HITACHI
Inspire the Next

An active-active high availability feature in shared nothing DBMS

Norihiro Hara

Hitachi, Ltd

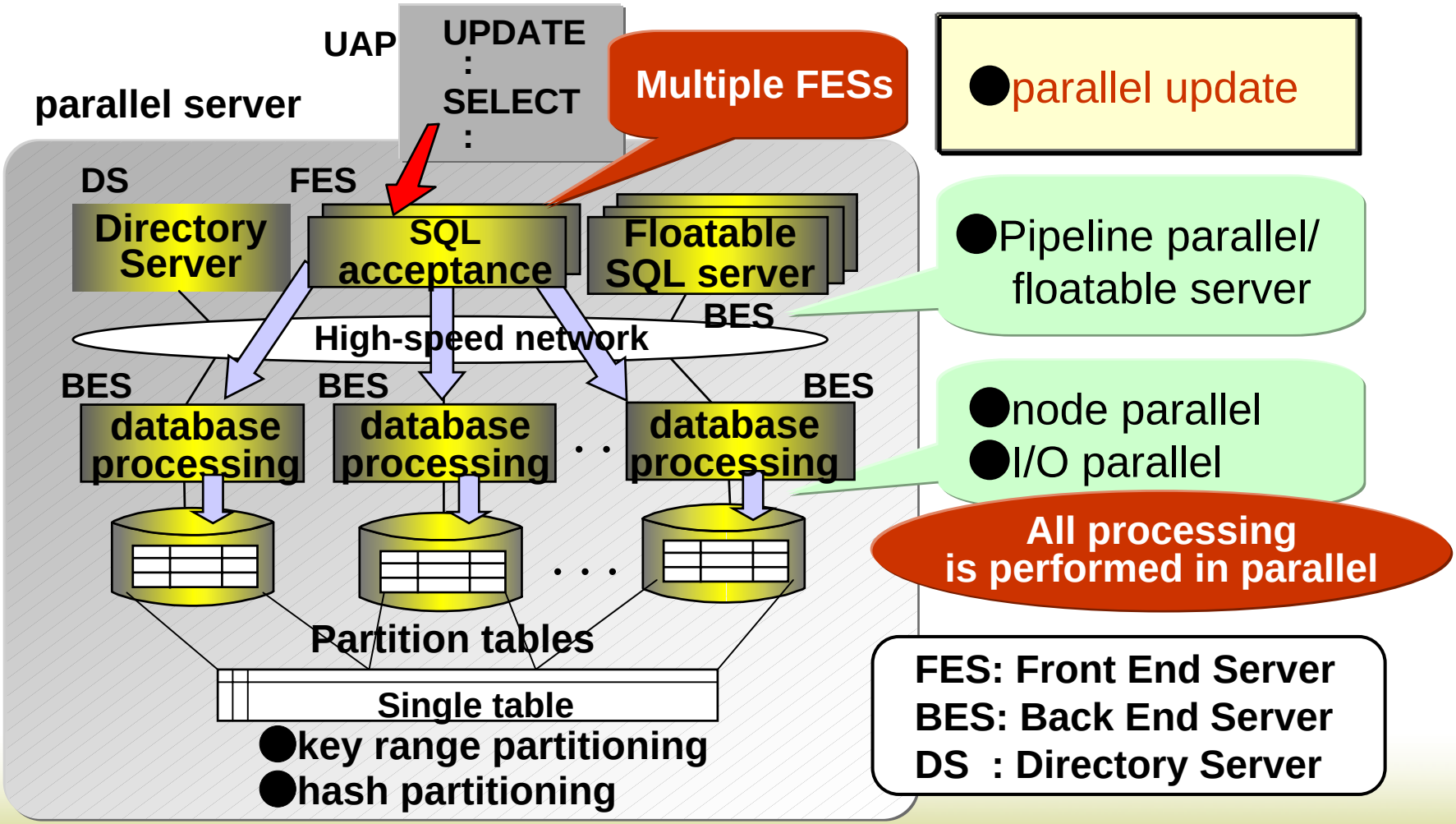
Oct. 13, 2003

Agenda

- HiRDB architectural overview
- HiRDB Version 7 Targets
- Conventional HA Approach
- What is our “Active-Active” ?
- Impact of a node failure
- Failover Time Example

HiRDB architectural overview

- "Parallel Server" realizes large-scale database processing.
- High-scalability achieved by shared-nothing architecture.



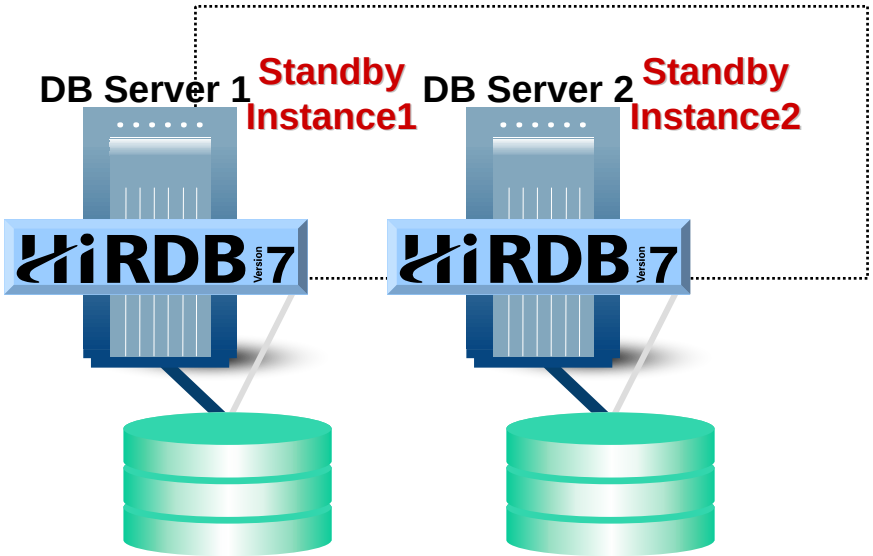
HiRDB Version 7 Targets

- ✓NON_STOP (No Downtime)
  - – Unplanned Downtime
    - ➢**HA Solutions**
    - • DR Solutions
  - – Planned Downtime
    - • Online Reconfiguration
    - • Online Reorganization

