World Class HA with Linux-HA

Alan Robertson

Project Founder, Leader – Linux-HA project

alanr@unix.sh / alanr@us.ibm.com

IBM Systems & Technology Group

Business Resilience and Security

Solution Architect

HA BLOG: http://techthoughts.typepad.com/

Overview

- HA Principles
- Introduction to Linux-HA
  - Who uses it?
  - What do they use it for?
  - What can it do?
  - Most interesting features
- Future Thoughts

Questions Given in Advance

- Please indicate the target areas of Linux HA Systems.
- What are the recent topics of Linux HA development?
- What development contributions you are expecting from Japanese developers.
- How Linux HA System beneficial to Virtualized environment?
- How Linux HA System grows in the near future?

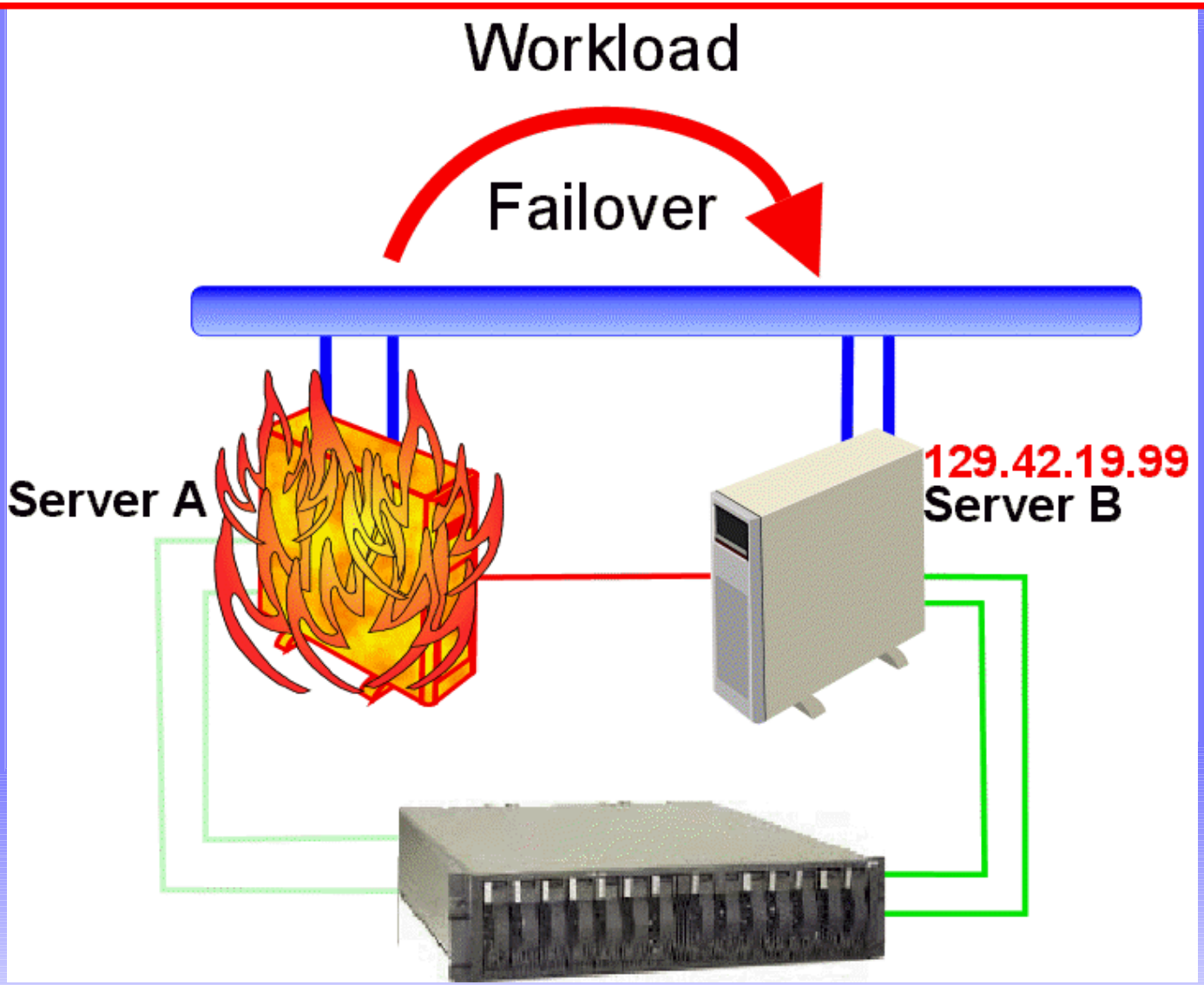
What Is HA Clustering?

- Putting together a group of computers which trust each other to provide a service even when system components fail
- When a server goes down, others take over its work
- When services fail, they get restarted
- This involves IP address takeover, service takeover, etc.
- New work comes to the “takeover” machine
- Not primarily designed for high-performance

What Can HA Clustering Do For You?

