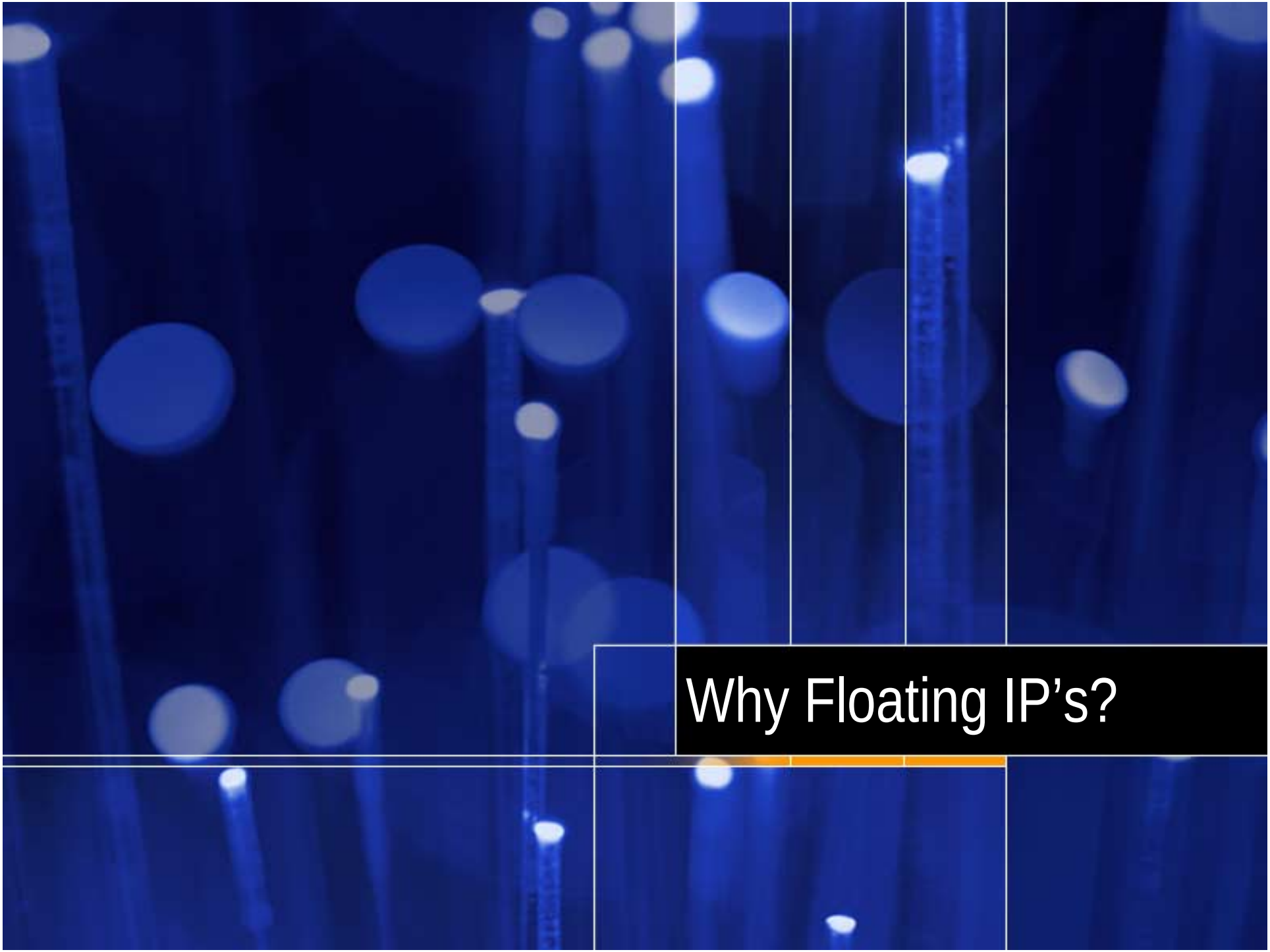
MC/ServiceGuard Case Study: Oracle Parallel Server with Floating IP's

Bill Marmagas
Metromedia Fiber Network Internet Solutions
485 Spring Park Place, Suite 1500, Herndon, VA 22031
703-796-3034
FAX: 703-467-5799
bill.marmagas@mmfn.com


Why Floating IP’s?