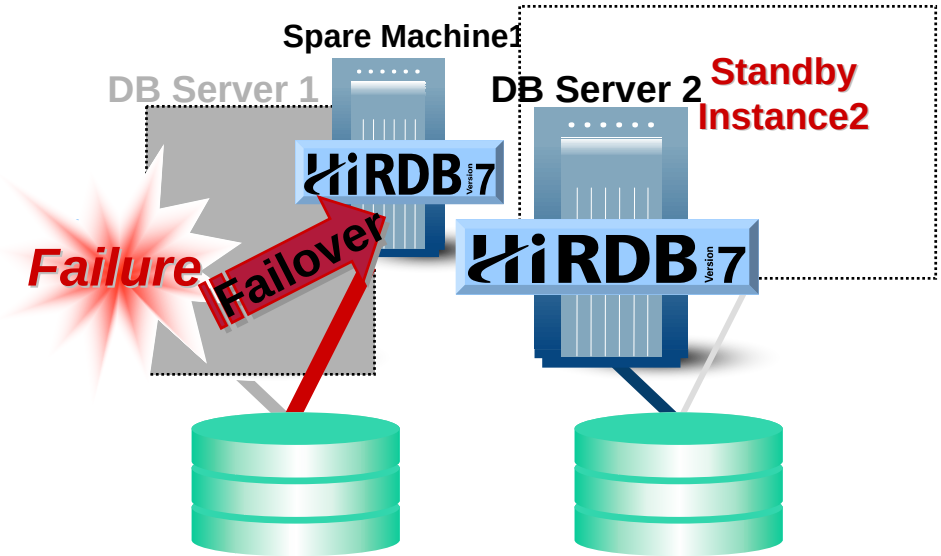
Conventional HA Approach

Active-Standby architecture: Hot Standby

- A spare server machine and a Standby database instance are preparing to failover on various failures
- It provides good failover time
- It has a big problem for TCO: resource utilization
  - A standby database instance for failover is usually inactive and sleeping



Normally: ***Inactive, Idle, Sleeping***

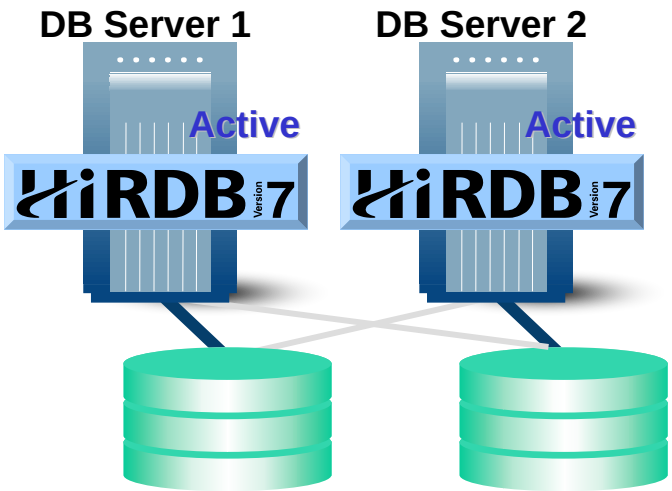


At failure: ***Standby to Active quickly***

What is our “Active - Active” ?

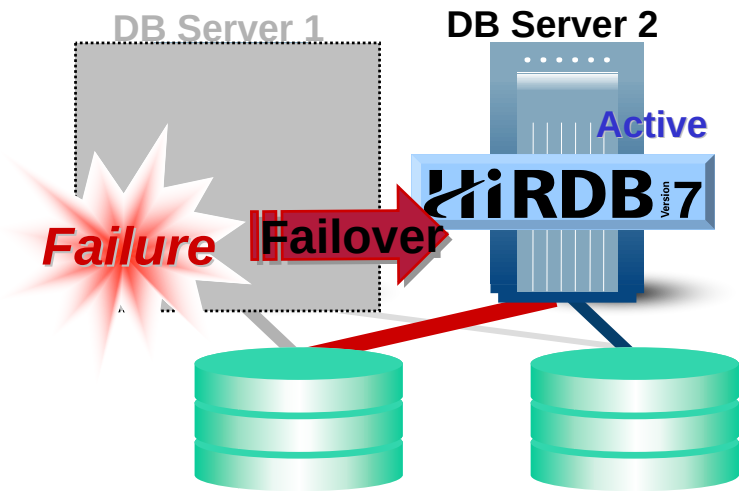
Our Active-Active architecture overview

- Fast failover：Single digit seconds
  - Failover to active servers
  - The surviving active server accesses to the database instead of the server on the failed instance
- Low Cost：No redundant resources
  - All Instances Active, No Instance Idle (Standby)



