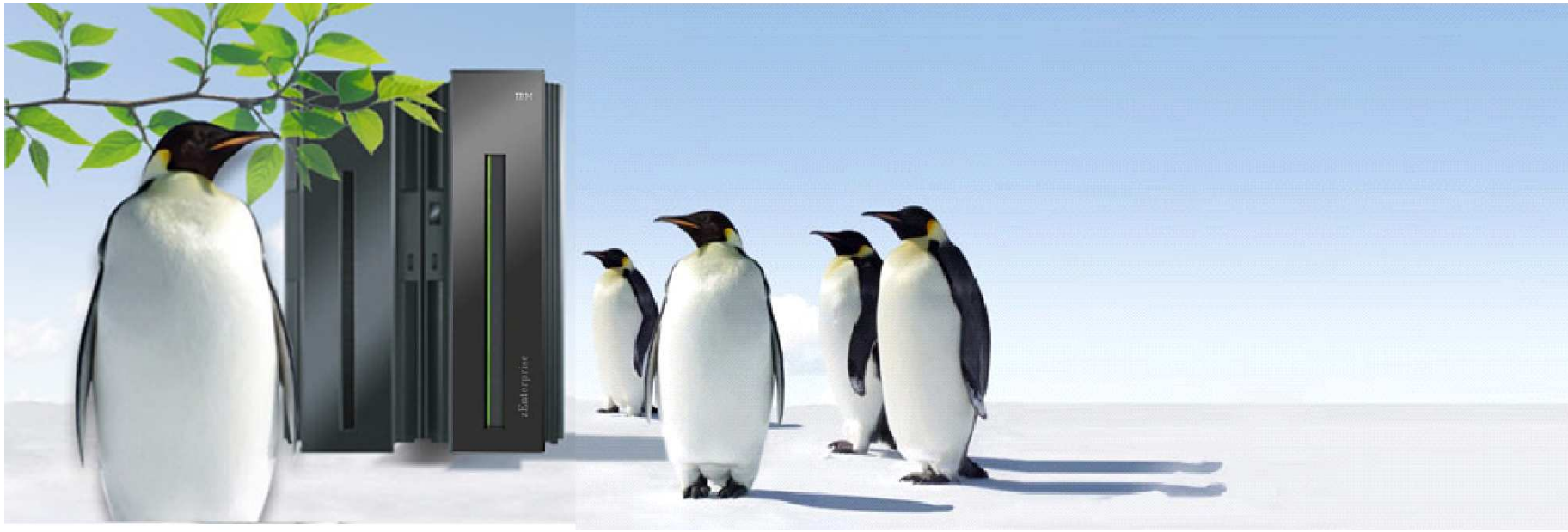
Dr. Stefan Reimbold
IBM Research & Development

Running Linux-HA on a IBM System z



Trademarks

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- **High Availability**
- Challenges
- Linux-HA
- Examples

Computer Cluster

A computer cluster consists of a set of loosely connected computers that work together so that in many respects they can be viewed as a single system.

(wikipedia: Computer Cluster)

High Availability Cluster

- When one node fails another node is taking over IP address, services, etc.
- The key of High Availability is avoiding single points of failure
- High Availability adds costs because you need redundant resources

High Availability

- Amazon
  - 2005 - 3 hours offline first European sites spreading to amazon.com
  - 2010 - 30 min offline for Europe during Christmas time
- protecting mission-critical applications
- 24x7 availability
- keep interruptions as short as possible

High Availability

- 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
- Adds one 9 to the availability

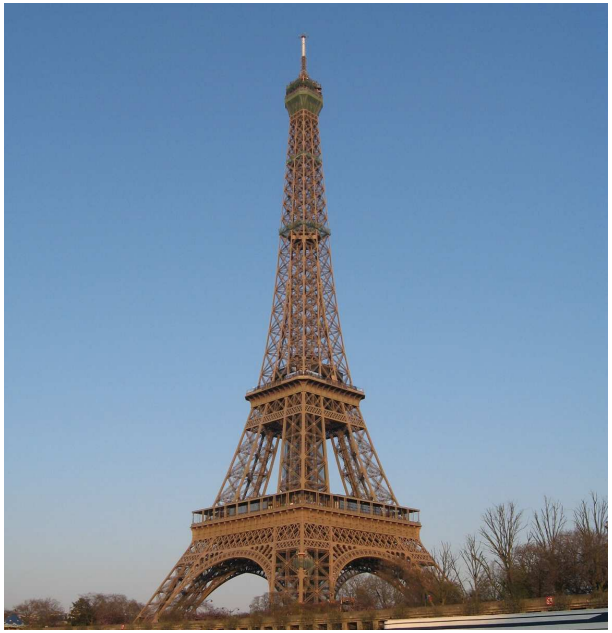
99.9%	9 h	
99.99%	53 min	
99.999%	5 min	System z Application Availability
99.9999%	32 sec	
99.99999%	3 sec	

High Availability

99.9% Washington DC 250 miles

High Availability

99.9%	Washington DC	250 miles
99.99%	Paris	2500 miles



High Availability

99.9%	Washington DC	250 miles
99.99%	Paris	2500 miles
99.999%	Around the world	25000 miles

