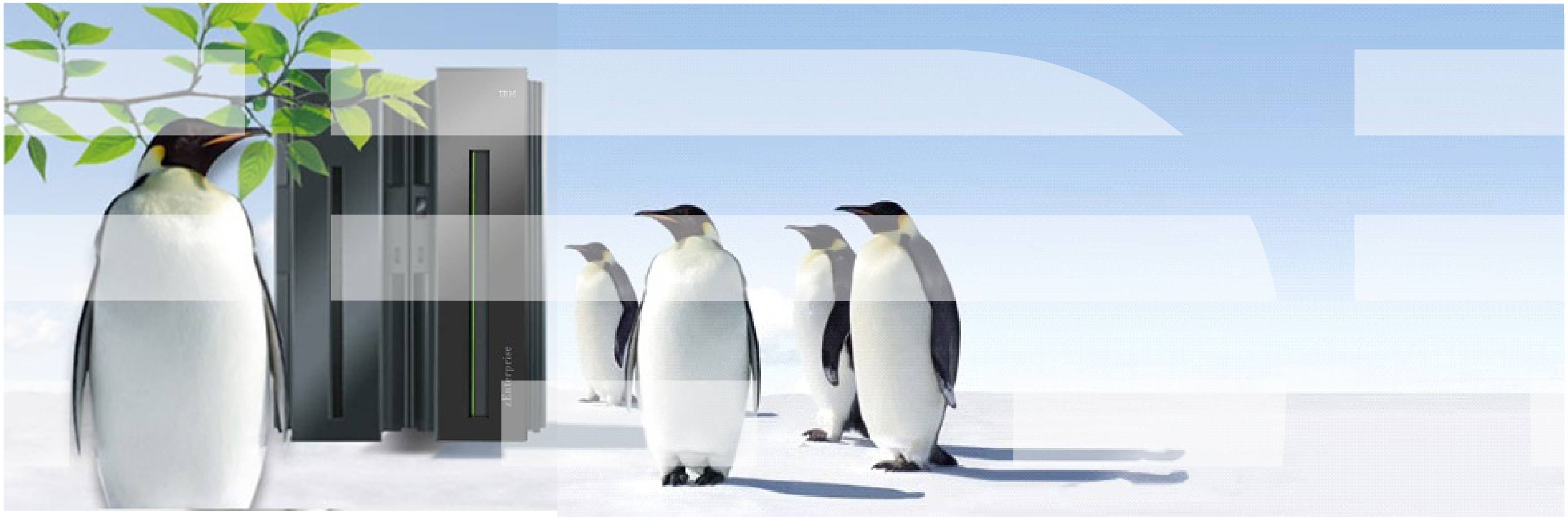
Running Linux-HA on a IBM System z

Martin Schwidefsky
IBM Lab Böblingen, Germany

February 8 2013



Trademarks & Disclaimer

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

IBM, the IBM logo, BladeCenter, Calibrated Vectored Cooling, ClusterProven, Cool Blue, POWER, PowerExecutive, Predictive Failure Analysis, ServerProven, System p, System Storage, System x , System z, WebSphere, DB2 and Tivoli are trademarks of IBM Corporation in the United States and/or other countries. For a list of additional IBM trademarks, please see http://ibm.com/legal/copytrade.shtml.

The following are trademarks or registered trademarks of other companies: Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries or both Microsoft, Windows,Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc. InfiniBand is a trademark of the InfiniBand Trade Association.
Other company, product, or service names may be trademarks or service marks of others.

NOTES: Linux penguin image courtesy of Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Users of this document should verify the applicable data for their specific environment. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
Information is provided “AS IS” without warranty of any kind. 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 are suggested US list prices and are subject to change without notice. Starting price may not include a hard drive, operating system or other features.
Contact your IBM representative or Business Partner for the most current pricing in your geography. Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use. The information could include technical inaccuracies or typographical errors.
Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any

Agenda

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

What is it ?

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

High availability is a system design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period.

(wikipedia: High Availability)

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 cost because you need redundant resources.

High Availability

- Amazon
  - 2005 – 3 hours offline, first the European sites, then spreading to amazon.com
  - 2010 – 30 minutes 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

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

- Add on 9 to the availability

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

High Availability

- Compared to distances

99.9%	Moon	250000 miles

High Availability

- Compared to distances

99.9%	Moon	250000 miles
99.99%	Around the world	25000 miles

High Availability

- Compared to distances

99.9%	Moon	250000 miles
99.99%	Around the world	25000 miles
99.999%	New York City	2500 miles

High Availability

- Compared to distances

99.9%	Moon	250000 miles
99.99%	Around the world	25000 miles
99.999%	New York City	2500 miles
99.9999%	Las Vegas	250 miles

High Availability

- Compared to distances

99.9%	Moon	250000 miles
99.99%	Around the world	25000 miles
99.999%	New York City	2500 miles
99.9999%	Las Vegas	250 miles
99.99999%	LA Airport	25 miles

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 technique to provide and manage redundancy