Normally: *All Instances Active*

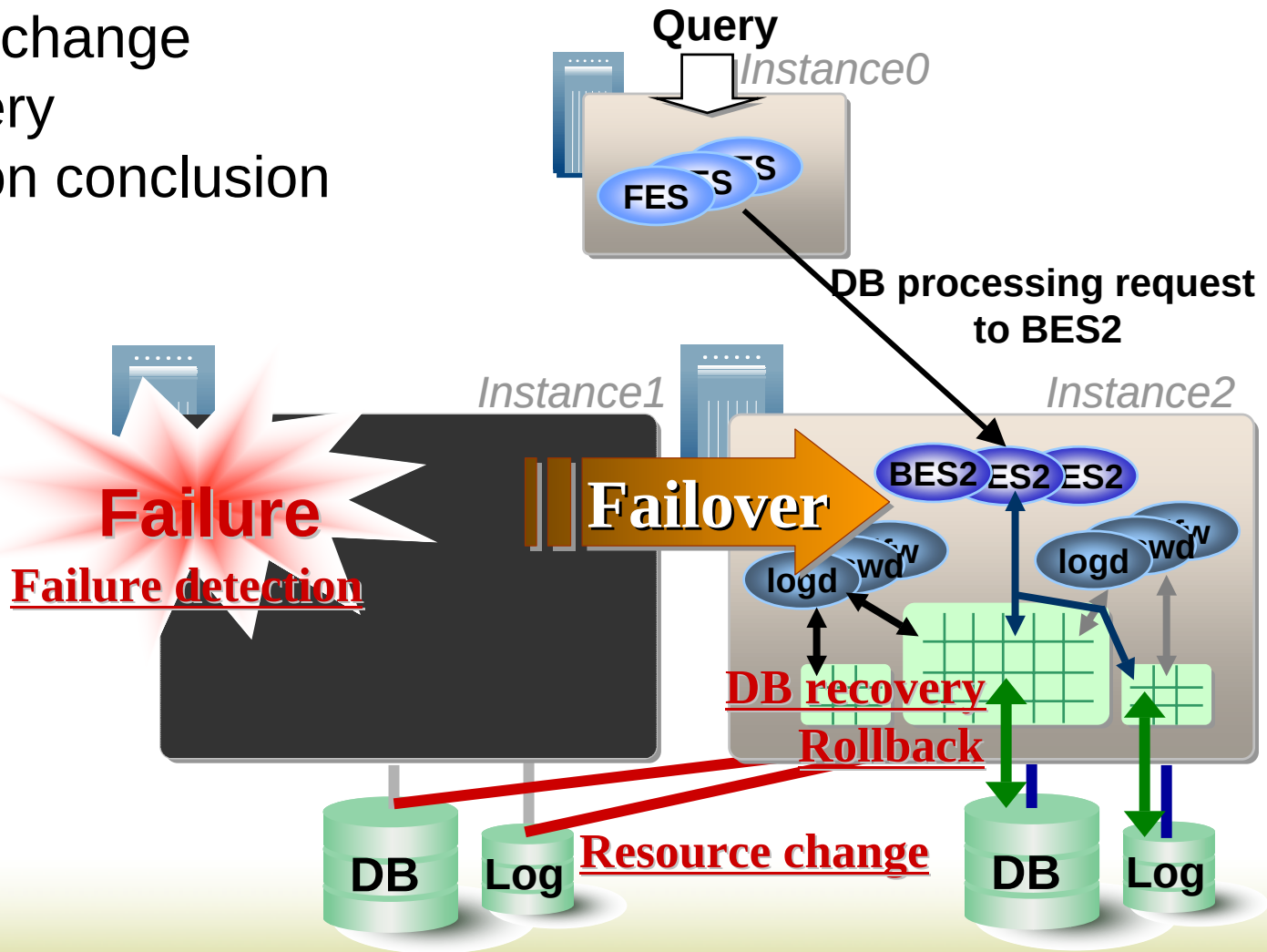
No Spare Server Machine, No Standby Instance



At failure: *Surviving Instances access all databases*

Failover Processing

- ✓ Failure detection
- ✓ Resource change
- ✓ DB recovery
- ✓ Transaction conclusion (rollback)