- **It cannot achieve 100% availability** – *nothing can.*
- HA Clustering designed to recover from single faults
- It can make your outages very short
  - From about a second to a few minutes
- It is like a Magician's (Illusionist's) trick:
  - When it goes well, the hand is faster than the eye
  - When it goes not-so-well, it can be reasonably visible
- A good HA clustering system adds a “9” to your base availability
  - 99->99.9, 99.9->99.99, 99.99->99.999, etc.
- **Complexity is the enemy of reliability!**

High-Availability Workload Failover



Lies, Damn Lies, and Statistics

Counting nines

99.9999%	30 sec
99.999%	5 min
99.99%	52 min
99.9%	9 hr
99%	3.5 day

Barriers to HA systems

- Hardware costs
- Software costs
- Complexity
- Standardization

Potential User Community

What would be the result?

- Increased hardware, software, services opportunities
- Drastically multiplying customers multiplies experience - products mature faster (especially in OSS model)
- OSS developers grow with customers
- Low-end customers are less prone to sue than high-end customers
- OSS Clustering is a disruptive technology

How is this like what you know?

- It's a lot like the current init startup scripts extended by:
  - Adding optional parameters to them
  - Running on a more than one computer
  - Adding policies for
    - what order to do things
    - how services relate to each other
    - when to run them
- HA systems are a lot like “init on steroids”

What is different?

- Data sharing isn't usually an issue with a single server – it's critically important in clusters
- HA Clusters introduce concepts and complications around
  - Split-Brain
  - Quorum
  - Fencing
- You need to tell us what applications run where, it's no longer implicit

Split-Brain

- Communications failures can lead to separated partitions of the cluster
- If those partitions each try and take control of the cluster, then it's called a split-brain condition
- If this happens, then bad things will happen
  - http://linux-ha.org/BadThingsWillHappen

Fencing

- Fencing tries to put a fence around an errant node or nodes to keep them from accessing cluster resources
- Fencing doesn't have to rely on correct behavior or timing of the errant node.
- We use STONITH to do this
  - STONITH: Shoot The Other Node In The Head
- Other techniques also work
  - Fiber channel switch lockout
  - etc

Quorum

- Quorum is an attempt to avoid split brain for many kinds of failures
- Typically one tries to make sure only one partition can be active
- Quorum is term for methods for ensuring this
- Most common kind of quorum is voting – and only a partition with > n/2 nodes can run the cluster
- This doesn't work very well for 2 nodes :-(

Split brain, quorum and fencing are discussed extensively on my blog

Single Points of Failure (SPOFs)

- A single point of failure is a component whose failure will cause near-immediate failure of an entire system or service
- Good HA design eliminates of single points of failure
- Redundant hardware is used to eliminate them – giving us “back up” components

