



Education

# Understanding High Availability in the SAN

Mark S Fleming, IBM

- The material contained in this tutorial is copyrighted by the SNIA.
  - Member companies and individual members may use this material in presentations and literature under the following conditions:
    - ◆ Any slide or slides used must be reproduced in their entirety without modification
    - ◆ The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
  - This presentation is a project of the SNIA Education Committee.
  - Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
  - The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.
- NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.**

## ➤ Understanding High Availability in the SAN

- ◆ This session will appeal to those seeking a fundamental understanding of High Availability (HA) configurations in the SAN. Modern SANs have developed numerous methods using hardware and software to assure high availability of storage to customers. The session will explore basic concepts of HA, move through a sample configuration from end-to-end, investigate HA and virtualization, and discuss some of the challenges and pitfalls faced in testing HA configurations.

# The 50,000 Foot View

## ➤ High Availability in the SAN

- ◆ Is a design protocol
- ◆ Requires **technology, skills, information** and **processes** to deliver
- ◆ *Promises* up to a certain amount of uptime
- ◆ *Promises* access to critical functions of the system
- ◆ Allows system to handle faults or failures in a system
- ◆ Involves redundant components
- ◆ Allows component upgrades or maintenance without impacting availability

## ➤ What This Is Not

- ◆ Not a guarantee of 100% availability
- ◆ Not an inexpensive solution
- ◆ Not failover
- ◆ **Not immune to poor implementation**

# The Language of HA

- Uptime
  - ◆ Measure of the time a computer system has been “up” and running (does not imply availability)
- Availability
  - ◆ The proportion of time a system is **production capable**
- High Availability (HA)
  - ◆ System design protocol and associated implementation intended to ensure a certain degree of operational continuity during a given measurement period
- Single Point of Failure (SPOF)
  - ◆ Part of a system which, if it fails, will stop the entire system from working
- Fault Tolerance
  - ◆ The ability to continue properly when a hardware or software fault or failure occurs. Designed for reliability by building multiples of critical components like controllers, adapters, memory and disk drives.
- Redundancy
  - ◆ The duplication of components to ensure that should a primary resource fail, a secondary resources can take over its function

# And Now For The Parts

- Storage Controller
  - ◆ The control logic in a storage subsystem that performs, among other things, command transformation and routing, I/O prioritization, error recovery and performance optimization
- Fabric
  - ◆ Interconnection method that allows multiple hosts and/or storage devices connected with a multi-port hub, simultaneous and concurrent data transfers. Can be a single or multiple switches.
- Adapter
  - ◆ Circuit board that provides I/O processing and physical connectivity between a server and storage device
- Multipathing
  - ◆ The use of redundant storage networking components (adapters, cables, switches) responsible for the transfer of data between the server and the storage using redundant, and possibly load balanced, connections

# The Seven R's Of High Availability

- Redundancy
  - ◆ Eliminate single points of failure
- Reputation
  - ◆ What's the track record of the key suppliers in your solution?
- Reliability
  - ◆ How dependable are the components and coding of the products? What is the Mean Time Between Failures (MTBF) for components?
- Repairability
  - ◆ How quickly and easily can suppliers fix or replace failing parts? What is the Mean Time To Repair (MTTR) for failing components?
- Recoverability
  - ◆ Can your solution overcome a momentary failure and not impact users?
- Responsiveness
  - ◆ A sense of urgency is essential in all aspects of High Availability
- Robustness
  - ◆ Can your solution survive a variety of forces working against it?

# Five Nines...And Then Some

<b>Availability</b>	<b>Downtime/Yr</b>	<b>Downtime/Mo</b>	<b>Downtime/Wk</b>
90%	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
98%	7.30 days	14.4 hours	3.36 hours
99%	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes
99.9%	8.76 hours	43.2 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99%	52.6 minutes	4.32 minutes	1.01 minutes
99.999%	5.26 minutes	25.9 seconds	6.05 seconds
99.9999%	31.5 seconds	2.59 seconds	0.605 seconds



# It Costs To Be Gone!

- Business Revenue
  - ◆ Lost revenue opportunities, transaction business value
- Productivity
  - ◆ Application/Business users idle, support personnel/systems active
- Brand
  - ◆ Negative publicity (RIM outage), lost customers (existing and potential), supplier relationships
- Legal
  - ◆ Contractual obligations, late fees, etc

Business Operation	Downtime (Avg. Cost per Hour)
Brokerage operations	\$6.5 million
Credit card/sales auth.	\$2.6 million
PPV television	\$150 thousand
Home shopping (TV)	\$113 thousand
Catalog sales	\$90 thousand
Airline reservations	\$89.5 thousand

Source: META, February 2004



- Redundancy Based Fault Tolerance
  - ◆ Designed to ensure it takes two independent local faults, in a short time period, to cause end-to-end failure
- Even the best designs can be defeated
  - ◆ Undetected faults : single local fault not detected and addressed
  - ◆ Dependent faults : two faults assumed to be independent are actually related
  - ◆ Out-of-scope faults : additional categories of faults not addressed by the design

## ➤ Undetected Faults

- ◆ Single local fault not detected and addressed
- ◆ Complexity of SAN access paths
  - › Often hundreds or thousands of SAN access paths
  - › Many thousand SAN configuration changes, often undocumented
  - › Relationship between components often not understood well enough to ensure that redundant fault tolerance is adhered to

## ➤ Example

- ◆ Planned switch upgrade brings down critical business because a fault in the redundant access path was not detected
- ◆ Vendor merges have combined previously separate redundant paths into one when combined under new controller

