Informix Replication

Use cases

Agenda

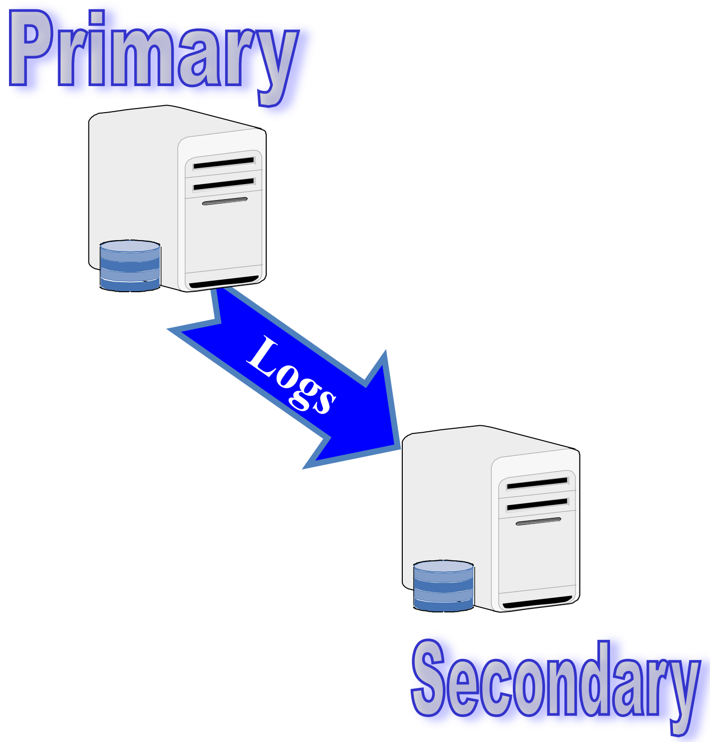
- High availability
- Disaster recovery
- Handling workload spikes
- Application Connectivity and Failover
- Multisite global business
- Zero downtime operations
- Database Scale-out

High Availability and Hot Standby

- Less than 30 seconds Recovery Time Object (RTO) and zero Recovery Point Objective(RPO) ?
- And also offload read activity, and occasional write activity from your business critical server ?
- And protect from disk corruption

High Availability Data Replication (HDR)

- Provides hot backup for the primary
- Can be used to offload read/write processing
- Easy to setup
  - Take a Backup of the primary and restore
- Works by automatically applying logs on secondary
  - Secondary is in recovery mode
- Supports synchronized commits



HDR Pros & Cons

Pros

- Very easy to set up & administer
- Automatic failover between primary & secondary node can be configured
- In synchronous mode, secondary is immediately ready to become primary

Cons

- Requires same hardware, OS, and Informix version for primary & secondary servers
- Secondary needs to be physically close to primary (same city/data center). Not great over WAN network
- Just one secondary

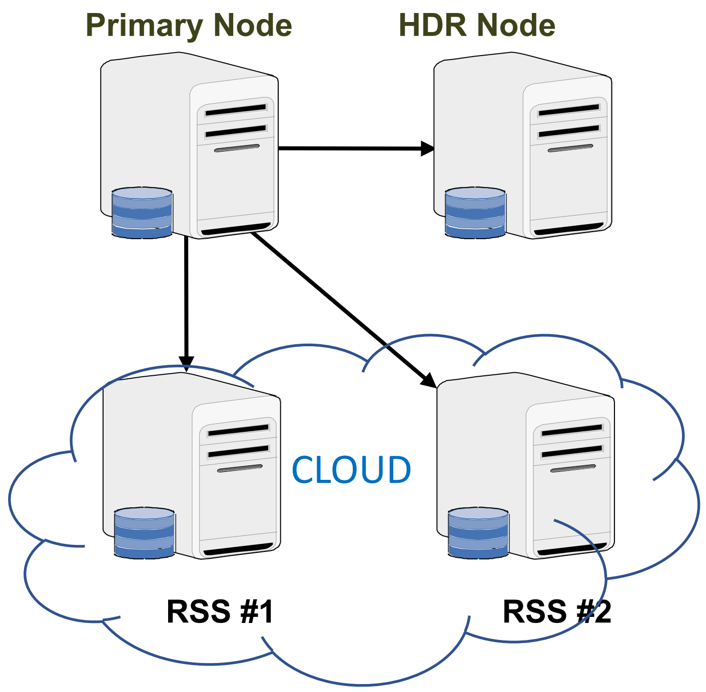
Offsite disaster

- Need to protect business operations from natural disasters ?
- Need to protect data from human errors ?
- Need to offload backups from 24/7 business critical server ?
- For offsite disaster solution, cannot afford dedicated network line, want to use cloud server to save cost?
- Cannot afford to use business critical server to stream OLTP data to business analytic server like Informix Warehouse accelerator ?