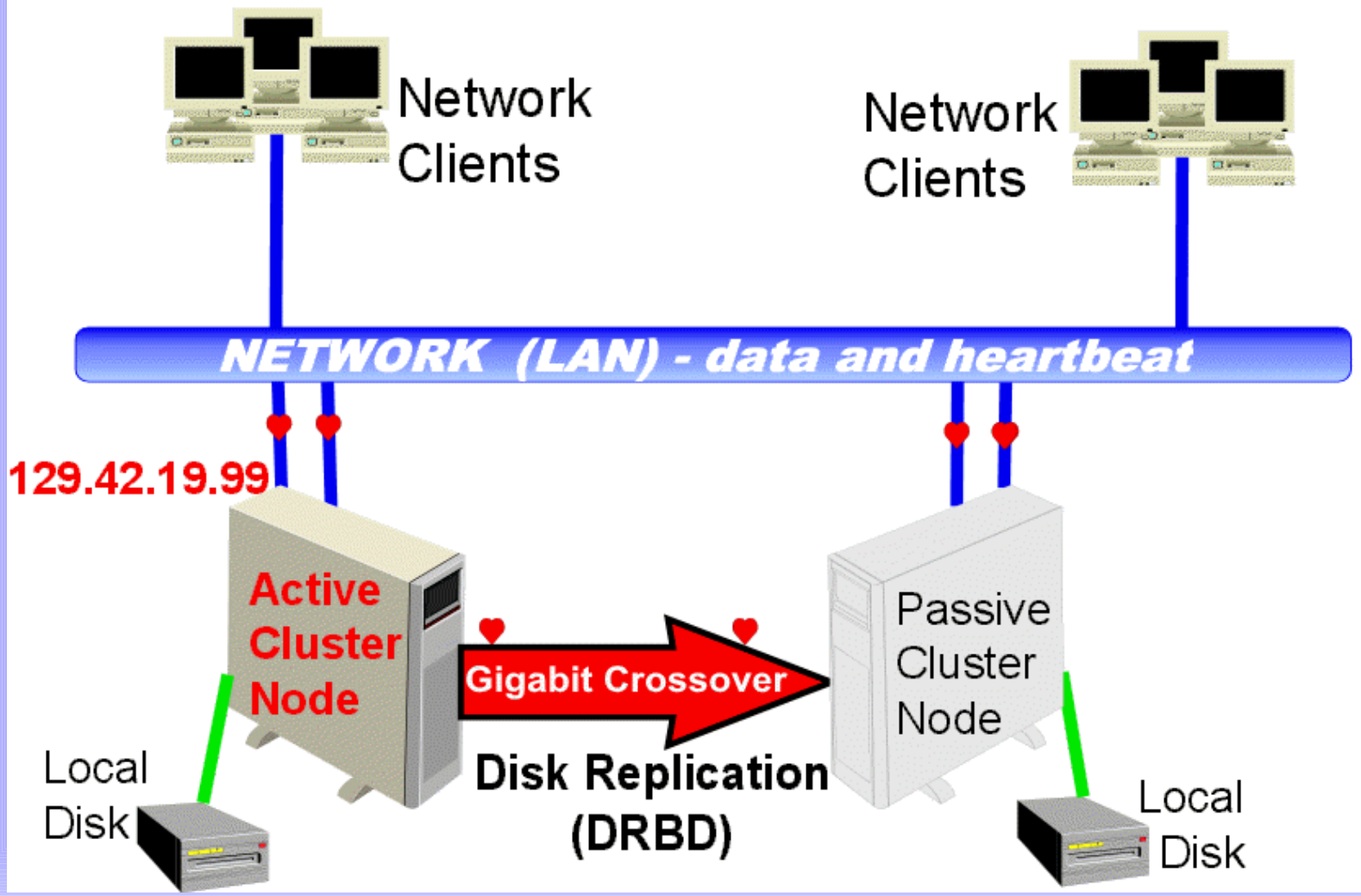
Redundant Communications

- Intra-cluster communication is critical to HA system operation
  - Most HA clustering systems provide mechanisms for redundant internal communication for heartbeats, etc.
- External communications is usually essential to provision of service
  - External communication redundancy is usually accomplished through routing tricks
  - Having an expert in BGP or OSPF is a help

Redundant Data
Data Sharing Methods

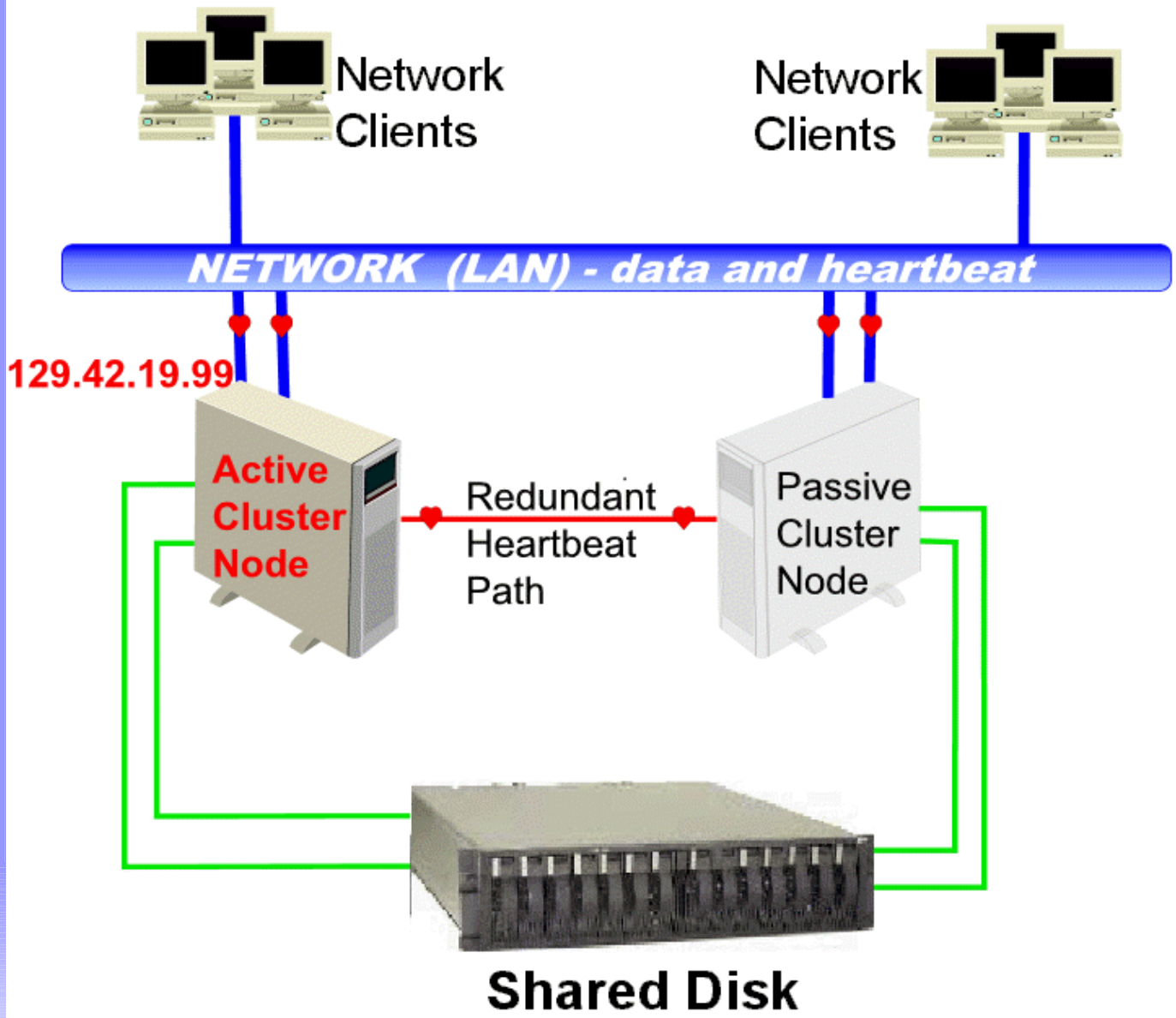
- None
  - Used for firewalls, proxy servers
- Shared Disk – fibre channel
  - Most common method for databases, etc.
- Replicated Data
  - Can be very inexpensive (low end)
  - Also used for Disaster Recovery (high end)
- “Back End Server”
  - Data sourced from “somewhere else” - not your problem

Two node Active/Passive HA Cluster Real-Time Disk Replication (*DRBD*)



Two node Active/Passive HA Cluster

Shared Disk (FasTT, ESS, etc.)