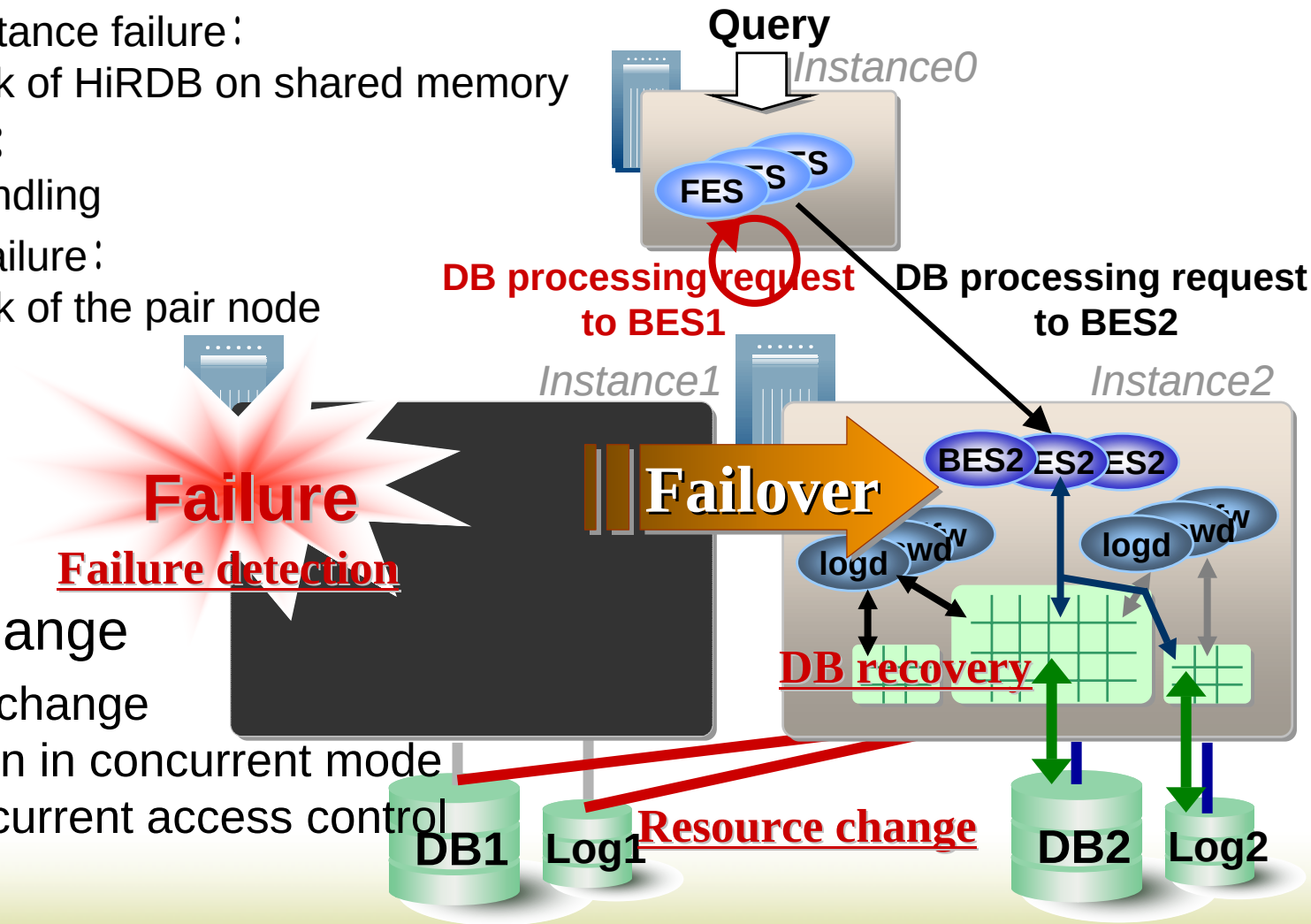


Failover Processing (Cont.)

✓Failure detection

– Early detection in cooperation with Clusterware

- HiRDB instance failure:
  Alive check of HiRDB on shared memory
- OS failure:
  Failure handling
- Machine failure:
  Alive check of the pair node



✓Resource change

– Quickly VG change
by varying on in concurrent mode
with no-concurrent access control

– DB, Log

Failover Processing (Cont.)

✓DB recovery

– Internal DB recovery in the surviving active instance

- During DB recovery, the instance (ex. Instance2) is performing DB processing to the own DB (ex. DB2)
- No impact to the original DB processing