Remote Standalone Secondary (RSS)

- New type of secondary – RSS nodes
- Can have 0 to N RSS nodes
- Can coexist with HDR secondary
- Supports delayed apply and “stop apply” functionality
- Similarities with HDR secondary node:
  - Receive logs from Primary
  - Has its own set of disks to manage
  - RSS has minimal impact on primary performance
- Differences with HDR secondary node:
  - Can only be promoted to HDR secondary, not primary
  - SYNC mode not supported

Replication to Multiple Remote Secondary Nodes



RSS Pros & Cons

Pros

- Simple way to build your cluster - add additional CPU resources that can work on its own data
- Support for delayed apply mode
- Any number of RSS nodes supported

Cons

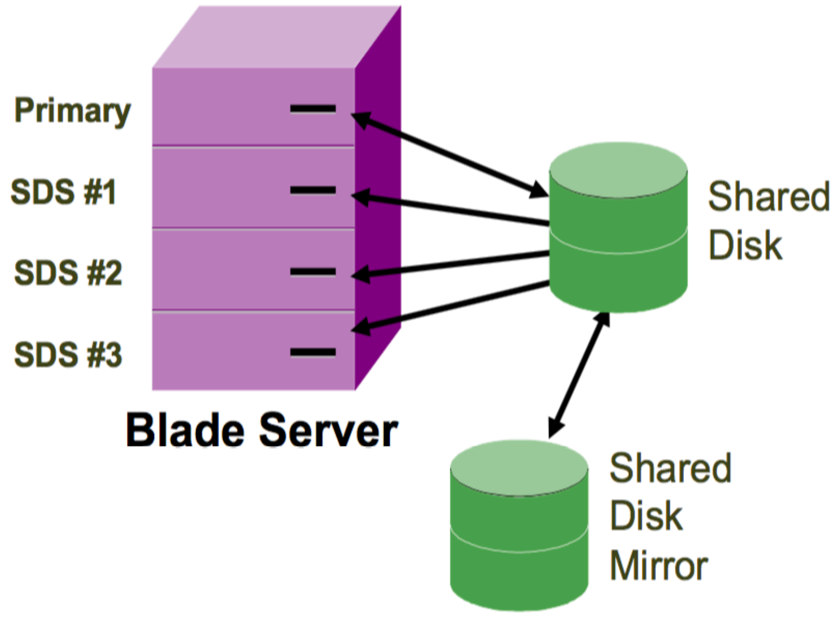
- Only supports asynchronous replication
- Same OS and Informix version required

*MUCH better with 14.10!

Handling workload spikes during special events

- Need to quickly scale out during special events like a Black Friday sale?
- Also supports zero RPO, and less than 30 seconds RTO

- Share Disk Secondary (SDS)
  - Does not duplicate disk space
  - No special hardware
    - Cluster mgr or SDS_LOGCHECK
  - Coexist with ER, HDR & RSS
  - Primary can failover to any SDS



SDS Pros & Cons

Pros

- Simple way to add additional CPU power to work with existing data
- Any SDS node can become primary
- Any number of SDS nodes supported

Cons

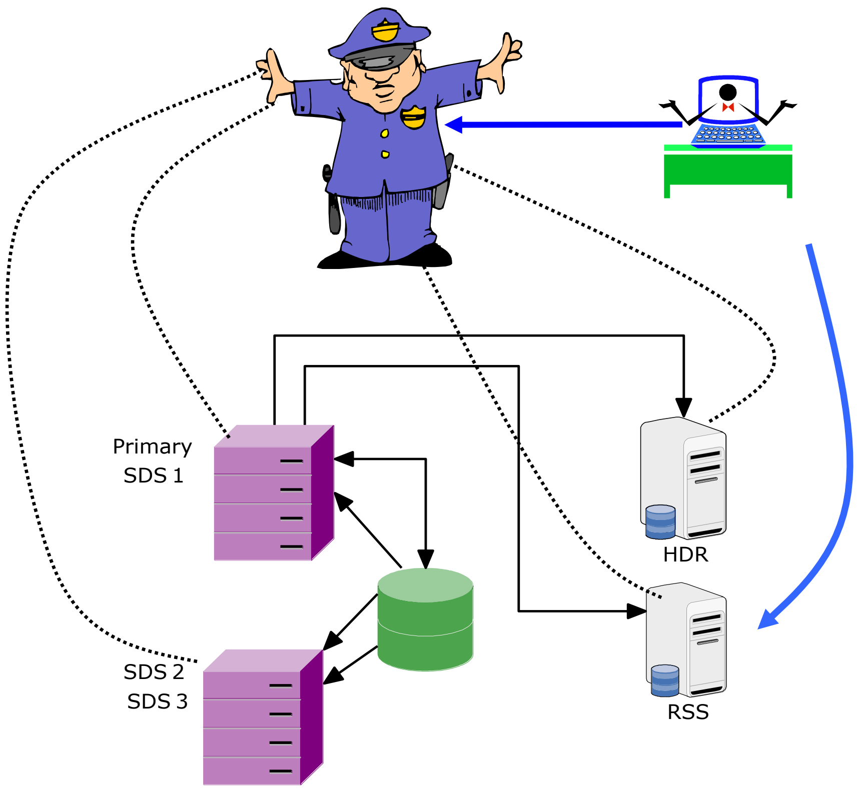
- Generally requires a more sophisticated storage management solution (RAID, etc.) to avoid the disk being a single point of failure
- Same OS and Informix version required

Application Connectivity and Failover

- Application connectivity based on business rules, and SLAs.
- Automated failover and cluster monitoring
- Need reverse proxy service to hide database cluster from internet and intranet ?