## ➤ Dependent Faults

- ◆ Two faults assumed to be independent are actually dependent
- ◆ Operations often performed twice, once for each redundant system
  - › Operation error initiated in one system can be replicated to redundant system

## ➤ Example

- ◆ Application downtime caused by zoning errors made in one fabric repeated across redundant dual-fabrics
  - ◆ Best practices suggest a time period for component to stabilize after changes made before making them on redundant component also
- ◆ Switch reboot bug hitting both fabrics at once

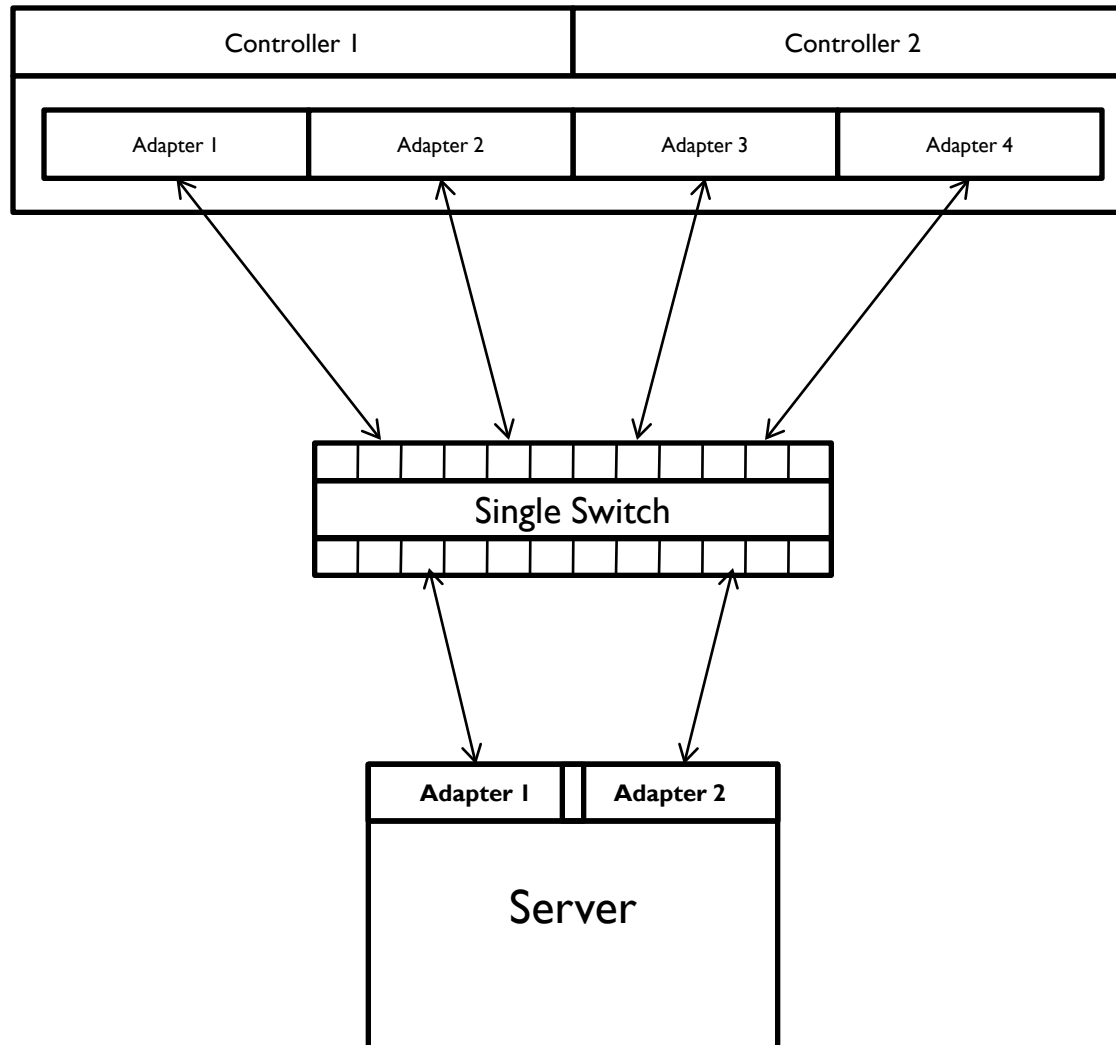
## ➤ Out-Of-Scope Faults

- ◆ Faults that were not anticipated in the original fault tolerant design
- ◆ Misconfigurations or failure to clean up old configurations

