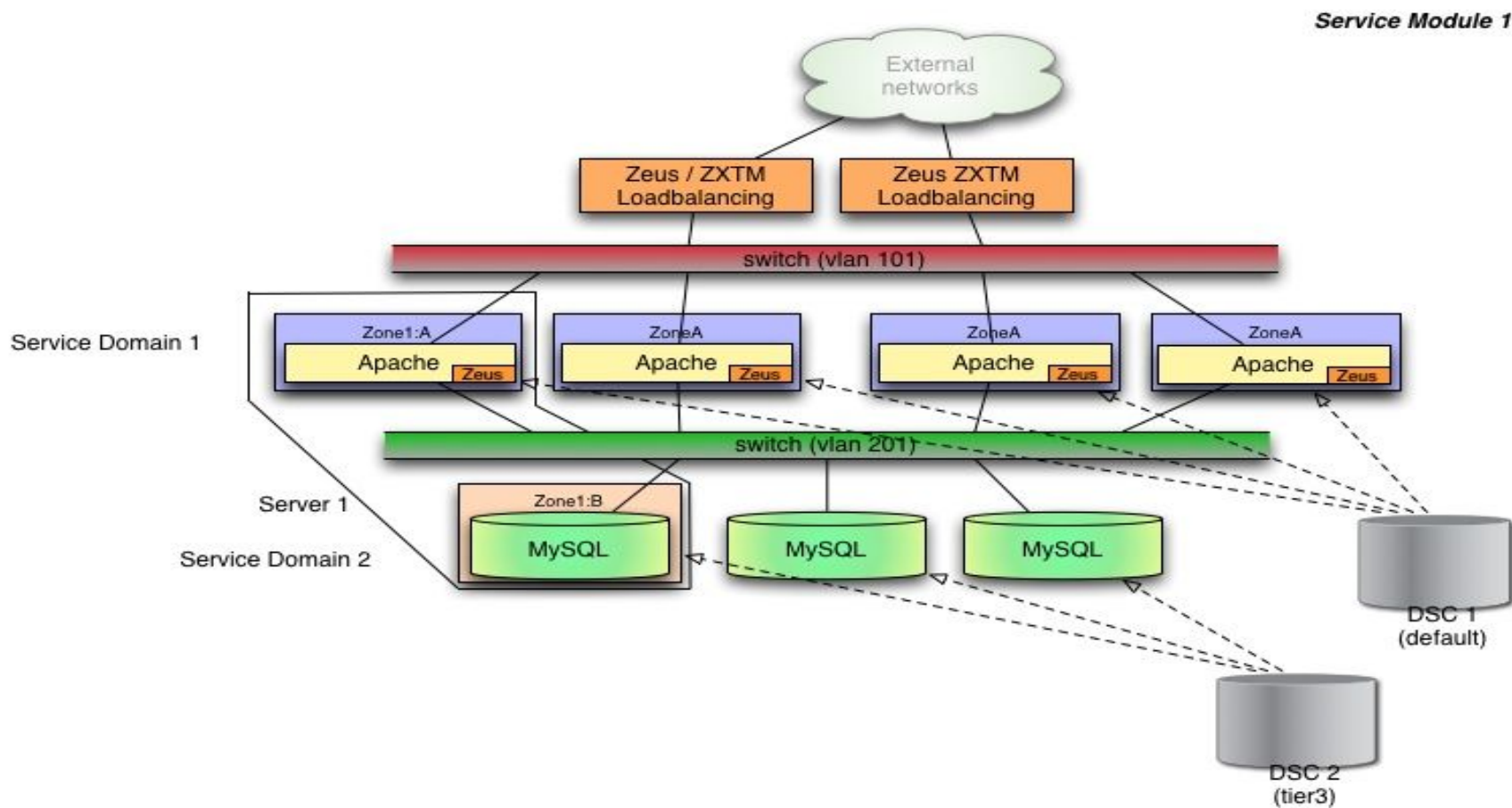
➤ Deploy a test environment

- Test workloads as “payloads”
- Use Cron or scheduler

➤ Build a cloud in a cloud

- Simplified management of a Solaris containers environment

SDN + DSC Example - Demo

