ECE568
Engineering Robust Server Software

Spring 2023

Business Continuity: High Availability

Brian Rogers (slides courtesy of Tyler Bletsch)
Duke University

Includes material adapted from the course "Information Storage and Management v2" (modules 9-12), published by EMC corporation.

What is Business Continuity?

Business Continuity

It is a process that prepares for, responds to, and recovers from a system outage that can adversely affects business operations.

- An integrated and enterprise-wide process that includes set of activities to ensure "information availability"
- BC involves proactive measures and reactive countermeasures
- In a virtualized environment, BC solutions need to protect both physical and virtualized resources

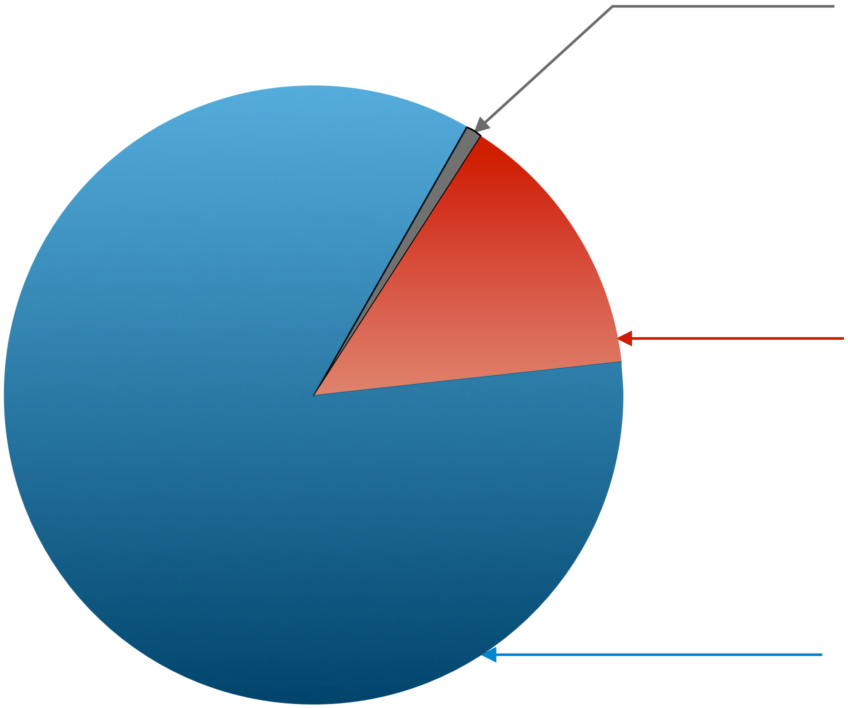
Information Availability

Information Availability

It is the ability of an IT infrastructure to function according to business expectations, during its specified time of operation.

- Information availability can be defined with the help of:
  - **Accessibility**
    - Information should be accessible to the right user when required
  - **Reliability**
    - Information should be reliable and correct in all aspects
  - **Timeliness**
    - Defines the time window during which information must be accessible

Causes of Information Unavailability



Disaster (<1% of Occurrences)

Natural or man made

- Flood
- Fire
- Earthquake

Unplanned Outages (20%)

Failure

- Database corruption
- Component (physical and/or virtual) failure
- Human error

Planned Outages (80%)

Competing workloads

- Backup, reporting
- Data warehouse extracts
- Application and data restore

Impact of Downtime

Lost Productivity

- Number of employees impacted x hours out x hourly rate

Know the downtime costs (per hour, day, two days, and so on.)

Lost Revenue

- Direct loss
- Compensatory payments
- Lost future revenue
- Billing losses
- Investment losses

Damaged Reputation

- Customers
- Suppliers
- Financial markets
- Banks
- Business partners

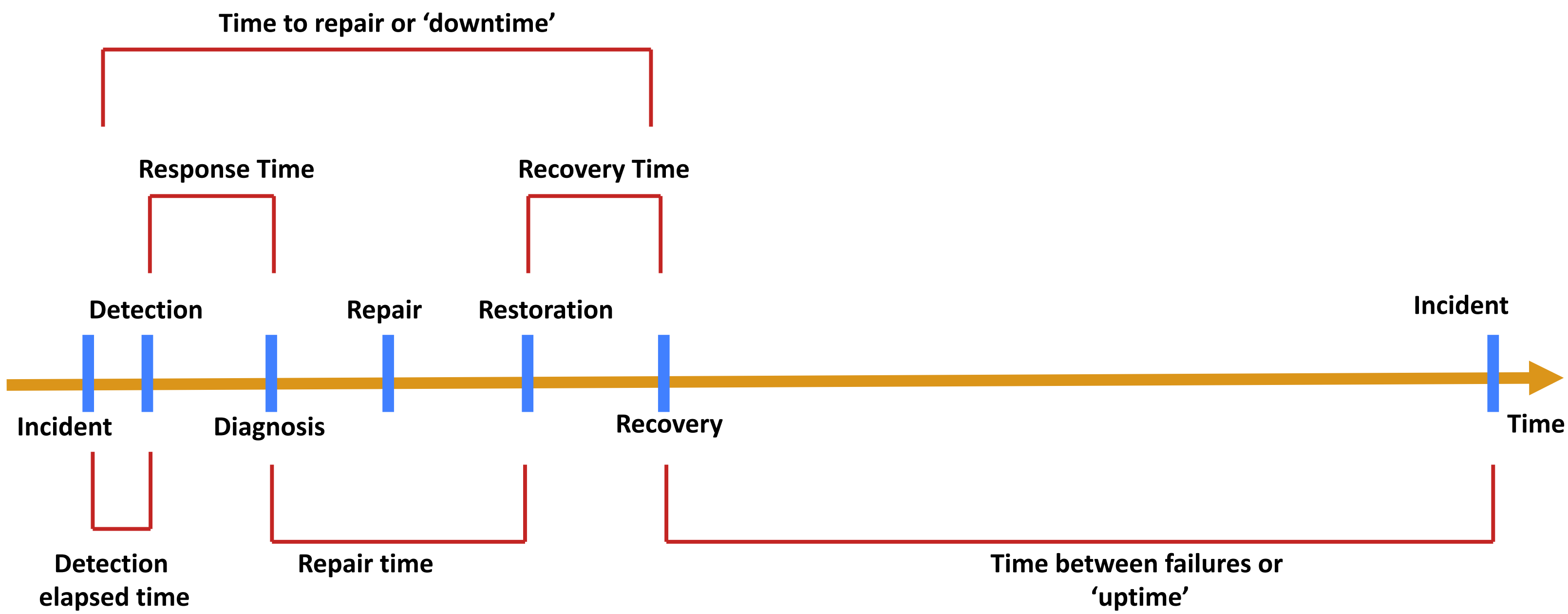
Financial Performance

- Revenue recognition
- Cash flow
- Lost discounts (A/P)
- Payment guarantees
- Credit rating
- Stock price

Other Expenses

- Temporary employees, equipment rental, overtime costs, extra shipping costs, travel expenses, and so on.

Measuring Information Availability



- MTBF: Average time available for a system or component to perform its normal operations between failures

  MTBF = Total uptime/Number of failures

- MTTR: Average time required to repair a failed component

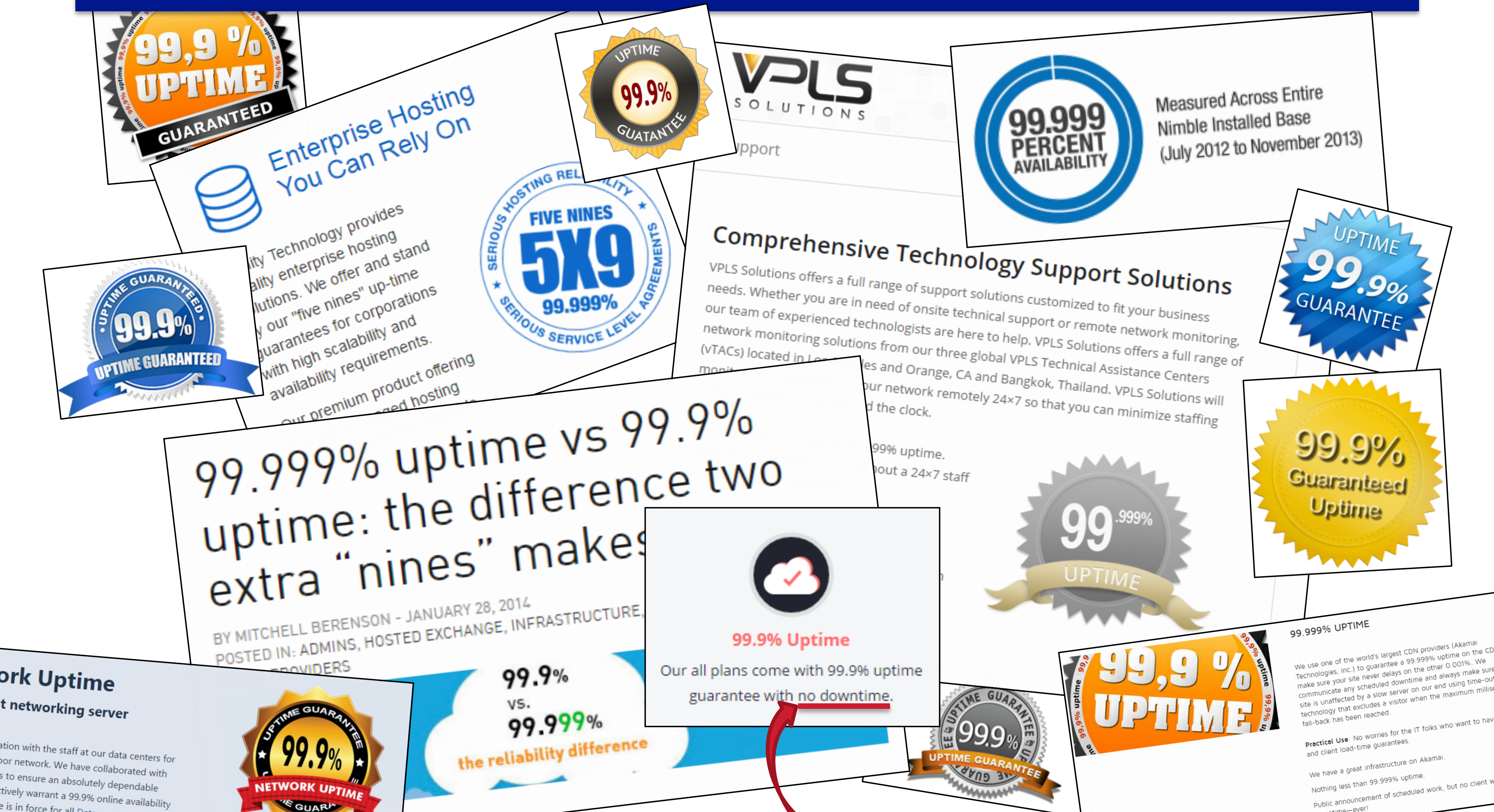
  MTTR = Total downtime/Number of failures

IA = MTBF/(MTBF + MTTR) or IA = uptime/(uptime + downtime)

Availability Measurement – Levels of '9s' Availability

Uptime (%)	Downtime (%)	Downtime per Year	Downtime per Week
98	2	7.3 days	3hrs, 22 minutes
99	1	3.65 days	1 hr, 41 minutes
99.8	0.2	17 hrs, 31 minutes	20 minutes, 10 secs
99.9	0.1	8 hrs, 45 minutes	10 minutes, 5 secs
99.99	0.01	52.5 minutes	1 minute
99.999	0.001	5.25 minutes	6 secs
99.9999	0.0001	31.5 secs	0.6 secs

Vendors love to brag about their nines



Even when they don’t actually know what that means...