Agenda

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

HA 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 fail-over
  - Keep offline time as short as possible (mean-time-to-repair MTTR)
  - Reliable detection by reliable internal communication
- Split-Brain
- Quorum
- Fencing
- Data Sharing

HA challenges

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

HA 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

HA 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

HA 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
  - Fail-over
- 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

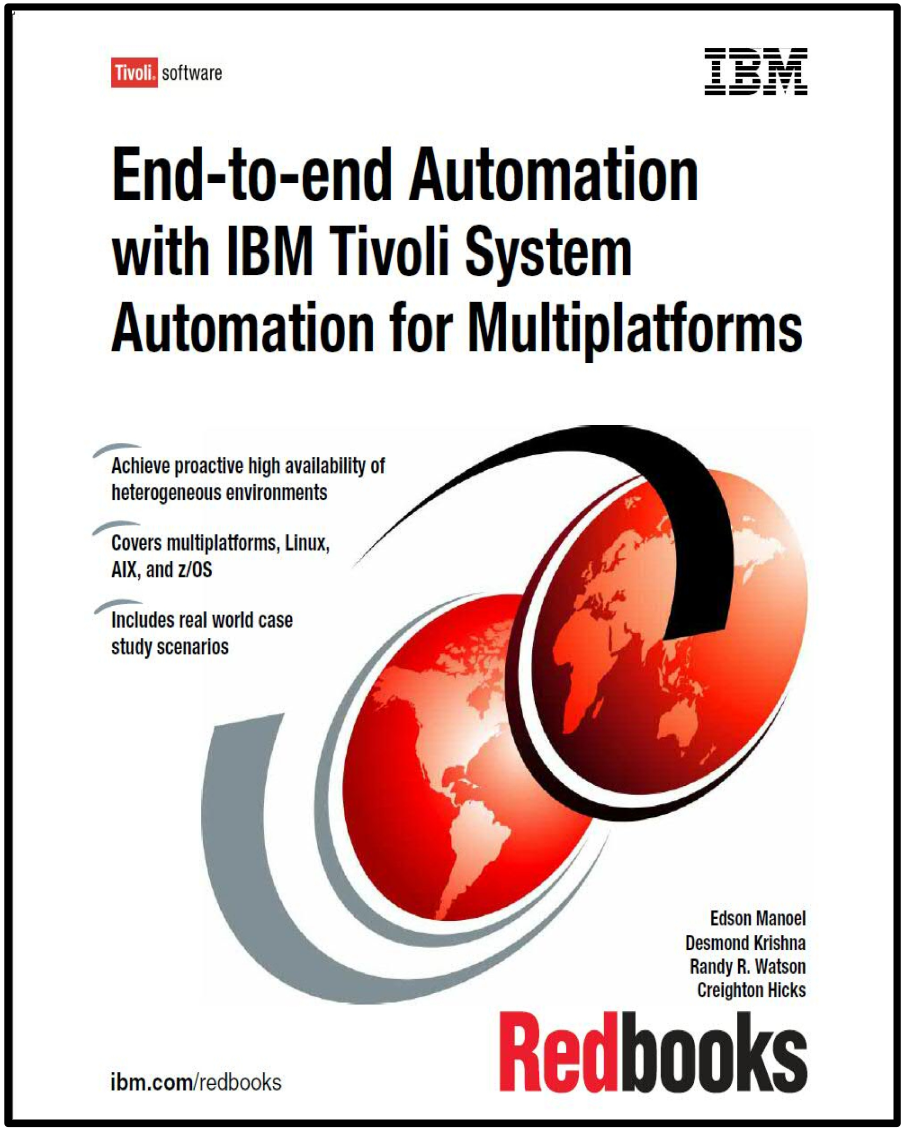
- Apache
- HTTP WebServer
- IBM Tivoli Directory Server
- inetd
- MaxDB SAP 7.5
- NFS Server
- Samba
- Sendmail
- TSM
- TWS 8.3
- WAS 6.0
- Websphere MQ 7
- DP for my SAP 5.3
- TSAM – Tivoli Service Automation Manager

Tivoli System Automation

samadmin tool

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

Tivoli System Automation RedBook



Tivoli. software IBM

End-to-end Automation with IBM Tivoli System Automation for Multiplatforms

- Achieve proactive high availability of heterogeneous environments
- Covers multiplatforms, Linux, AIX, and z/OS
- Includes real world case study scenarios

Edson Manoel
Desmond Krishna
Randy R. Watson
Creighton Hicks

ibm.com/redbooks

Redbooks

Linux-HA components

- 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

- 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)
- Optional
  - STONITH
    - Shoot The Other Node In The Head
    - Fence a node to ensure unique access to data and reliably manage shared storage

Linux-HA heartbeat

- Heartbeat connection between nodes
  - HiperSockets
  - VLAN
  - OSA Ethernet
- Heartbeat timeout determines MTTR
- Integrated IP address takeover
- Integrated file system support

Linux-HA applications

- Examples
  - IP address
  - Webserver
  - Firewall
  - DNS
  - DB2
  - Complex scenarios can be managed with constraints and dependencies