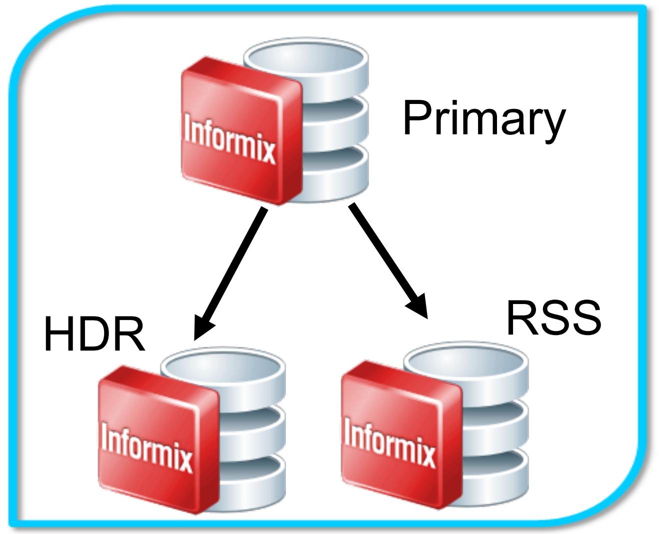
Connection Manager

- Maintains knowledge of all nodes within the cluster
- Records adding/removal of nodes
- Monitors type of node
- Monitors workload of nodes
- Routes the client application to optimal target node based on SLA