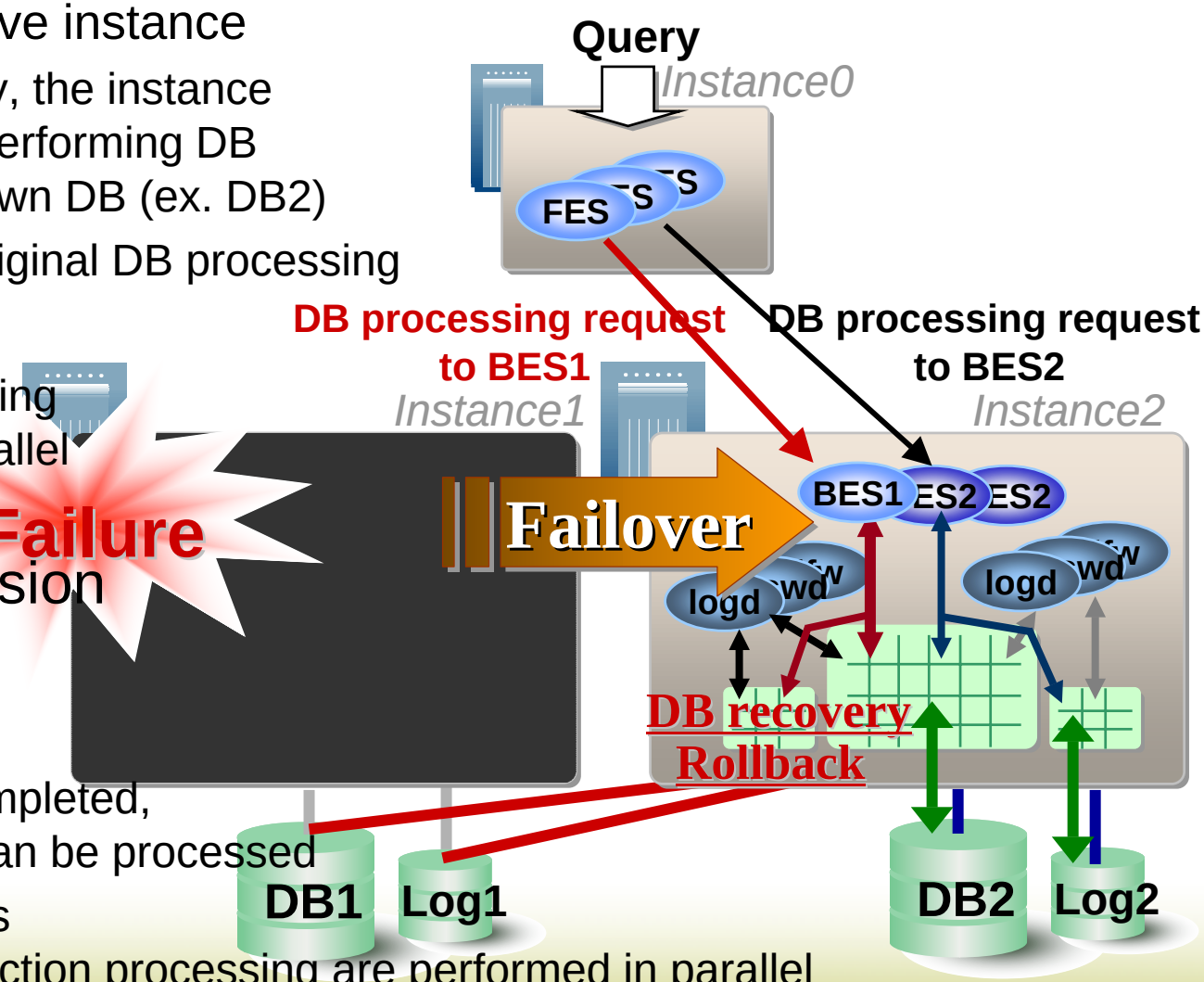
– Parallel REDO

- Rollfoward processing is performed in parallel

✓Transaction conclusion (rollback)

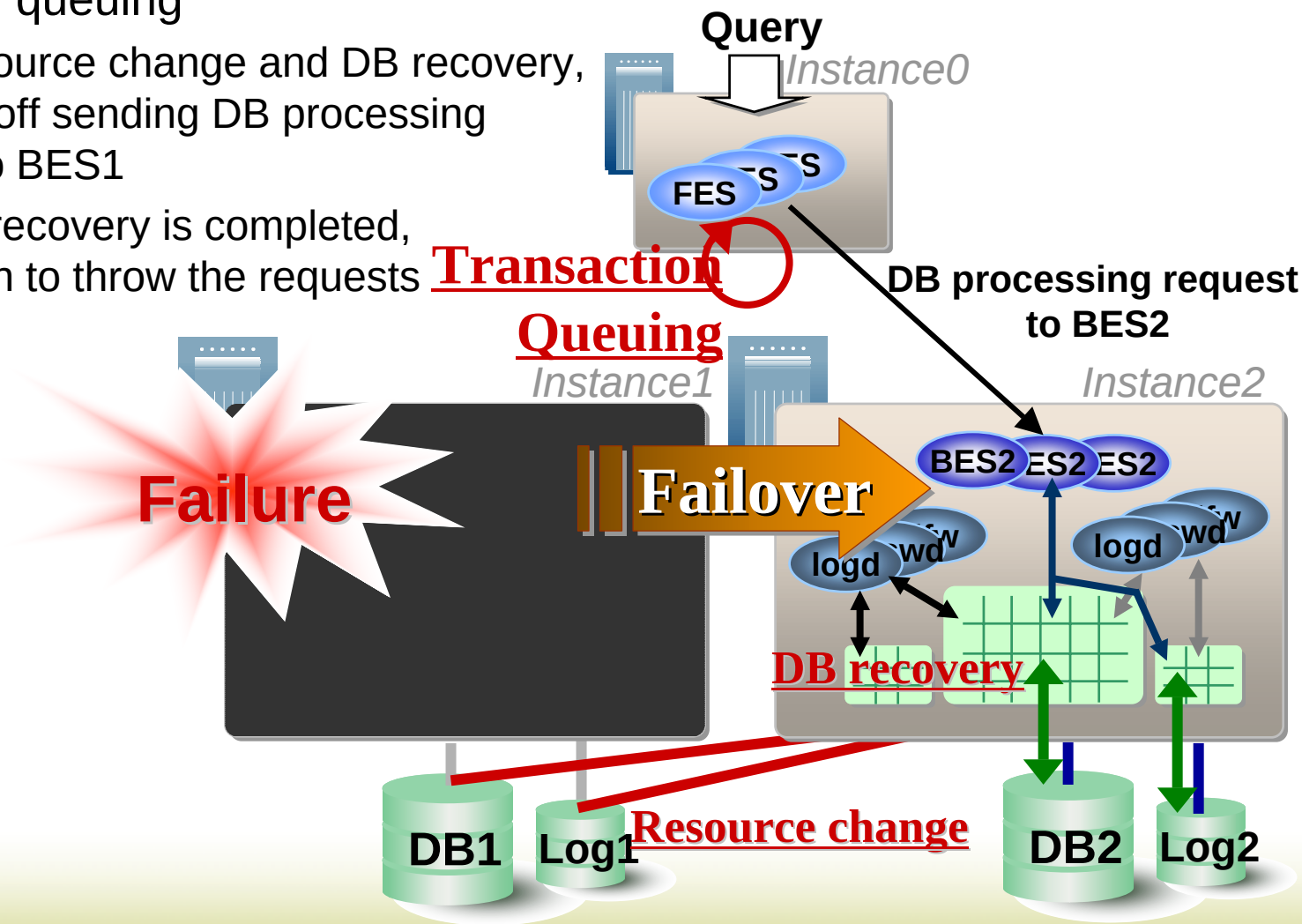
– "Delayed rerun"

- When REDO is completed, new transactions can be processed
- Rollback operations and the new transaction processing are performed in parallel