Linux-HA advantages

- Strongly authenticated communication
- Highly extensible
- Connectivity monitoring using voting protocol
- Sub-second failure detection
- SAF data checkpoint API
  - store application state to disk used to restore state in fail-over
  - 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

Linux-HA 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 fail-over detection
- TCP connection is broken

Linux-HA on System z

- System is redundant and highly available already
- Hardware is redundant and highly available
- 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

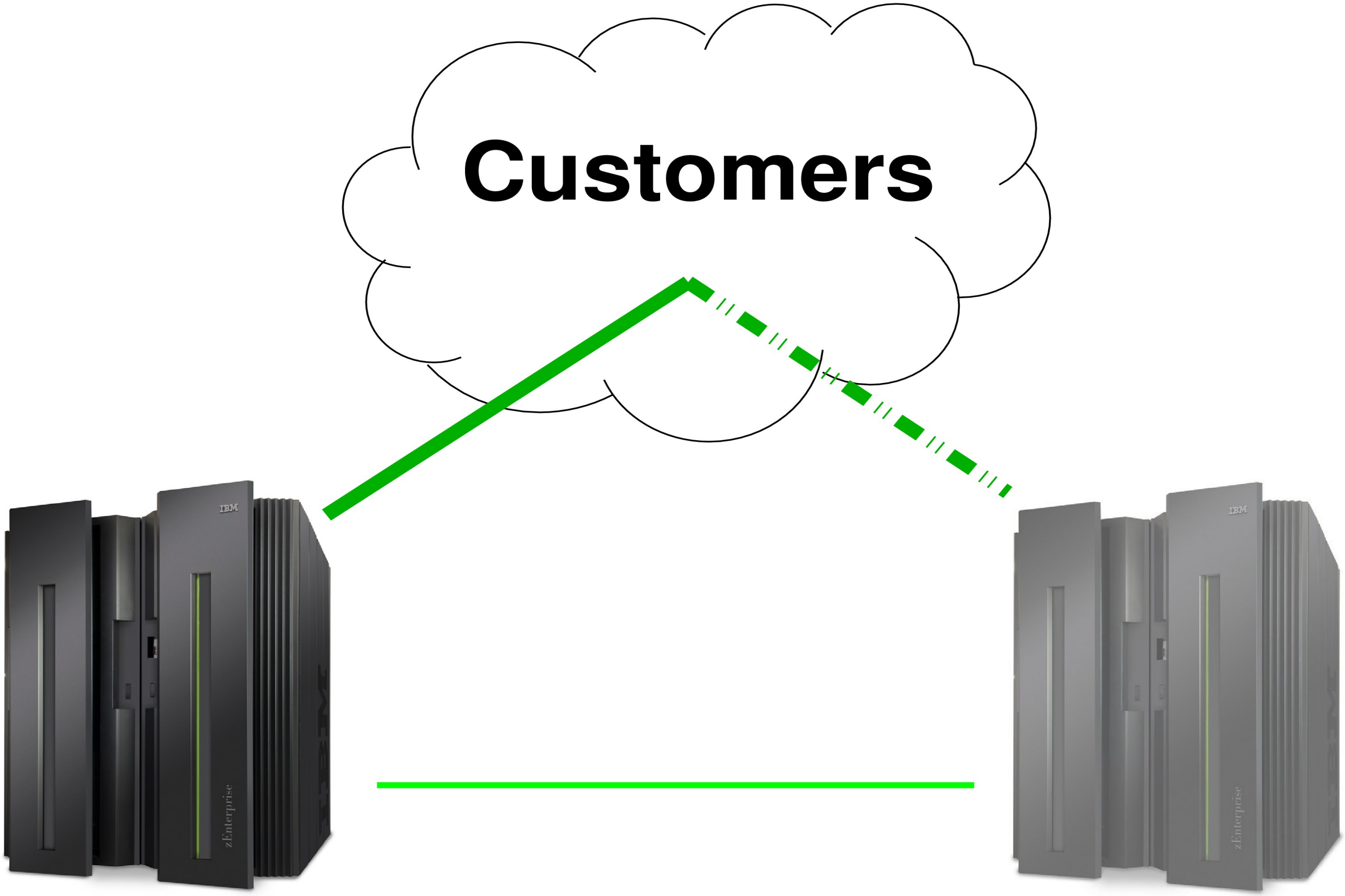
Linux-HA on System z

- Packages are available as extension for SuSE
  - SLES 10
  - SLES 11
- Packages can be compiled for RedHat
  - RHEL 4
  - RHEL 5
  - RHEL 6