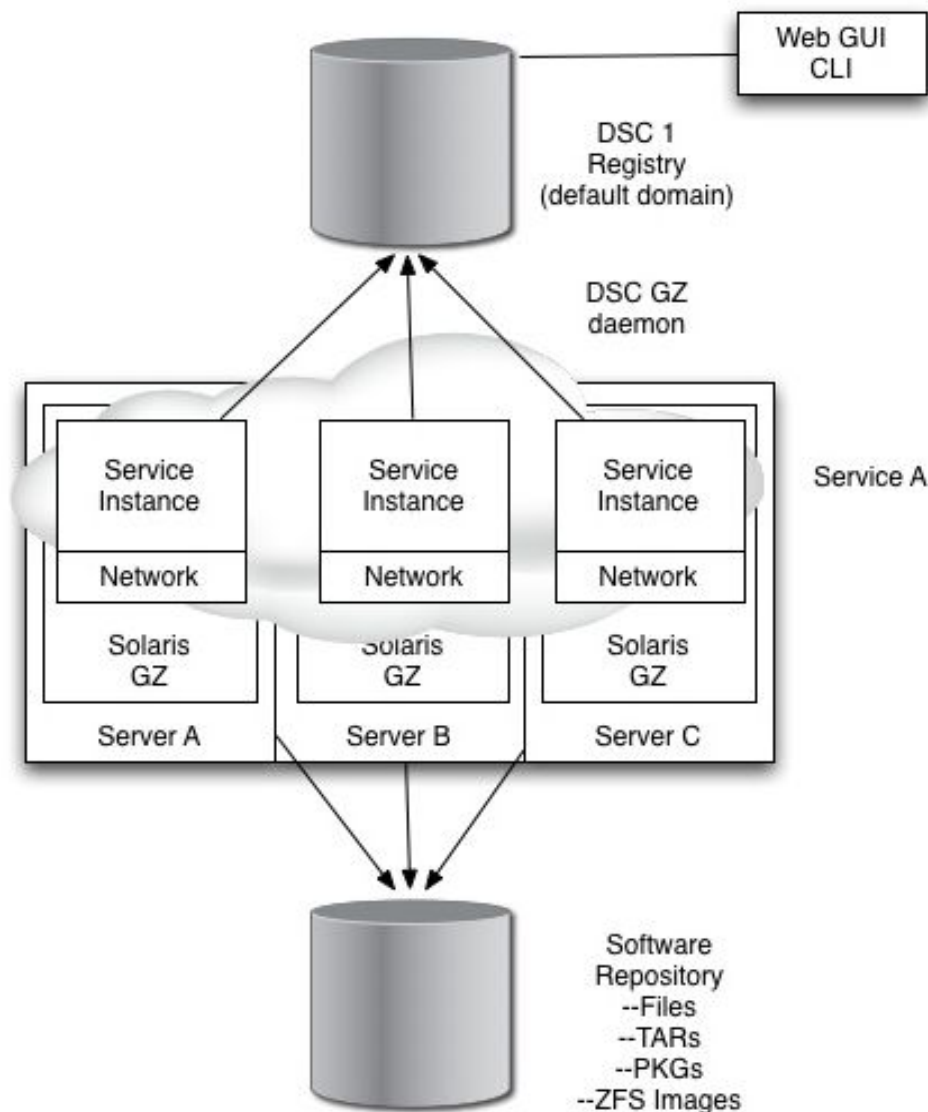


Demo Part 1

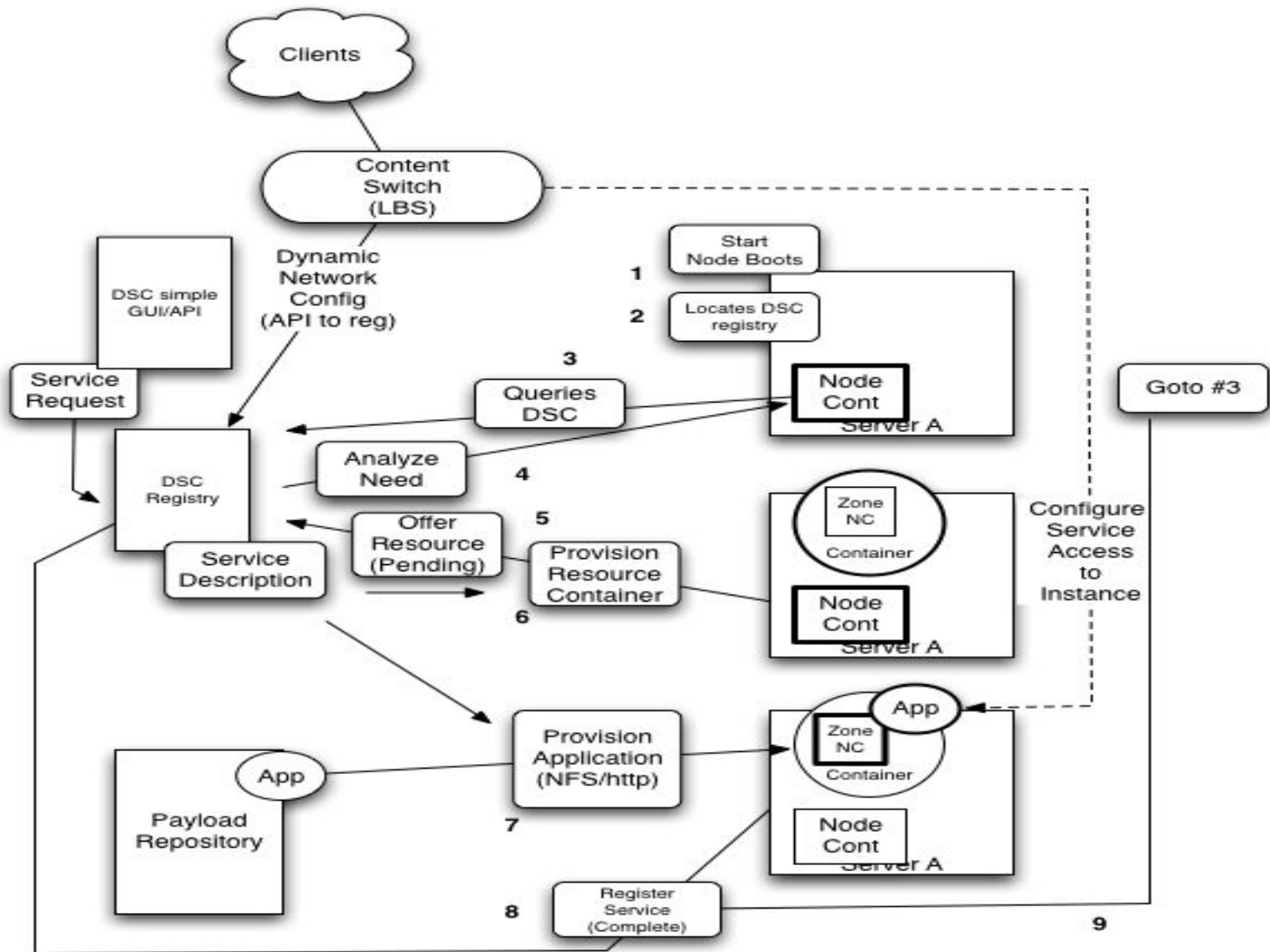
demo

DSC Features

- If you can package it, we can manage it
- Simplified SLOs
 - instance min, max
 - High Availability?
- Multi-node, scalable
 - Pull vs push
- Self-managing
- Networking support...
 - VNICs, Zeus, etc...
- Simple...
 - Packaging
 - Extensible
 - Discovery (WIP)

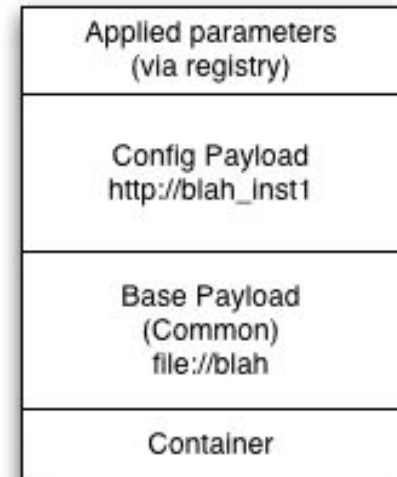


Initial Implementation Details



Packaging/Config Options (WIP)