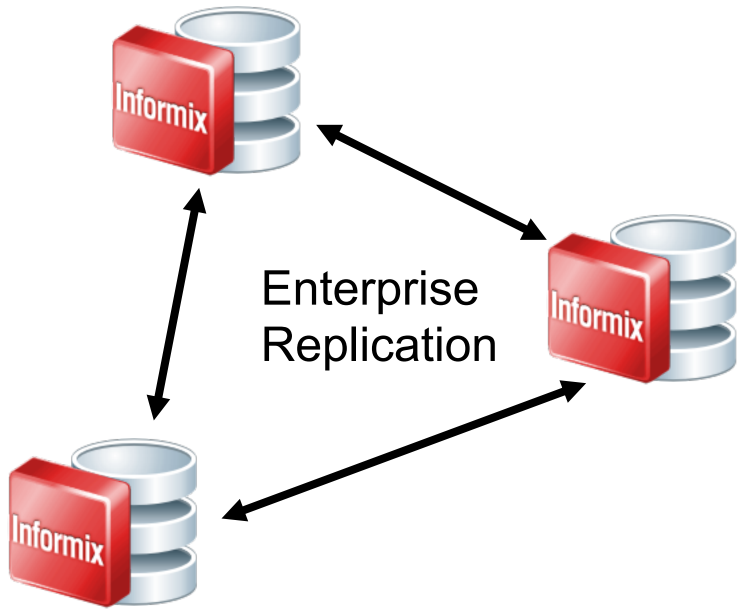


Connection Manager

- Connection unit types

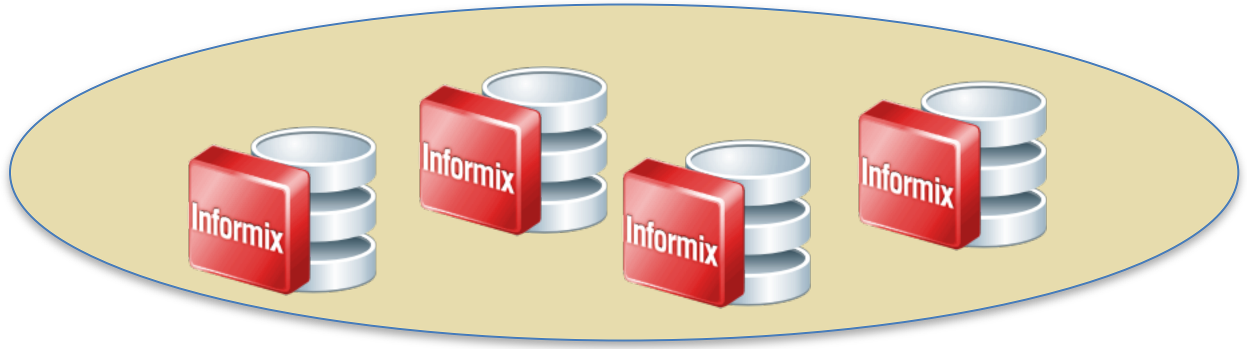


1) CLUSTER



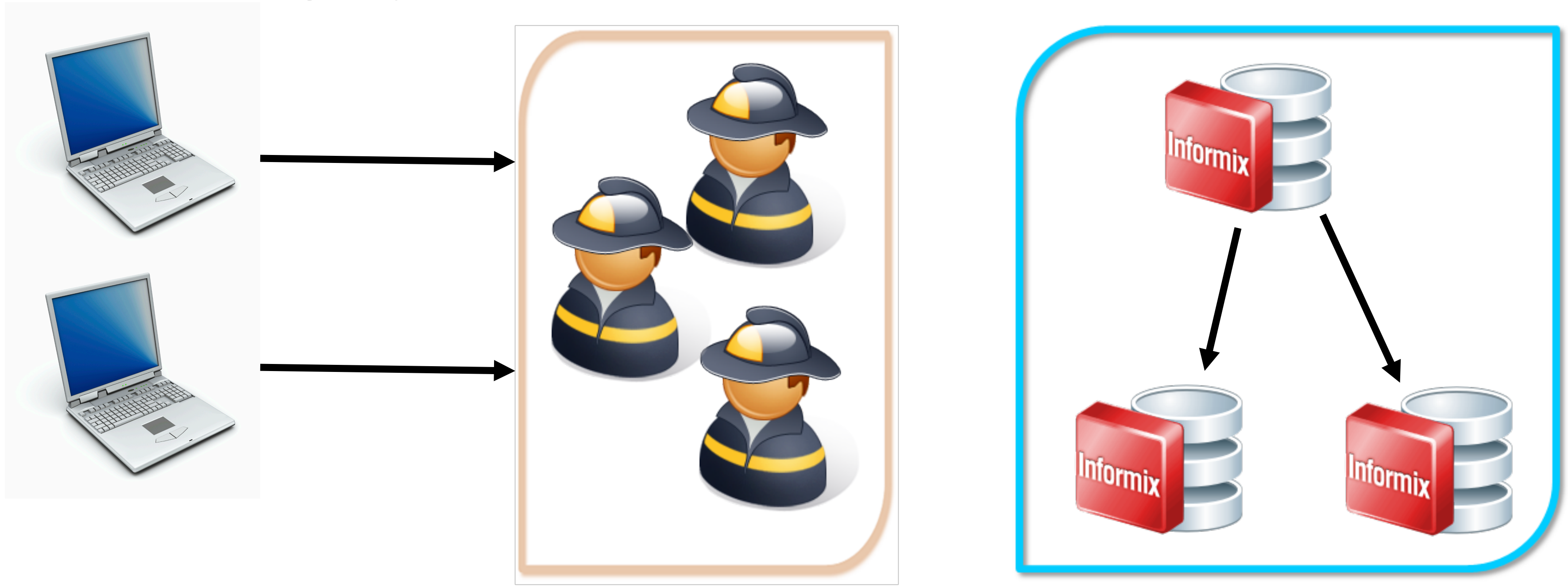
2) REPLSET
3) GRID

4) SERVERSET



Connection Manager

- Avoid single point of failure



Client's INFORMIXSQLHOSTS:

```
g_mySLA      group        -         -        c=1,i=123456
cm1_mySLA    onsoctcp     cm1Host   cm1Port  g=g_mySLA
cm2_mySLA    onsoctcp     cm2Host   cm2Port  g=g_mySLA
cm3_mySLA    onsoctcp     cm3Host   cm3Port  g=g_mySLA
```

Multisite Global business

- Need to consolidate data from branch offices to central location ?
- Need to route clients to closest server based on GEO location ?
- Need protect local business operation from network failures ?
- Heterogeneous Hardware requirement ?

Near zero downtime maintenance activity

- Rolling server upgrades between Informix major versions
- Rolling schema and application upgrade
- Database codeset conversion – like from en_us.819 to en_us.utf8
- Migrating data from legacy hardware to newer hardware and OS.
  - 32bit to 64bit data migration

Enterprise Replication basic functionality (1)

- Log-based replication
- Rows identified by:
  - Primary key
  - Enterprise Replication key
  - Unique index
- Supports selective replication
  - Row selectivity
  - Column selectivity
- Supports various replication strategies
  - Primary/Secondary
  - Dissemination
  - Consolidation
  - Update-Anywhere

Enterprise Replication basic functionality (2)

- Supports various rules for conflict resolution
  - Timestamp
  - Ignore
  - Stored procedure
  - Always apply
  - Deletewins
- Supports various topologies
  - Fully connected
  - Hub and spokes
  - Hierarchy
  - Forest of trees

ER Pros & Cons

Pros

- Heterogeneous environments – operating systems & Informix versions
- Data to be replicated can be established by SELECT statements
- **Very** flexible →foundation for many other features – rolling upgrade, smart triggers