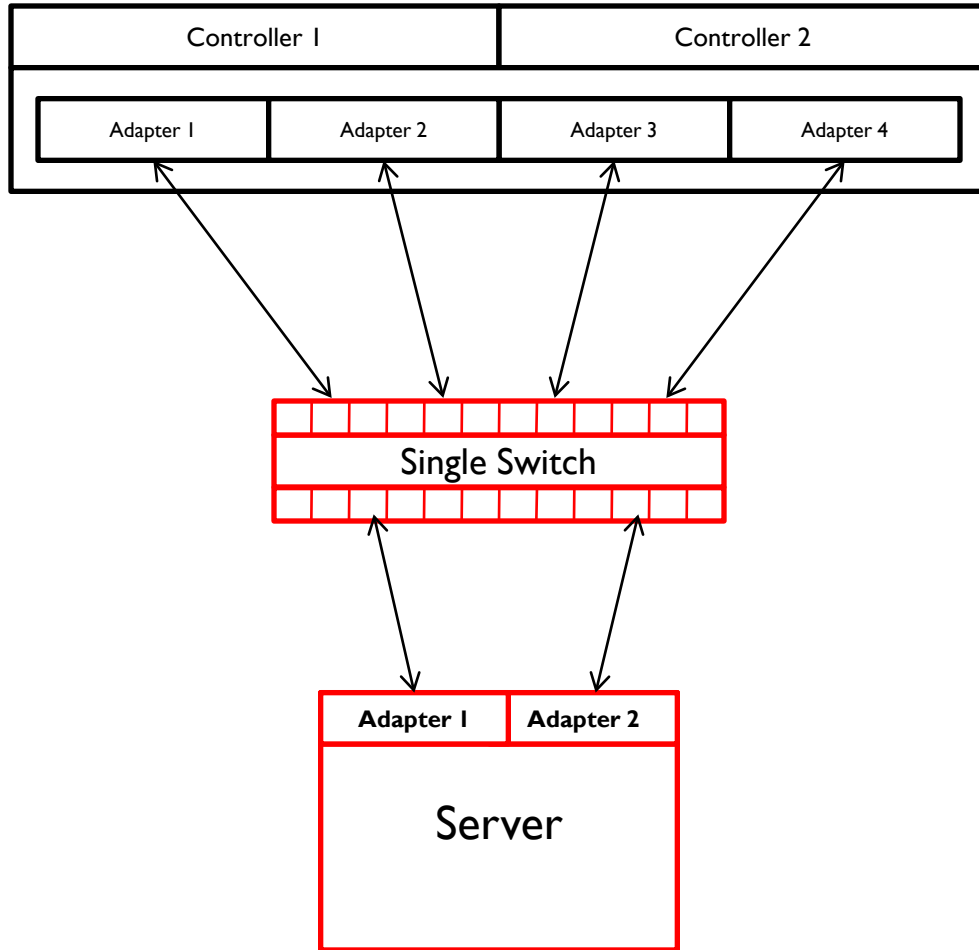
## ➤ Example

- ◆ LUN accidentally assigned to two different hosts, resulting in data corruption
- ◆ Reuse of an old HBA in a new server caused downtime because previous zonings using that HBA had not been cleaned up
- ◆ Uncontrolled changes, like an oversight of another scheduled change the user was not aware of
  - ◆ Process control important!!