High Availability

99.9%	Washington DC	250 miles
99.99%	Paris	2500 miles
99.999%	Around the world	25000 miles
99.9999%	Moon	250000 miles

High Availability

- 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
- Adds one 9 to the availability

99.9%	9 h	
99.99%	53 min	
99.999%	5 min	System z Application Availability
99.9999%	32 sec	
99.99999%	3 sec	

- It's like respawn on a cluster-wide scale
  Like init on steroids
- HA Clustering is designed to recover from single faults

High Availability

- The Three R's of High Availability
  - Redundancy
  - Redundancy
  - Redundancy
- This might sound redundant, but that's probably ok
- Most Single Points of Failure are managed by redundancy
- HA Clustering is a technigue to provide and manage redundancy

HA vs DR

- High Availability
  - Fast and reliable inter-node communication
  - Failover is cheap
  - Failover needs to be fast - measured in seconds
- Disaster Recovery
  - No special requirements for inter-node communication
  - Failover is expensive - sometimes not automatic
  - Automatic failback may be impossible
  - Failover times often longer - can be hours

Agenda

- High Availability
- **Challenges**
- Linux-HA
- Examples

Challenges

- Early detection
  - To keep the offline time as short as possible a failure has to be detected fast
  - Risk of false positive interpretation and unnecessary failover
  - Keep offline time as short as possible (mean-time-to-repair MTTR)
  - Reliable detection by reliable internal communication
- Split-Brain
- Quorum
- Fencing
- Data sharing

Challenges

- Early detection
- Split-Brain
  - When the connection between the nodes fails, all nodes can still be active but detect the other as failing
  - The status of an unreachable node is unkonwn
  - Especially in geographical displaced systems
- Quorum
- Fencing
- Data sharing

Challenges

- Early detection
- Split-Brain
- Quorum
  - Algorithms to decide which part of the cluster is active
  - A remote quorum server can decide more reliably
  - Quorum server is in client perspective
- Fencing
- Data sharing

Challenges

- Early detection
- Split-Brain
- Quorum
- Fencing
  - Keep a node that was detected as failed from working to prevent damage
  - self-fencing
  - STONITH
- Data sharing

Challenges

- Early detection
- Split-Brain
- Quorum
- Fencing
- Data sharing
  - Mirror data e.g. DRBD
  - Synchronize database

Agenda

- High Availability
- Challenges
- **Linux-HA**
- Examples

High Availability Solutions

- Tivoli System Automation
- Linux-HA
- HACMP for AIX

Tivoli System Automation

- Automation Manager
  - Starting
  - Stopping
  - Restarting
  - Failover
- Supports
  - Quorum
  - Dead-man switch
  - disk and network tiebreaker
- Advantages
  - Policy-based and Goal-driven automation
  - Integrated in Tivoli Systems Management Portfolio

Tivoli System Automation

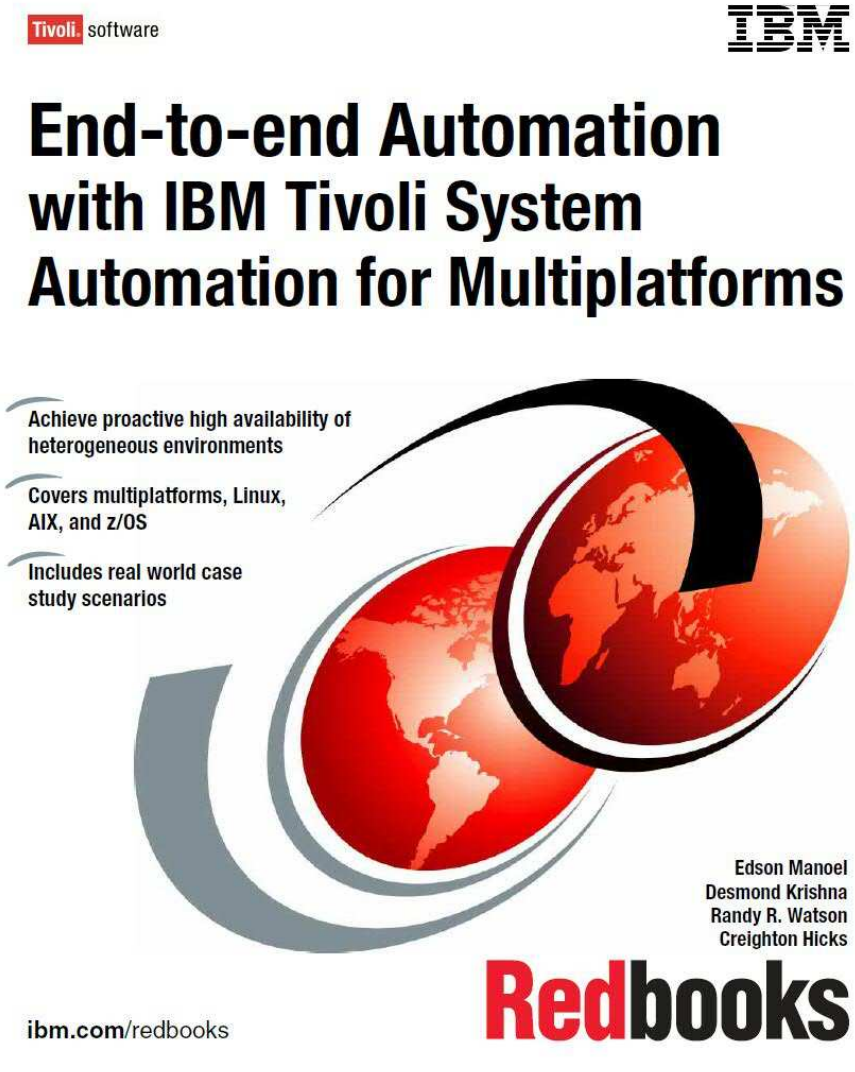
- Apache
- HTTP WebServer
- IBM Tivoli Directory Server
- inetd
- MaxDB SAP 7.5
- NFS Server
- Samba
- Sendmail
- TSM
- TWS 8.3
- WAS 6.0
- WebSphere MQ 7
- DP for my SAP 5.3
- TSAM - Tivoli Service Automation Manager