How is HA Clustering Different from Disaster Recovery ("geographic clustering")?

- **HA (single-site):**
  - Reliable inter-node communication
  - Failover is cheap
  - Failover times measured in seconds
- **DR (split-site):**
  - Unreliable inter-node communication assumed
  - Failover is expensive
  - Automatic failback may be impossible
  - Failover times often longer, sometimes measured in hours
- Linux-HA provides special features to deal with "geographic clustering" (aka disaster recovery)

When are you in a DR situation?

Alan's DR “rule of thumb”:

- Once you bury your wires in the ground, you've crossed over from HA to DR

Why?

- You lose the ability to ensure the reliability of inter-node communication
- You typically cannot use shared storage, and must use data replication instead

What happens differently in DR?

- Data must be replicated, cannot be shared
- You can't rely on fencing
- Quorum typically becomes problematic – particularly for a 2-site DR arrangement
  - Linux-HA provides a quorum daemon to deal with this

Linux-HA Background

- The oldest and most well-known open-community HA project - providing sophisticated fail over and restart capabilities for Linux (and other OSes)
- In existence since 1998; ~ 30k mission-critical clusters in production since 1999
- Active, open development community led by IBM, Novell and NTT
- Wide variety of industries, applications supported
- Shipped with most Linux distributions (all but Red Hat)
- No special hardware requirements; no kernel dependencies, all user space
- All releases tested by automated test suites

Linux-HA Success Stories

- NTT – enterprise applications
- Sony - manufacturing
- Japanese Weather Bureau
- DFS – German Air Navigation – airport tower information
- Nationwide Insurance
- Emageon (medical imaging)
- BBC
- Motorola
- The Weather Channel
- Man Nutzfahrzeuge AG
- Thessaloniki Port Authority
- ISO New England (power grid mgmt)
- Sanger Institute (genome research)
- Agilent Technologies

Nearly every industry uses Linux-HA

Linux-HA Application Areas

- ERP (SAP et al)
- Databases
- Websphere Application Server
- File Servers
- Web Servers
- Load balancers
- Firewalls
- DNS, DHCP, etc.
- Proxy caching
- Manufacturing
- Custom applications

Nearly any type of application can be supported

Linux-HA Capabilities

- Supports n-node clusters – where 'n' <= something like 16
- Can use UDP bcast, mcast, ucast comm.
- Fails over on node failure, or on service failure
- Fails over on loss of IP connectivity, or arbitrary criteria
- Active/Passive or full Active/Active
- Built-in resource monitoring
- Support for the OCF resource standard
- Sophisticated dependency model with rich constraint support (resources, groups, incarnations, master/slave)
- XML-based resource configuration
- Configuration and monitoring GUI
- Support for OCFS2 cluster filesystem

Linux-HA Strengths

- Very powerful Policy Engine – fine grained, attribute based
- Failover on arbitrary conditions
- Primitive Resources
  - LSB – most services with LSB init scripts immediately supported without additional scripting
  - OCF – Very powerful Standard Interface for creating resources
- Clone Resources
- Master/Slave Resources
- Virtualization Support – adds "migrate" operation
- Split-site support for Disaster Recovery

HA and Virtualization

- Virtualization is often used for server consolidation
- Failure of a single physical server can cause the failure of dozens of consolidated virtual servers
- In these circumstances, High Availability is essential to keep massive service outages from occurring when a single server fails
- Linux-HA can manage virtual machines as resources
- Linux-HA will transparently migrate virtual machines when possible
- Linux-HA comes with a resource agent for Xen DOMU instances

Making Outages Disappear!

- Many IBM servers provide predictive failure analysis – announcing expected failures before they occur
- It is simple to connect that indication to Linux-HA to cause preemptive, transparent migration of virtual machines *before the server failure occurs*
- Linux-HA transparently migrates virtual machines off the affected server without interruption
- The result? Nothing is on it when it crashes. Service outage ***completely avoided!***
- This Linux-HA integration with IBM servers is coming in 2008

Linux-HA as an Open Source Project

- Linux-HA is a true community open source project
- Contributors come from many companies, the number of contributors is large (around 100)
- Contributors come from many countries.
- One of my favorite patches came from the Japanese weather service
- NTT, NEC, VA Linux Japan and other Japanese firms are active in the Linux-HA community
- Much of the web site has been translated into Japanese, and there is a Japanese mailing list

Future Thoughts

Predicting the future of open source projects is difficult

- Improve virtual machine management
- MUCH larger clusters (hundreds or thousands of nodes)
- Better Documentation / Educational materials
- Improved usability – GUI, ciblint, logging, etc.
- Integrate with Unified Linux cluster filesystem stack
- Better hardware integration – server, storage
- Better configuration history, authorization
- Add “green computing” power management capability

Future Questions...