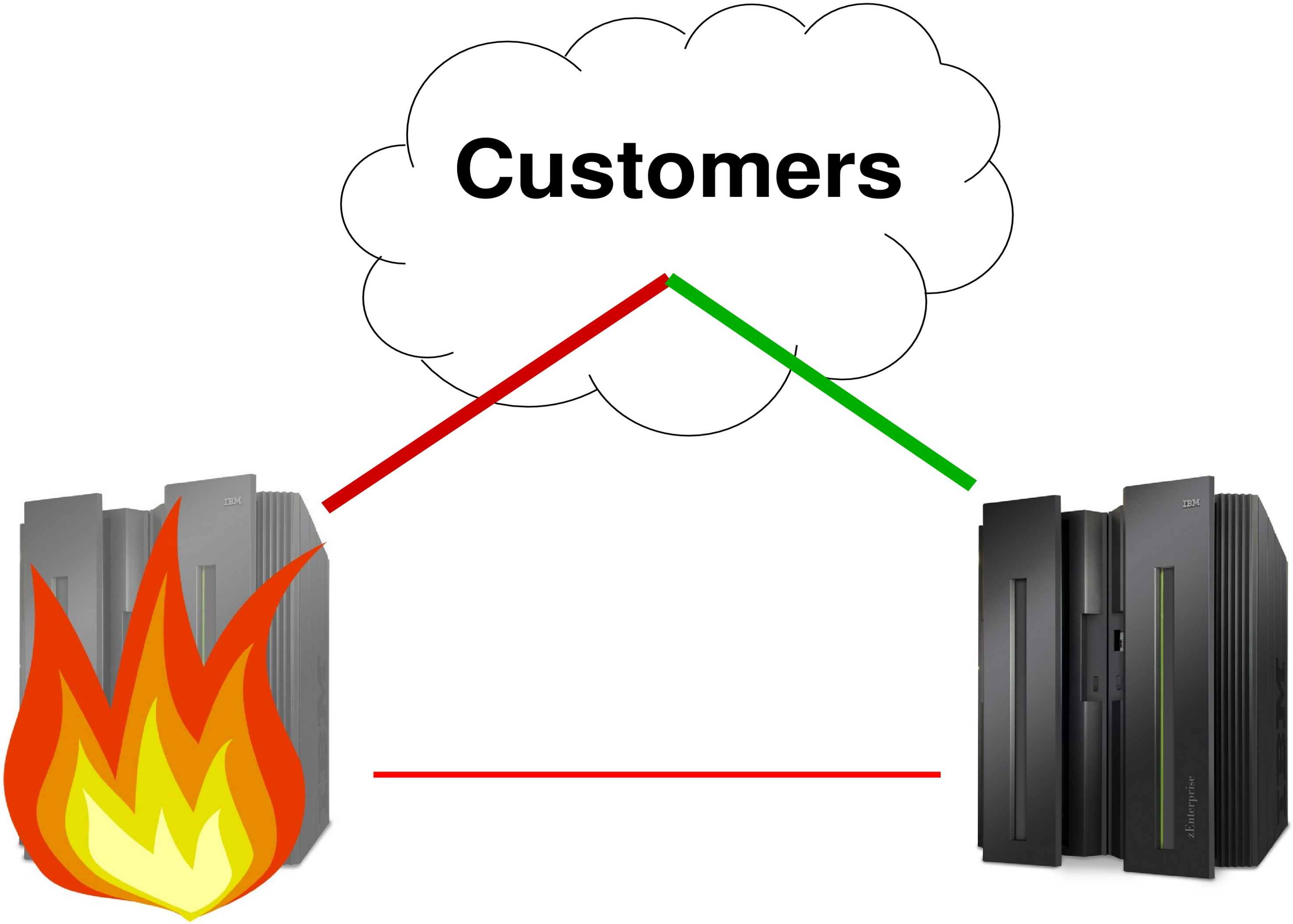
Agenda

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

2 Node - Active-Passive



2 Node - Active-Passive

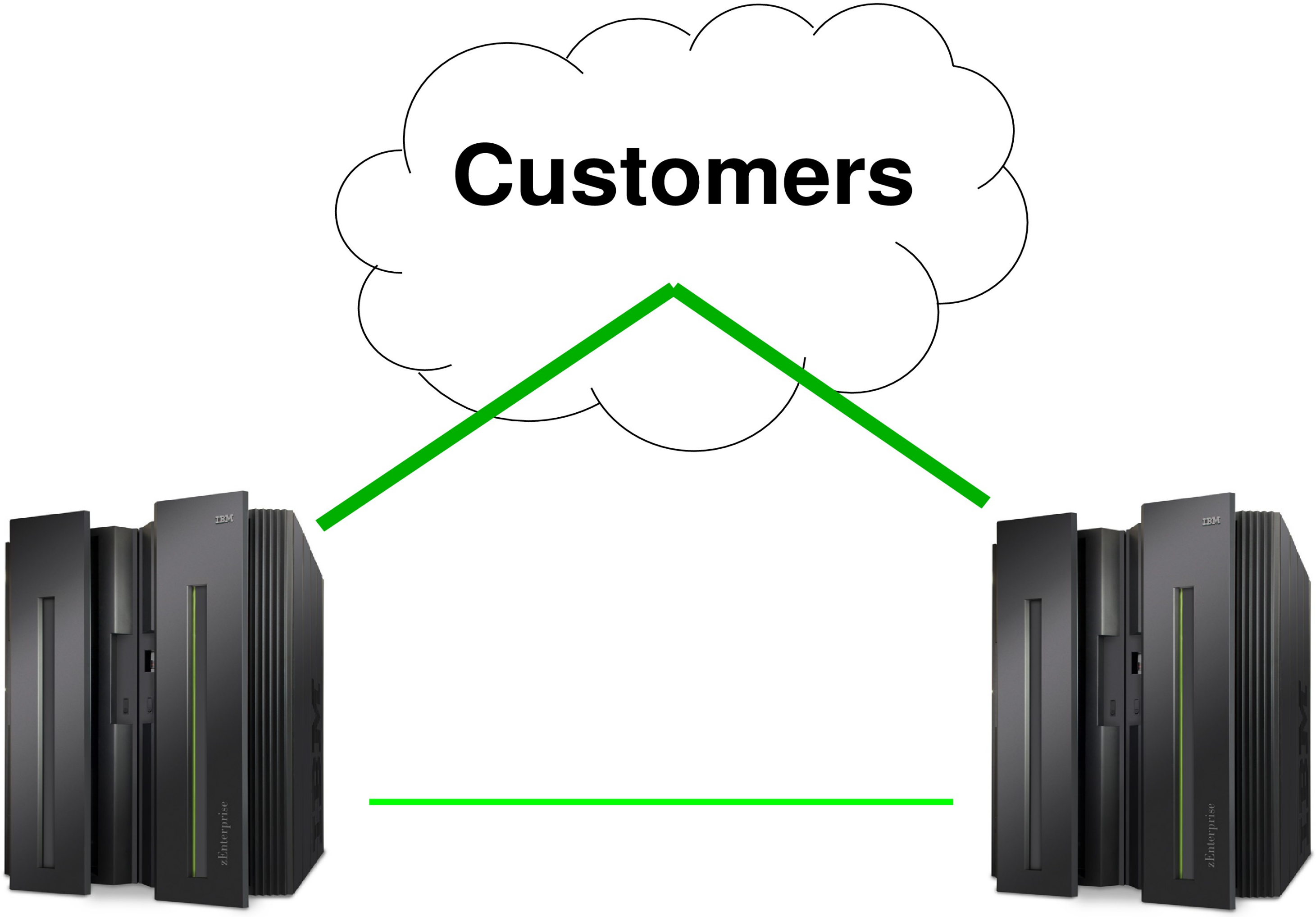