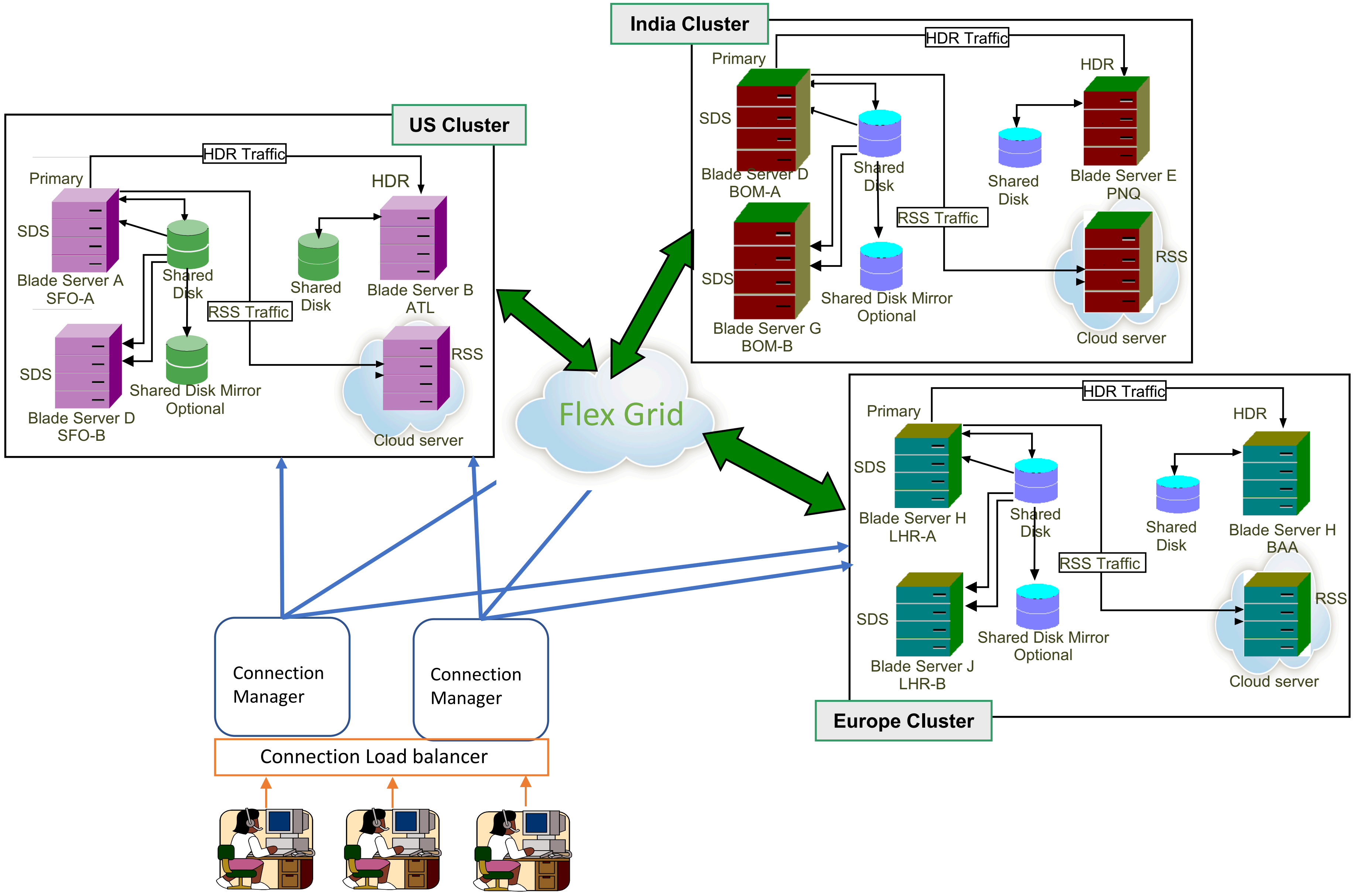
Cons

- Setup can require many steps – may be viewed as complex

Flexible grid

- DDL replication
- Replication of SQL statement execution
  - Useful while purging data from large replicated table
- Replication of function and procedure execution
  - Useful to administer large number of Informix servers.

Informix Replication Reference Architecture



Sharding

- Sharding is a type of database partitioning
  - Each database piece is called a shard
  - Data can be spread across multiple computers
  - A special case of horizontal partitioning
- Offers rapid horizontal scalability
  - Ability for the application to grow by adding low cost hardware to the solution
  - Ability to add or delete nodes dynamically
- Application transparent elasticity

Why shard your data?

- Grows with a company/application
  - Additional capacity means adding an additional computer
  - No need to pre-buy hardware for expected growth
- Reduce costs
  - Use smaller less expensive hardware
- Great for applications which are regional
  - The shards can be spread across multiple computers
  - Retail stores are a great example…

vs.