What would ***you*** like to see it do in the future?

Come to the Birds-Of-A-Feather (BOF) session and we will discuss it!

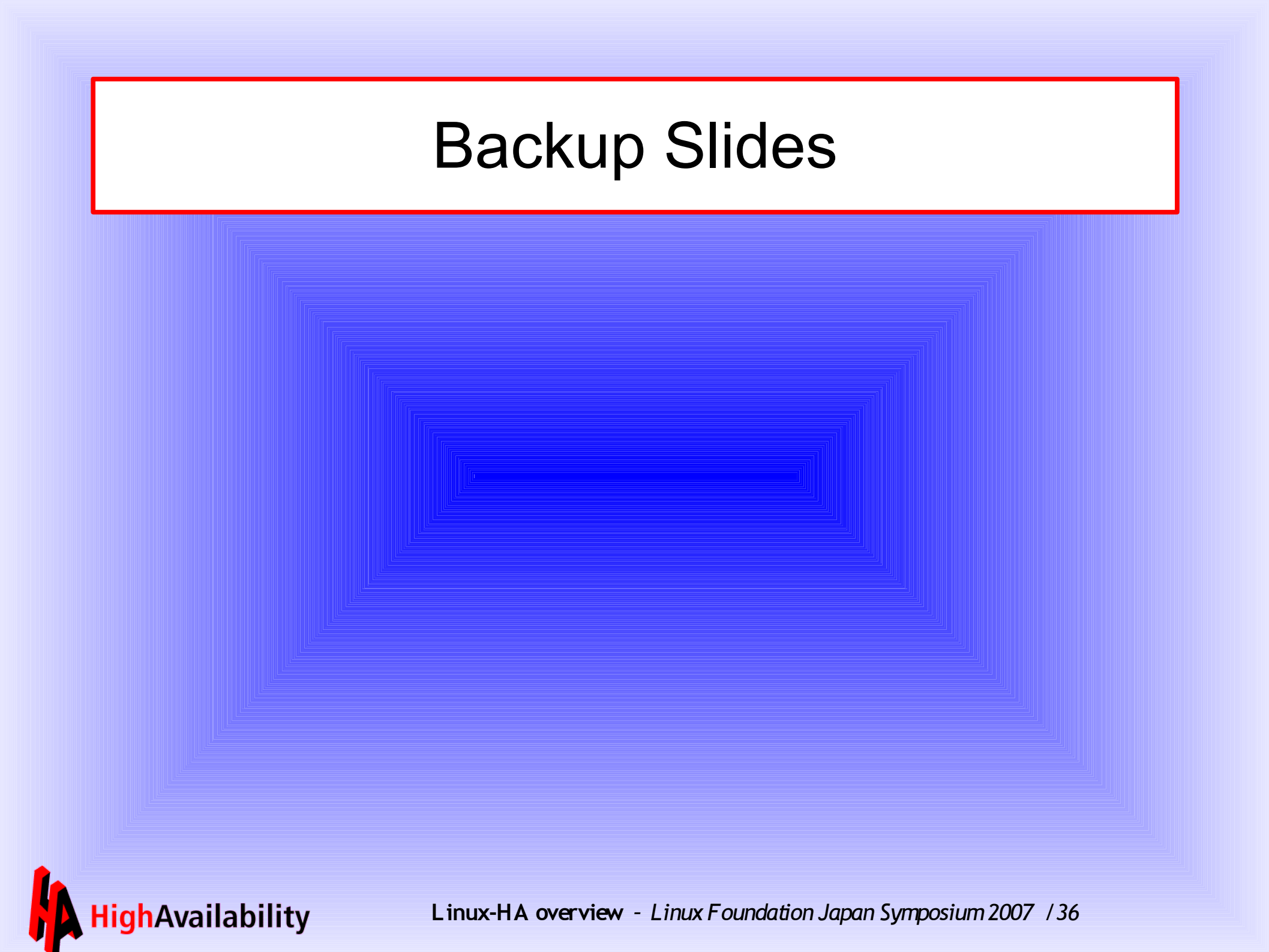
References

- http://linux-ha.org/ - ENGLISH
- http://linux-ha.org/ja/HomePage_ja - JAPANESE
- http://linux-ha.org/download/
- http://linux-ha.org/SuccessStories
- HA BLOG: http://techthoughts.typepad.com/

Legal Statements

- IBM is a trademark of International Business Machines Corporation.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
- This work represents the views of the author and does not necessarily reflect the views of the IBM Corporation.

Backup Slides



First working code: 18 March 1998

Alan Robertson <alanr@bell-labs.com>
Wed, 18 Mar 1998 07:58:16 +0100

I put together the heartbeat mechanism I'd described and coded it up and, amazingly enough, it works! Yes, I got a heartbeat -- signs of life!
To be perfectly honest, I only have access to one machine right now, but there's still a lot I can test. Here's what I know works:

- Basic cluster config file parsing and error checking
- Local machine heartbeat satisfies "keepalive" requirements
- Remote machine (not connected :-)) has no heartbeat => it's dead!
- Invocation of external scripts on state changes
- Ability to easily update state of local machine (via echo!)