Tivoli System Automation

samadmin tool

- Domain Management
- Resource and Group Management
- Equivalency Management
- Relationship Management
- TieBreaker Management
- Cluster Overview

RedBooks



Linux-HA

- Components
  - heartbeat
    - Messaging between nodes to make sure they are available and take action if not
  - cluster-glue
    - Everything that is not messaging layer and not resource manager
  - resource-agents
    - Scripts that start/stop clustered services
    - Templates and scripts for many applications
  - pacemaker
    - cluster resource manager (CRM)

Linux-HA

- Components
  - heartbeat
    - Messaging between nodes to make sure they are available and take action if not
  - cluster-glue
    - Everything that is not messaging layer and not sesource manager
  - resource-agents
    - Scripts that start/stop clustered services
    - Templates and scripts for many applications
  - pacemaker
    - cluster resource manager (CRM)
- Optional
  - STONITH
    - Shoot The Other Node In The Head
    - Fence a node to ensure unique access to data and reliably manage shared storage

heartbeat

- Heartbeat connection between nodes
  - HiperSockets
  - VLAN
  - OSA Ethernet
- Heartbeat timeout detemines MTTR
- Integrated IP address takeover
- Integrated filesystem support

Applications

Examples

- IP address
- Webserver
- Firewall
- DNS
- DB2

- Complex scenarios can be managed with constraints and dependencies

Advantages

- Strongly authenticated communication
- Highly extensible
- Connectivity monitoring using voting protocol
- Subsecond failure detection
- SAF data checkpoint API
  - store application state to disk
    used to restore state in failover
  - not working if state changes to fast for disk
  - SAF provides an API to replicate data
    without storing to disk
- Standard init scripts as resource agents
- API for monitoring and control

Limitations

- Linux-HA can not provide 100% availability
- Applications which can not deal with the timeout need to be cluster aware
  - i.e. store the state to disk for restore
  - or use SAF data checkpoint API which provides a replication API for faster change rates
- Short outage due to failover detection
- TCP connection is broken

Linux-HA on System z

- seems like the system is redundant and highly available

but

- Hardware is redundant and highly available

Linux-HA on System z

- Improve availability of applications
- Shared Resources in z/VM
  - Standby nodes can use overcommitment of memory and PUs
- z/VM Guests as test systems
- Use HiperSockets for reliable cluster communication
- Take care about scheduling issues
- Time to page in inactive guest

Linux-HA on System z

- Packages are available as extension for SUSE
  - SLES 10
  - SLES 11
- Packages can be compiled for RedHat
  - RHEL 4
  - RHEL 5