Sharding strategies

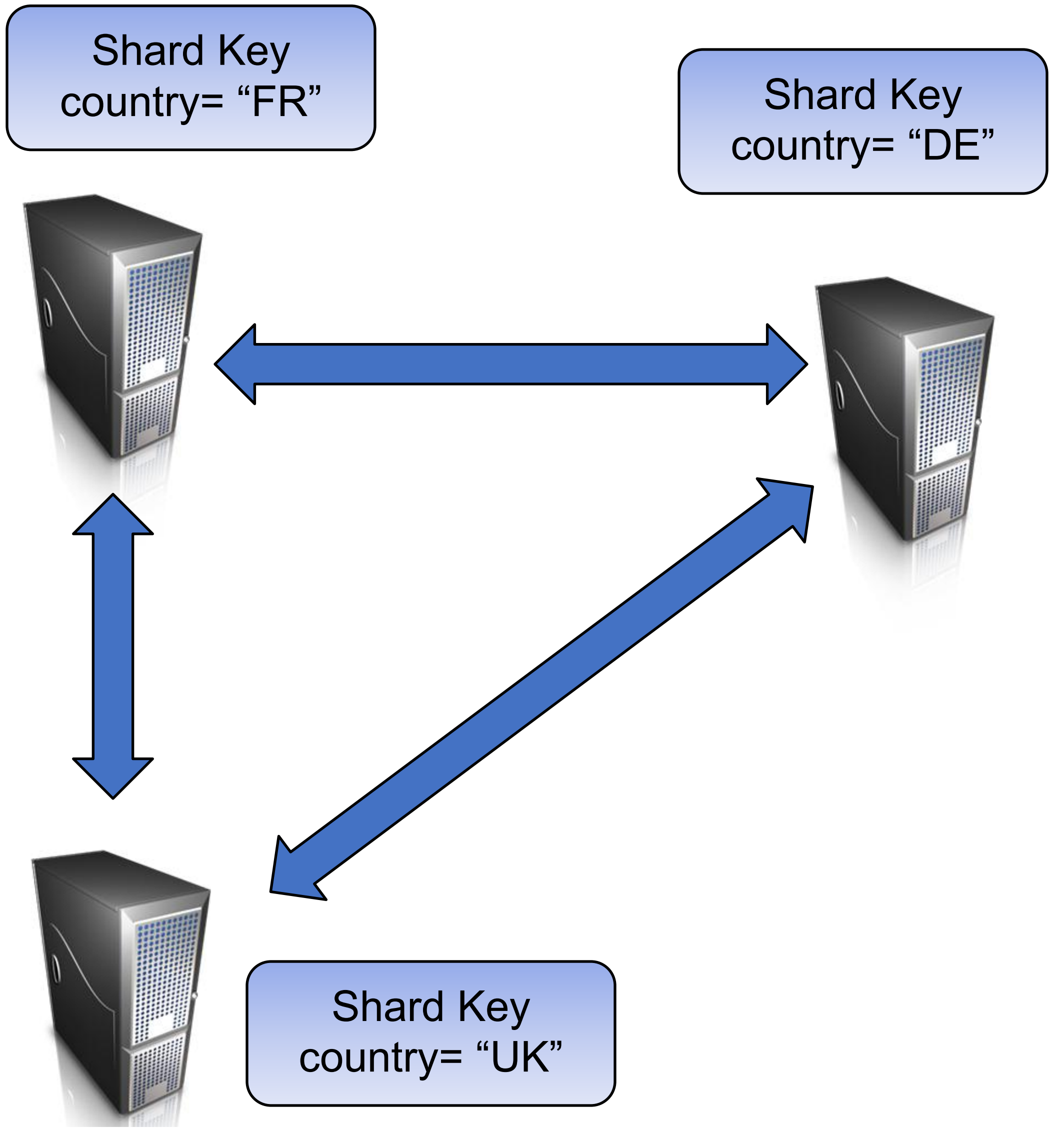
- Shard by <u>hash</u>
  - Simplest mechanism
  - Hash function determines which node is home for the data
  - Adding and removing nodes changes the hash algorithm, results in significant data movement to the new node
  - Best for data where hash key is used just for uniqueness
- Shard by <u>expression</u>
  - Essentially shard by WHERE clause
  - SQL functions can be used (IN, EQ, GT, LT, NE, etc.) to specify condition
  - Data that does not match the expression can go into its own node
  - Best for collecting meaningful data together (country=”UK”, for example)

Sharding strategies (2)

- Consistent hash based sharding
  - Mathematical consistent hash algorithm distributes data to the nodes in a consistent pattern
  - Adding/removing nodes does not result in massive data shuffling
  - Better hashing algorithm, especially in an elastic environment where nodes are added and removed often

- No Key sharding
  - Strategy where data is inserted into the node to which it is sent
  - Best where insert speed is critical. This results in fast inserts, perhaps at the cost of querying more nodes to satisfy queries

Scaling Out – Sharding Data



- Each node in the environment holds a portion of the data
- When inserting data it is automatically moved to the correct node
- Queries automatically aggregate data from the required node(s)
- Join optimization for replicated and sharded tables
- Sharding can be used along with other Informix HA solutions (HDR, SDS, RSS) to provide redundancy for disaster recovery, backup, etc.