- URL-based model for payload provisioning
- `file://` -- local file system assume mounted (could also include mount parameter in DSC db for node)
- `http://` -- wget
- `nfs://` -- NFS-based access (user pass??)
- Package Model for payload, scripts
 - > TAR bundles
 - > PKG – IPS?
 - > ZFS
 - > simple file
- Parameters from Registry
 - > e.g. instance IP
- Supports multiple levels (1 big “tar” or layered model)



Design Methodology

➤ KISS: Keep it Small, Simple, Separable

- no code module > 800 lines
- uses in BASH, PHP, a small MySQL database and a little XML for service descriptions in SMF.

➤ Modular

- Multiple services used for different parts of functionality
- built in stages, continues to evolve

➤ Understandable

- Any mid-level to good Solaris Admin should be able to setup, understand or modify the code

➤ Open Source

- Open Sourced under CDDL 1.0 license

DSC 2.0 Elements / Services / Features

➤ dsc-global

- create zones, IPs, initial resource / performance management

➤ dsc-refresh

- ongoing workload check & update if appropriate
- ongoing resource / performance management

➤ dsc-control

- Emergency-stop, DSC software update

➤ dsc-status

- reports node status back to the registry

➤ dsc-local

- runs in a created zone, loads the payload, installs and starts it

➤ dsc-integrity

- maintains integrity of cloud and model in Registry

2.0 Elements / Services Registry

➤ Various tables :

- Host, Service & Cloud Policy
- Host, Service individual, Service aggregate & Cloud Status
- Suspect Service Instances

➤ Admin Interface :

- Relatively simple html interface
- Package transfer/management interface

Provisioning Test Results (1 hour run time)

Run	Start	End	Increate	What	1 st Instance	Zone CAPS	Results
1	8:57	9:57	100	glassfish 2.2	2 minutes	CPU=4, SWAP=4GB	650
2	11:20	12:20	1000	glassfish 2.2	2 minutes	CPU=20, SWAP=10GB	888
3	1:26	2:26	30	MySQL x86		SWAP=3GB	86
3	1:26	2:26	30	MySQL SPARC Apache/ZXTM		SWAP=3GB	129
3	1:26	2:26	30	x86/Crossbow		CPU=10, SWAP=4GB	98
3	1:26	2:26	30	Apache/ZXTM SPARC/Crossbow		CPU=10, SWAP=4GB	118
3	1:26	2:26	30	2-Tier Architecture	3 minutes		431/hr
4	4:40	5:40	12	1.3GB Payload	2 minutes	none	358
5	2:35	3:35	40	1.3GB Payload	2 minutes	none	282

Node level
limits

Repo
Network
BW limit

Workload nodes :

4x T5220	64 cores @ 1.4 Ghz	64GB RAM	4 disks
6x T5440	128 cores @ 1.4GHz	64GB RAM	4 disks
2x Sfx4450	24 cores @ 2.7GHz	24GB RAM	4 disks
3x Sfx4600	32 cores @ 2.7GHz	64GB RAM	4 disks

Demo Part 2

demo

Nimsoft

- Monitor server health
 - assist in capacity management/trending
- Monitor and report on service levels of business services – dashboards, etc.
 - Easy discovery of new services
- Monitor the health of the cloud itself
 - When do I need to add more resources?
 - Are key services up and running?
 - Additional out of band capabilities beyond DSC