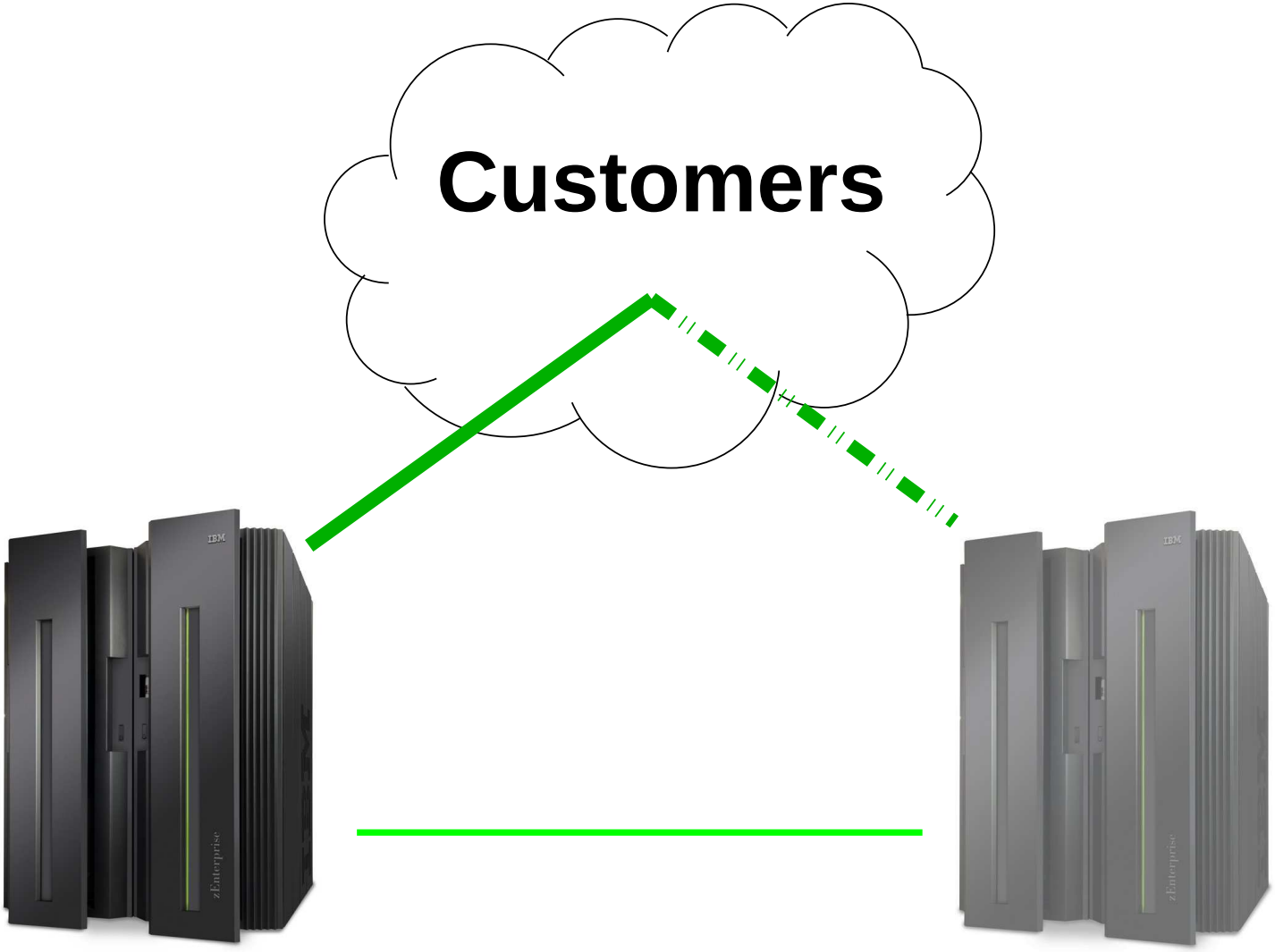
Linux-HA Tools

- `crm`
- `crm_mon`
- `crm_verify`
- `crm_simulate`
- `crm_resource`
- `crm_gui`

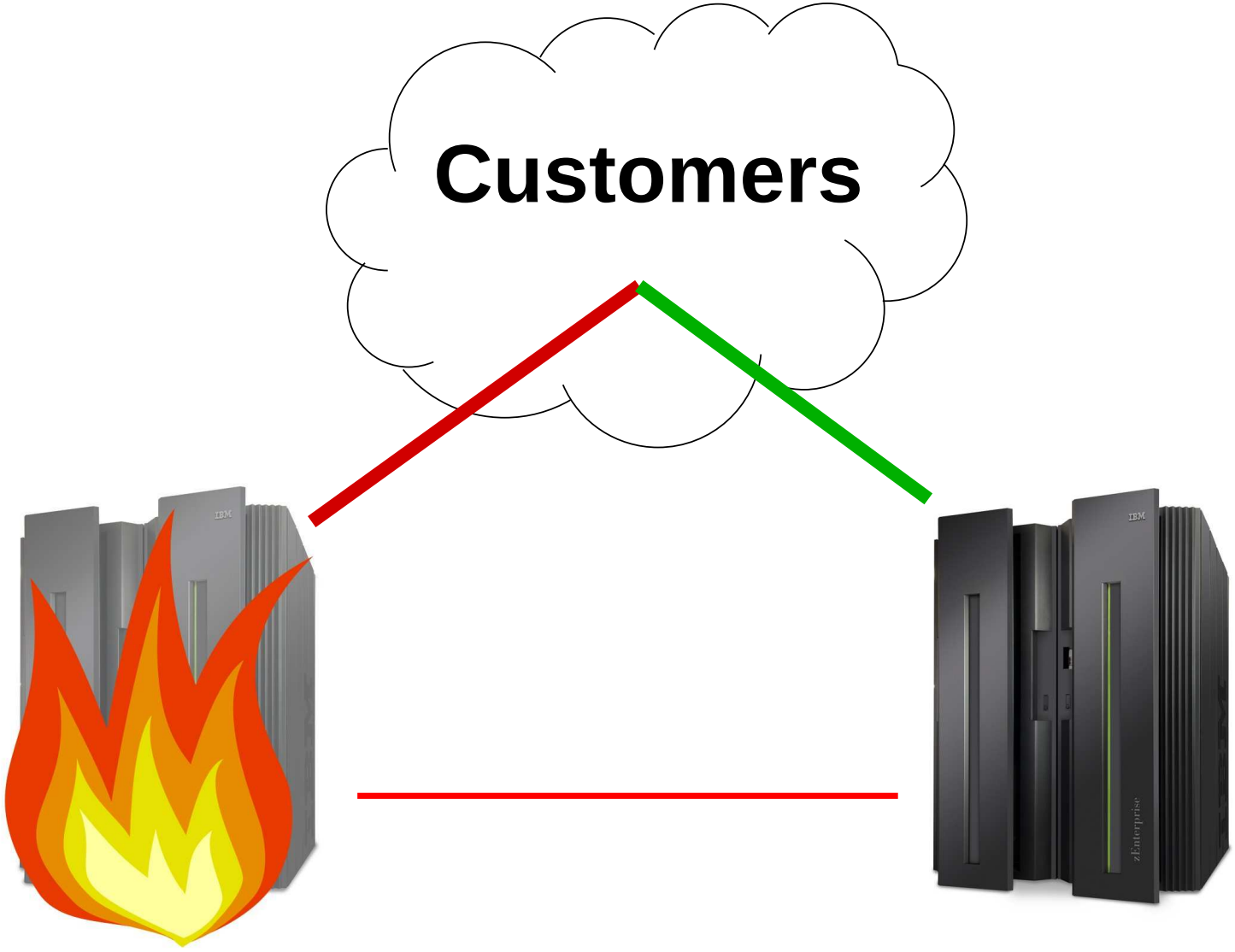