Maximizing availability

- Two complementary approaches:

- **High Availability (HA)**
  - Design systems so that failures do not result in *any* downtime
  - Keys: **redundancy** and **failover**
  - "No single point of failure"

- **Disaster Recovery (DR)**
  - If the HA techniques are overwhelmed (e.g. due to a site failure, major human error, etc.), be able to recover data and restore functionality
  - Keys: **replication** and **restore/failover**
  - "Survive the inevitable multiple failure"

High Availability (HA)

Redundancy

- Core HA concept:
  - Identify **single points of failure**
  - Add redundancy to eliminate
  - Need policy for how to interface with redundant system

Active/active vs. active/passive

- **Active/active**: Both redundant components used in normal operation; symmetric design
  - + Higher utilization and capacity/performance
  - – Capacity/performance is reduced on failure

- **Active/passive**: A "primary" and "secondary" system; secondary only does work if primary fails; asymmetric design
  - + Failures don't affect capacity/performance
  - – Half the hardware is idle most of its life (low utilization)

The split brain problem

- Imagine an active/passive system
- What if the two redundant systems lose contact with each other, and each thinks its time to “take over”?
- Both are serving traffic and issuing commands!
- Result: chaos!

- Redundant computer systems must have protocol to govern takeover

Layers on which to apply redundancy

Power

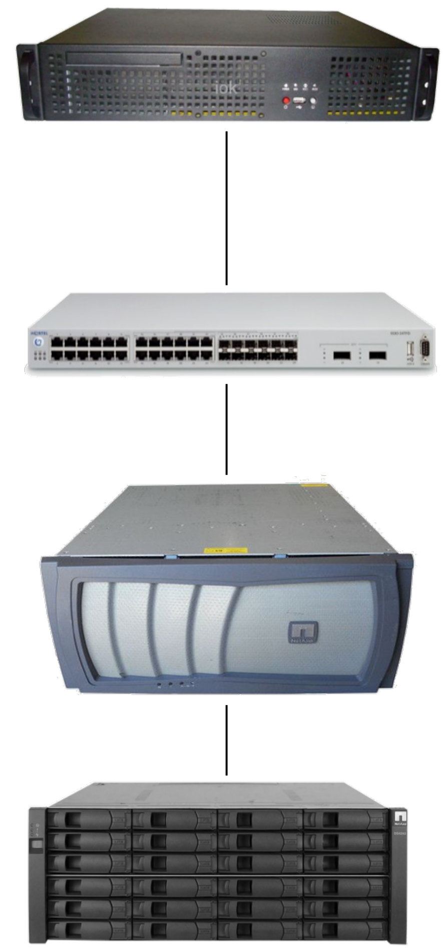
Cabling

Servers

Network switch

Storage controller

Disks



First, let's deal with power

- Everything has 2+ power supplies
  - Equipment can survive with half its power supplies dead
  - This protects against power supply failure
- Power comes from Power Distribution Units (PDUs) (basically rackmount power strips)
- HA power: Racks have two PDUs.
  - PDU 1 hooked to "left" power supply, PDU 2 hooked to "right" power supply
- Power supplies usually hot-swappable
  - Replace on fault without downtime

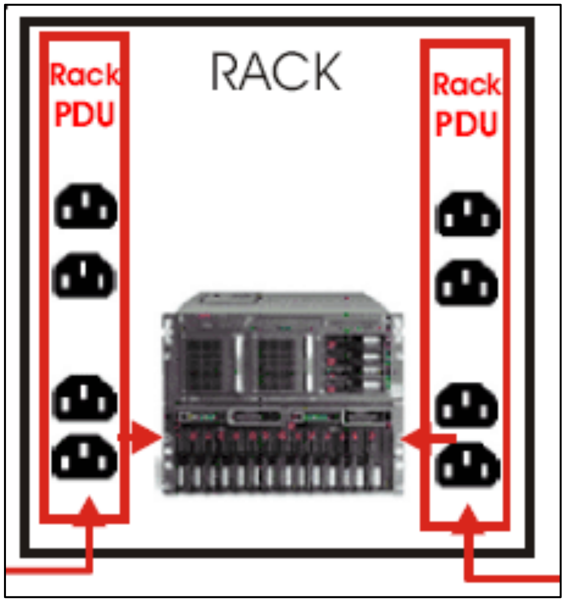
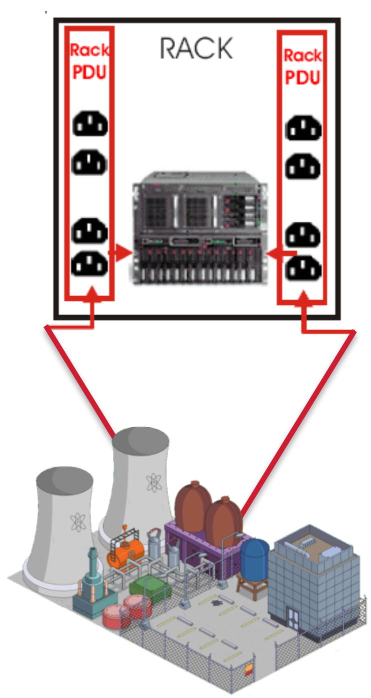


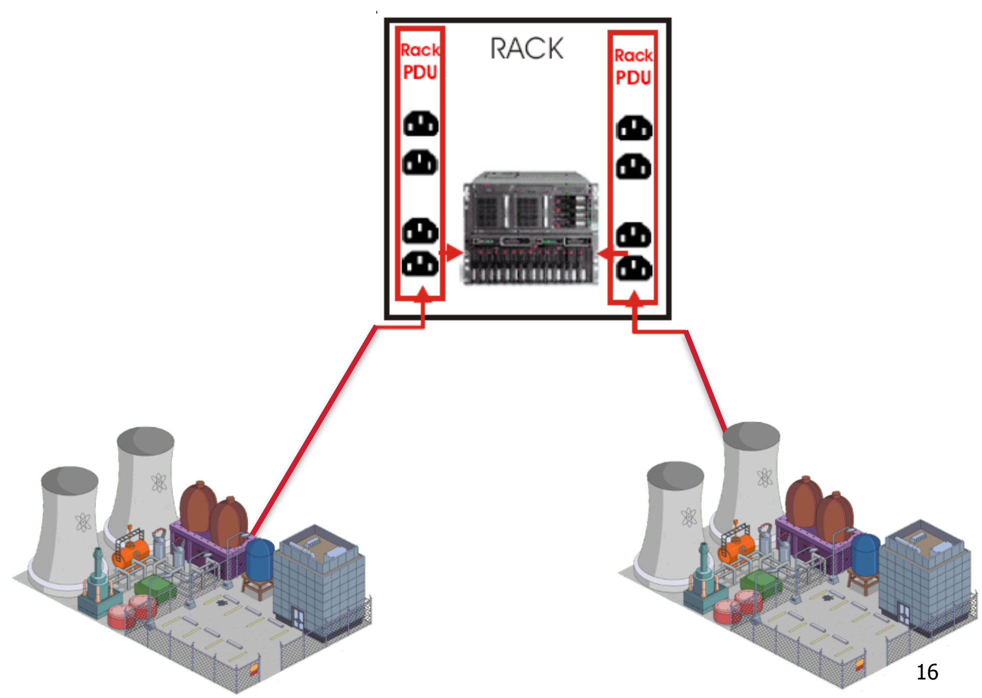
Figure from "Do dual-power supply servers increase redundancy?", TechTarget. Julius Neudorfer.

Additional power redundancy options omitted for time;
for details, take my ECE566, Enterprise Storage Architecture

Utility power

- Single-feed environment: both strips get power from same utility
  - Power outage? All gear goes down.
  - Still protects against accidental disconnect, power supply failure, local tripped breaker
- Double-feed environment: two separate feeds from two separate power sub-stations
  - Also protects against utility power outage
  - Might even draw from two different power plants!





UPS: Uninterruptable Power Supply

- UPS: Uninterruptable Power Supply
  - Takes AC power in, gives AC power out
  - Keeps a big battery array charged
  - If AC power-in fails, AC power-out comes from battery array without interruption
    - DC power from batteries must be converted to AC with an inverter
  - Rated by **battery capacity (total energy)** and inverter **current capability (max power)**
- Use cases
  - Smooth power "**blips**" (momentary interruptions that would reboot everything)
  - Keep things running for a few **minutes**, long enough for graceful shutdown
  - Run things for a few **hours**
  - Keep things running long enough to start a **gasoline/diesel generator**