Nimssoft Monitoring Solution
Overview for Managed
Cloud Computing

Service Delivery Portal
Customer, Enterprise IT, Service Provider

**Cloud Framework/
Orchestration**

Business Service Management

Service Desk

Service Levels

AppLogic
EC2+Right Scale
Sun
Others

End user Response Time

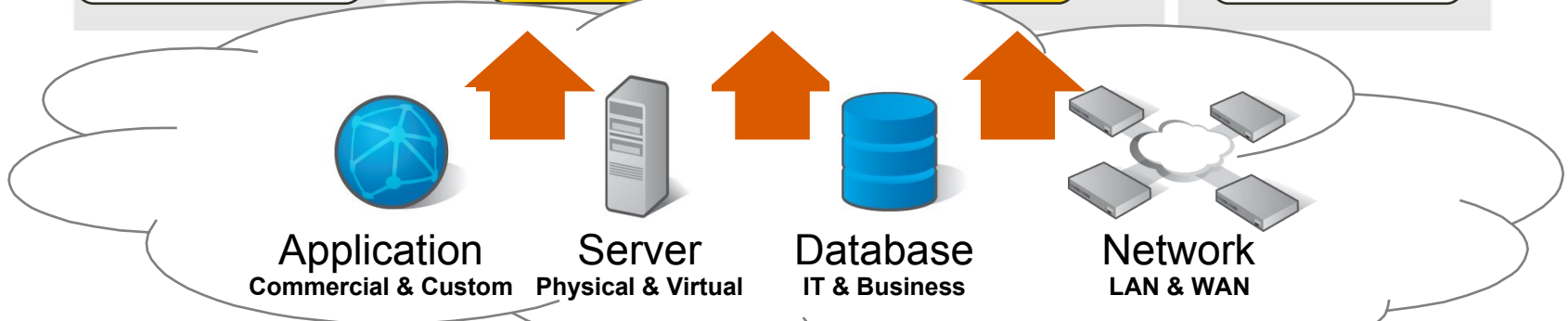
Performance & Availability