Agenda

- High Availability
- Challenges
- Linux-HA
- **Examples**

2 Node - Active-Passive



2 Node - Active-Passive

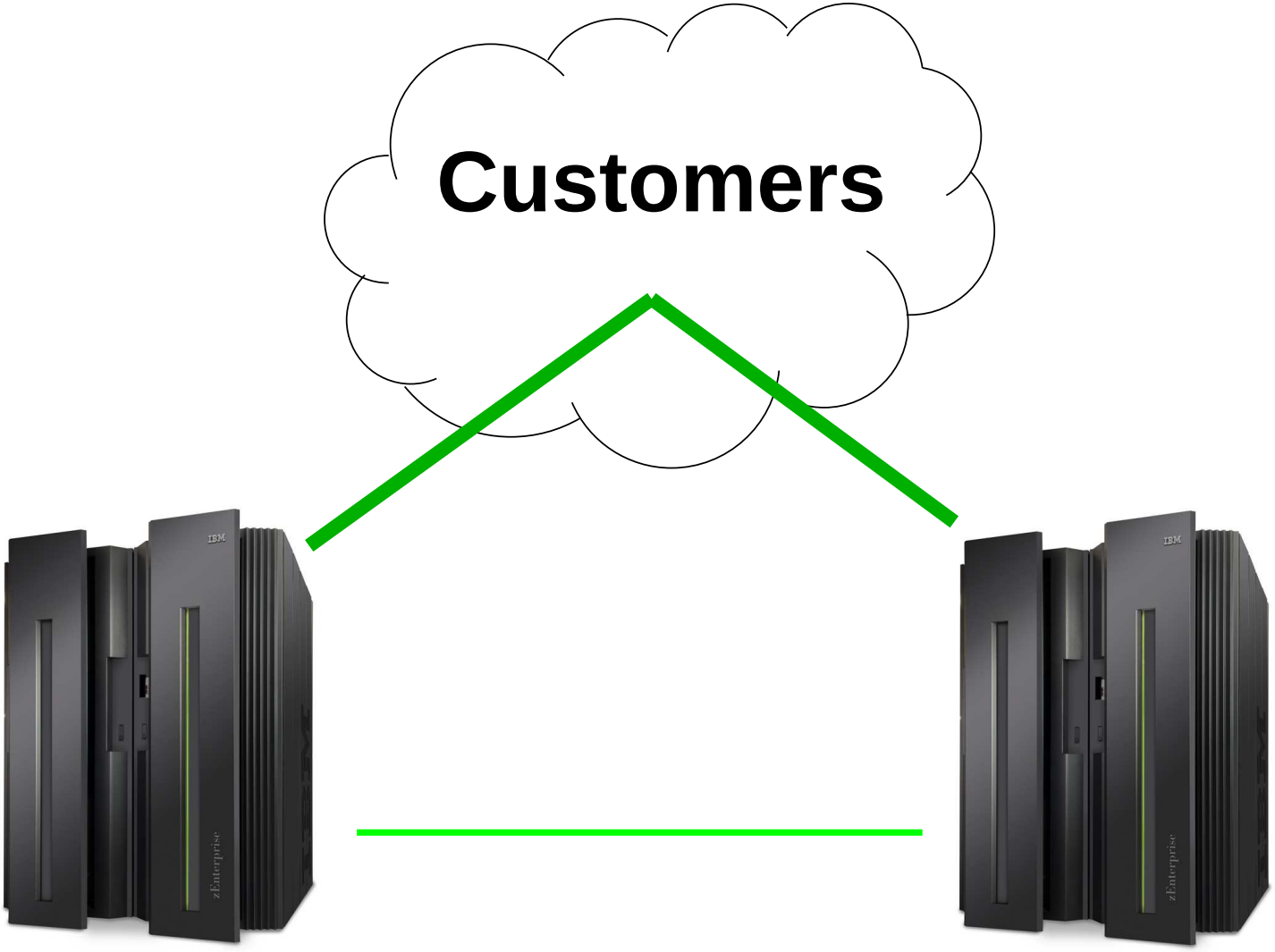


2 Node - Active-Passive