# Looks Good...Feels Good...But...

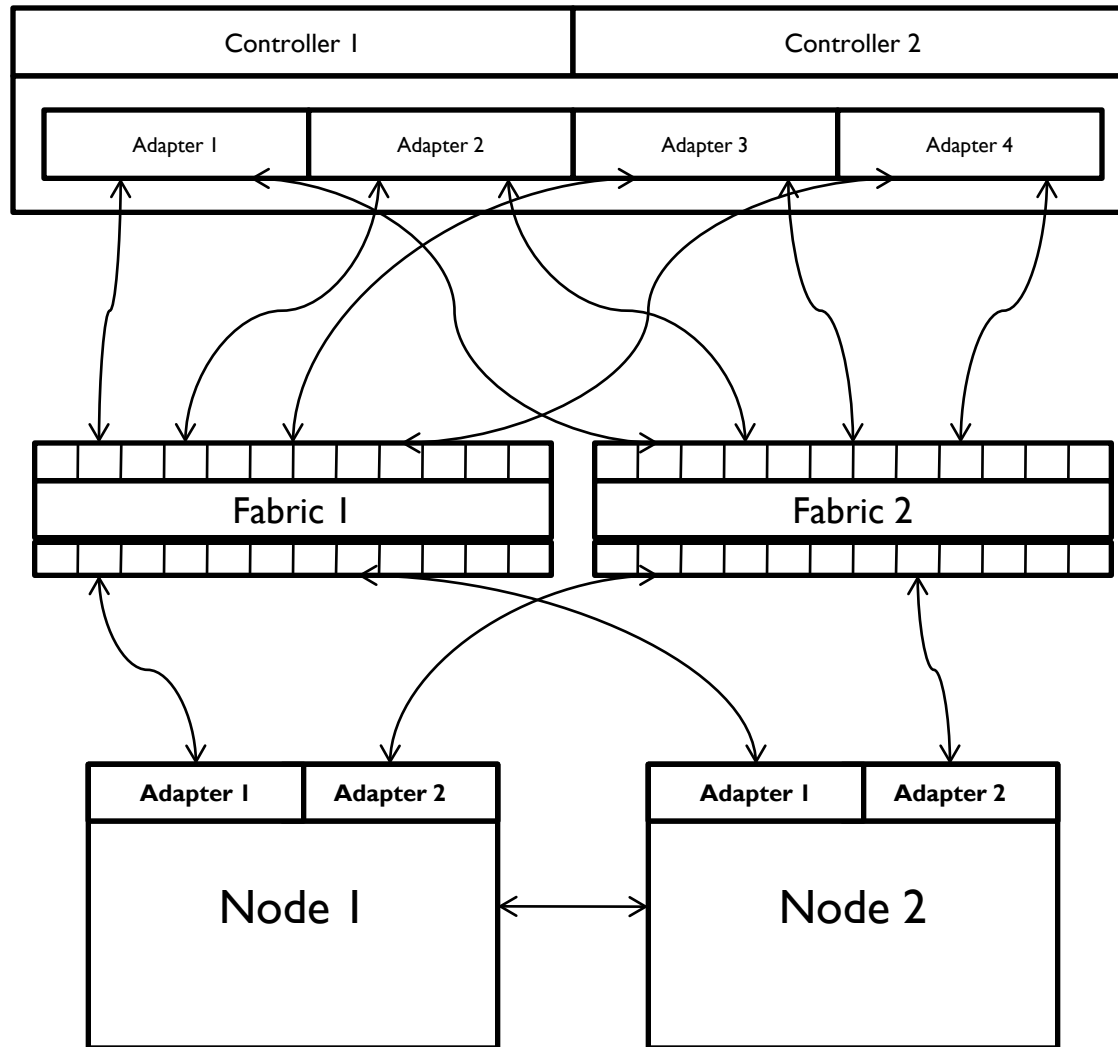


# You're Halfway There



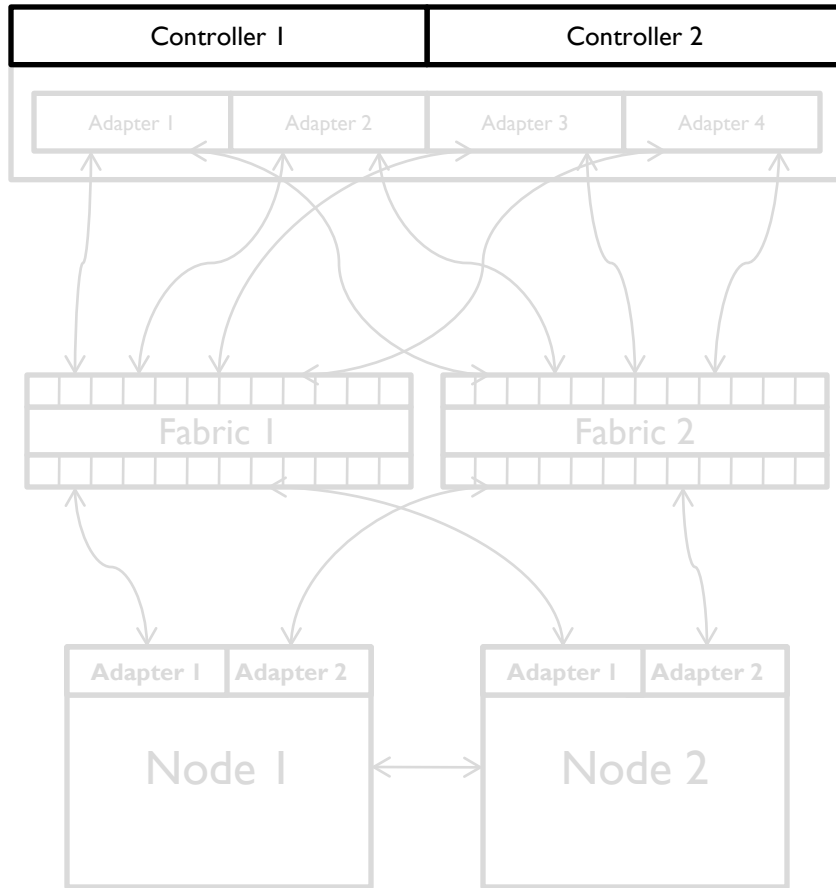
- Despite redundant paths and adapters, there are several single points of failure
- Fabric
  - ◆ Single point of failure with only one switch
  - ◆ Unable to take unit offline for maintenance, etc, without impacting users
- Server
  - ◆ Single server offers no redundancy
  - ◆ Unable to take unit down for maintenance or upgrade
  - ◆ Multiple paths out, but at mercy of system as a whole

# A “Simple” High Availability SAN



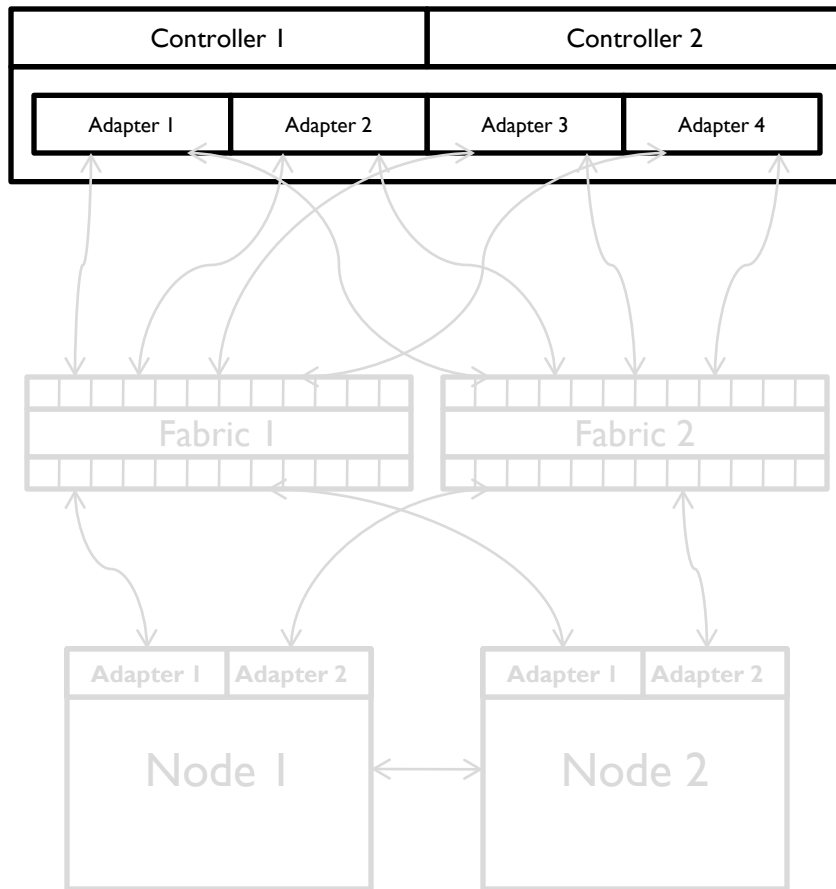


# Can't Have Too Much Control



- Storage Controller/Controller
  - ◆ Typical features
    - ◆ Redundant internal HDDs
    - ◆ Fault Tolerant Internal Fabric
    - ◆ Hot swappable components
    - ◆ Predictive failure analysis
  - ◆ Redundant controllers allow for
    - ◆ Scheduled maintenance
    - ◆ Single controller faults
    - ◆ System upgrades
  - ◆ Active/Active vs. Active/Passive
    - ◆ May be a delay in LUN ownership to be moved, which might affect I/O
  - ◆ Single controller boxes can result in downtime if they experience a fault
  - ◆ Can single controller handle workload?