Service-now
Remedy
HP, CA,
NetSuite,
Sf.com

APIs
SDK
Web

APIs
SDK
Gateways

Events

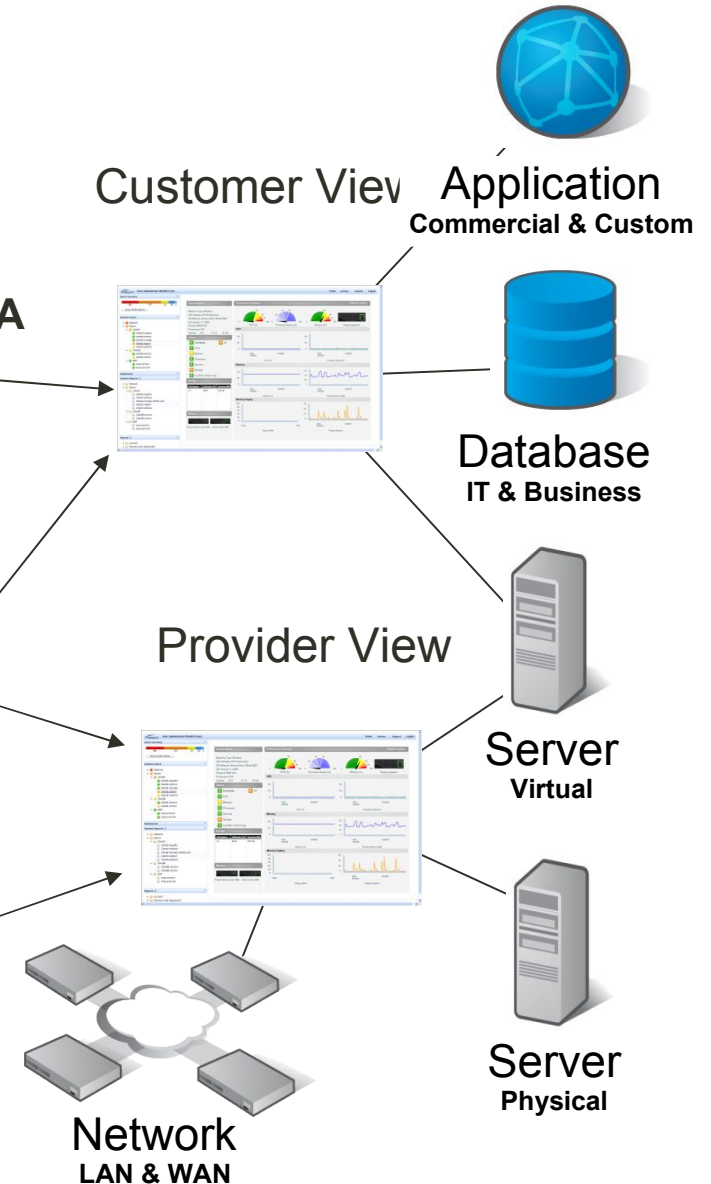


Nimsoft Monitoring

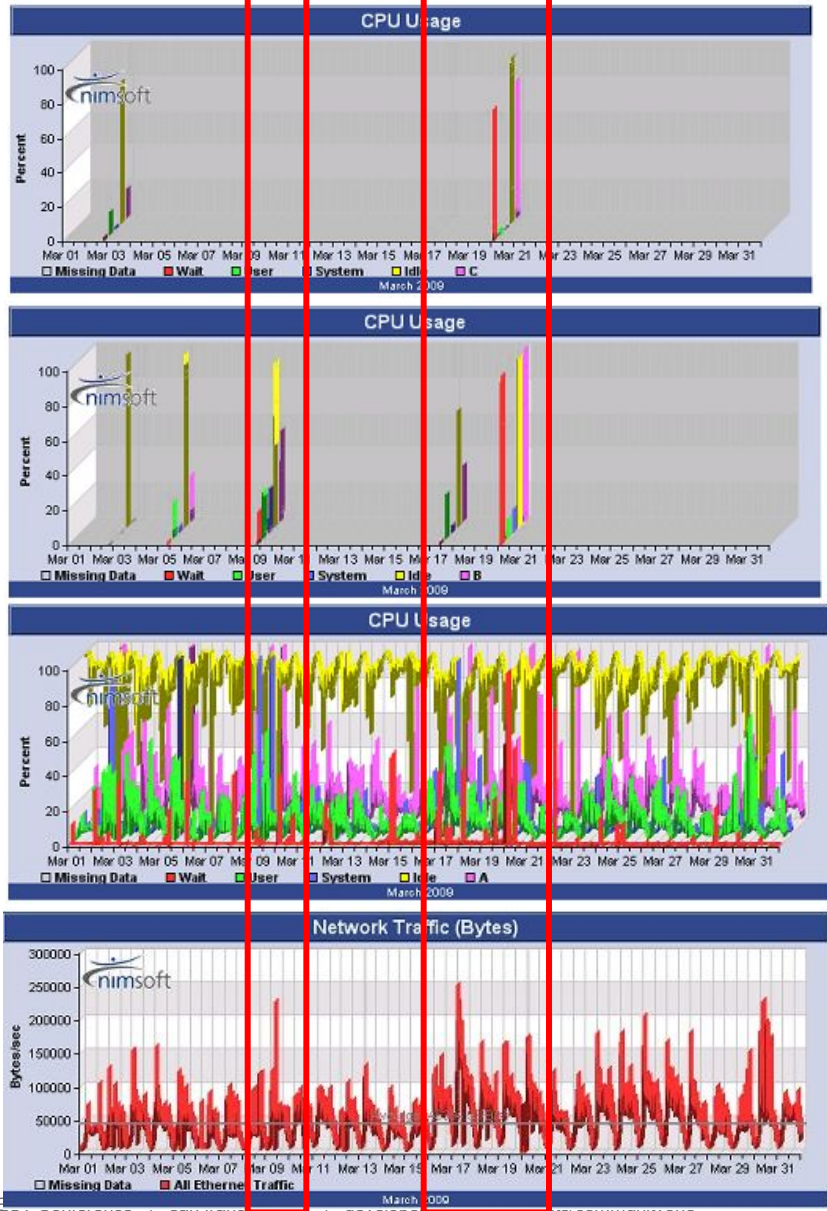
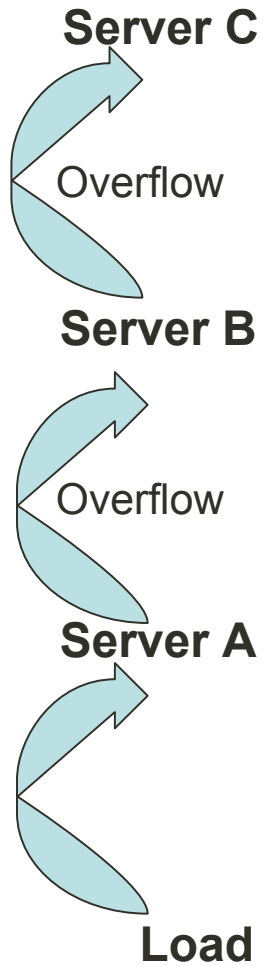
- 1) Customer Monitors their own application for its SLA
 - End-to-end service monitoring
 - URL response monitoring

