Xen Cluster without BloatStack

Linus van Geuns

about:speaker

- Linus van Geuns
- Freelancer
- Linux, Virtualization, LDAP, Networking, ..

about:talk

- Present “MHCoDi” cluster concept
  - designed by Markus Hochholdinger
  - In production since 2006
  - Based on Xen hypervisor (mostly)
- First contact of speaker in ~2009
  - through Ulrich Hochholdinger
  - As a “student trainee” at Netpioneer GmbH

Contents

- Introduction
- Motivation
- Concept
- Evolution

Introduction

- public, private and hybrid “cloud”?
- OpenStack? Ganeti? ..
- Virtualization
  - Abstraction from actual hardware
  - “Container” or “guest operating systems”
  - One container for each service?

Contents

- Introduction
- Motivation
- Concept
- Evolution

Motivation

- Efficiency of hardware utilisation
- Decrease impact of hardware maintenance
- Optimize infrastructure maintenance
- Strong seperation of services / containers

Motivation

- Efficiency of hardware utilisation
  - Decrease number of hardware servers required (cost, maintenance)
- Decrease impact of hardware maintenance
  - Redundancy in case of hw failure
  - Hardware maintenance without service impact
    - Spare parts, upgrades, ...

Motivation

- Optimize infrastructure maintenance
  - Use FOSS Linux/UNIX CLI tools
  - Maintain flexibility
  - Reduce possibility of “split brain” situations
- Strong seperation of services / guests
  - Ressource scheduling between guests
  - Use same infrastructure for multiple customers or different service setups

Contents

- Introduction
- Motivation
- Concept
- Evolution

Concept

- Xen as a hypervisor (“type 1”)
  - FOSS
  - Small codebase
  - “Clean” architecture (claimed)
  - More efficient “para-virtualization” mode for Linux guests
  - Concept has also been set up with KVM

Concept

- Xen as a hypervisor (“type 1”)
  - Requires a “management domain” (dom0)
  - Management of hardware, drivers
  - Management of configuration (network, etc)
  - Management of “guests” (domU)
  - Main attack vector: dom0 (Linux instance :o)

Concept

- Hardware (Redundancy)
  - At least one hardware node allowed to fail
  - Simplest set-up: Two hardware servers
  - Guests allowed to fail on hardware failure
    - Quick recovery on redundancy node

Concept

- Guests / services (redundancy)
  - Sync persistent data to two hardware nodes
  - One “disk” from each hw node available to guest (domU)
  - Sync “disks” via RAID1 within guest
    - Software RAID
  - Monitor that the guest is running on one hardware node only (no automatic start-up)
  - Split brain monitoring based only on guest creation

Concept

- “Disks” (redundancy)
  - Partition disks on hw node via logical volumes
  - Export logical volumes to other hw nodes via network (currently iSCSI)
  - Import logical volumes from other hw nodes via network (..)
  - Manage consistent device names / paths across all hardware nodes
  - Transparent devices available on all cluster nodes

Concept

- “Disks” (backup and performance)
  - Stripe logical volumes over several local disks to increase throughput
  - Backup: snapshots of logical volumes and sync filesystem content to backup volumes

Concept

- Network
  - Seperate SAN and service traffic
  - Bridge interfaces of guests to service traffic (v)LANs
  - Use redundant physical links to redundant switch setup via bonding

Concept

- Service state
  - Leave replication of service state to applications / services
    - too complex for infrastructure layer

Contents

- Introduction
- Motivation
- Concept
- Evolution

Evolution

- SAN
  - Further analysis of iSCSI 10GBit Ethernet performance
  - Analyze FCoE performance on 10G Ethernet

Evolution

- Network automation
  - SDN/OpenFlow for network hardware
  - SDN/OpenFlow for dom0 networking
    - Open vSwitch
  - Find approriate FOSS controller for SDN
  - active/active redundant links

Evolution

- Automation of Maintenance
  - Automatic provisioning of service guests
  - Automatic provisioning of services / service configuration
  - Puppet / Saltstack / …?