



2015 Connect OpenVMS Bootcamp

**ROBERT GEZELTER**  
SOFTWARE CONSULTANT  
*Bringing Details Into Focus™*  
*Focused Innovation*  
*Focused Solutions*

# **Using OpenVMS Technologies to Build an Agile Computing Base From Experiment to Production without Interruption**

**Robert Gezelter, <http://www.rlgsc.com>**

As a courtesy to your fellow attendees:

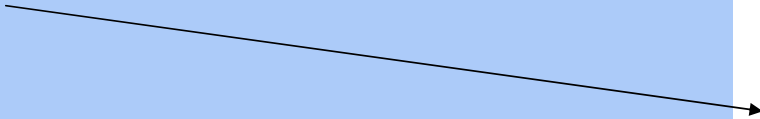
Please take this opportunity to check that **ALL** portable electronic devices are on the silent or vibrate settings.

If you need to answer a call, please leave the room to avoid disturbing your fellow attendees.

This is a NOT a Non-Disclosure Session.

This session, and the material therein may be referenced and reproduced without limitation, provided that proper credit is given and the copyright notice is retained unaltered.

# The goal – Seamless operation from Experiment through Production



# Do you use “cloud computing”?

- ◆ Scalability
- ◆ Configuration independence
- ◆ Maintainability
- ◆ Upgradeability
- ◆ Transparent failover



## 2015 Connect OpenVMS Bootcamp

**ROBERT GEZELTER**  
SOFTWARE CONSULTANT  
*Bringing Details Into Focus™*  
*Focused Innovation*  
*Focused Solutions*

Maintainability

Upgradeability

Transparent Failover

Scalability

Configuration Independence

Using OpenVMS Technologies to Build An Agile Computing Base

© 2014, Robert Gezelter, All Rights Reserved

## “ility’s” are results; not causes

- ◆ Specific engineering create results
- ◆ Most “cloud” presentations omit what creates the results
- ◆ Many “cloud” computing models are nothing more than “virtualized” versions of non-cloud platforms (e.g., Windows™, Linux)
- ◆ Virtualization does not solve problems (e.g., virtual machine migration)

# Is “cloud computing” new?

- ◆ The term is of recent origin
- ◆ Computing independent of being “in front of the machine” is by no means new
  - ◆ SaaS
  - ◆ ASP
  - ◆ Remote Access (1970’s)
  - ◆ Timesharing (Project MAC, circa 1963)



### Six blind men and an elephant

- ◆ What you feel depends on where you are
- ◆ Perspectives are only a single point or slice



From Martha Adelaide Holton & Charles Madison Curry (1914), *Holton-Curry readers*,  
Rand McNally & Co. (Chicago), p. 108

# Often, what appears different is merely a question of perspective

- ◆ Not unlike the elephant
- ◆ Circles, ellipses, parabolas, hyperbolas, and other curves are all “conics”
- ◆ “conics” are all slices of a cone
- ◆ Analyses are all related
- ◆ Understand general case, all of the special cases are solved

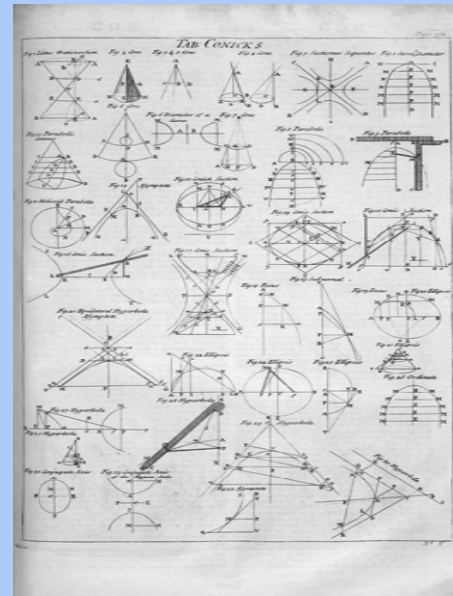


Table of Conics, *Cyclopaedia* (1728), volume 1, pp 304

# Difference between clairvoyance and reality

- ◆ Controlled and uncontrolled changes are fundamentally different
- ◆ Example: Processor upgrade
  - ◆ Known in advance
  - ◆ At “Time and Place chosen”
  - ◆ Can always be aborted