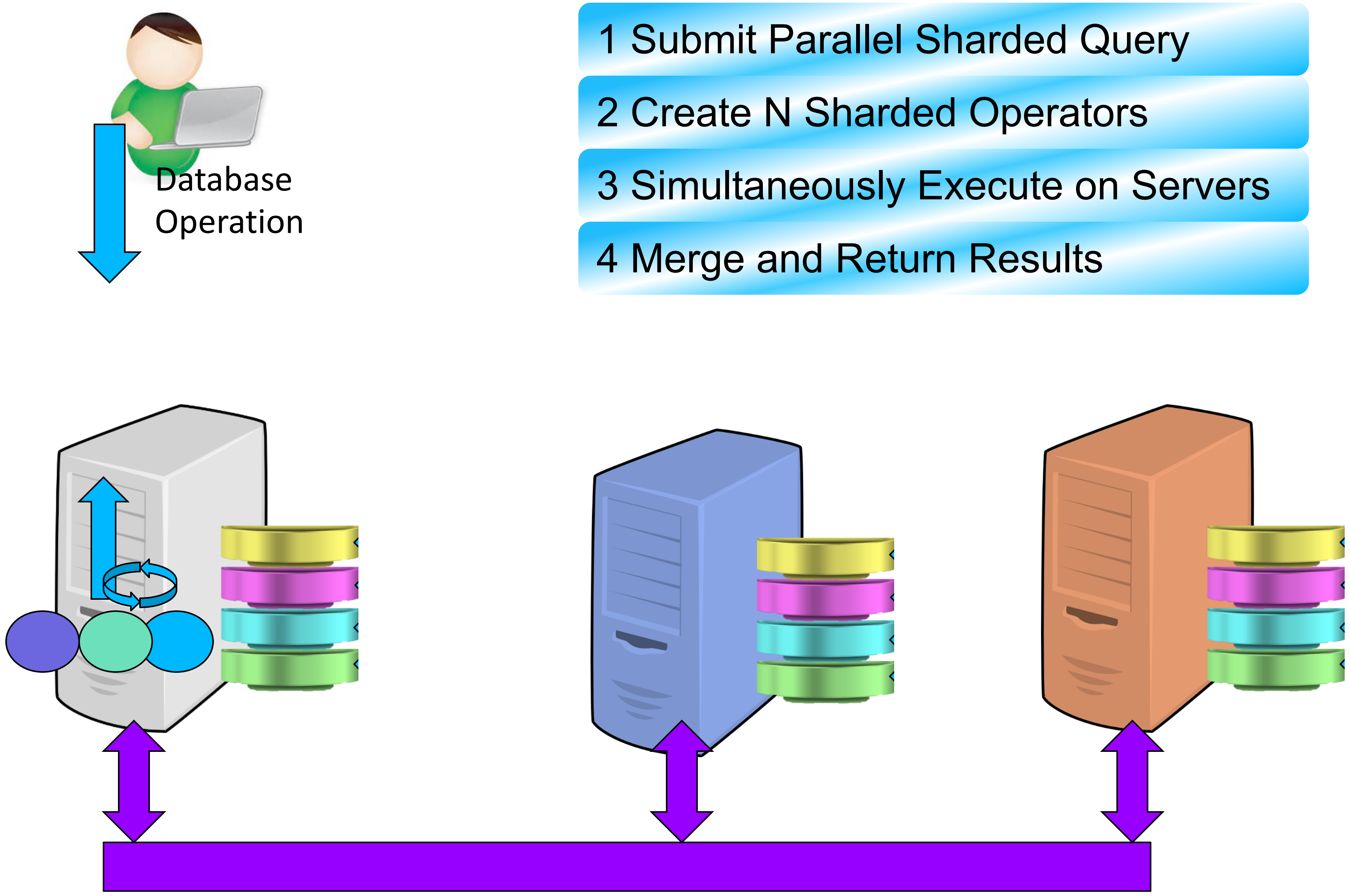
Informix Parallel Sharded Query

1 Submit Parallel Sharded Query

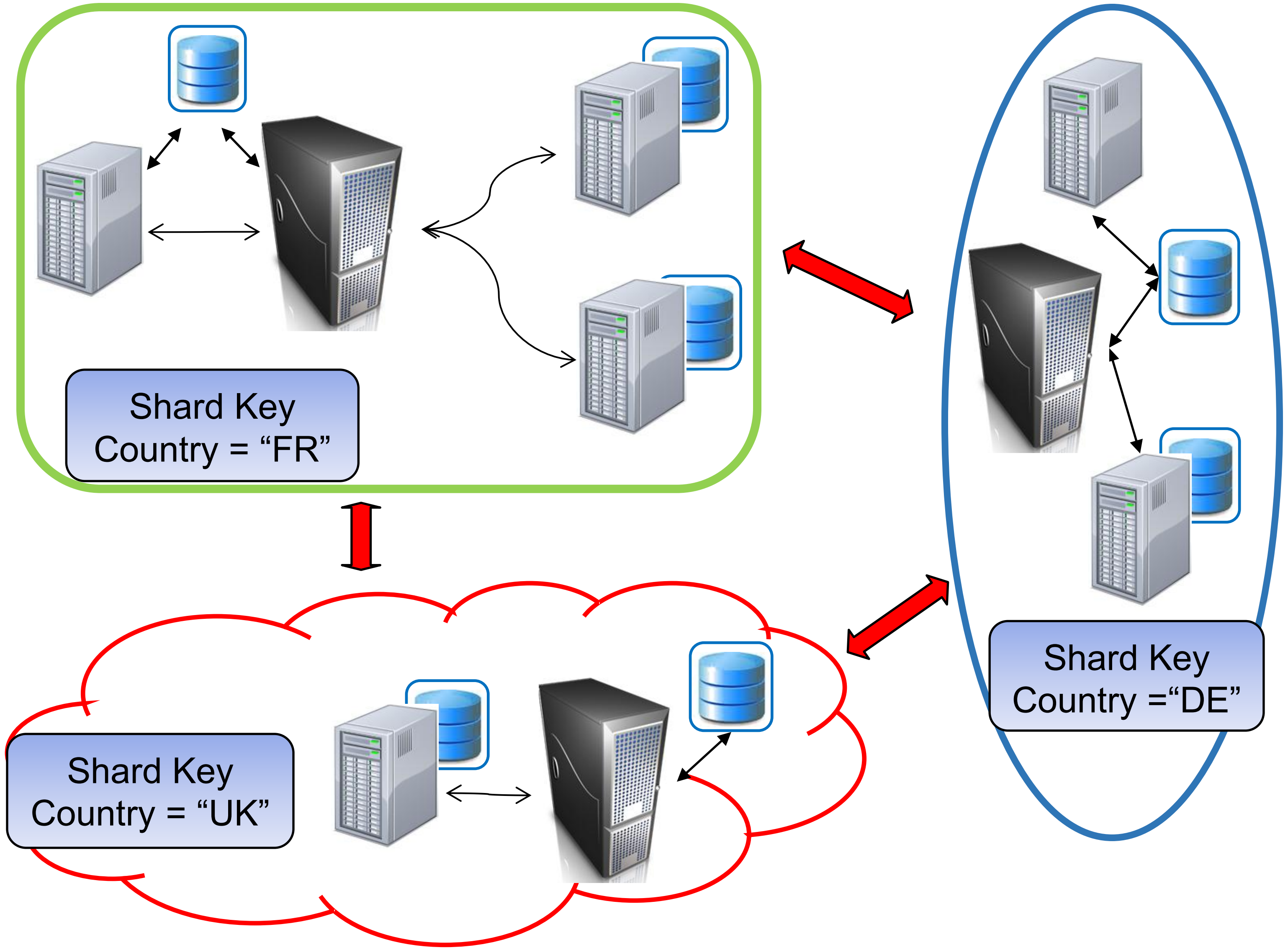
2 Create N Sharded Operators

3 Simultaneously Execute on Servers

4 Merge and Return Results



Sharding + Informix High Availability ➔ Your data is always available!



Sharding Pros & Cons

Pros

- Elasticity to add/remove nodes as business needs change
- Scale out not up OR scale out **and** up
- Commodity hardware
- Built on ER means heterogenous hardware and IFX versions supported

Cons

- Sharding an existing database could take time
- Eventual consistency*
- Still need HA solution since shards could be single point of failure
- Sharding strategy is important as changing *could be* very difficult

New Projects based on ER

- Smart Triggers
- Asynchronous Post Commit Triggers
- Loopback Replication
- *cdr migrate server*
- *…*

Shawn Moe – smoe@hcl.com
Nagaraju Inturi - nagaraju.inturi@hcl.com

Backup slides

Flexible Grid Use Cases

Flexible Grid Use Case 1

- Sue has an ER domain of 20 nodes.
- She needs to purge 4 million rows fro a replicated table.
- She could rely on basic ER functionalit to do this, but she is concerned about the impact to the network that such a operation would have.

Grid Based Statement Execution

- An individual statement can be executed as a grid statement as well as a stored procedure/function

```
execute procedure
ifx_grid_execute('grid1',
 'delete from orders where order_year <
2009');
```

  - The execution is replicated, not the results of the execution
  - The table 'orders' could be a raw table, or even contained within a non-logging database.
  - By default, the results would not be replicated by ER

Flexible Grid Use Case 2

- Jim needs to add 3 rows to a table “job_class" on each of the servers that he is responsible for.
- This table is not replicated and contains job classification for some new position that his company is creating.