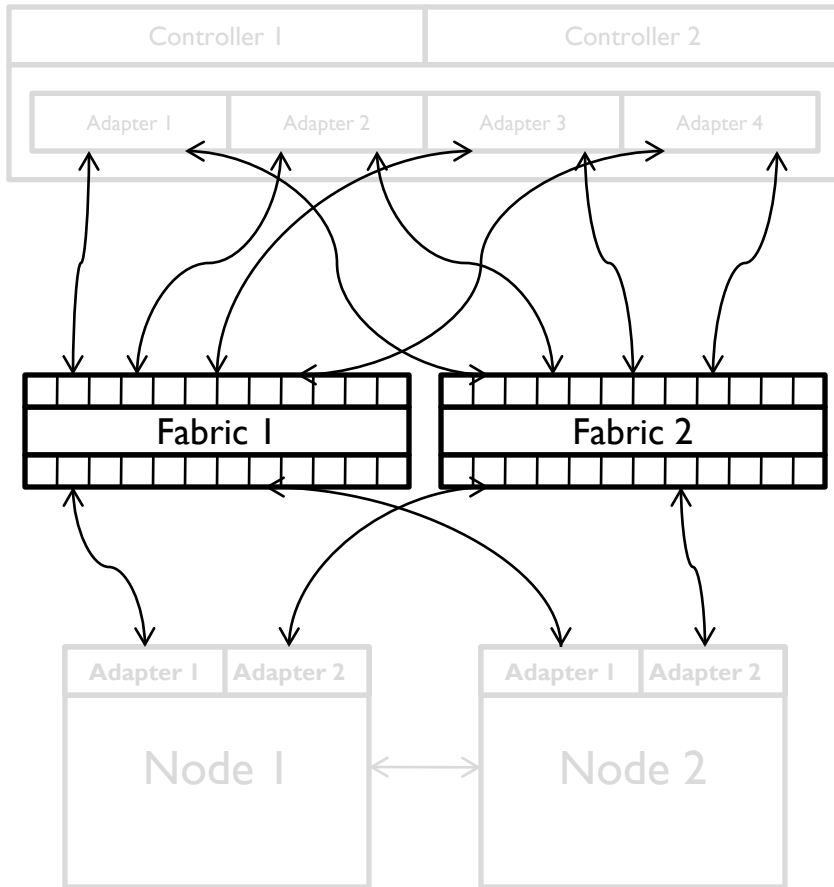
# Like A Six Lane Highway



## ➤ Adapters

- ◆ Multiple adapters allow redundant access and often load balancing
- ◆ In a good fault tolerant system, the adapters can be managed by either controller, providing greater availability

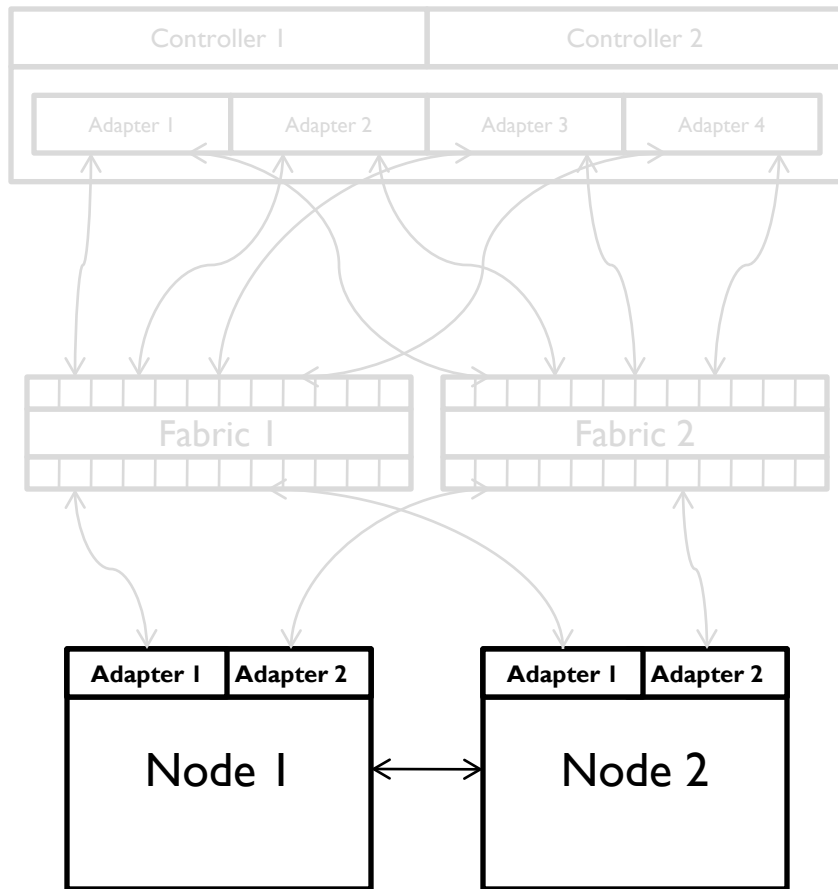
# Are We In “The Matrix?”