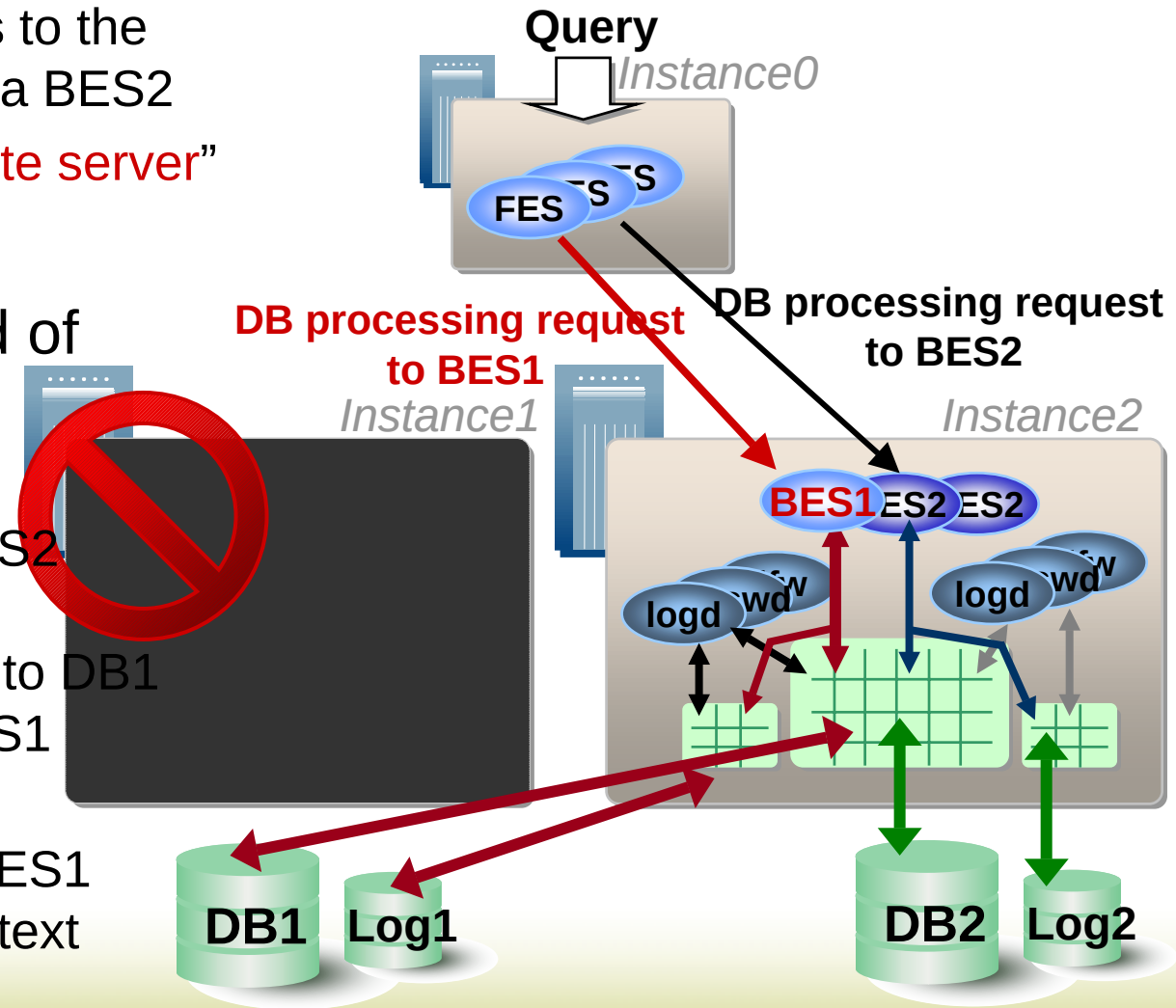
Failover Processing (Cont.)

- Impact to requests to BES1:No error
  - "Transaction queuing"
    - During resource change and DB recovery, FESs put off sending DB processing requests to BES1
    - When DB recovery is completed, FESs begin to throw the requests