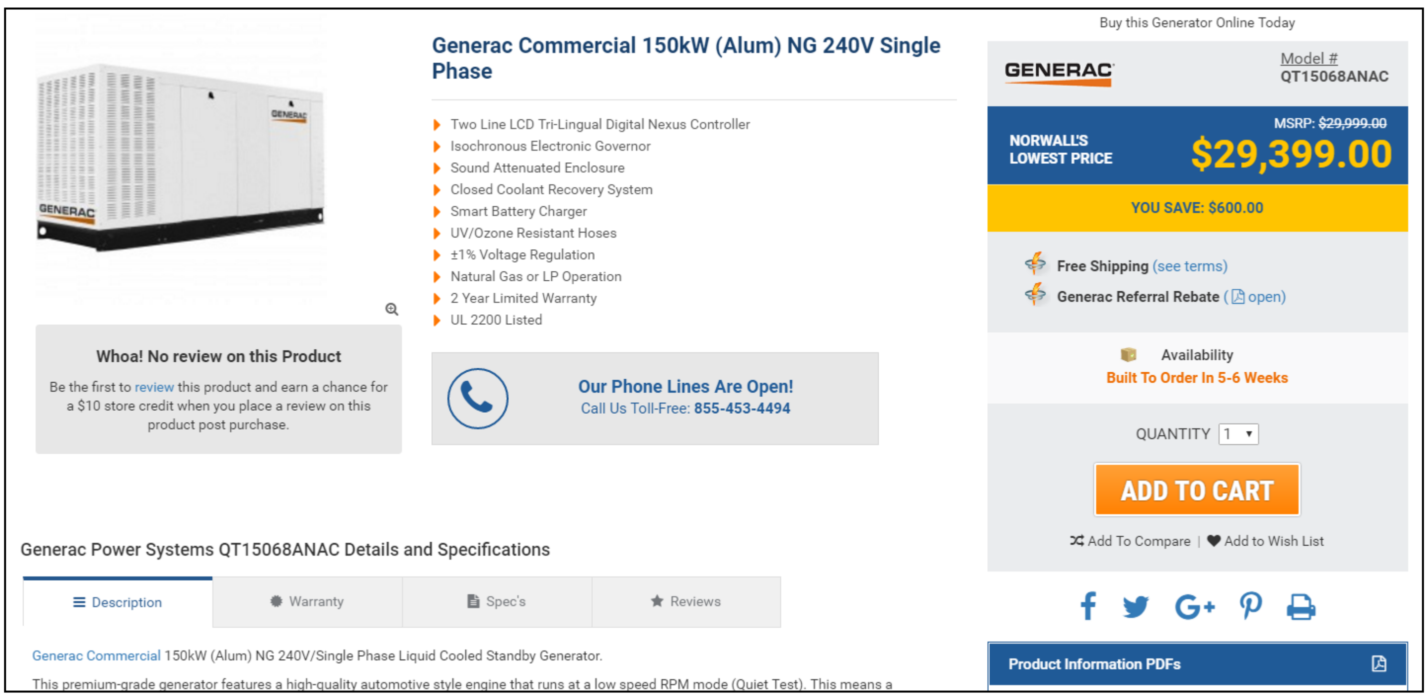
Small consumer UPS

Rackmount UPS

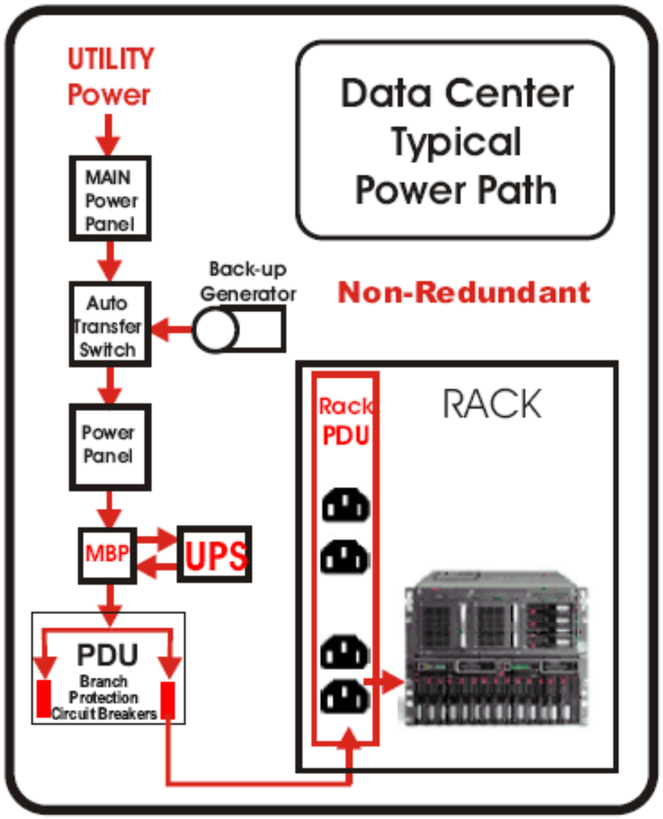
Building-scale battery array

Electric generators

- Need to survive long-term power loss?
- **Gasoline or diesel generator**
- Typically sized for whole or part of data center
- Large fuel tank on site, run time for days
- Can contract to have additional fuel brought during extended emergency
  - If fuel can't be brought, society is probably broken enough that it's okay your server isn't up...



Total redundant power picture



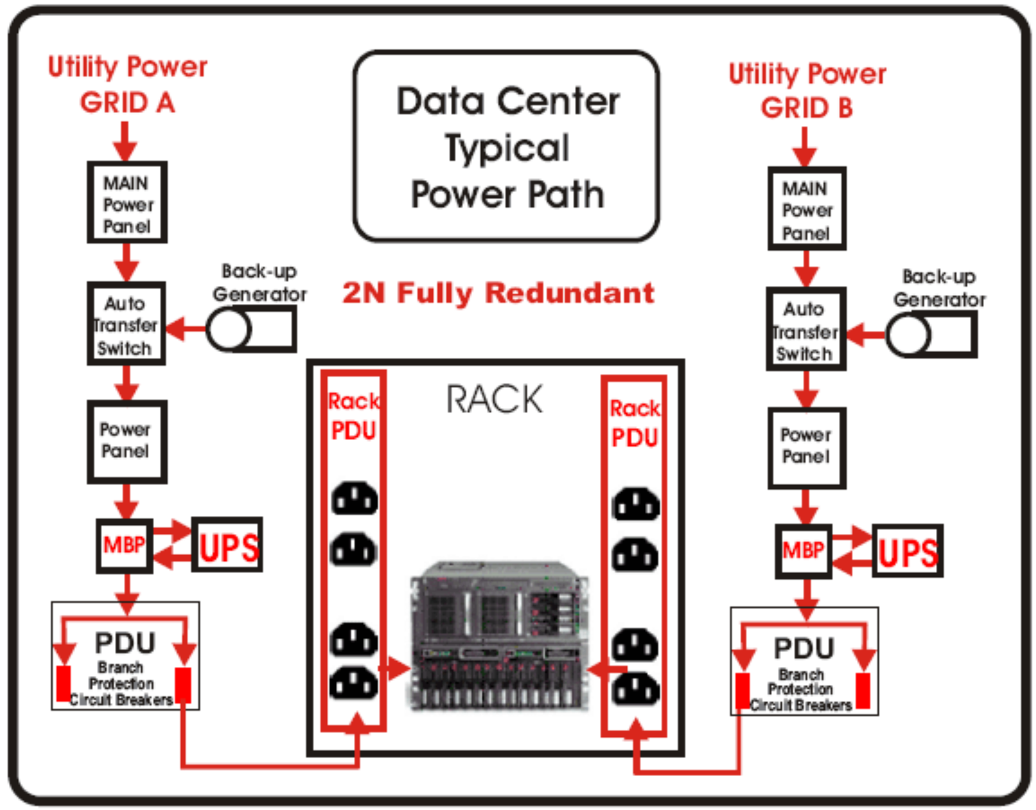


Figure from "Do dual-power supply servers increase redundancy?", TechTarget. Julius Neudorfer.

Layers on which to apply redundancy

Power

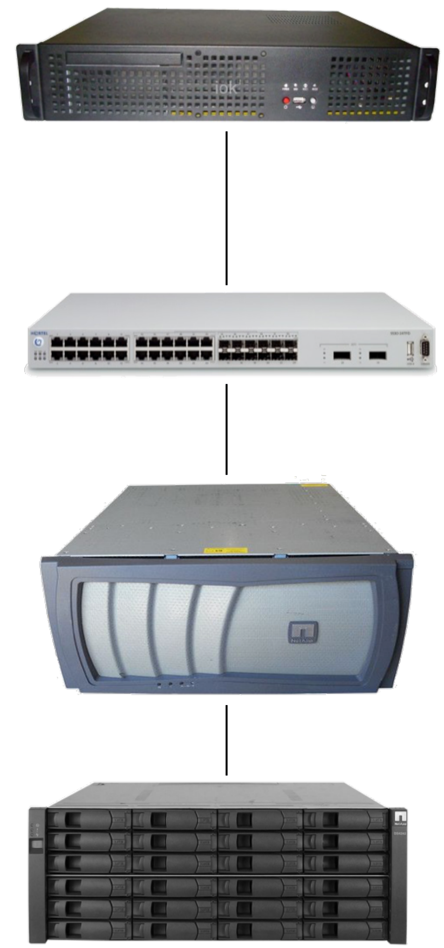
Cabling

Servers

Network switch

Storage controller

Disks

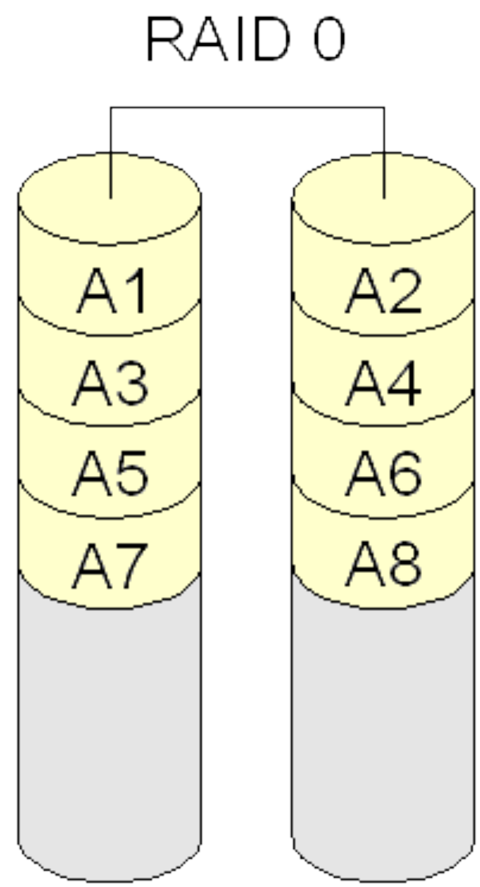


Disk-level redundancy RAID

- **RAID (Redundant Array of Inexpensive Disks)**
- Two RAID aspects taken into consideration:
  - **Data striping** : leads to enhanced bandwidth
    - Distributes data transparently over multiple disks
    - Appears as a single fast large disk
    - Allows multiple I/Os to happen in parallel.
  - **Data redundancy** : leads to enhanced reliability
    - Mirroring, parity, or other encodings