2 Node - Active-Passive

- Higher costs
- In good case
  - One side idle

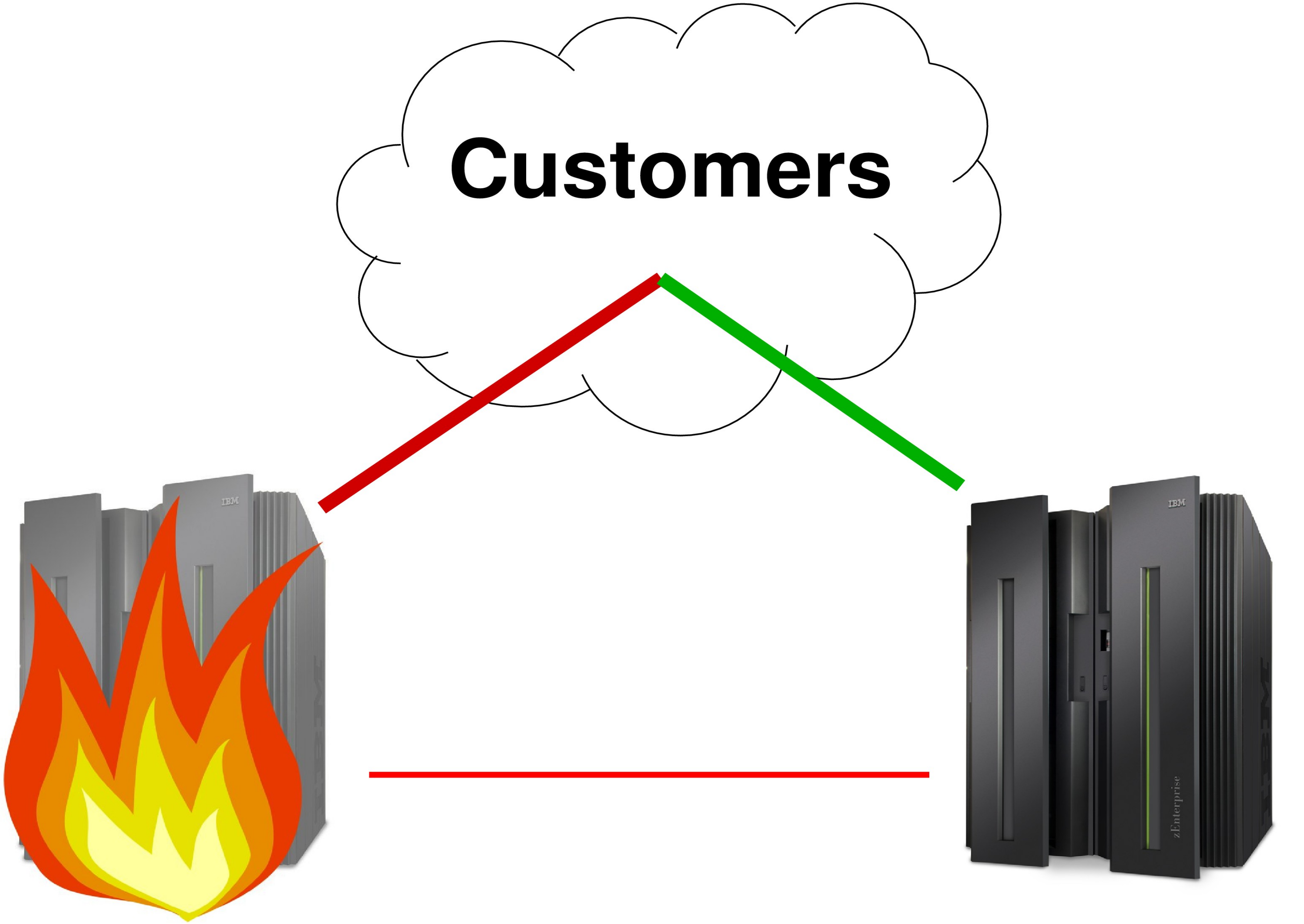
- In case of failure
  - Constant performance
  - Application topology remains unchanged