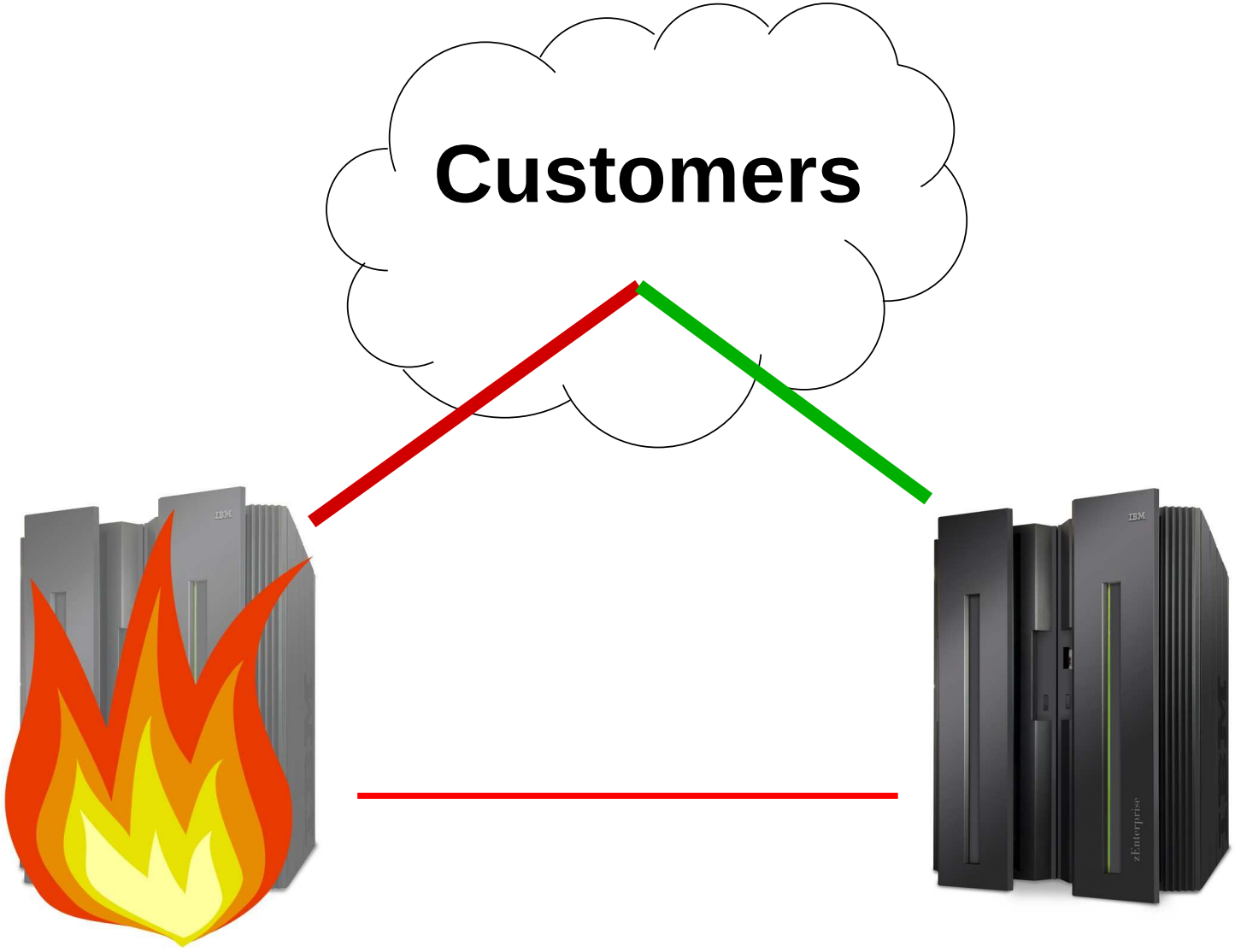
- Higher costs
- In good case
  - No idle resources

- In case of failure
  - Constant performance
  - Application topology remains unchanged