## Difference between clairvoyance and reality (cont'd)

- ◆ Example – Uncontrolled
  - ◆ “Time and task not of my choosing” (?) – Chester Nimitz, Admiral, USN, Spring 1942
  - ◆ No advance warning
  - ◆ No reschedule
  - ◆ No inherent fallback
- ◆ Cases in point: World Trade Center, 9/11; Blade-out in a jet turbine; Spring 2004 HPTF NE US power outage

## The difference – In short

The difference can be summarized as that between a ordinary switch and a circuit breaker

- ◆ Switches work when thrown
- ◆ Circuit breakers work either when:
  - ◆ Manually
  - ◆ Automatically (when an overload occurs)
- ◆ Circuit breakers are more embrative than switches

# Back to computing: An example – Virtual machine migration vs. OpenVMS Clusters

- ◆ Comparing apples to oranges
- ◆ Virtual machine migration is a “switch”
- ◆ OpenVMS cluster failover is a “circuit breaker”
- ◆ Virtual migration is useful ***WITHIN*** the context of an OpenVMS cluster; it is not a substitute

# Combining existing fundamental facilities in new ways

- ◆ OpenVMS clusters
  - ◆ Shared locking domain
    - ◆ Shared system volumes
  - ◆ Logical names
  - ◆ Rolling reboot
  - ◆ Volume Shadowing for OpenVMS (aka HBVS)

Using OpenVMS Technologies to Build An Agile Computing Base

# Combining existing fundamental facilities in new ways (continued)

- ◆ HP Virtual Machines (and other virtualization products from Stromasys and Migration Specialties)



# Each of these technologies is independent

- ◆ These technologies are independent
- ◆ In concert, they create an extremely malleable environment
- ◆ This flexibility allows us to transition the hosting and capacity of a cluster in any way we choose

# The fourth dimension: Time

- ◆ Hindsight is always 20/20 (if not better)
- ◆ Foresight, somewhat less so
- ◆ Capacity projects are fallible; both high/low

# Employ technology to remove shortfalls

- ◆ OpenVMS clusters address capacity up/down
- ◆ Volume Shadowing for OpenVMS allows us to change storage platforms
- ◆ Virtual machines allow:
  - ◆ Fractional provisioning
  - ◆ +(fractional second) Ready Reserve capacity
- ◆ Dynamic Volume expansion allows expansion of file volumes
- ◆ Logical names hide hardware dependencies

Using OpenVMS Technologies to Build An Agile Computing Base

# Not new technologies: Change Perspective

- ◆ “short sightedness” is a common hazard
- ◆ Manuals often reinforce with “on point” examples
- ◆ The general case is often under explained and thus under appreciated



## Technologies from a high perspective

- ◆ “Not seeing the forest for the trees”
- ◆ A more global perspective aids comprehension