- There are several hundred servers that this needs to be done on.
- He’s concerned that it will take forever to complete this task.

Grid Based Procedure Execution

In addition to DDL propagation, we can perform the execution of a procedure, function, or statement as a grid operation.

Create a new procedure across the grid

```
execute ifx_grid_connect('mygrid');
create procedure newjobclass()
insert into job_class values ("class1",…);
insert into job_class values ("class2",…);
insert into job_class values ("class3",…);
end procedure;
execute ifx_grid_disconnect();
```

Execute the new procedure across the grid

```
execute procedure
ifx_gird_procedure("mygrid","newjobclass()"
);
```

Flexible Grid Use Case 3

- Sam has been asked to add a new table on the databases that he is responsible for.
- Because of the potential size of the table, he wants to isolate it in a dbspace.
- The only problem is that he needs to first create that dbspace on each of the 500 nodes that he is responsible for.

Grid Based Function Execution

- The only difference between a function and a procedure is that a function will have a return value.
  - The return is saved in the syscdr database and can be viewed from `cdr list grid`

    ```
    database sysadmin;
    execute function ifx_grid_function(‘grid1’,
        ‘task(“create dbspace”,”dbsp3”,
    “/db/chk/chk3”,”8G”,”0”)’);
    ```

  - The above command would create a new 8GB dbspace “dbsp3” on all nodes within the grid.
  - By default the results of the function execution would not be replicated by ER

Easy cloning of new server

- Need to quickly setup test server from production server?
- Need to easily setup HDR, RSS, ER/Flexible grid Server ?

ifxclone

- Allows a snapshot copy of a system
- Utilizes backup/recovery logic
- Final clone can be an independent server, an RSS node, or an ER clone of the source node
- Requires that source allows the clone to be taken