- 2) Customer monitors provider's virtual infrastructure
 - Server Monitoring (of virtual servers)
 - Application Monitoring

- 3) Provider monitors their own datacenter and collective customer base
 - Network device monitoring (SNMP-based)
 - Server Monitoring



>Cloud Computing Auto- Provisioning & Nimsoft Monitoring in Action



Demo Part 3

demo

Nimsoft Benefits

- SLA Management for provider service availability, which they can publish to their customers
- Support for resource allocation and to scale resources and determine optimal cost per customer
- Automatic incident response: provider knows before end customer that there is a problem
- Predictive failure analysis

Summary/Lesson's Learned

- Simple tools to do complex things
 - Centralized resource management is a challenge at scale
 - Model what you need – not everything
 - ...how do I get to the next hop?
- Leverage partners to fill in the architecture...
 - Service level monitoring, reporting, “executive” dashboards
 - Health of the cloud itself (e.g. Nimsoft)
 - Content and load balancing (zadapter & Zeus Tech)
- Evolving “cloud-like” management...
 - Trust the cloud to manage itself as long as it has the resources (headroom) to adjust
 - “good enough?” is pretty good at scale
 - (± 1-2%? 99.99% in 5 min?)
- Use it, get involved!

Futures/Help

➤ Network Automation

- Scale up / scale down multi-tier app use case
- Automatic Service Discovery (layer 4+)

➤ Better Node and Service Monitoring

- In progress – beta2 & 2.0 release

➤ LDOMS/Other VMs (Virtualbox, etc)

➤ Payload create/upload script

➤ 10,000+ workload nodes test

➤ mDNS for registry discovery

	Mgd Wload	Mgd Wload
	DSC Zone	DSC Zone
Managed Workload	DSC Node Controller	
DSC LDOM/OS	DSC LDOM	
DSC Node Controller		
Control LDOM/OS		
Physical Hardware		

Sun Professional Services for Cloud Computing

➤ Get Started:

- Cloud Strategic Planning Service:

- This service engagement results in a business readiness analysis (gap analysis), 24 month cloud roadmap, risk versus reward analysis and potentially some initial architectures and budgetary quotes as makes sense.

➤ Complementary Services:

- Customized planning engagements to address important cloud considerations

➤ Follow-on Professional Services for Cloud

- Methodology: The overall program management (customer facing) methodology is Sun Scope. The AIM methodology is used as the delivery methodology.

Sun Solution Centers for Benchmarking

Sun Solution Centers are a global community.....

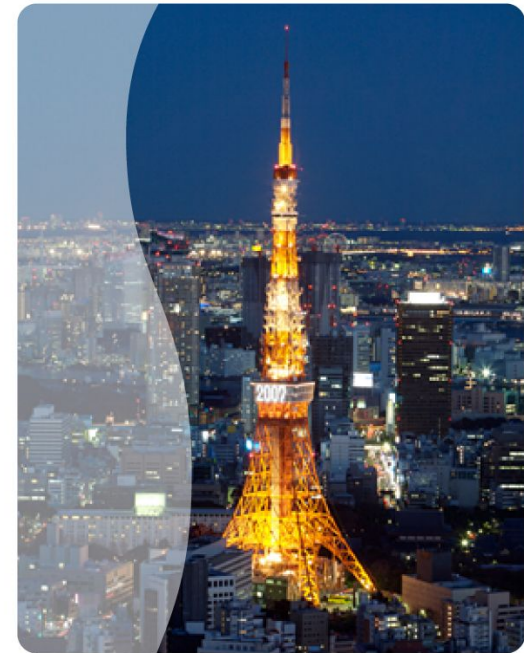
- We bring Sun technology, partner solutions, and best industry
- Around 24 SSCs and 15 Authorized SSC
- Comprehensive, Coordinated Network Services(AAA)
Wide VPN Network – IPSEC Over MPLS IPVPN/Internet

that provides heterogeneous environment.....

- We provide the environment to solve customer's integration and interoperability challenges across multi-vendor solutions

so that customers can test before they invest

- We reduce risk and time-to-market for customers by allowing them to try-before-they-buy towards their business needs



Move Rapidly from vision, value to customized business solutions.
Think of the Solution Center as a place for mind-share creation.