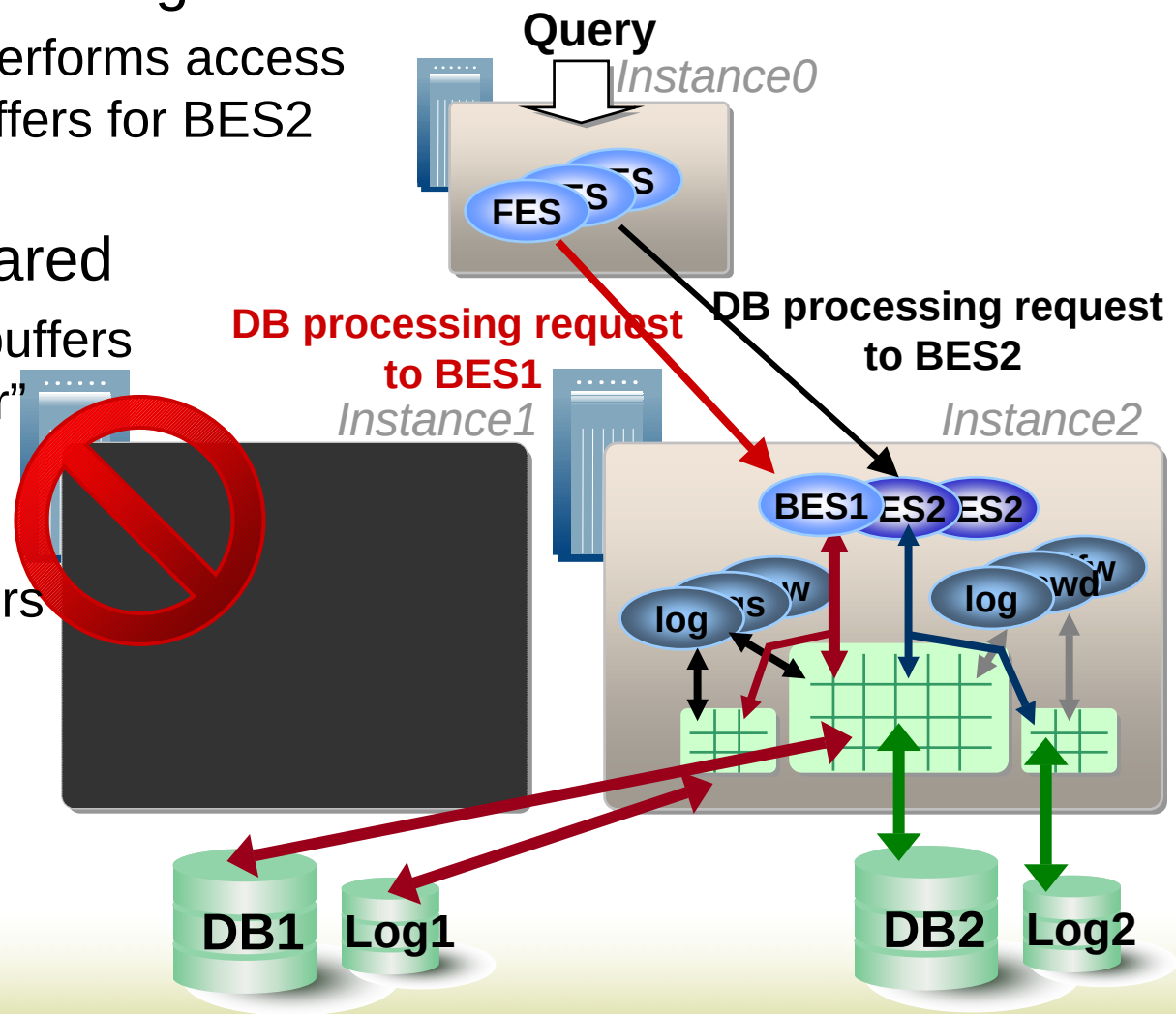
Database Processing After Failover

- ✓Requests routing
  - DB processing requests to the failed BES1 are sent to a BES2
  - BES2 is called "substitute server"
- ✓DB processing instead of the failed BES
  - On demand, a "substitute server" BES2 process performs DB processing with access to DB1 instead of the failed BES1
  - change the processing environment:BES2 to BES1 (DB server process context switch)



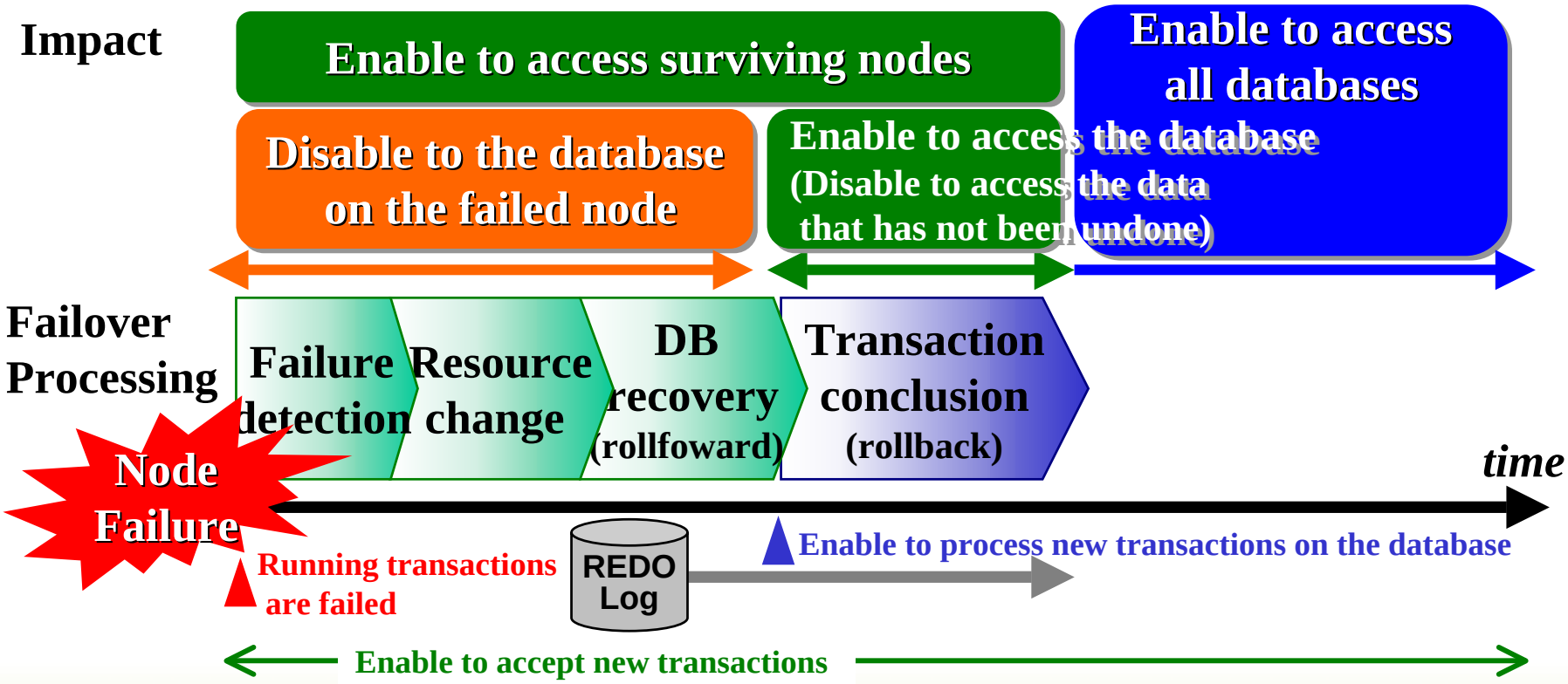
Database Processing After Failover (Cont.)