2 Node - Active-Active



2 Node - Active-Active

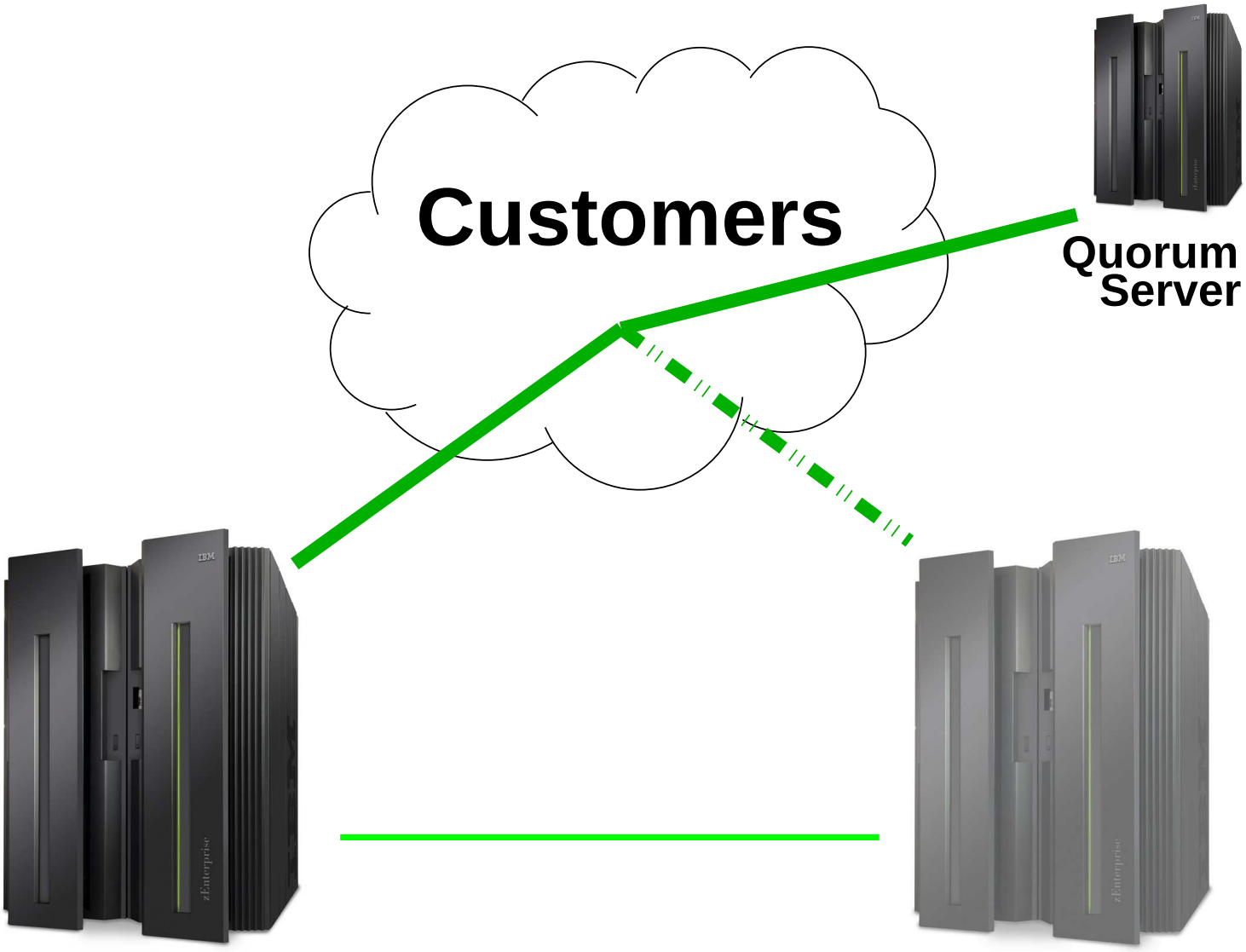


2 Node - Active-Active

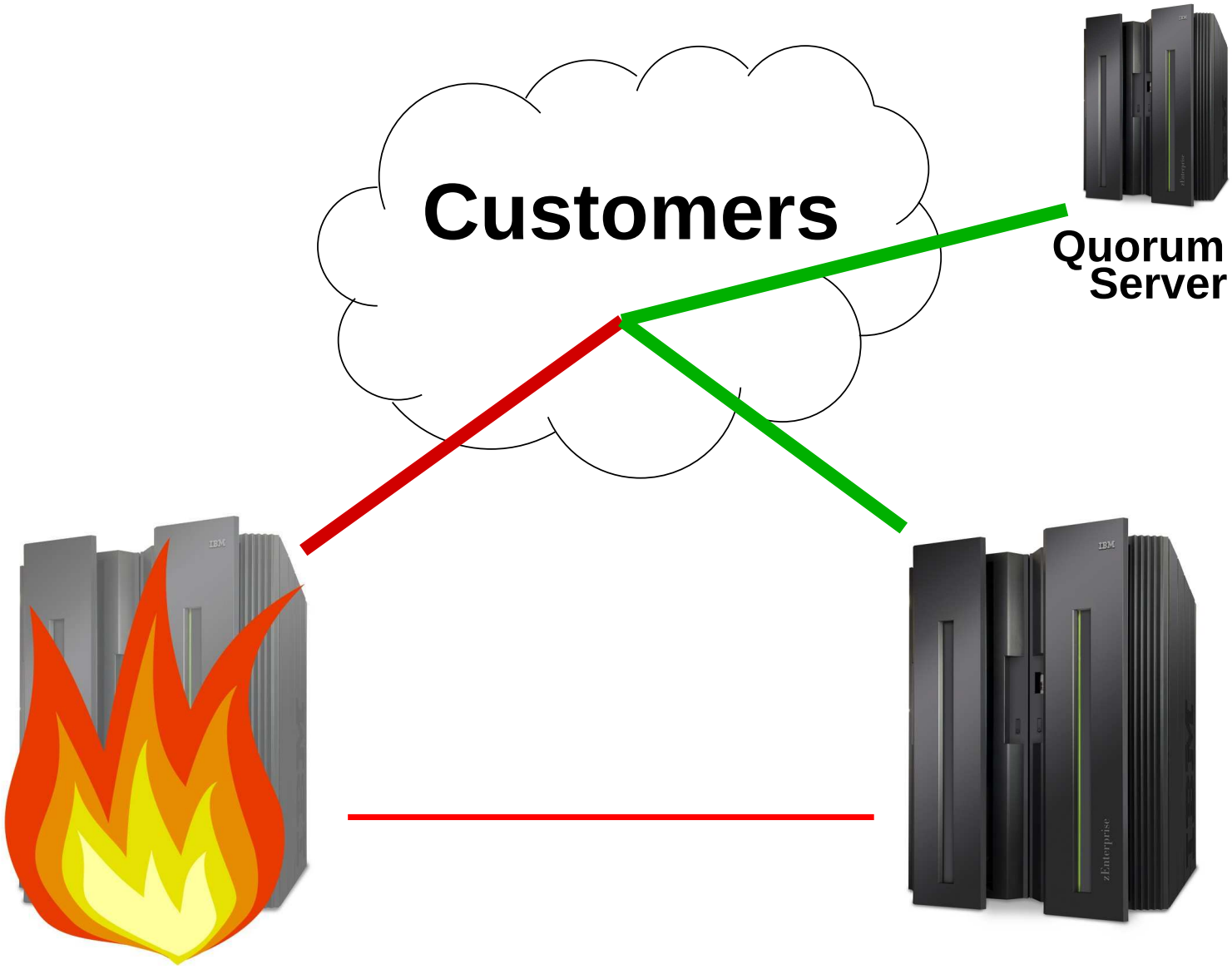
- Lower costs
- In good case
  - No idle resources