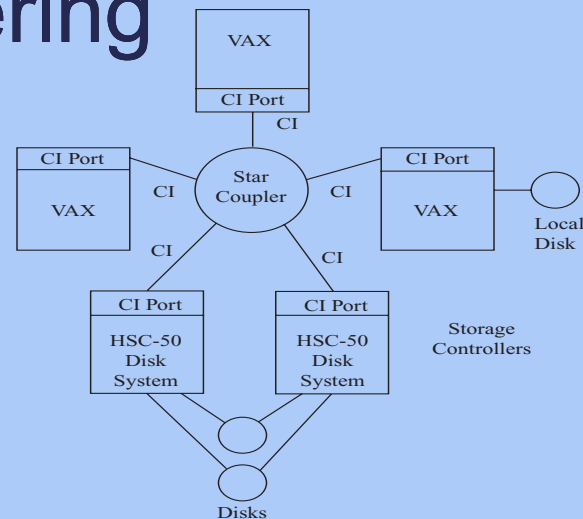


Then look at point cases as one point in a long-term continuum



# In this vein, revisit OpenVMS clustering

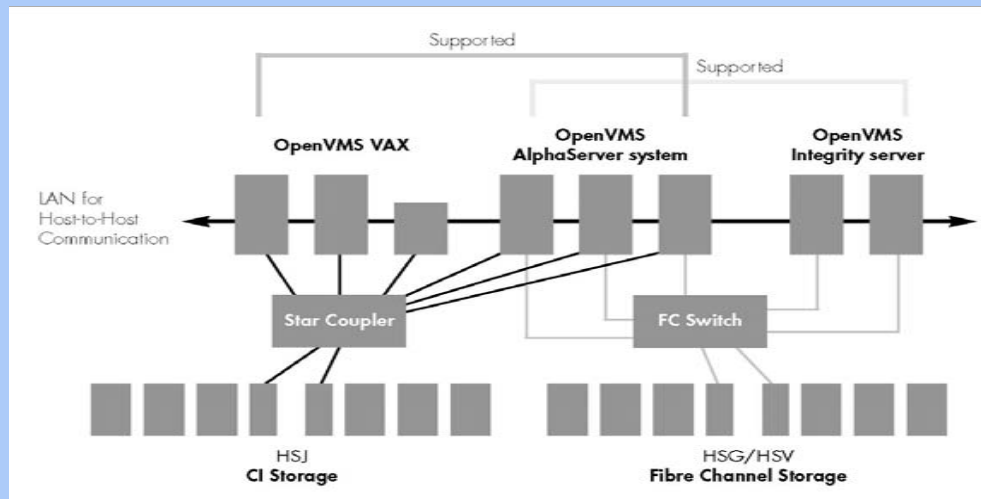
- ❖ Classic VAX cluster (Kronenberg, Levy, Strecker, 1986)
- ❖ Certainly valid
- ❖ Not the entire concept
- ❖ Does not illustrate the potential of the “OpenVMS cluster gestalt”



From Kronenberg, Levy, & Strecker, (1986)  
*VAXcluster: A closely-coupled distributed system*  
ACM Transactions on Computer Systems 4(2)

# Current OpenVMS Clusters

- ◆ Even today's examples are far too restrictive
- ◆ Cluster nodes remain hardware tied
- ◆ This is an unneeded and incorrect belief





# Both classic and present are snapshots

- ◆ Both are individual moments in time
- ◆ Over time
  - ◆ a cluster node may be small, large, or non-existent
  - ◆ Over time, nodes matter
  - ◆ Nodes are independent of their hardware

|            |               |
|------------|---------------|
| Monday     | Fractional VM |
| Tuesday    | BL 860        |
| Wednesday  | <none>        |
| Thursday   | Fractional VM |
| ...        | <none>        |
| Monday + n | Superdome     |

An OpenVMS cluster node is **NOT** a :

- ◆ CPU, blade, box, or virtual partition
- ◆ System disk (or root thereof)

# If a node is not a machine, what is it?

A member belonging to an OpenVMS cluster is identified by its Cluster ID (`SCSSYSTEMID`) and Cluster Node name (`SCSNAME`). At any given point in time, a member can exist on at most one “processor” with communications to the OpenVMS cluster. The current host processor may be real or virtual.

# An active OpenVMS cluster member has a:

- ◆ Host processor(s)
- ◆ A system volume or shadow set
- ◆ A specific system root on the system volume (**SYS\$SPECIFIC**)
- ◆ Files specific to that root
- ◆ Files specific to that node (note the difference with the preceding)

# Types of nodes in an OpenVMS cluster

- ◆ Core nodes (voting)
- ◆ Satellite nodes (non-voting)

Both types of nodes may be individually virtualized at various times.

## New logical name needed: **SYS\$NODE\_SPECIFIC**

- ◆ New root on system volumes: **[NODE\_SPECIFIC]** [Gezelter, 2009]
- ◆ Each member has a directory below this root (e.g., **[NODE\_SPECIFIC.ALPHA]**)
- ◆ Add logical name definition early in startup process by entering definition file in user side of **STARTUP** database (**STARTUP\$STARTUP\_LAYERED**)
- ◆ Inserted in **SYS\$...** search lists behind **SYS\$SPECIFIC** and before **SYS\$COMMON**

## New logical name needed: **SYS\$SITE\_SPECIFIC**

- ◆ Logical names specific to local site [Gezelter, 2004]
- ◆ May have separate directory tree, e.g. [**SITE.<location>**]
- ◆ Add logical name definition early in startup process by entering definition file in user side of **STARTUP** database (**STARTUP\$STARTUP\_LAYERED**)
- ◆ Inserted in **LN\$FILE\_DEV** ahead of **SYS\$COMMON** and behind **SYS\$NODE\_SPECIFIC**