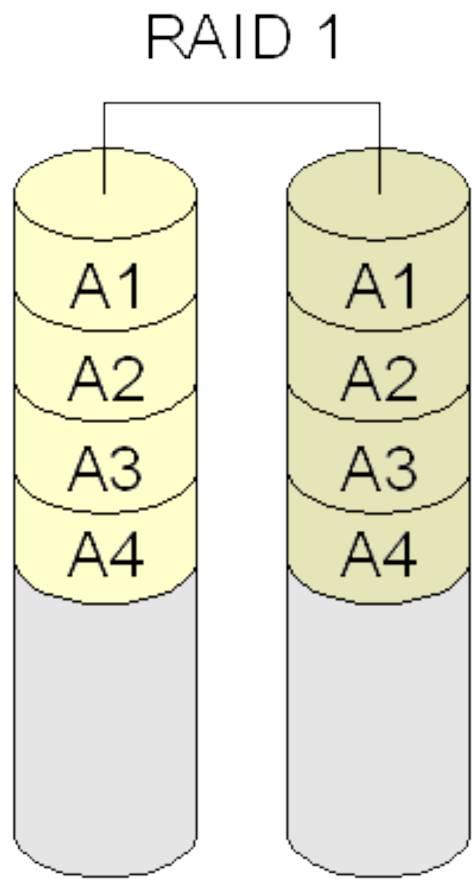
RAID 0 ("Striping")

- Non-redundant
  - Stripe across multiple disks
  - Increases throughput
- Advantages
  - High transfer
  - Cost
- Disadvantage
  - No redundancy
  - Higher failure rate



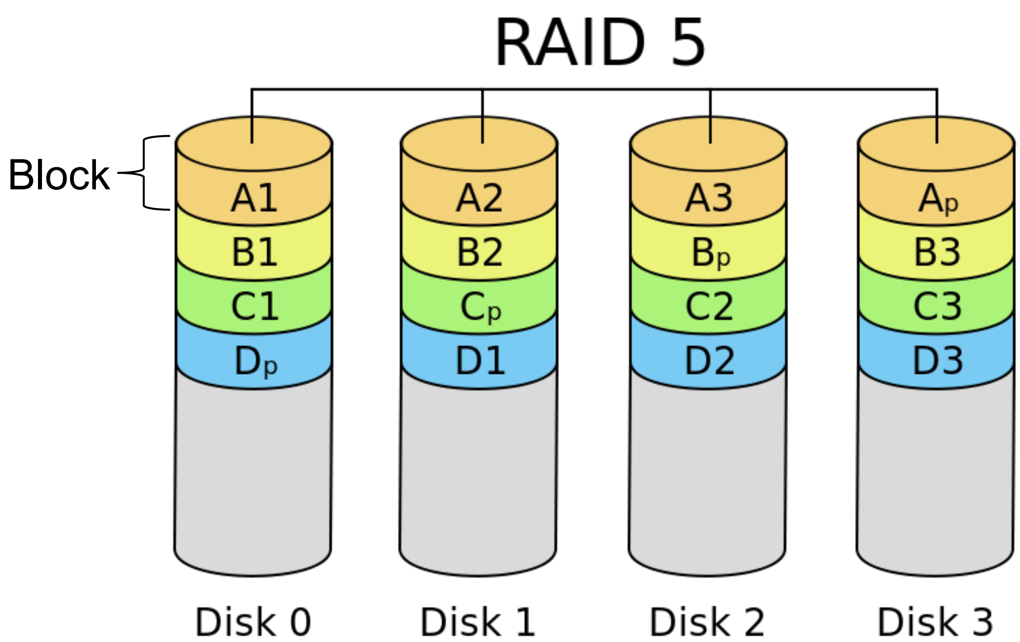
RAID 1 (“Mirroring”)

- Mirroring
  - Two copies of each disk block
- Advantage
  - Simple to implement
  - Fault-tolerant
- Disadvantage
  - Requires twice the disk capacity



RAID 5 (“Distributed parity”)

- For each stripe, a drive stores it’s parity (XOR)
- Can lose ANY drive, and using parity, restore its data
- Parity is evenly distributed across drives,
  so two independent writes will usually engage two separate sets of disks.



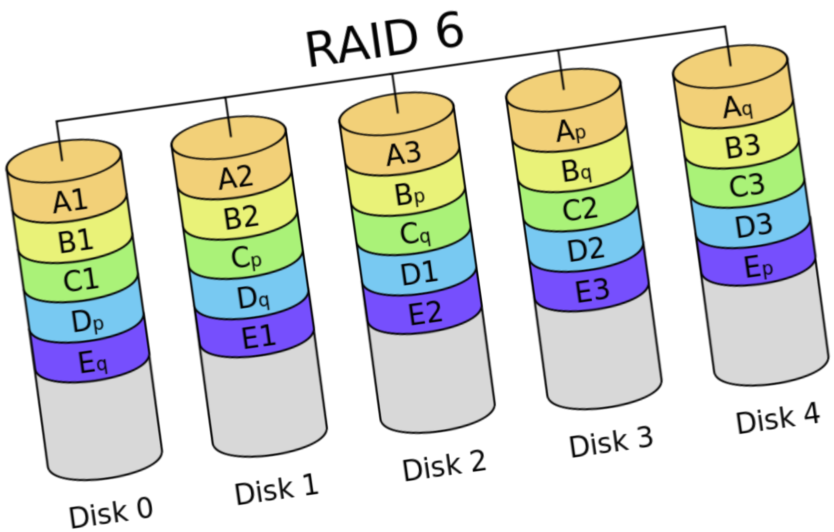
XOR parity demo

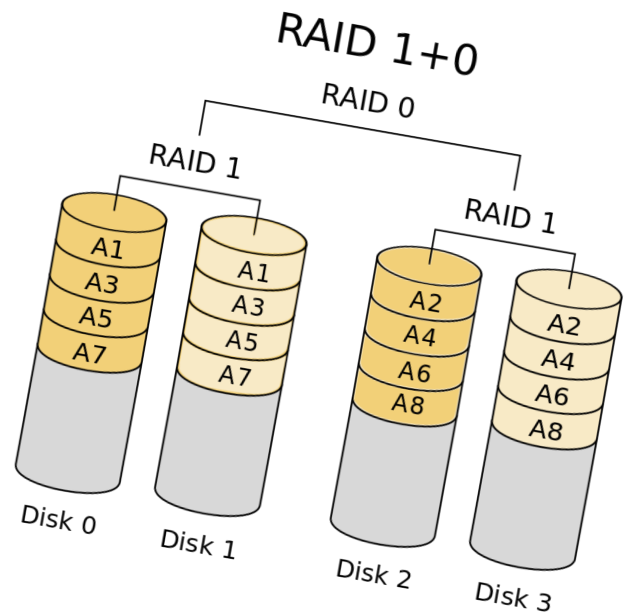
- Given four 4-bit numbers: [0011, 0100, 1001, 0101]

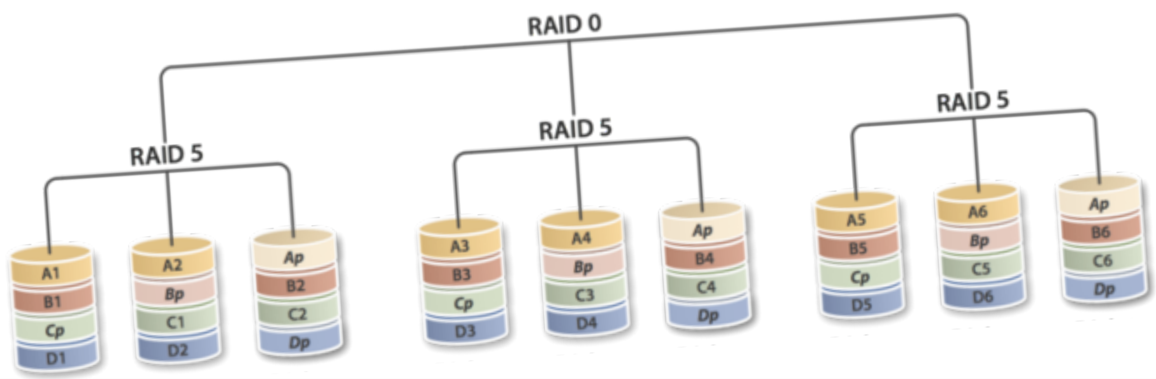
<u>XOR them</u>	<u>Lose one and XOR what's left</u>
~~0011~~	1011
0100	0100
1001	1001
⊕ 0101	⊕ 0101
1011	0011
	Recovered!

- Given N values and one parity,
  can recover the loss of *any* of the values

More RAID levels exist...