2 Node - Active-Active



Customers

2 Node - Active-Active

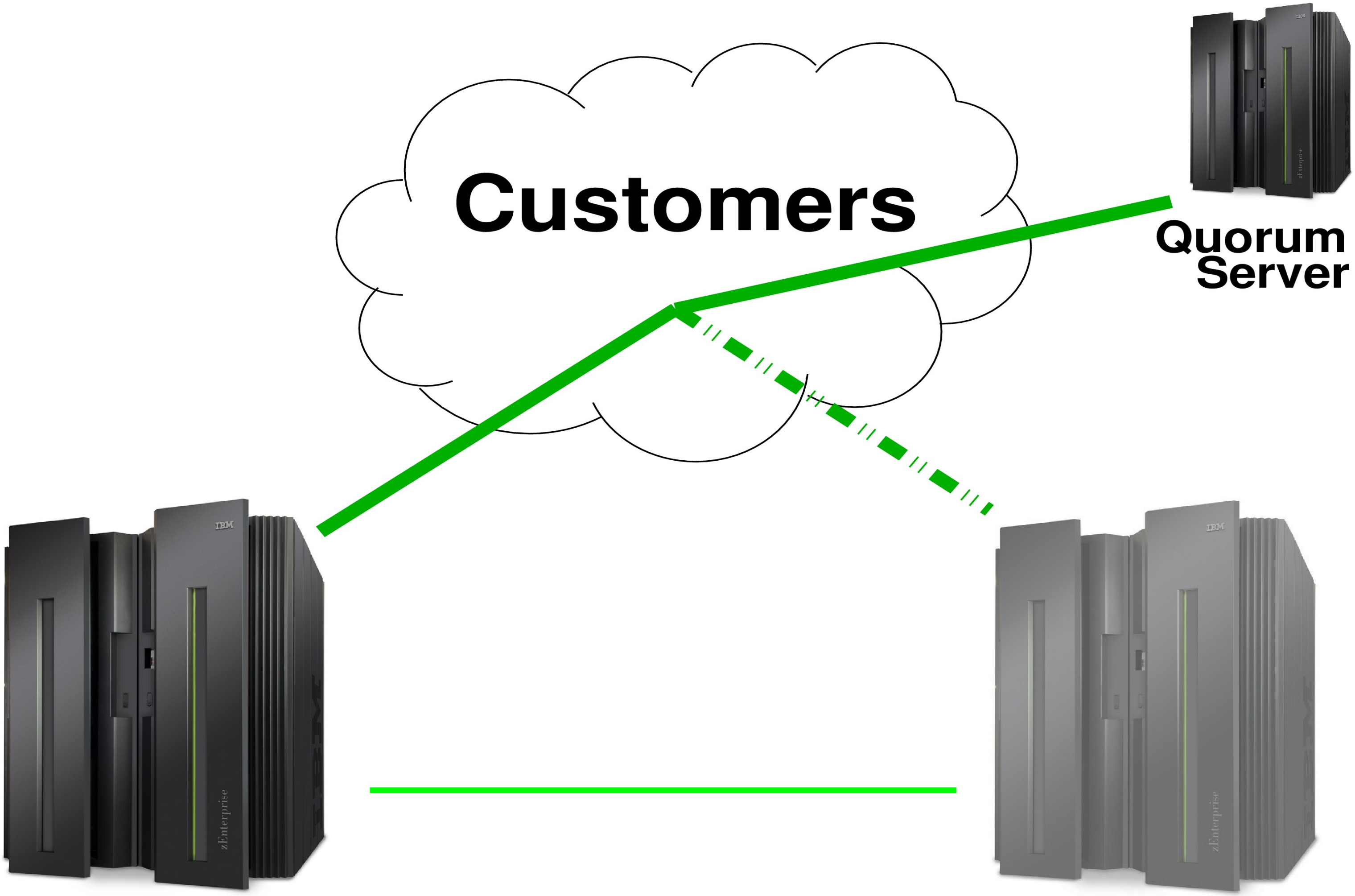


2 Node - Active-Active