Here's some things I haven't a clue about whether they work or not:

- Correct line settings
- Write/Read serial port function

Right now I'm running a test to see if lack of a machine (and corresponding serial port signals) on a tty port causes deadlock.

I've put in a request with my company (Lucent Technologies) to allow me to release it under the GNU public license. I'll let you know when that comes through. FYI, it's about 1200 lines of 'C' code including comments.

It's Alive!

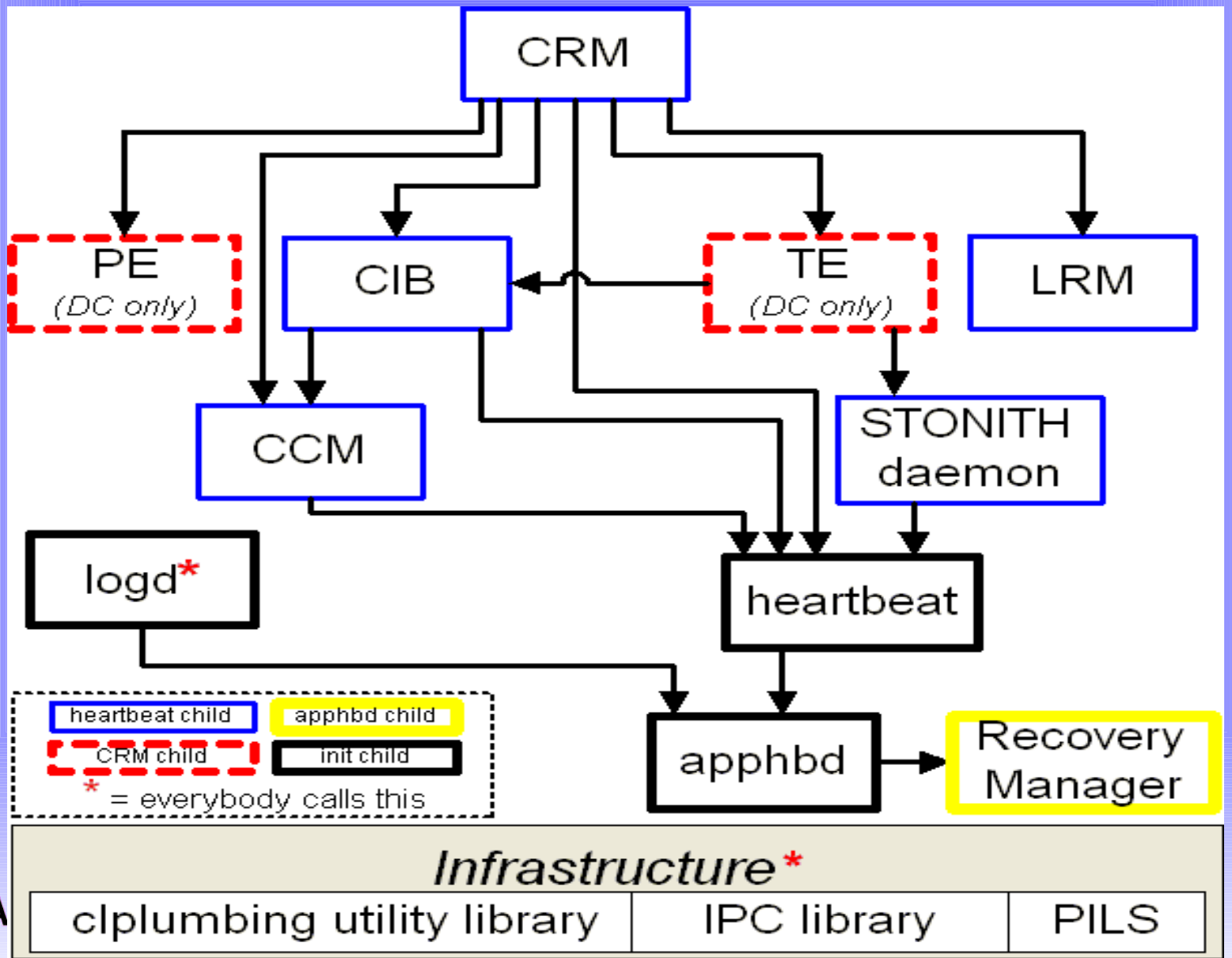
Some Linux-HA Terminology

- **Node** – a computer (real or virtual) which is part of the cluster and running our cluster software stack
- **Resource** – something we manage – a service, or IP address, or disk drive, or whatever. If we manage it and it's not a node, it's a resource
- **Resource Agent** – a script which acts as a proxy to control a resource. Most are closely modelled after standard system init scripts.
- **DC** – Designated Coordinator – the “master node” in the cluster
- **STONITH** – Acronym for **Shoot The Other Node In The Head** – a method of fencing out nodes which are misbehaving by resetting them
- **Partitioned cluster** or **Split-Brain** – a condition where the cluster is split into two or more pieces which don't know about each other through hardware or software failure. Prevented from doing BadThings by STONITH
- **Quorum** – normally assigned to at most one single partition in a cluster to keep split-brain from causing damage. Typically determined by a voting protocol