Want to know more?
Take my ECE566, Enterprise Storage Architecture

Layers on which to apply redundancy

Power

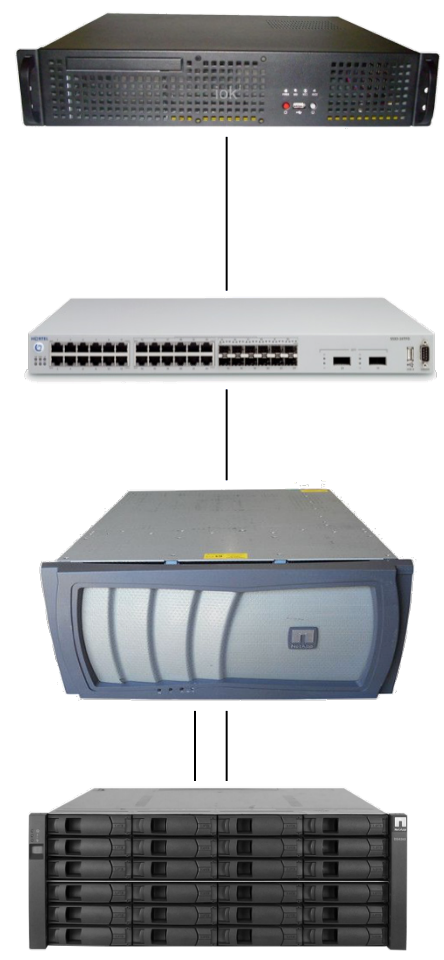
Cabling

Servers

Network switch

Storage controller

Disks

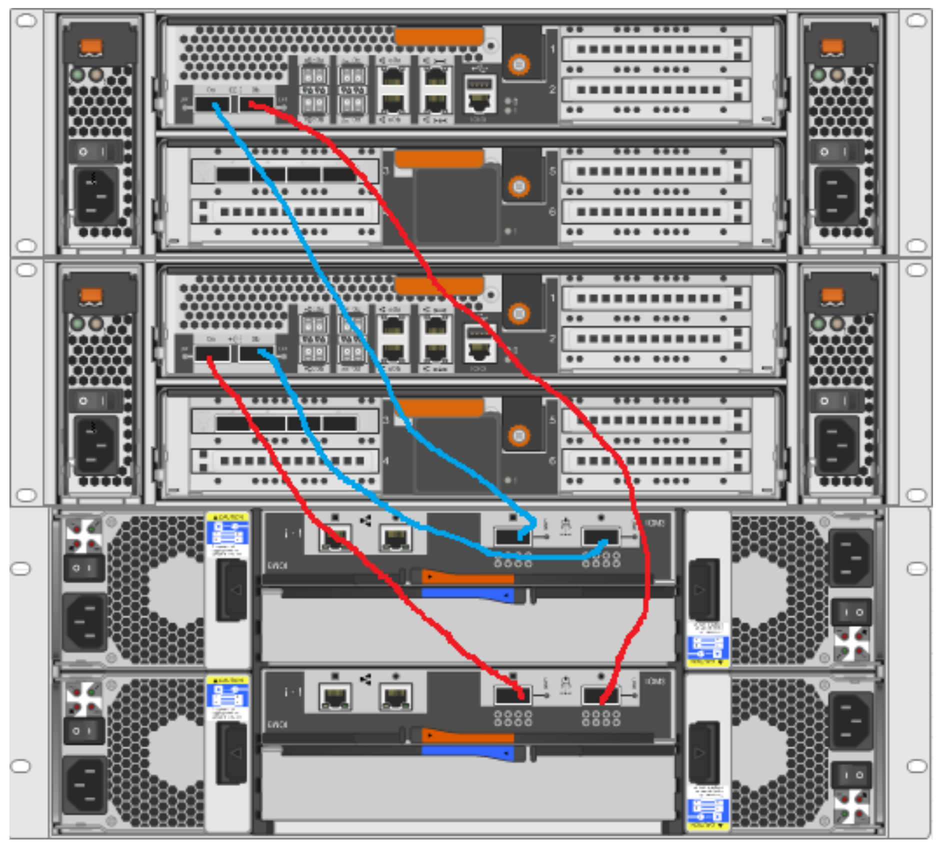


Redundancy: Storage controller

- So we want something like this.

Note: you almost always want storage controllers to be highly available, so they're often sold as "two headed" units, where there's two of everything in one box. To keep it simple, we'll ignore that and use an example where there's literally two boxes.

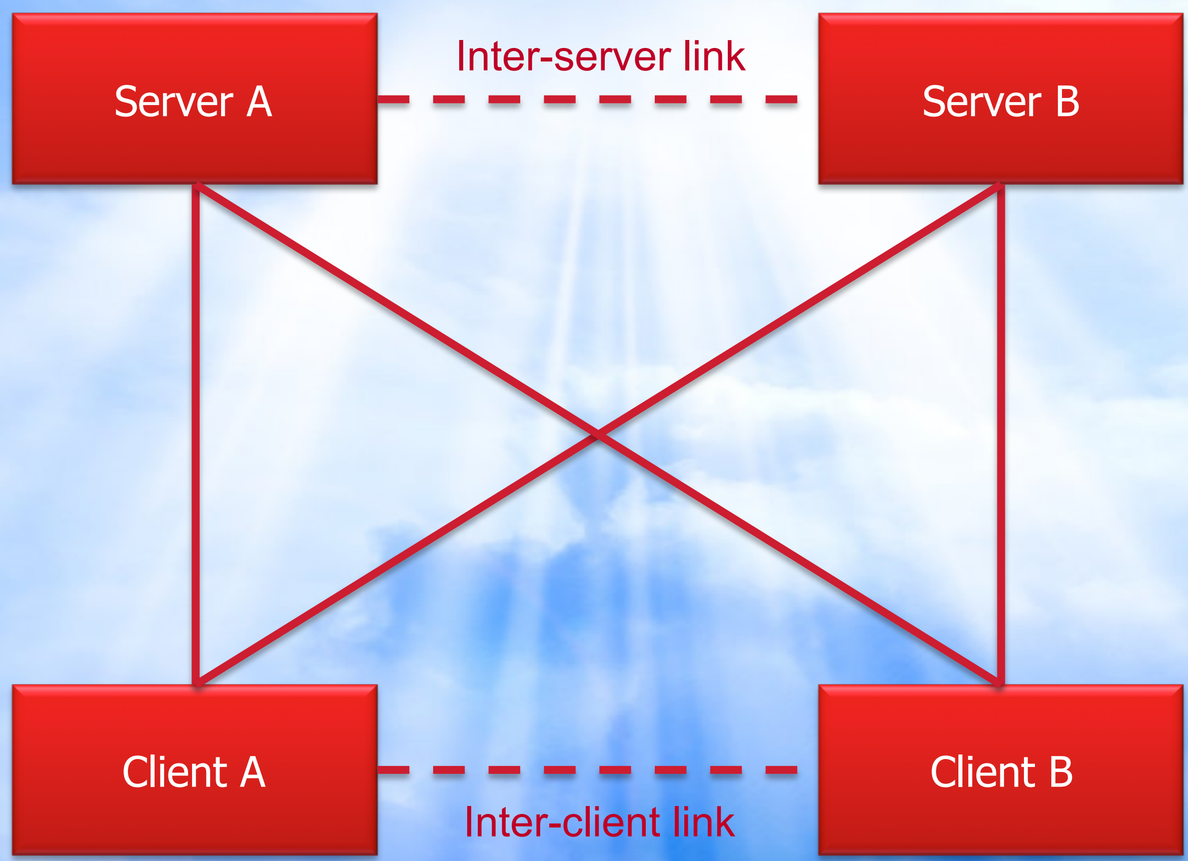
Actual back of that storage controller and its connection to a disk shelf:



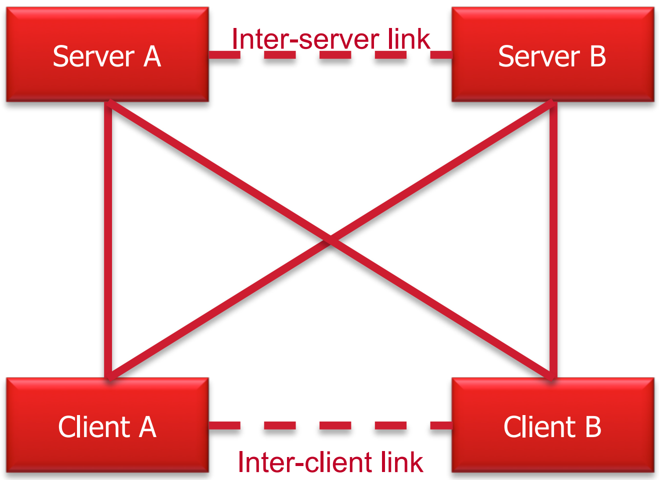
This is an example of what I call the "**universal HA topology**"

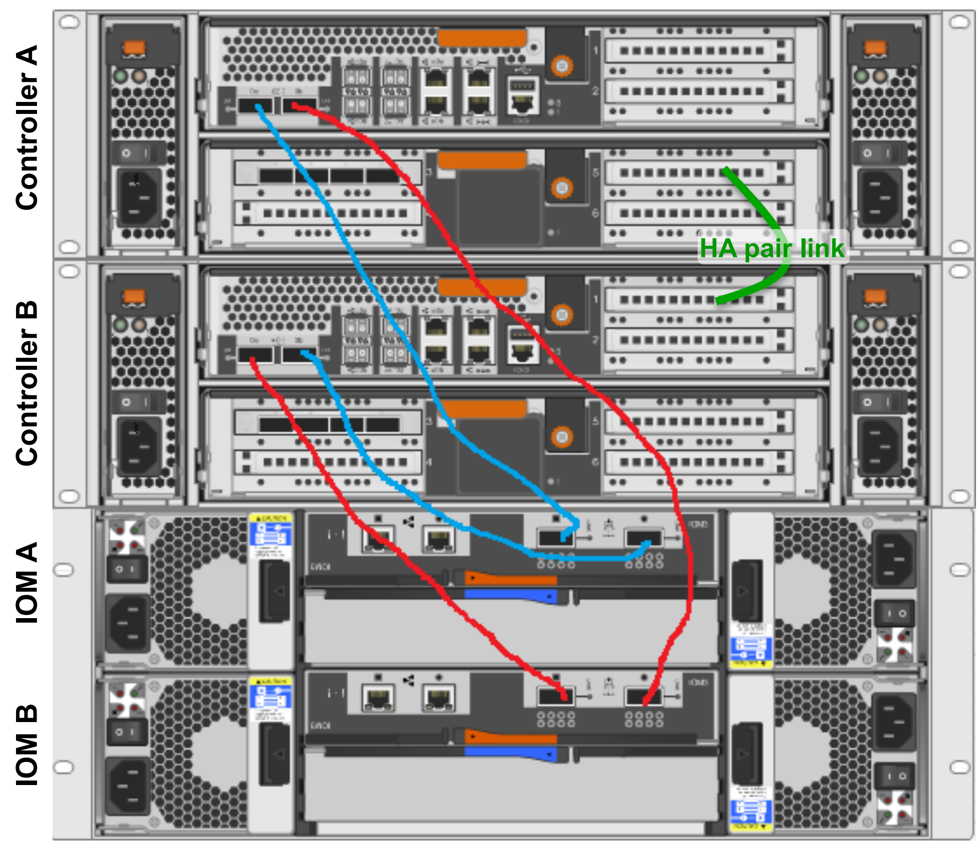
The universal HA topology

(a term Prof. Bletsch made up)



See the topology?