- ✓DB buffers are shared with the "substitute server" processing
  - The BES1 processing performs access to DB2 using the DB buffers for BES2
- ✓Log buffers are not shared
  - Instance2 has the Log buffers for the "substitute server"
  - The BES1 processing writes log into Log1 using the own Log buffers
  - no need to merge Log1 to Log2



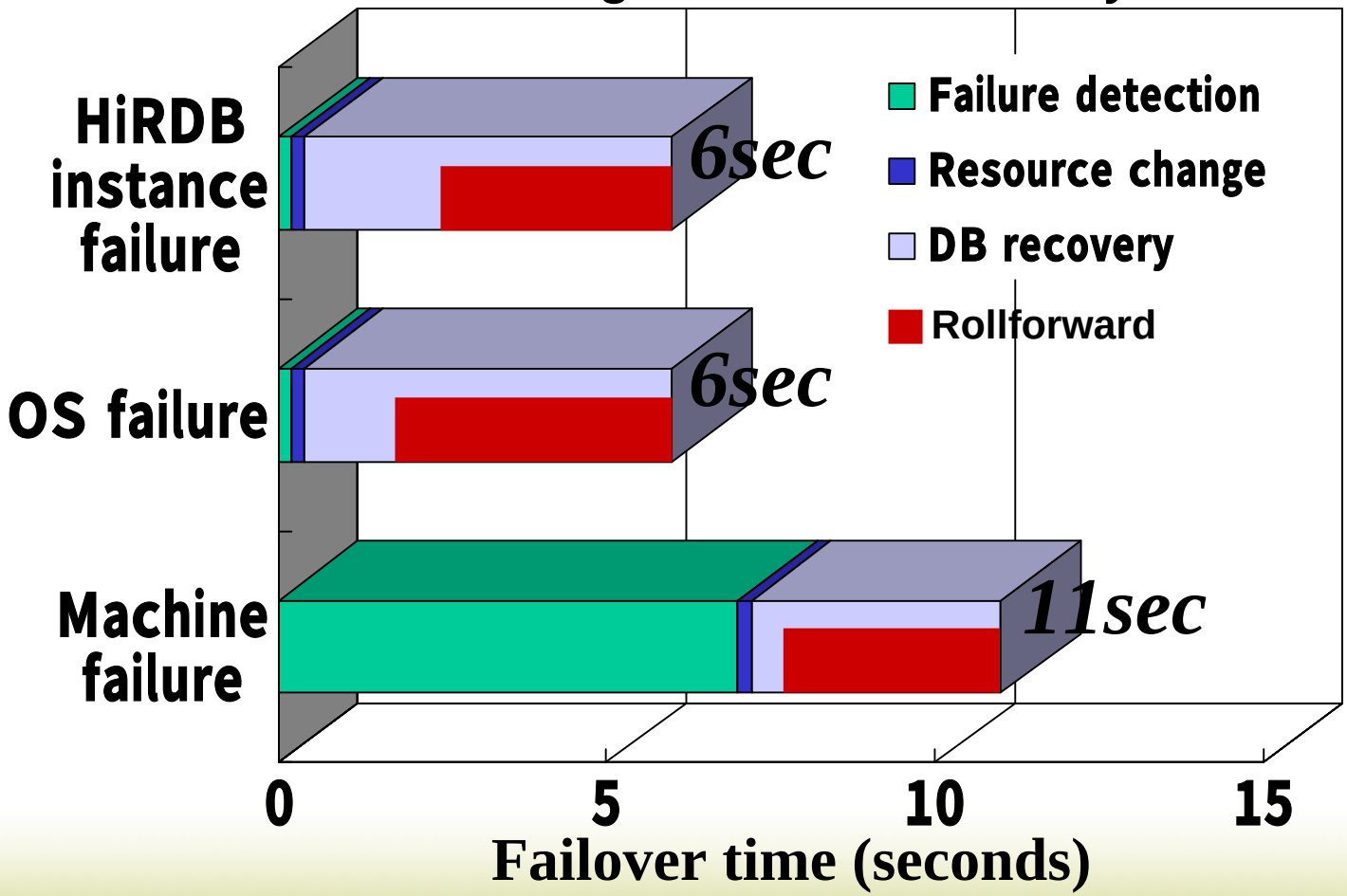
Impact of a node failure

- During failover the other “surviving” instances (nodes) remain up and running
- No period during which all databases can not be accessed
- Always new transactions can be accepted



Failover Time Example

- Fast failover : 6sec~11sec
  - Early failure detection
  - Fast resource change and DB recovery



Conclusions

- When nodes are failed, the other surviving instances perform DB processing instead of the failed servers as if it were shared disk architecture
- Fast failover and Good resource utilization by substitute processing by active server etc.
- Applications never notice failures and the services are provided to end-users continuously
- Of course highly scalability: advantages of shared nothing architecture are inherited
- Therefore our “Active-Active HA architecture” is good
- Which is winner, Shared Disk Cluster or Shared Nothing Cluster?
  > No winner! Getting advantages of the other

Other Important Features and Some Challenges

- **Other Important Features**
  - Superior solutions for disaster recovery
    - Real-time SAN replications
    - Real-time SQL replications
    - Relocatable Tables
  - Online Reconfiguration
    - add instance/server
    - add DB buffer
    - add DB space
    - change configuration
  - Online Reorganization
    - Online rebalance utility
  - Online Software Maintenance

- **Some Challenges**

Thanks!