Key Linux-HA Processes

- **CRM** – Cluster Resource Manager – The main management entity in the cluster
- **CIB** – The cluster Information Base – keeper of information about resources, nodes. Also used to refer to the information managed by the CIB process. The CIB is XML-based.
- **PE** – Policy Engine – determines what should be done given the current policy in effect – creates a graph for the TE containing the things that need to be done to bring the cluster back in line with policy *(only runs on the DC)*
- **TE** – Carries out the directives created by the PE – through it's graph *(only runs on the DC)*
- **CCM** – Consensus Cluster Membership – determines who is in the cluster, and who is not. A sort of gatekeeper for cluster nodes.
- **LRM** – Local Resource Manager – low level process that does everything that needs doing – not cluster-aware – no knowledge of policy – ultimately driven by the TE (through the various CRM processes)
- **stonithd** – daemon carrying out STONITH directives
- **heartbeat** – low level initialization and communication module

Linux-HA Release 2 Architecture



HighA

Resource Objects in Release 2

- Release 2 supports “resource objects” which can be any of the following:
  - Primitive Resources
    - OCF, heartbeat-style, or LSB resource agent scripts
  - Resource Clones – need *“n”* resource objects - somewhere
  - Resource Groups – a group of primitive resources with implied co-location and linear ordering constraints
  - Multi-state resources (master/slave)
    - Designed to model master/slave (replication) resources (DRBD, et al)

Basic Dependencies in Release 2

- Ordering Dependencies
  - start before *(normally implies stop after)*
  - start after *(normally implies stop before)*
- Mandatory Co-location Dependencies
  - must be co-located with
  - cannot be co-located with

Resource Location Constraints

- **Mandatory Constraints:**
  - Resource Objects can be constrained to run on any selected subset of nodes. Default depends on setting of *symmetric_cluster*.
- **Preferential Constraints:**
  - Resource Objects can also be preferentially constrained to run on specified nodes by providing weightings for arbitrary logical conditions
  - The resource object is run on the node which has the highest weight (score)

Resource Clones

- Resource Clones allow one to have a resource which runs multiple (*"n"*) times on the cluster
- This is useful for managing
  - load balancing clusters where you want *"n"* of them to be slave servers
  - Cluster filesystems
  - Cluster Alias IP addresses

Resource Groups

Resource Groups provide a simple method for creating ordering and co-location dependencies

- Each resource object in the group is declared to have linear *start-after* ordering relationships
- Each resource object in the group is declared to have co-location dependencies on each other
- This is an easy way of converting release 1 resource groups to release 2

Multi-State (master/slave) Resources

- Normal resources can be in one of two stable states:
  - **`started`**
  - **`stopped`**
- Multi-state resources can have more than two stable states. For example:
  - **`stopped`**
  - **`running-as-master`**
  - **`running-as-slave`**
- This is ideal for modelling replication resources like DRBD, HADR (IBM DB2) and Oracle DataGuard

Cluster Information Base (CIB) Intro

- The CIB is an XML file containing:
  - Configuration Information
    - Cluster Node information
    - Resource Information
    - Resource Constraints
  - Status Information
    - Which nodes are up / down
    - Attributes of nodes
    - Which resources are running where
- We only provide configuration information

Classes of Resource Agents in R2

- **OCF** – Open Cluster Framework - http://opencf.org/
  - take parameters as name/value pairs through the environment
  - Can be monitored well by R2
- **Heartbeat** – R1-style heartbeat resources
  - Take parameters as command line arguments
  - Can be monitored by **`status`** action
- **LSB** – Standard LSB Init scripts
  - Take no parameters
  - Can be monitored by **`status`** action
- **Stonith** – Node Reset Capability
  - Very similar to OCF resources

Resource Groups

- Resources can be put together in groups a lot like R1 resource groups or those of other HA systems
- Groups are simple to manage, but less powerful than individual resources with constraints

```
<group id="webserver">
   <primitive/>
   <primitive/>
</group>
```

- By default, groups imply co-location and ordering, these properties are optional

Resource "clone" Units

- If you want a resource to run in several places, then you can "clone" the resource

```
<clone id="MyID">
   <instance_attributes>
     <attributes/>
   </instance_attributes>
   <primitive>
     <operations/>
     <instance_attributes/>
   </primitive/>
</clone>
```