Layers on which to apply redundancy

Power

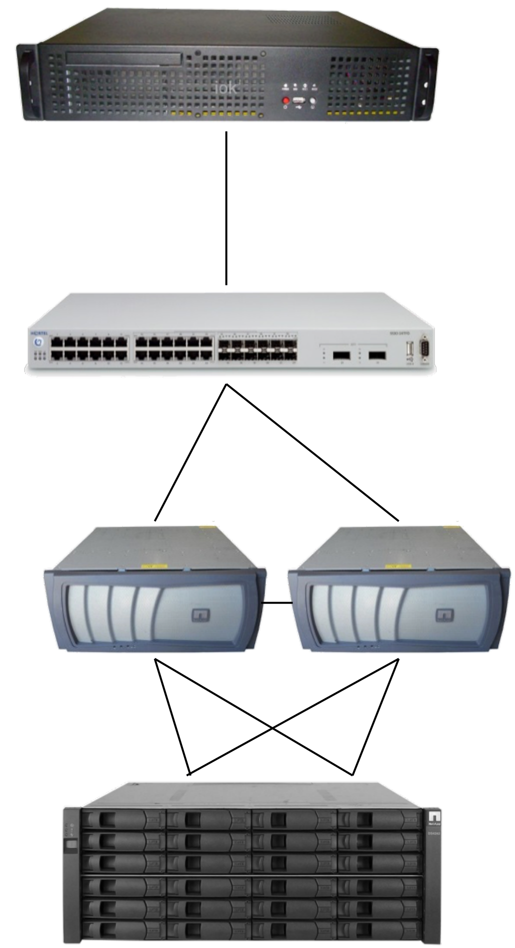
Cabling

Servers

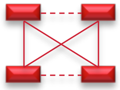
Network switch

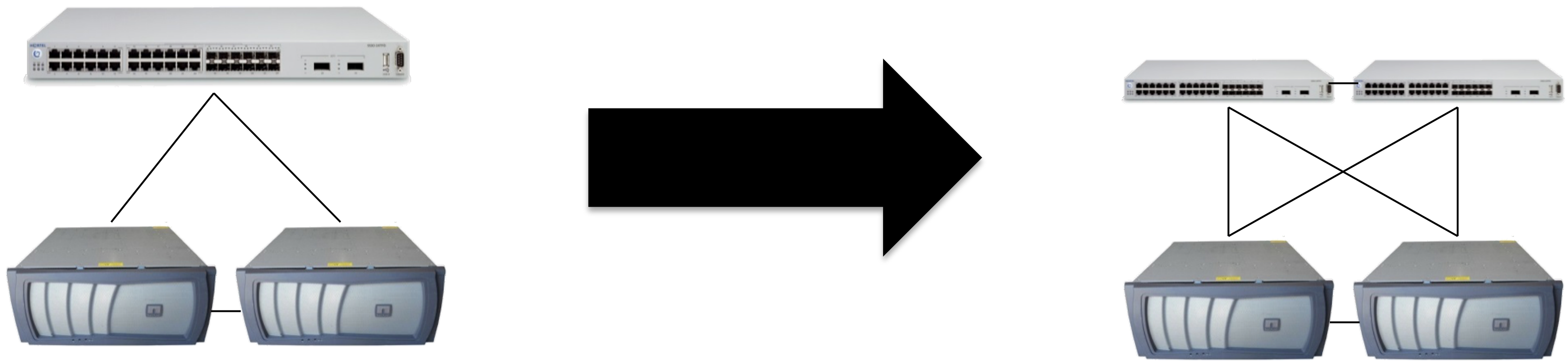
Storage controller

Disks



Network redundancy

- Apply the Universal HA topology 



- In networking, this is known as **multipathing**

Network configuration details that make this possible omitted for time;
for details, take Prof. Bletsch's ECE566, Enterprise Storage Architecture

Ethernet Multipathing (1)

- Known as **Ethernet Link Aggregation**
- Inter-switch configuration:
  - Stackable switches
    - Two switches connected by a proprietary cable to extend the backplane
    - Both switches managed as single entity
  - Peer Link for "Virtual Port Channel" support (Cisco term)
    - Switches are separate, but can cooperate over an Ethernet link between them

Ethernet Multipathing (2)

- Downlinks:
  - Ports on each <u>switch</u> heading to the same target are logically combined (a “port channel” or “virtual port channel” in Cisco terminology)
  - Ports on each <u>endpoint</u> (e.g. storage controller) are similarly bonded, this is often called “link bonding”
- Port bonding situation can be configured statically on both switch and endpoint
- Alternative: Link Aggregation Control Protocol (LACP)
  - Standardized protocol to allow automatic negotiation of link bonding

Layers on which to apply redundancy

Power

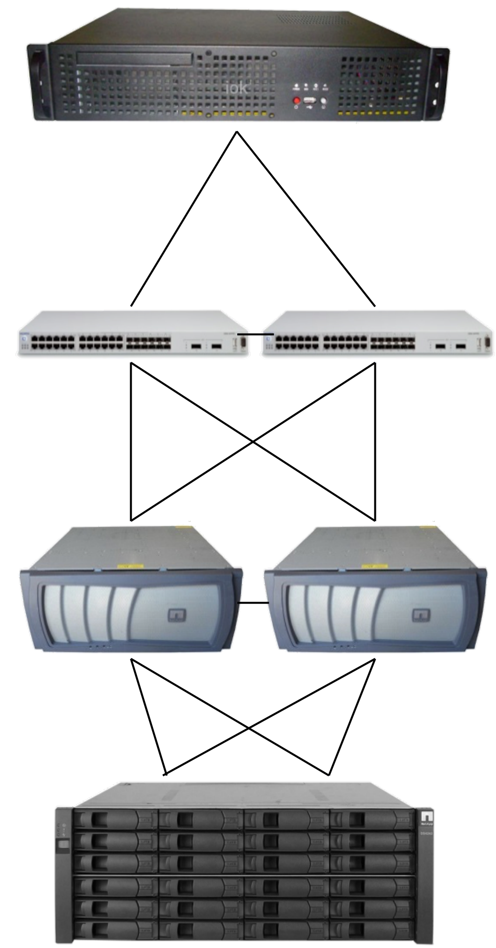
Cabling

Servers

Network switch

Storage controller

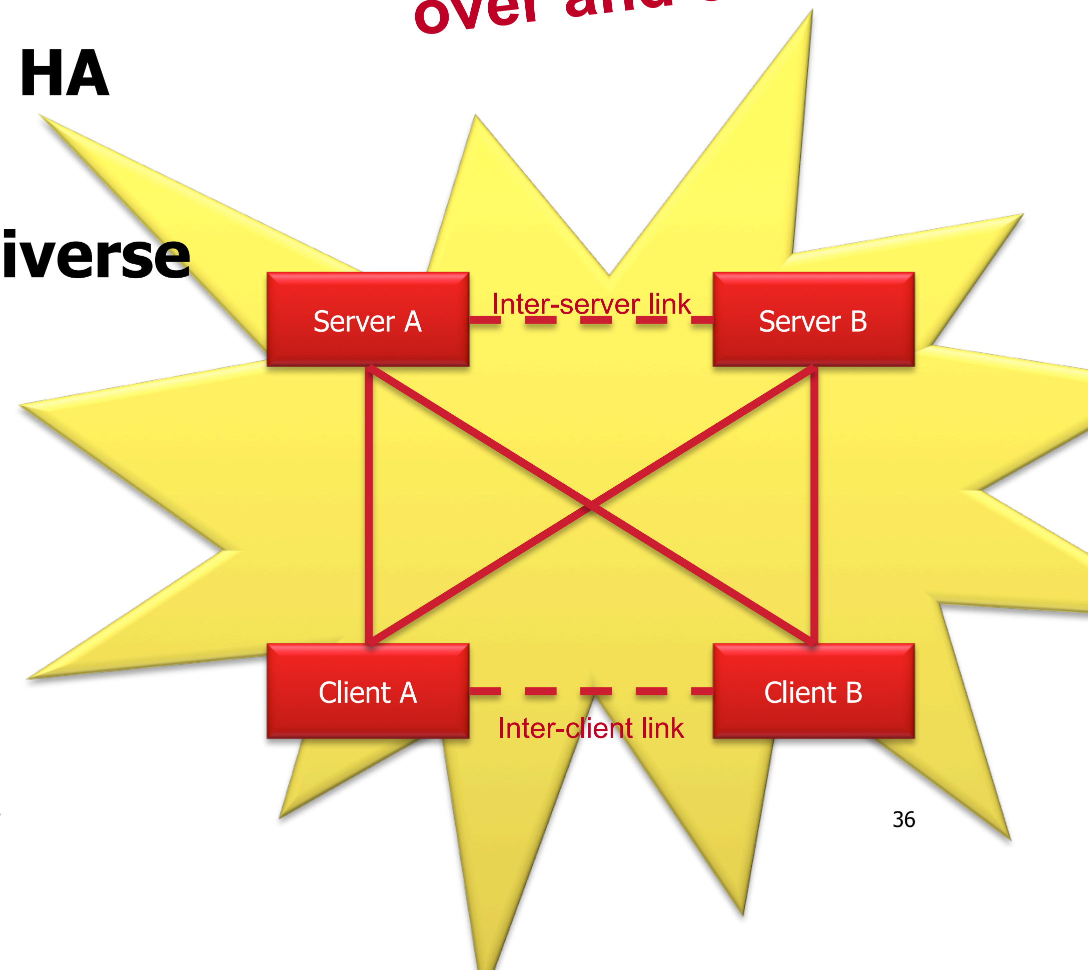
Disks



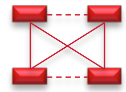
Server redundancy

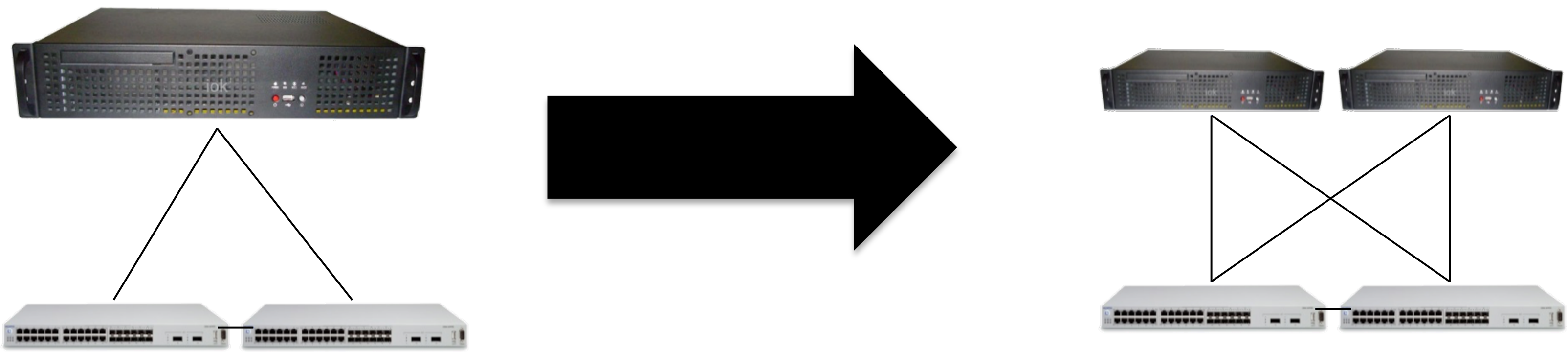
- We need to make servers redundant
- We need a **topology** that does this
- The topology will make them **HA**
- This will take place in our **universe**
- what can we choose
- help
- what can we chooooooooooooooooooose???????