## ➤ Fabric

- ◆ Fabric redundancy allows for fault tolerance as well as multiple paths for delivering I/O
- ◆ Need at least two switches
- ◆ Multipathing
  - ◆ When path fails, it's disabled and I/O routed to other paths
  - ◆ Active/Active or Active/Passive
  - ◆ Load balancing

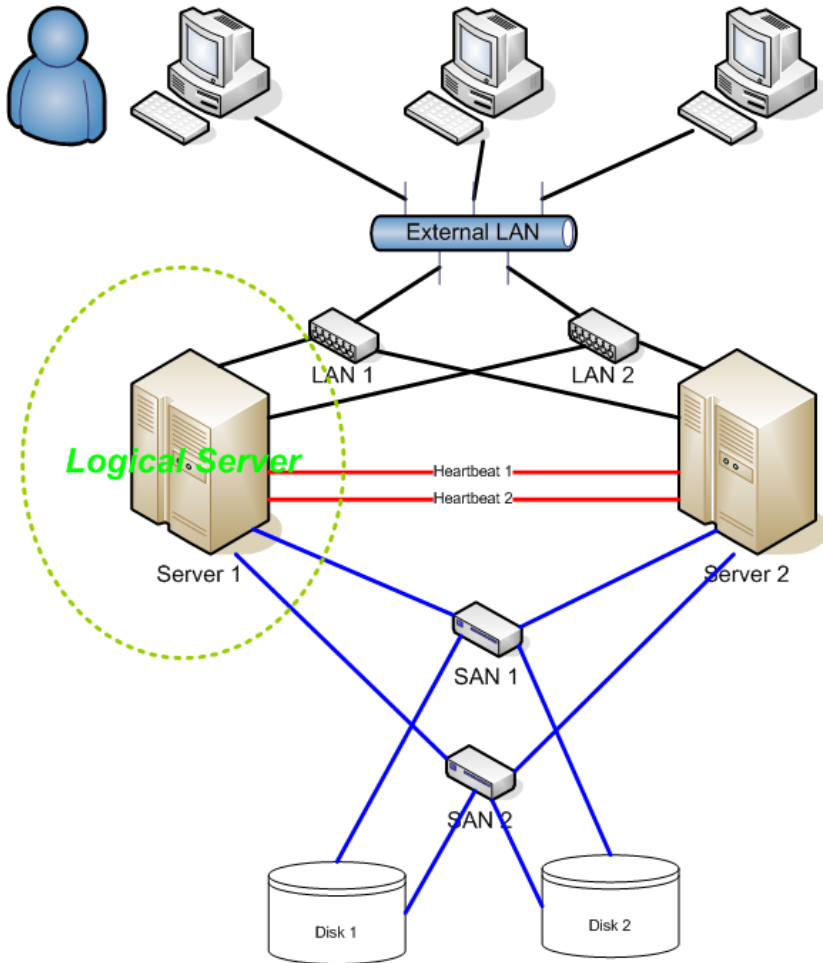
# All The User Really Cares About



## ➤ End Servers

- ◆ There are different ways to provide fault tolerance on end servers
  - ◆ Configure a high availability cluster
  - ◆ Multiple adapters in each server
  - ◆ Fault tolerant hardware configuration on each servers (disks, CPUs, etc)
  - ◆ Application HA

# High Availability Failover Cluster



## ➤ Server-side Protection

- ◆ Ensure service by providing redundant nodes
- ◆ Builds redundancy through multiple network connections and multiple SAN connections
- ◆ Detects hardware or software faults on one node, restarts application on alternate node
- ◆ Minimum two nodes per cluster, but can scale depending on vendor
- ◆ Active/Active
  - I/O for failed node passed on to surviving nodes, or balanced across remaining
- ◆ Active/Standby
  - Redundant node brought online only when a failure occurs

# HA And Virtualization

- Non-virtualized environments
  - ◆ HA typically only applied to environments hosting critical workloads
    - ◆ Loss of single server in most cases acceptable (though changing with economy resulting in trimming)
- Virtualized environments
  - ◆ Companies moving to virtualized environments
    - ◆ Single server hosting multiple virtual machines
  - ◆ Loss of single server will result in loss of multiple workloads
  - ◆ HA should be focus of any virtualized environment, regardless of scale or scope, especially in a production environment



## Check out SNIA Tutorial:

Storage Virtualization I - What, Why, When, Where, How? (Rob Peglar)