## Each OpenVMS cluster node has several alternative boot roots

- ◆ Base node definition information (**SCSNAME**, **SCSSYSTEMID**, DECnet node address, etc.) in **SYS\$NODE\_SPECIFIC**
- ◆ Individual boot roots hold system parameter file
- ◆ Writeable logs
- ◆ Possibly page file (could be in **SYS\$NODE\_SPECIFIC** or elsewhere)
- ◆ Possibly dump file (could be in **SYS\$NODE\_SPECIFIC** or elsewhere)



# Why separate node specific and boot roots?

- ◆ Production version
- ◆ Test version
- ◆ Previous production version
- ◆ Experimental version
- ◆ Different hardware scenarios (e.g., blade, virtual, rx2660, AlphaServer DS10)

## Separate roots – Example

Cluster member **GREEN** has:

- ◆ node specific files in **[NODE\_SPECIFIC.GREEN]**
- ◆ Port Production BL860c boot root of **sys1**
- ◆ Starboard Production BL860c boot root of **sys11**
- ◆ Emergency rx2660 boot root of **sys21**
- ◆ Test BL860c boot root of **sys31**
- ◆ Experimental boot root of **sys41**
- ◆ etc ...

# Specific boot roots invoke Node-specific files

- ◆ **STARTUP** series command files (e.g., **LAT\$SYSTARTUP.COM**)
- ◆ **AUTOGEN** files
- ◆ Test within “Experimental Boot root”, promote to “Production” roots or Node-specific directories
- ◆ Similarly, promote from Node-specific to **sys\$COMMON** as appropriate

# About system volumes

- ◆ Characterizing OpenVMS as a “single system image” cluster understates the case
- ◆ “single system images” (e.g., shared system disk) is a possibility; but it is only one of many
- ◆ “a copy of the system that may be used by zero or more nodes at any point in time” may be a more appropriate description
- ◆ At least one (preferably more) per architecture per cluster at any moment in time

# System volumes are similar to boot roots

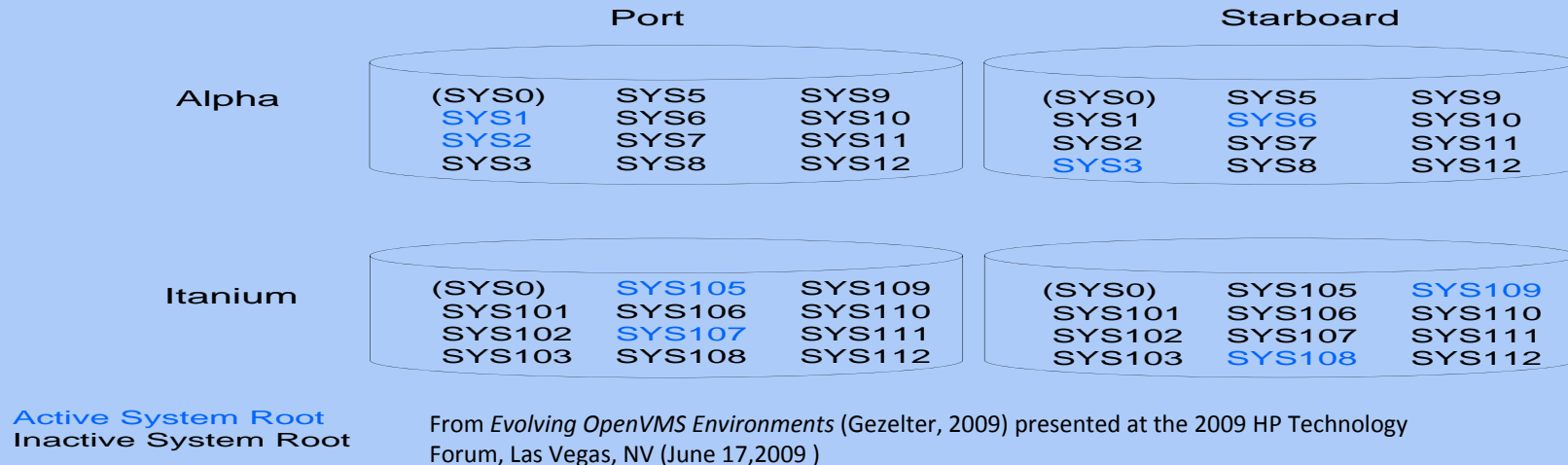
Per architecture:

- ◆ Port/Starboard Production (or more depending on load) copies
- ◆ Test copies for upgrading
- ◆ Previous copies for fallback
- ◆ Experimental copies as needed
- ◆ Master copy

# Treat system volumes same as applications

- ◆ Clone masters for “Production” copies
- ◆ For “Upgrades” or “Installations”
  - ◆ Clone master creating test system volume
  - ◆ Perform update/installation
  - ◆ Following test; promote Test to master
  - ◆ Create one/more new Production clones
  - ◆ Phase in use of new Production clones; phase out previous set of Production clones

### Steady state:



## About cluster members:

OpenVMS clusters are often incorrectly described as being a “***n***-node cluster”. A better phrasing would be “normally a ***n***-node cluster”.

Why?

- ◆ Sporadically operating test nodes
- ◆ Scheduled expansion (daily) nodes (e.g., “Wildfile”)
- ◆ Pre-configured expansion nodes (often non-voting)



## Surge capacity (“call up the reserves”):

- ◆ Pre-defined satellite “worker” nodes
- ◆ May be physical (e.g., blade, test system, quality assurance systems, training systems)

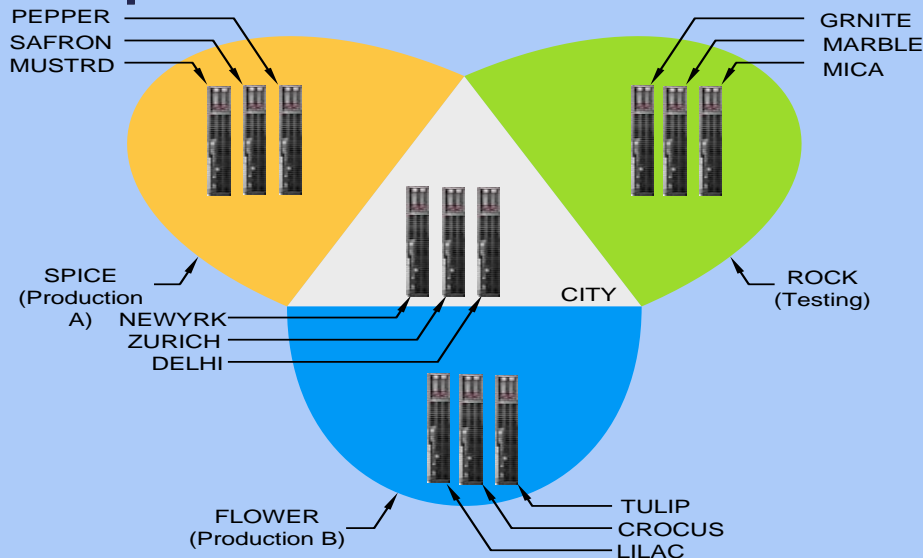
## 2015 Connect OpenVMS Bootcamp

“Oh &^%\*&\$#; get 10,000 (or more) VUPS  
online now!!!!

- ◆ Remember those pre-configured reserve production roots?
- ◆ Consider:
  - ◆ Virtualizing test/quality/assurance/training systems
  - ◆ Creating a nominally, high priority reserve production cluster member instance in a different VM on the same physical host hardware.
  - ◆ 90+% return of capacity in under one second; reduced impact on normal users (test, QA, students)

# Hardware assets become a “pool”:

- ◆ Assets are fungible
- ◆ Reallocate as needed
- ◆ Virtual slices can be quickly pre-empted



From *Evolving OpenVMS Environments* (Gezelter, 2009) presented at the 2009 HP Technology Forum, Las Vegas, NV (June 17, 2009)

# This is not theoretical

- ◆ This is all completely legal OpenVMS
- ◆ Nothing has been done which has not been supported
- ◆ Fall forward; not fall back
- ◆ Shortened downtime
- ◆ Agility  $\equiv$  pre-provisioned and prepared
- ◆ This is an “OpenVMS” private cloud with all of the attributes of a virtually hosted servers on other platforms

# Back to the original problem – Prototype to Production without Interruption

- ◆ There are multiple variables, each of which can prevent success
- ◆ Look at successful episodes, is there a common thread?