oh yeah, that thing we keep using over and over

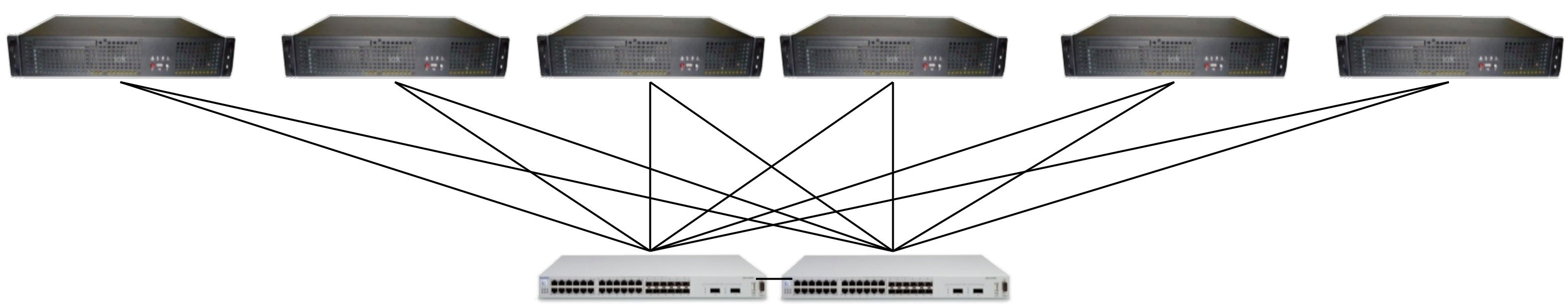


Server redundancy

- Apply the Universal HA topology 



- However, typically have more than 2 servers – storage/ network usually serves pool of many servers



Software support for redundancy

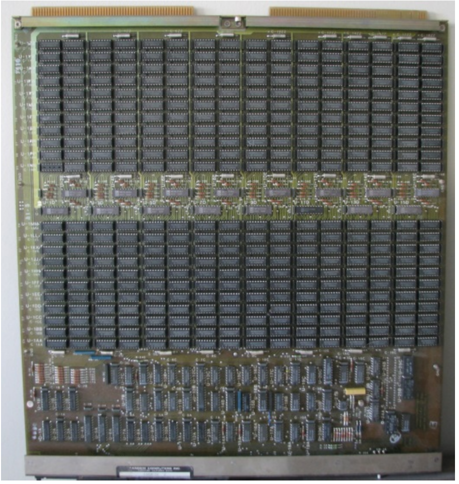
- Physical connectivity is simple; software side is complex:
  - What is the effect of running two copies of the software?
    - Depends on the software!

- Many techniques/mechanisms to take advantage of redundancy:
  1. Truly redundant hardware
  2. Redundancy via hardware abstraction
  3. Redundancy via hypervisor abstraction
  4. Hypervisor-based virtual fault tolerance
  5. Application-based fault tolerance

The greyed out approaches omitted for time;
for details, take Prof. Bletsch's ECE566, Enterprise Storage Architecture

Mechanisms for server redundancy

- **Truly redundant hardware**
  - Not common; very expensive and proprietary
  - Redundancy at the hardware level; software is oblivious
  - Requires very fine-grained synchronization, lots of bandwidth
  - **Result**: Hardware failure produces no measurable effect to the outside world; failed hardware can be hot-swapped afterward
  - **Example**: Tandem Systems shared-nothing message-passing system; now sold as "HP Integrity NonStop X"[1]



A Tandem T/16 memory board; it just has two of everything.

[1] The larger the company, the stupider the naming.

Mechanisms for server redundancy

- **Redundancy via hardware abstraction**
  - Whole servers can “fail over” to spares
  - All server-unique configuration is managed by a server manager
    - Ethernet MAC addresses
    - Fibre Channel World-Wide Names (WWNs)
    - BIOS/firmware settings (e.g. boot order)
  - Servers are of uniform configuration; some servers are “hot spares”
  - On failure:
    - Management system writes hardware config onto a spare server; boots it
    - The replacement server is indistinguishable from the original; software works the same
  - **Result**: Server hardware failure is just like a simple reboot
  - **Example**: Cisco Unified Computing System (UCS) servers

Mechanisms for server redundancy

- **Redundancy via *hypervisor* abstraction**
  - Virtual machines can "fail over" to other physical hosts
  - Physical servers need not be uniform
  - Don't need entire "hot spare" hosts; just need "spare" compute/ memory capacity on overall cluster
  - On physical server failure:
    - VM management system identifies affected VMs; boots them on another server
    - This is possible because VM virtual disks are on shared storage (SAN or NAS)
  - **Result**: Physical server failures act like VM reboot events
  - **Example**: The VMware HA Clustering feature

Mechanisms for server redundancy

- **Hypervisor-based virtual fault tolerance**
  - A VM is "run" on two separate physical hosts
  - On the "primary" host, the VM's system calls actually happen as normal
  - On the "secondary" host, the VM's system calls are faked – responses actually come from whatever the primary system got
  - Result: because computers are deterministic, both systems' computations proceed identically
  - On failure of the primary physical host, the secondary is simply made primary, and its system calls begin happening "for real"
  - Outside world cannot tell that changeover has happened
  - **Result**: Zero perceived downtime
  - **Example**: VMware Fault Tolerance feature

Mechanisms for server redundancy

- **Application-based fault tolerance**
  - The user application has built-in support for some kind of HA clustering
  - May work with performance-based clustering (i.e. scaling application performance by adding more servers) or be totally separate (e.g. an active/passive app)
  - Pro: Application does its own consistency, can achieve higher performance than the previous application-oblivious techniques
  - Con: Developers must consciously design application with this in mind
  - **Result**: Depends on how app is built, but typically fault-tolerant apps allow server failure without measurable effect to outside world.
  - **Example**: Microsoft SQL Server Failover Clustering

Best practices for software HA (1)

Reliable restart

- Software comes up into correct state on reboot no matter the order of bring-up.

- **No RAM-only changes**: All changes are reflected on disk. Example: don't start the web server manually, make it a configured daemon.

Best practices for software HA (2)

Change carefully

- Only apply changes that are tested in an *identical* deployment configuration
  - Staging environment
  - Virtualization and containers can help with this
- No "cowboy" live updates!
  - It "shouldn't" affect anything != it *won't* affect things
  - Assumptions are the enemy

- Includes patches/updates – test before going to production!

Best practices for software HA (3)

Monitoring

- External system that watches key software.
- Measures:
  - if it's alive,
  - if it's functioning, and
  - if its performance is sufficient.
- Alert on failure
- Report periodically (make sure the monitor isn't dead too!)

We did it!