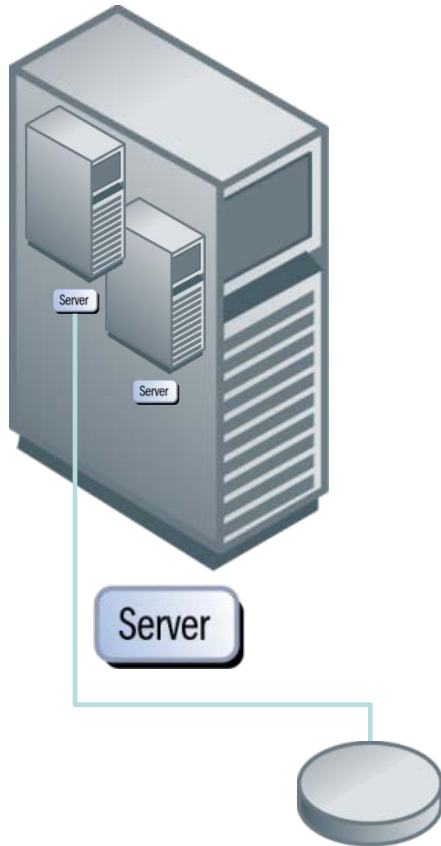


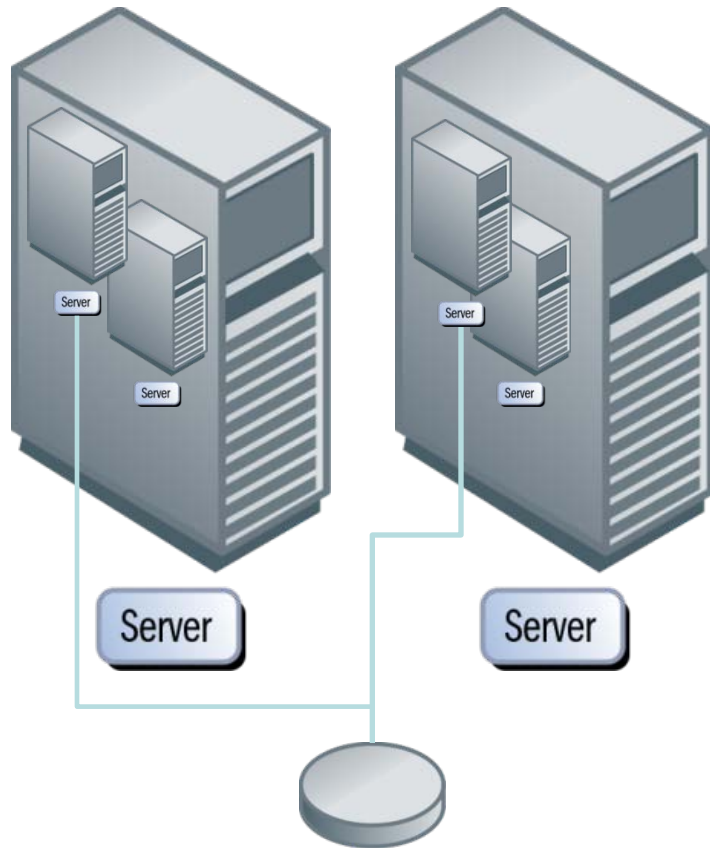
## Check out SNIA Tutorial:

Storage Virtualization II - Implementing and Managing Storage Virtualization (Rob Peglar)

## ➤ Cluster Guest VMs

- ◆ By clustering guest VMs you help protect applications in the event that one of the VMs goes down
- ◆ Single point of failure is the host server itself. If that goes down, everything is lost
- ◆ Good first step if you don't have enough hardware to cluster host server itself





## ➤ Cluster Guest Hosts/Guests

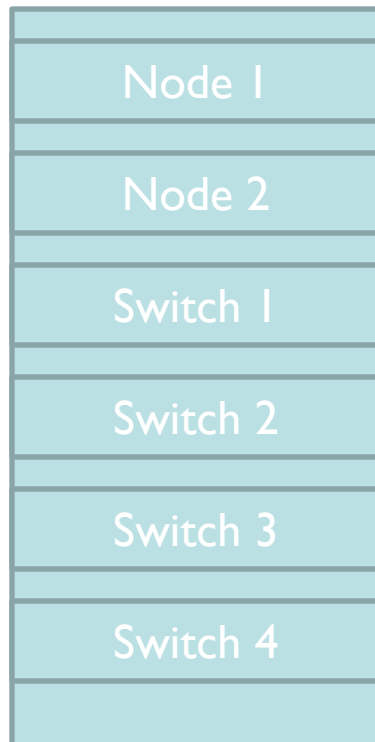
- ◆ By clustering guest VMs you help protect applications in the event that one of the VMs goes down
- ◆ By clustering hosts, you help protect the guest VMs that carry your applications
- ◆ Removes single point of failure of the host
- ◆ VMs can be clustered on host server itself, and can also be clustered across host servers



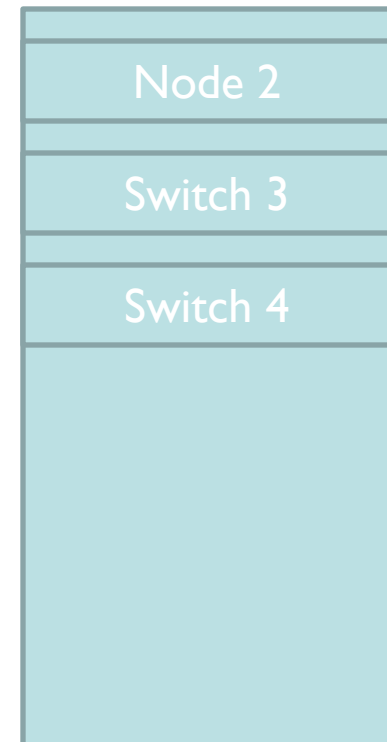
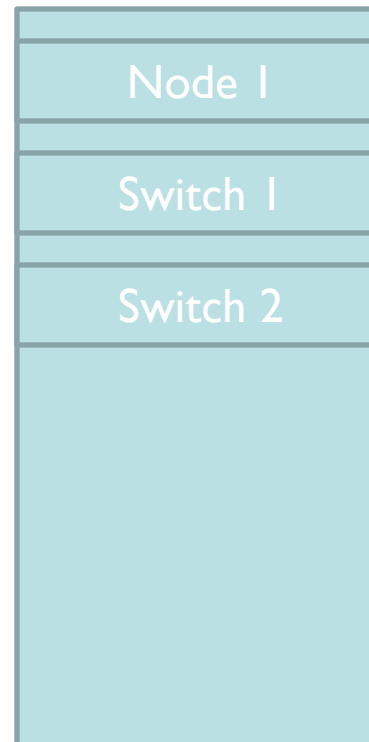
## ➤ HA Planning and Capacity Considerations

- ◆ Large, robust HA environments brought to its knees
  - ◆ Surviving components did not have enough bandwidth to handle additional workload
- ◆ Most capacity and performance planning assume all SAN paths and fabrics are available
- ◆ The best storage planners take into consideration what happens if one controller in a pair fails and all I/O gets directed to surviving controller
- ◆ Ensure controllers, paths, fabrics, adapters, servers, applications are all scaled to handle the entire load if necessary

➤ Which is better?