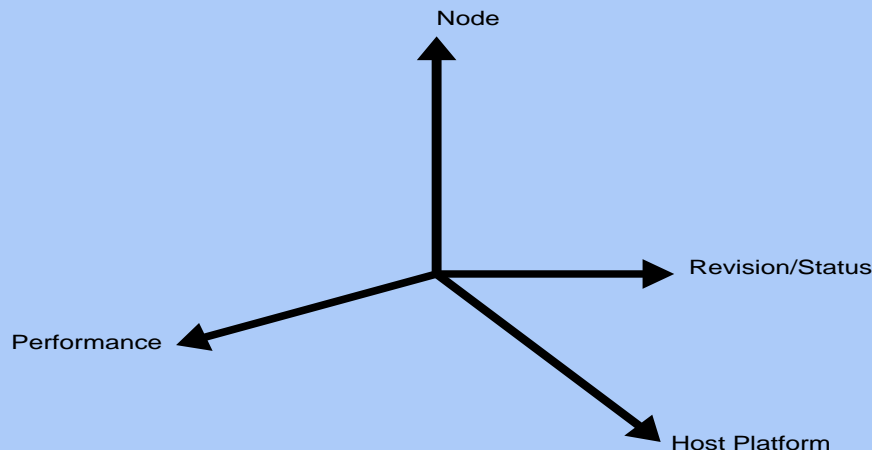
# How does OpenVMS do it?

- ◇ Since 1976, OpenVMS has run on
  - ◇ VAX
  - ◇ Alpha
  - ◇ HP Integrity™
- ◇ Some users and engineering have done this without disruption
- ◇ What is the “secret sauce”?



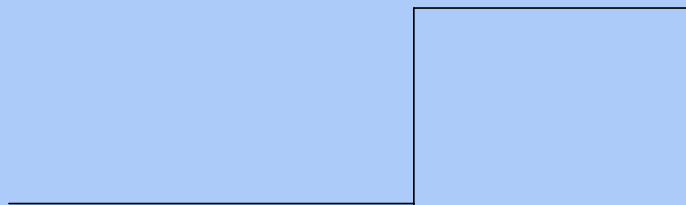
# Difference issues are independent, not linked

- ◆ Each one is independent



# What is the challenge?

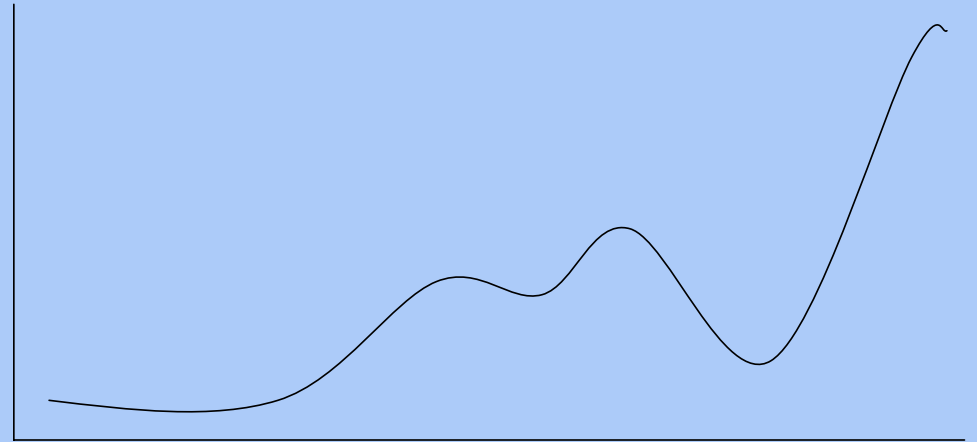
- ◆ Quantum transitions
- ◆ High risk
- ◆ No control
- ◆ Difficult to retreat





## A better approach – Incrementalism or Gradualism

- ◆ Calibrated changes
- ◆ Do change as can be accommodated
- ◆ Amount at risk is calibrated by business and technical considerations



# Continuity is the goal

- ◆ The OpenVMS trademark – rolling upgrade
  - ◆ Switch architectures
  - ◆ Switch system disks
- ◆ The constant is the “cluster member”, not the disk, CPU, or architecture