Power

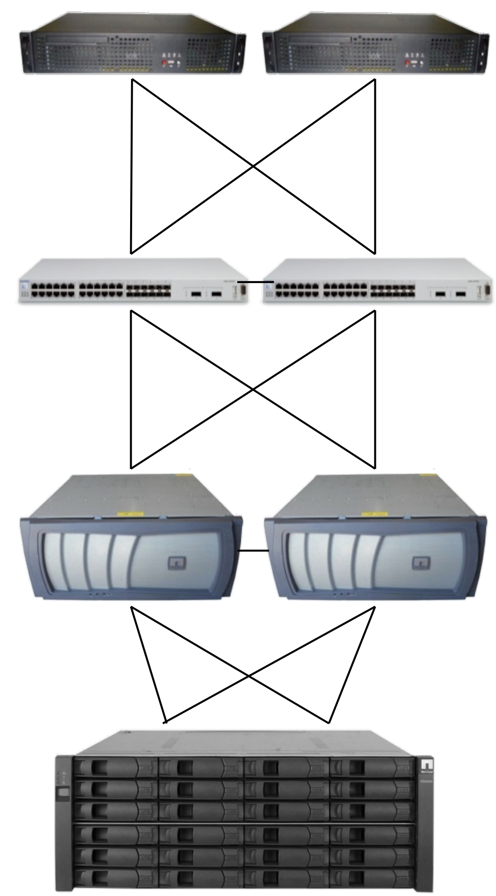
Cabling

Servers

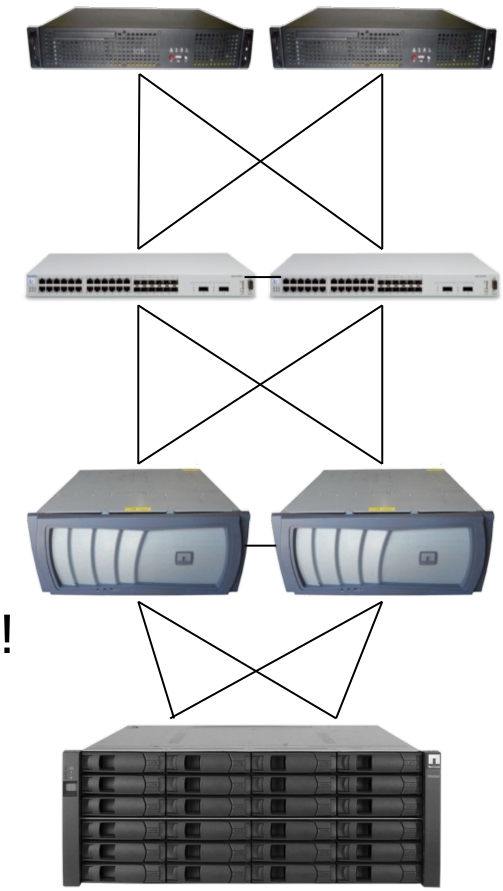
Network switch

Storage controller

Disks

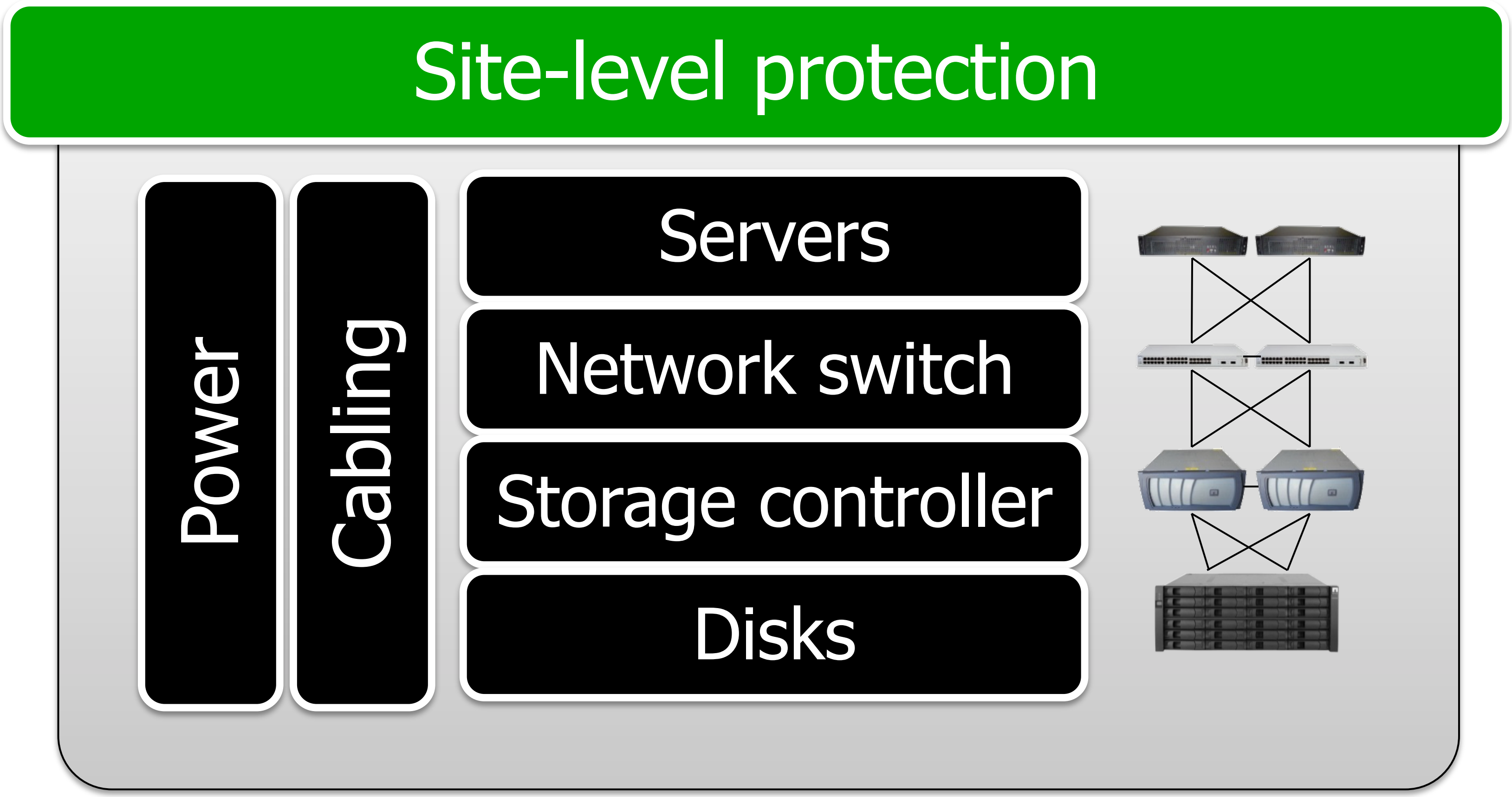


But what if...?



- A meteor lands on our datacenter?
- Can we be HA against that?
  - Surprisingly, **yes** (for a small enough meteor)!

Zooming out some...

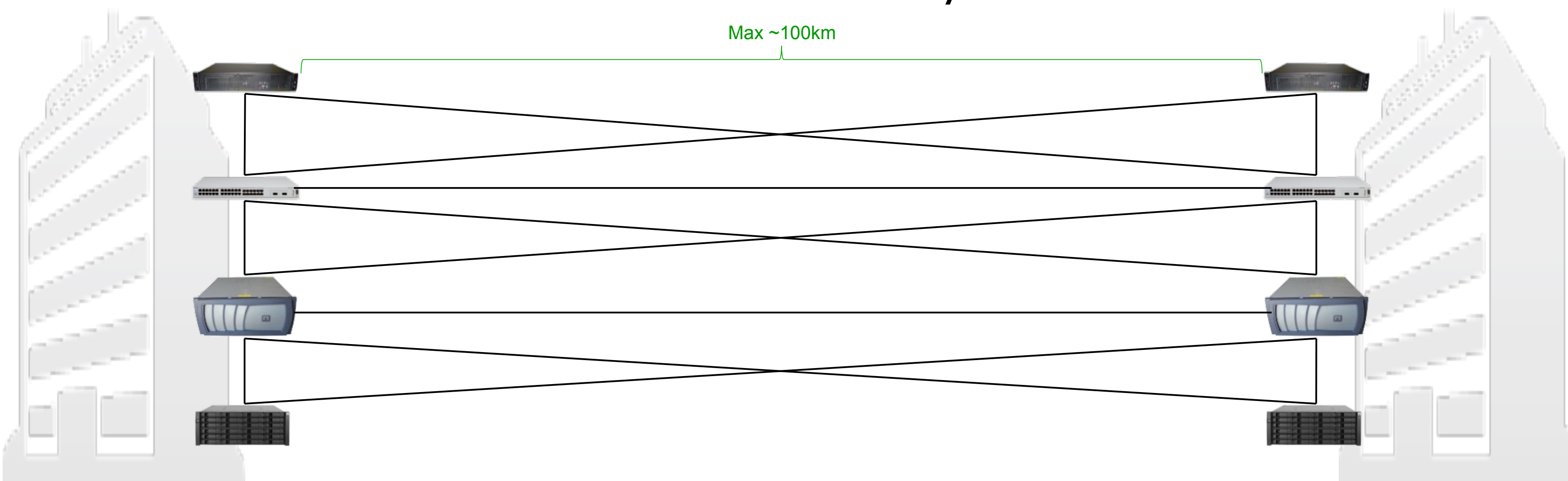


Thought experiment

- HA works if each redundant pair is 1 meter away
- Does it still work at 2 meters?
- Does it still work at 4 meters?
- Does it still work at 8 meters?
- ...
- What's the limit?
- What affects the limit?
  - Latency
  - Ability of cable to carry data that far
- Practical answer: around 100km (depends on many things)
  - FYI: (100 kilometers) / the speed of light = 333 microseconds

What if we put our two halves far apart?

- Result: Metro-scale clustering
  - "Metro-scale" = Around the size of a city



- Often deployed just at campus-scale (a few kilometers); sometimes deployed all the way between cities (especially in Europe, where cities are closer)
- Can also be applied to just storage: then it's a form of backup/replication, which we'll cover when we talk about disaster recovery
- **Result:** You can lose an ENTIRE DATACENTER and keep serving traffic with little to no interruption
- **Example**: NetApp Metrocluster plus VMware Stretch Cluster

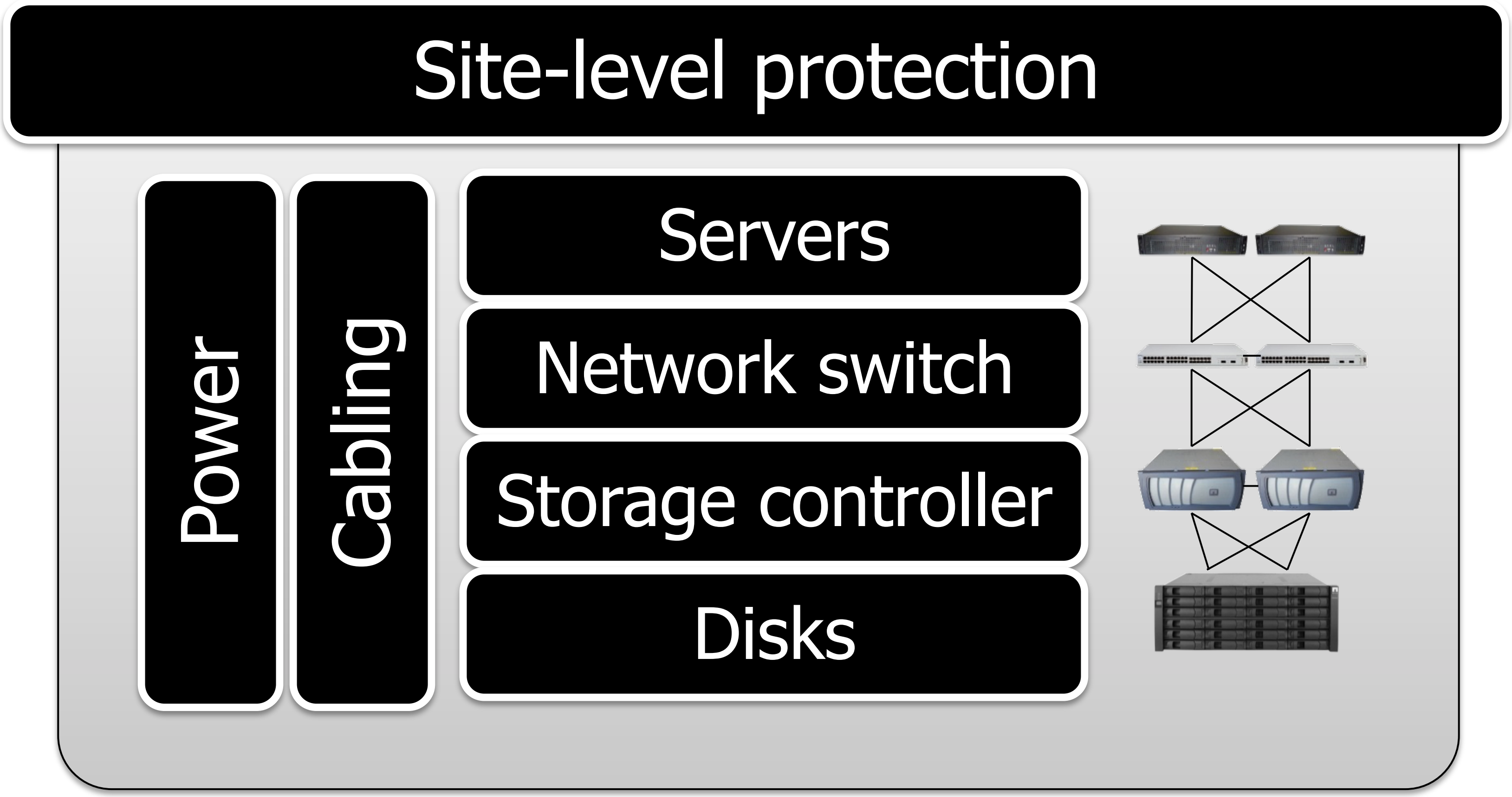
Connectivity

- Connection between sites typically dedicated optical fiber
  - Fiber optics can run data faster over much longer distance than copper

- How to get?
  - Dark fiber: abandoned pre-existing line
  - New fiber: pay huge cost to run a new buried line
  - Leased line: pay for bandwidth on existing dedicated lines

- Can also tunnel over existing network or the internet
  - Performance penalty, or even unpredictable performance

Conclusion



But what happens if something overwhelms these protections? Need ***<u>disaster recovery</u>*** (next).