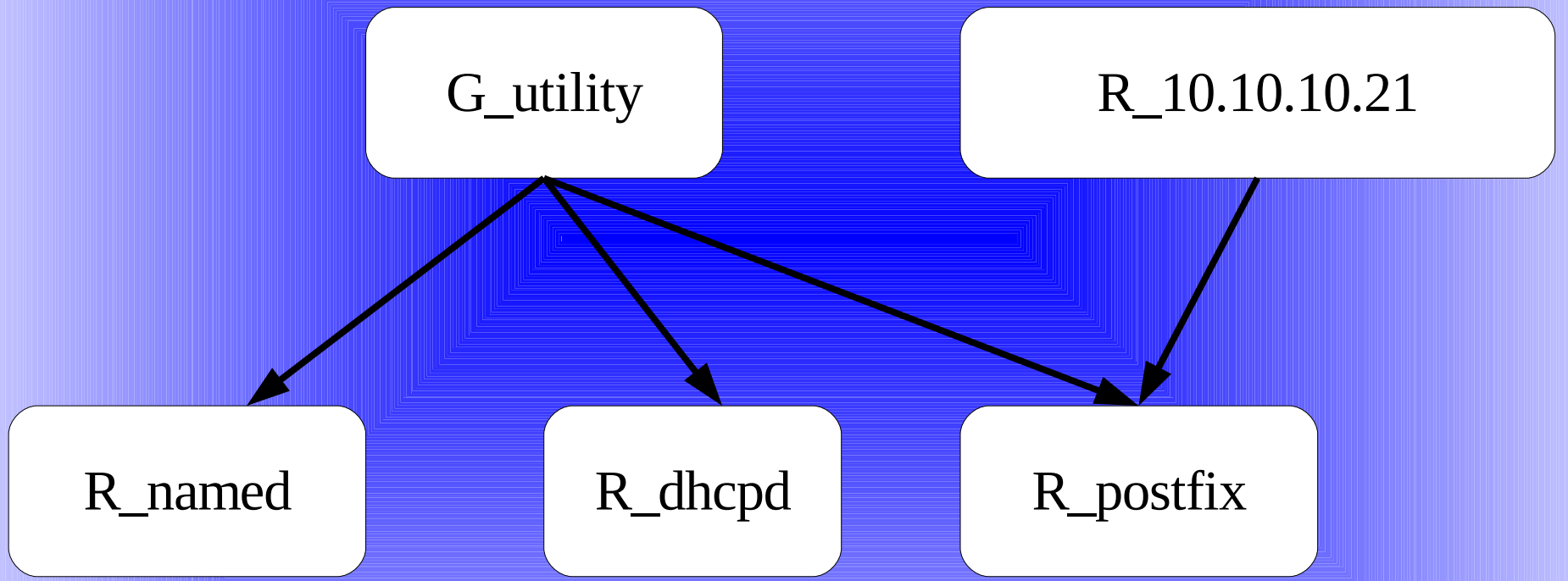
STONITH "clone" resource(s)

```
<clone id="fencing">
  <instance_attributes>
    <attributes>
      <nvpair id="1" name="clone_max" value="2"/>
      <nvpair id="2" name="globally_unique" value="false"/>
    </attributes>
  </instance_attributes>
  <primitive id="fencing_op" class="stonith" type="ibmhmc">
    <operations>
      <op id="1" name="monitor" interval="5s" timeout="20s"
                 prereq="nothing"/>
      <op id="2" name="start" timeout="20s" prereq="nothing"/>
    </operations>
    <instance_attributes>
      <attributes>
        <nvpair id="1" name="ip" value="192.168.224.99"/>
      </attributes>
    </instance_attributes>
  </primitive>
</clone>
```

Creating Detailed Ordering Constraints

- Ordering constraints can apply between any two resource objects – primitive, group or clone
  - The main kind of ordering constraint that is used is start_after
    - There is also a start_before constraint
    - There may also be stop_after, and stop_before constraints :-D
  - Although these others provide flexibility, they're not commonly used
- Ordering constraints can make start and stop actions complete faster than groups

Sample Ordering Constraint Graph



Co-location Constraints

- The XML DTD permits both mandatory and optional co-location constraints
- As of 2.0.8, both mandatory co-location constraints are supported.
- As of 2.0.8, co-location constraints are fully asymmetric.

Sample Co-location Constraints

```
<rsc_co-location id="C_10.10.10.21"
from="R_10.10.10.21" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_postfix"
from="R_postfix" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_dhcpd"
from="R_dhcpd" to="G_utility" score="INFINITY"/>

<rsc_co-location id="C_named"
from="R_named" to="G_utility" score="INFINITY"/>
```

Node attributes

- Nodes can be assigned arbitrary attributes, which can then be used in resource location rules

```
<node id="uuid1" uname="nodeA" type="normal">
  <instance_attributes id="uuid1:attrs">
    <attributes>
      <nvpair id="uuid1:installed_ram"
                name="installed_ram" value="1024"/>
      <nvpair id="uuid1:pingcount”
                name="pingcount" value="2"/>
    </attributes>
  </instance_attributes>
</node>
```

Failing over on arbitrary conditions

- **pingd** is a worked example of how to fail over on arbitrary conditions
- **attrd_updater** is what **pingd** uses to modify the CIB
- **attrd** implements the idea of hysteresis in setting values into the CIB – allowing things to settle out into stable configurations before failing over – to avoid false failovers
- **pingd** asks heartbeat to notify it when ping nodes come and go. When they do, it invokes **attrd_updater** to make the change, and **attrd** updates the CIB – after a delay
- You can use **attrd_updater** yourself to do this for any condition you can observe

Master/Slave Resources

- Master/Slave resources represent resources whose “running state” is manged by heartbeat, and you can use this to create dependencies which are state-dependent
- The states a master/slave resource can be in are:
  - master
  - slave
  - stopped
- All resources are initially started in slave mode, then *promoted* to master afterwards

Split-site (“stretch”) clusters

- Geographic-scale communications are never as reliable as local communications
- Fencing techniques (STONITH, SCSI reserve) all require highly reliable communications, don't work remotely
- Split-site clusters cannot rely on fencing in most cases
- Quorum without fencing must be used instead
- Two-site quorum without fencing is problematic
- Linux-HA introduces a quorum server to solve this problem