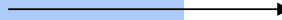
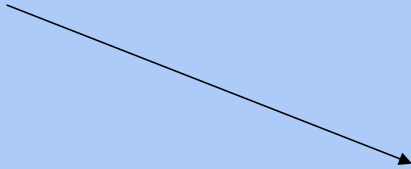
# Toward the future is often a teleological trap

- ◇ The future is inherently unclear and unknowable
- ◇ Evolution is in the current, not the future. Effort will not be expended for something that is not an immediate advantage
- ◇ Change is constant
- ◇ Positioning for change is the foundation of agility

## Dealing with load

- ◆ Pre-configured worker members
- ◆ Instant availability surge capacity as already active members on slices of virtual processors
- ◆ Difference between activity surge and flash spike
- ◆ Flash spike created by
  - ◆ Member hardware failure or crash
  - ◆ Flash spike in demand
- ◆ Long term phenomena are different

# Back to our goal: Experiment through Production without interruption



# Each step in the lifecycle is not significant

- ◆ Each increment is nothing more than a change in
  - ◆ Capacity
  - ◆ Host
  - ◆ Architecture
  - ◆ Version or revision
- ◆ “rolling reboot” is the core:
  - ◆ Add new member to cluster
  - ◆ Remove/reboot old member

## 2015 Connect OpenVMS Bootcamp

# Initial configuration

- ◆ “Soloist OpenVMS Cluster” [Gezelter, 2009]
- ◆ Configuration
  - ◆ Single node OpenVMS cluster
  - ◆ Single member shadow sets (system disk, data disk)
  - ◆ Fractional CPU hosting
    - ◆ HPVM
    - ◆ Stromasys Charon
    - ◆ Migration Specialties Avanti
- ◆ De minimis capital costs for prototype applications

# Capacity increases over time

- ◆ Increase virtual slide
- ◆ When appropriate, add real hardware
  - ◆ Boot in second member
  - ◆ Member may be spare free-standing; or it may be a blade
  - ◆ Up to a certain point, it can be increasing slices of a virtual processor
  - ◆ Business decision, the technical architecture is agnostic on the details of the provisioning



# Disk storage

- ◆ All volumes members of shadow sets
- ◆ For ordinary disks
  - ◆ Use 1-member shadow sets
  - ◆ Transition to different hardware or array by temporarily creating 2-member shadow sets
- ◆ For all shadow sets
  - ◆ Dynamic volume expansion enabled
- ◆ See “Migrating OpenVMS Storage Without Interruption” [Gezelter, 2007] HP Tech Forum 2007

# Operational considerations

- ◆ Volumes that are shadow sets can be migrated without interrupting normal operations
- ◆ User indistinguishable
  - ◆ File resident virtual disks
  - ◆ Real disks
  - ◆ MSA
  - ◆ EVA
  - ◆ Reconfiguration thereof (RAID)

# The key underlying principle

- ◆ Changes in all cases are user indistinguishable.
- ◆ If no user perception of change, change did not happen

## Where to start?

- ◆ Start process where appropriate
- ◆ If “budget challenged” the “on-ramp” (entry point) is
  - ◆ Fractional virtual CPU slice (VAX, Alpha, Integrity)
  - ◆ One/two single member host based shadow sets (may be containers a.k.a. file based “virtual disks”)
- ◆ Anywhere in between, this is a business decision

# Summary

- ◆ “Highly agile” is the result of preparation
- ◆ Many “cloud” offerings have substantial undisclosed and undocumented approaches, e.g., “Trust us”
- ◆ Infinite capacity is physically impossible
- ◆ Calling on reserves quickly without user disruption is the long term key



## 2015 Connect OpenVMS Bootcamp

**ROBERT GEZELTER**  
SOFTWARE CONSULTANT  
*Bringing Details Into Focus™*  
*Focused Innovation*  
*Focused Solutions*

Questions

Slides and other materials:

<http://www.rlgsc.com/openvms-bootcamp/2015/index.html>



2015 Connect OpenVMS Bootcamp

**ROBERT GEZELTER**  
SOFTWARE CONSULTANT  
*Bringing Details Into Focus™*  
*Focused Innovation*  
*Focused Solutions*

## **Robert Gezelter Software Consultant**

35-20 167<sup>th</sup> Street, Suite 215  
Flushing, New York 11358-1731

gezelter@rlgsc.com  
<http://www.rlgsc.com>

Voice: +1 (718) 463 1079