Get Involved

➤ Documentation

- <http://wikis.sun.com/display/DSC/>

➤ Code Repository / Community

- <http://dsc.kenai.com>
-
- Project Stratus (Simplified Cloud Networking)
- <http://wikis.sun.com/display/DSDC/>
- Sun Cloud -- <http://www.sun.com/cloud>
- Nimsoft -- <http://www.nimsoft.com>

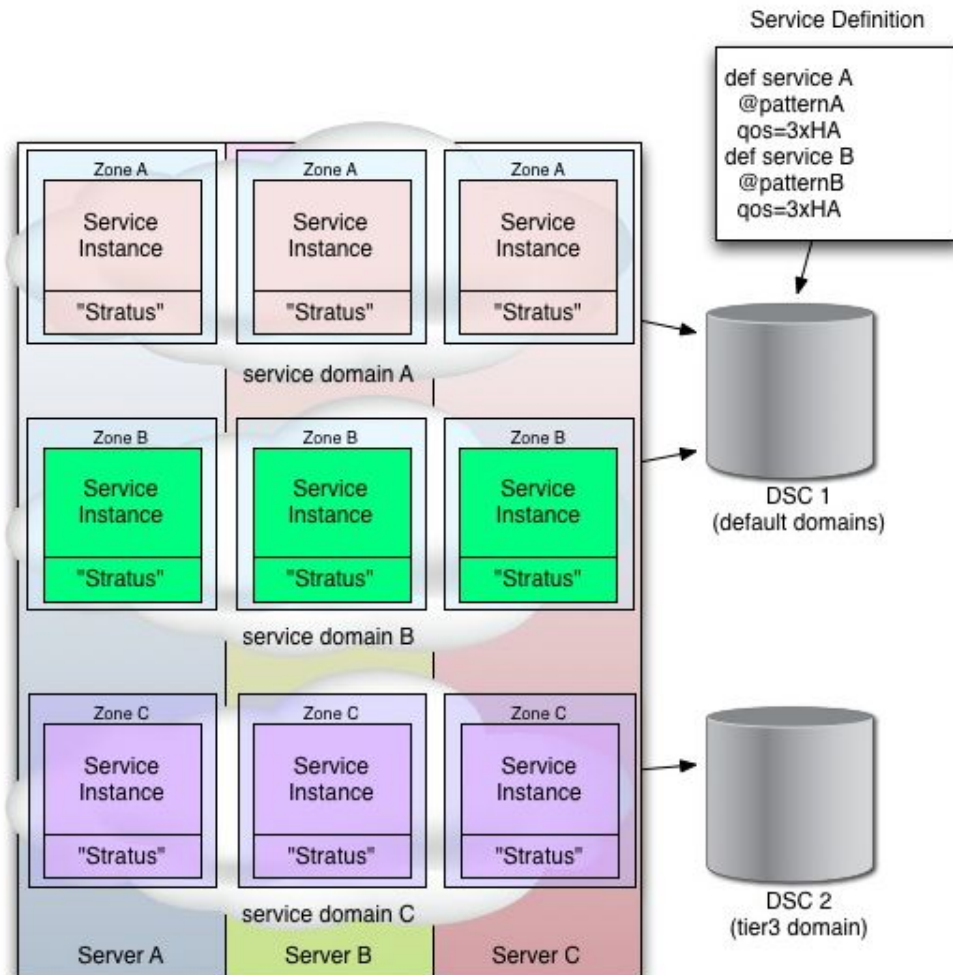


Project OpenSolaris™ Dynamic Service Containers featuring Nimsoft SLM

Session: S311526

Thank You!

DSC Topology Example



Failure Mode Examples (WIP)

➤ Local Instance

- Monitored, restarted by SMF

➤ Complete Node (Server) Failure

- Could be detected by “neighbor”
- Neighbor reports node failure to registry
- DSC provisioning cycle starts up new instances

➤ Network Service Instance Monitoring

- Load balancer could monitor service health and report state to registry

➤ DSC registry failure

- Working on distributed model to prevent failure
- Can cluster today for HA
- Also if nodes loose comms to registry – no changes