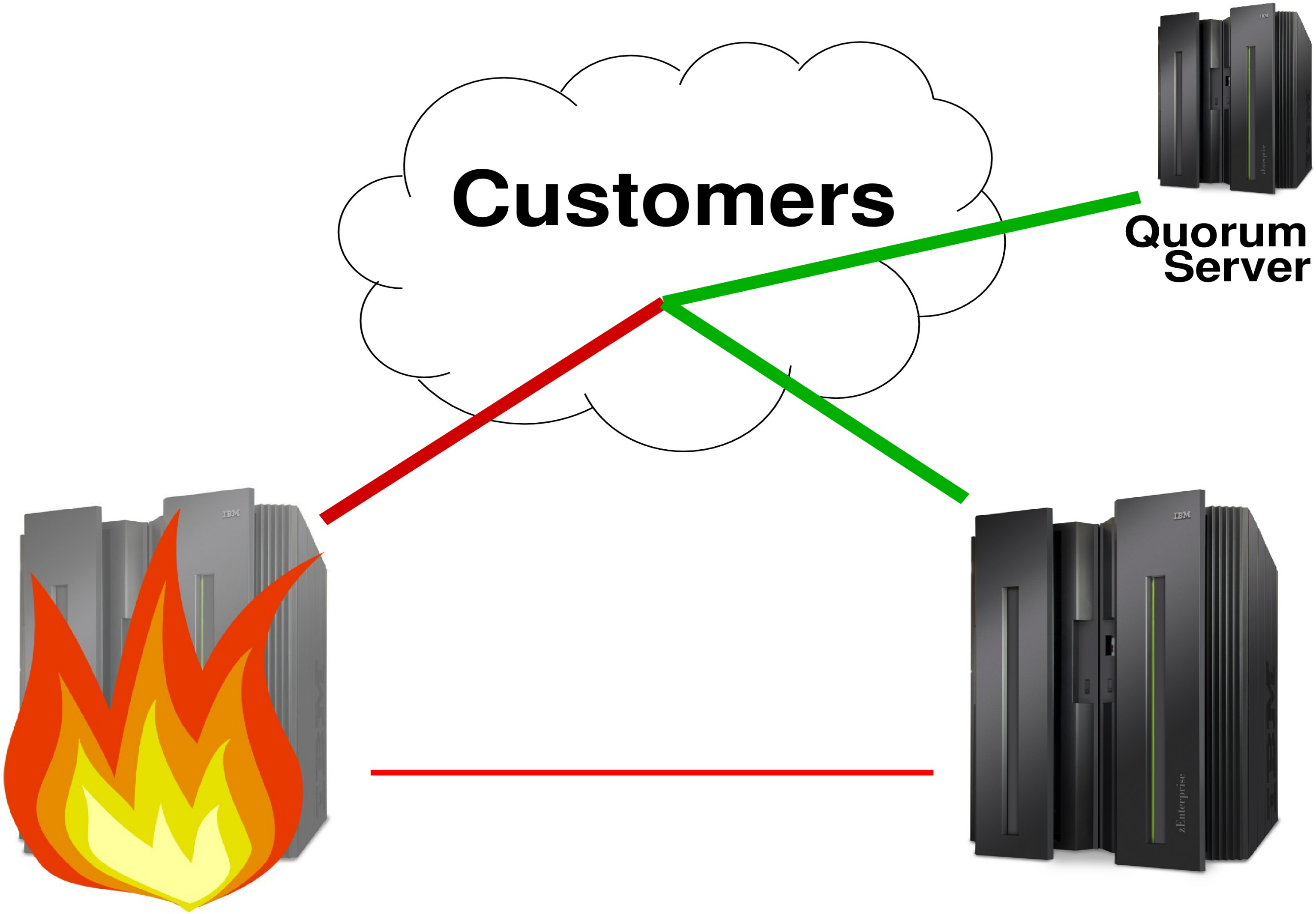
- Lower costs
- In good case
  - No idle resources
- In case of failure