- In case of failure
  - Degradation of performance
  - Different application topology

3 Nodes with Quorum



Quorum Server



Quorum Server

- Costs for Quorum server
- Monitoring from customer/service perspective

- In case of failure
  - No split brain situation
  - Application topology remains unchanged

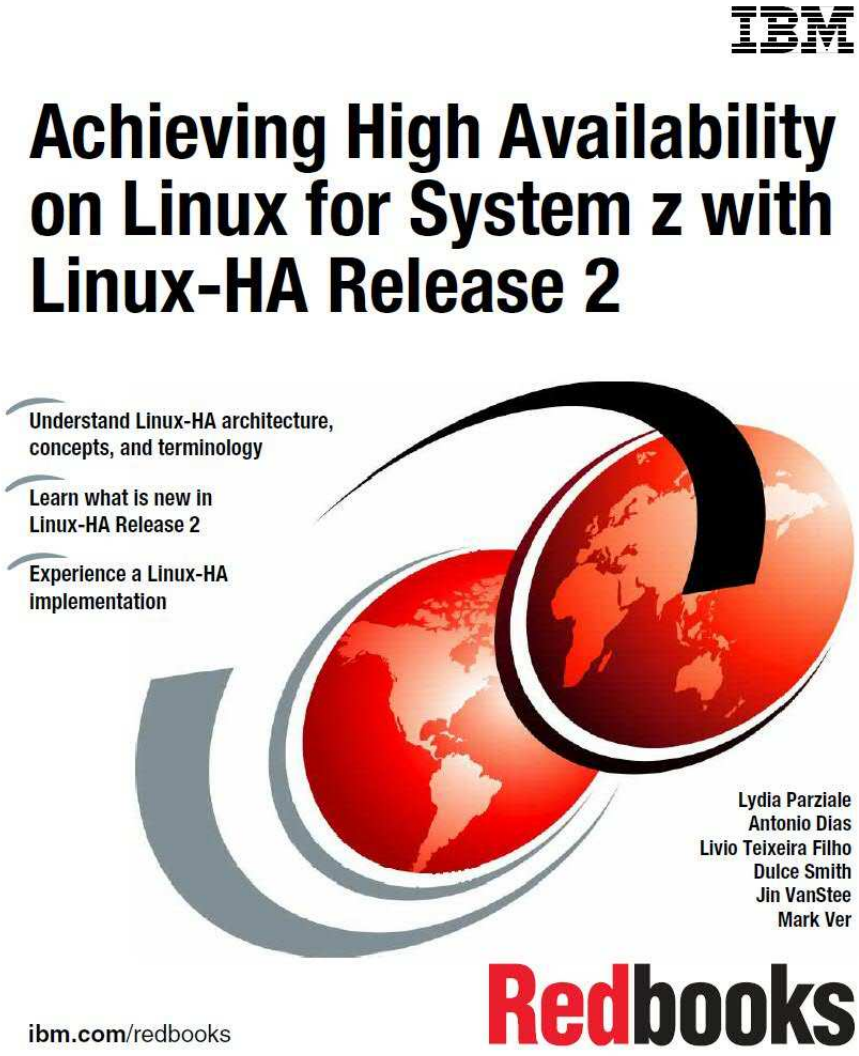
Summary

- Linux-HA can improve application availability
- Resource Agents for many applications
- Leverage z/VM resource sharing
  - Redundant resources
  - z/VM guests as test systems
- Systems have to be carefully designed and thoroughly tested

Links

- Linux-HA Wiki - Talks and Papers
  `http://linux-ha.org/wiki/Talks_and_Papers`
- IBM Redbooks
  `http://www.redbooks.ibm.com`

RedBooks



Thank You !

- Alan Robertson
  for using his Linux-HA Tutorial

Questions ?

Dr. Stefan Reimbold

Linux on System z Service Team Lead

Schoenaicher Strasse 220
D-71032 Boeblingen
Mail: Postfach 1380
D-71003 Boeblingen

Phone +49-7031-16-2368
Stefan.Reimbold@de.ibm.com