**vs.**



- “There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know.”
  - ◆ High Level Pentagon Official, February 12<sup>th</sup>, 2004 DOD News Briefing
- Good summary of what HA testing and configuration involves!

- High Availability configuring/testing is hugely complex
  - ◆ Multiple components from multiple vendors
  - ◆ Only subsets of entire solution often tested together
  - ◆ Portion may be officially supported, but entire configuration is not
  - ◆ Test matrix is absolutely huge, riddled with holes, requiring SMEs
- Knowns vs. Unknowns
  - ◆ There are often known problems in a configuration, but only to SMEs, and only for those components
  - ◆ Unknown problems exist around every corner, if configuration components have not been tested together
  - ◆ Details of configuration rules not known to everybody
- Everybody Getting Too Smart
  - ◆ As more products bring their version of “self-healing,” integration issues increase
  - ◆ For any given single point of failure, only one product should be responsible for protecting configuration

## ➤ Somebody's Gotta Win!

- ◆ Clustering configured to stop/restart services when outage on node discovered
- ◆ DBA takes down node for routine maintenance
- ◆ Clustering takes down DB and filesystems, then restarts applications
- ◆ DBA confused
- ◆ DBA tries again
- ◆ Cycle repeats until clustering gives up, cancels future checking
- ◆ DBA able to perform maintenance actions
- ◆ Clustering has disabled future checking, now entire cluster exposed!

## ➤ Triggers and Inhibitors

- ◆ In 1+1 redundancy, a failure sometimes implies two failures
  - ◆ Trigger Event – first failure that triggers the outage
  - ◆ Inhibitor Factor – combines with trigger to cause second failure
- ◆ These two events combine to fail redundancy
- ◆ Sometimes inhibitor is already there, lying dormant
  - ◆ Transforms simple trigger event into an outage
- ◆ Redundancy sometimes hides failures in the system, which keeps them from being discovered. Those become inhibitor factors for future failures.

# Vital Lessons

## ➤ Planning is **essential**

- ◆ All parties together, make requirements clear
- ◆ Good advice, pick right components for the solution
- ◆ Single points of failure can exist outside production servers

## ➤ Testing is **vital**

- ◆ Test everything, even what you think will work
  - ◆ Maintain an active test environment, separate from production environment
- ◆ Expect problems with initial test, allow time to fix
- ◆ Periodic testing after solution live to ensure nothing has broken

## ➤ Manage **change**

- ◆ New requirements may cause problems with other parts of solution
- ◆ Process is super important in managing change
- ◆ Many hands do not always make light work

## ➤ Document **everything**

- ◆ Document the plan, stick to it
- ◆ Document the testing to show how it worked. If the results change, something in the solution has changed.

## ➤ Configure for **redundancy**

- ◆ Use proven systems for servers
- ◆ Make servers resilient by providing for better cooling, redundant power supplies, etc
- ◆ Redundant network paths for cluster heartbeats
- ◆ Implement multipathing to shared storage
- ◆ Mirror storage volumes



## ➤ High Availability Using Fault Tolerance

- ◆ Requires at least two of each component
- ◆ Increases system complexity
- ◆ Increases administrative responsibility
- ◆ Increases capital expenditures
- ◆ Can be incorrectly configured

## ➤ Questions You Should Ask

- ◆ Is it worth it?
- ◆ How important is your data to you?
- ◆ How much downtime can your business stand?
- ◆ What is acceptable recovery time?

- Please send any questions or comments on this presentation to SNIA: [trackstoragemgmt@snia.org](mailto:trackstoragemgmt@snia.org)

**Many thanks to the following individuals  
for their contributions to this tutorial.**

**- SNIA Education Committee**

**Mark S Fleming**

**Lisa Martinez**

**Brad Powers Robert Jodoin**

**Jeffry Larson Dan Braden**

**David Spurway**

**Mark E Johnson**

**Stephanie Lengelle**

**Anthony Vandewerdt**

**Frans G Versteeg**

**David R Nicholson**

**Janet Hamilton**

**Doug Brigance**

## ➤ Resources

- ◆ [http://en.wikipedia.org/wiki/Graceful\\_degradation](http://en.wikipedia.org/wiki/Graceful_degradation)
- ◆ [http://en.wikipedia.org/wiki/Redundancy\\_%28engineering%29](http://en.wikipedia.org/wiki/Redundancy_%28engineering%29)
- ◆ [http://en.wikipedia.org/wiki/High\\_availability](http://en.wikipedia.org/wiki/High_availability)
- ◆ [http://searchstorage.techtarget.com/news/article/0,289142,sid5\\_gci1080870,00.html#](http://searchstorage.techtarget.com/news/article/0,289142,sid5_gci1080870,00.html#)
- ◆ [http://en.wikipedia.org/wiki/Myth\\_of\\_the\\_nines](http://en.wikipedia.org/wiki/Myth_of_the_nines)
- ◆ <http://www.informit.com/articles/article.aspx?p=28299&seqNum=3>
- ◆ [http://en.wikipedia.org/wiki/High-availability\\_cluster](http://en.wikipedia.org/wiki/High-availability_cluster)