  - Degradation of performance
  - Different application topology

3 Nodes with Quorum



3 Nodes with Quorum



3 Nodes with Quorum

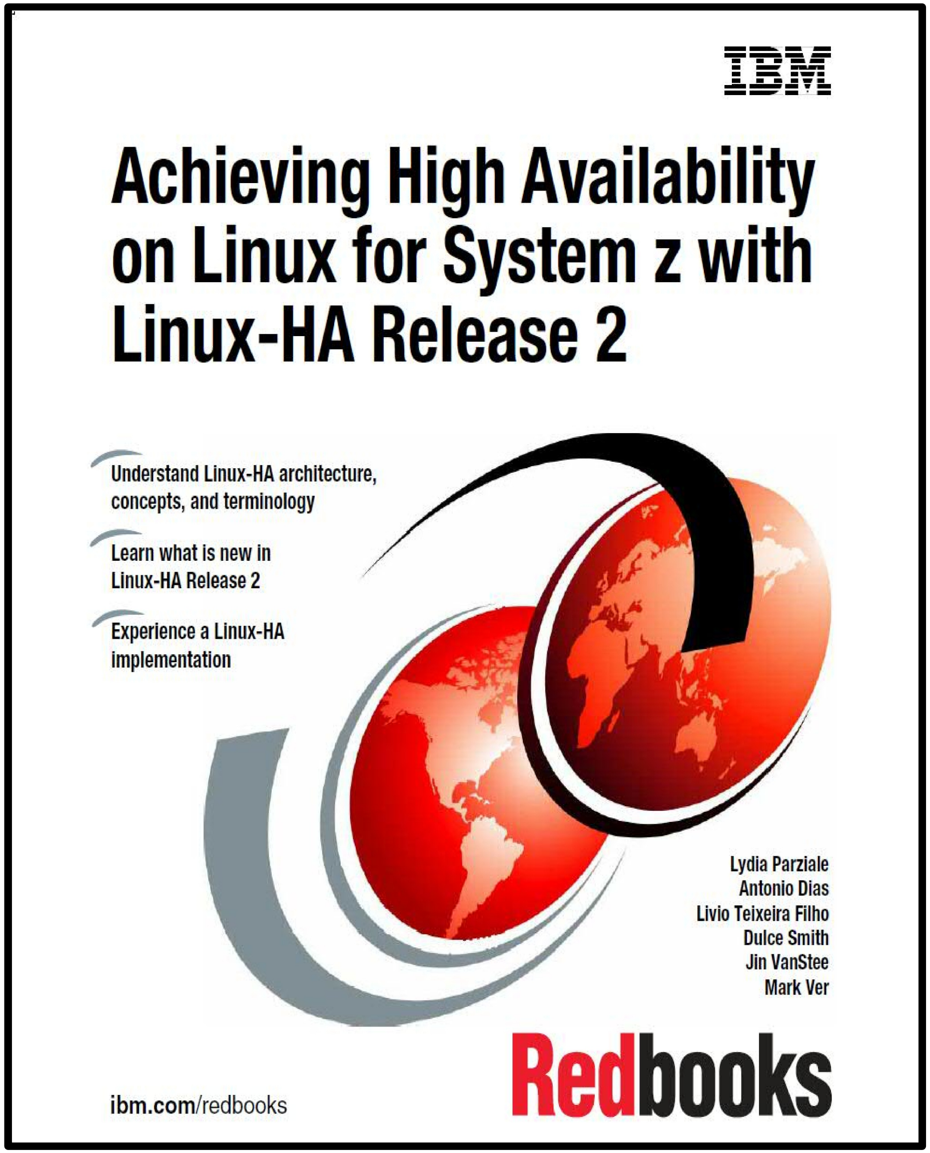
- Costs for Quorum server
- Monitoring from customer/service perspective

- 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

Linux-HA RedBook



IBM

Achieving High Availability on Linux for System z with Linux-HA Release 2

- Understand Linux-HA architecture, concepts, and terminology
- Learn what is new in Linux-HA Release 2
- Experience a Linux-HA implementation

Lydia Parziale
Antonio Dias
Livio Teixeira Filho
Dulce Smith
Jin VanStee
Mark Ver

ibm.com/redbooks

Redbooks

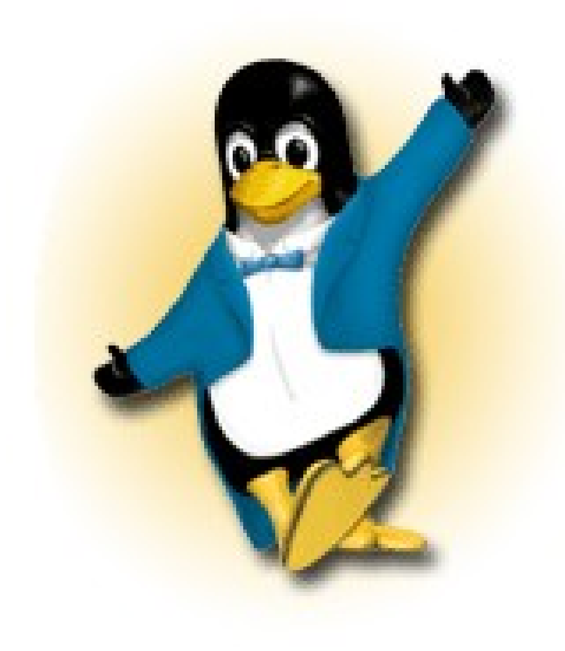
Links

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

Thank You !

- Alan Robertson
  for using his Linux-HA Tutorial
- Stefan Reimbold for creating this presentation

Questions?



Martin Schwidefsky

Linux on System z Development

Schönaicher Strasse 220
71032 Böblingen, Germany

Phone +49 (0)7031-16-2247
schwidefsky@de.ibm.com

Please Evaluate