Three Ways to ServiceGuard OPS

- The “Right” Way
  - For those well behaved applications
- The Wrong Way
  - Anyone can do this one, folks
- Bill’s Way!
  - For many complex e-commerce sites

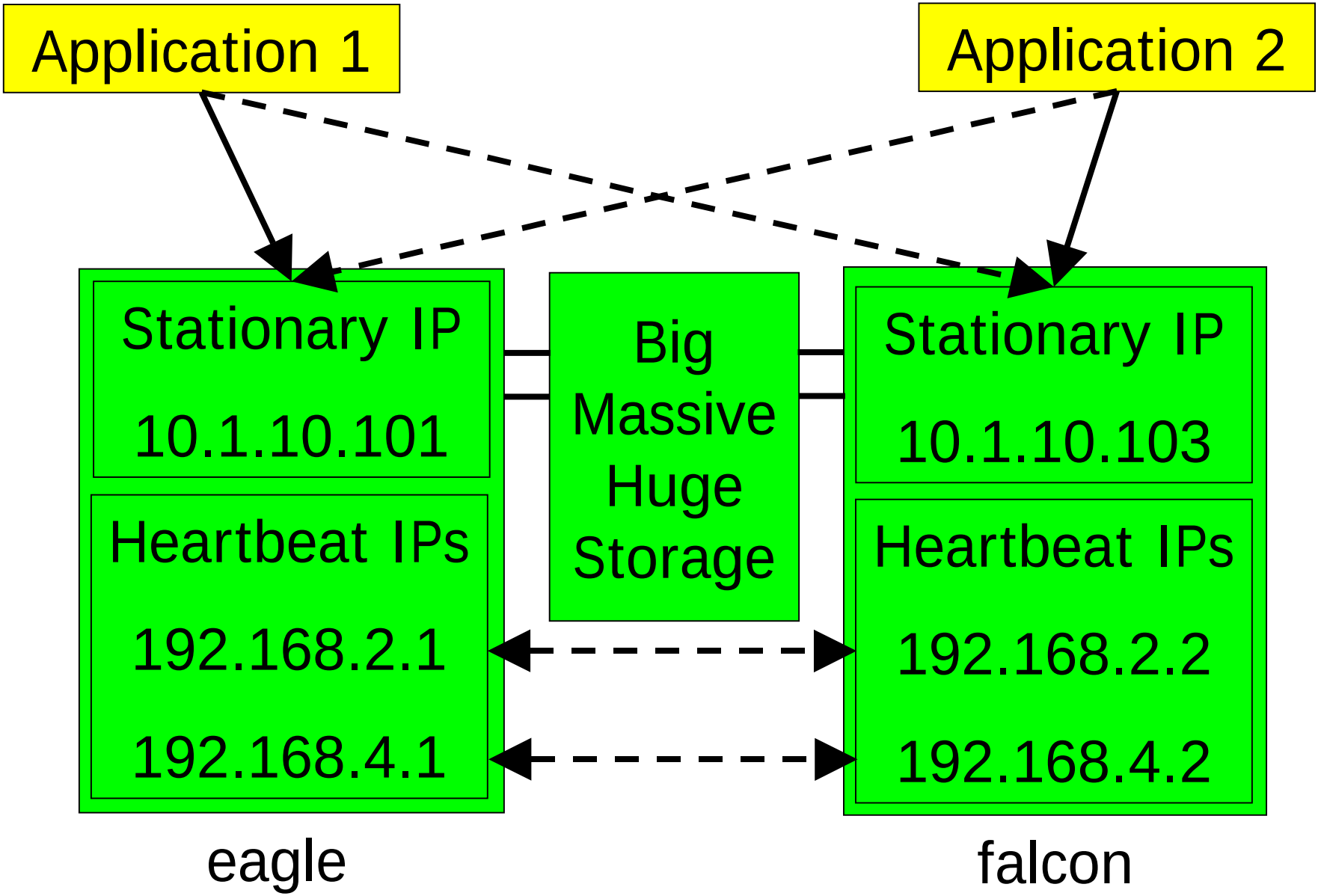
The Right Way

- **Based on tnsnames load balancing**

```
ops=
      (DESCRIPTION=
            (ADDRESS_LIST=
                  (ADDRESS= (PROTOCOL=TCP)
  (HOST=ops_node1) (PORT=1521))
                  (ADDRESS= (PROTOCOL=TCP)
  (HOST=ops_node2) (PORT=1521))
               )
               (CONNECT_DATA= (SID=ops_db))
```

Note: The ADDRESS lines have wrapped here.

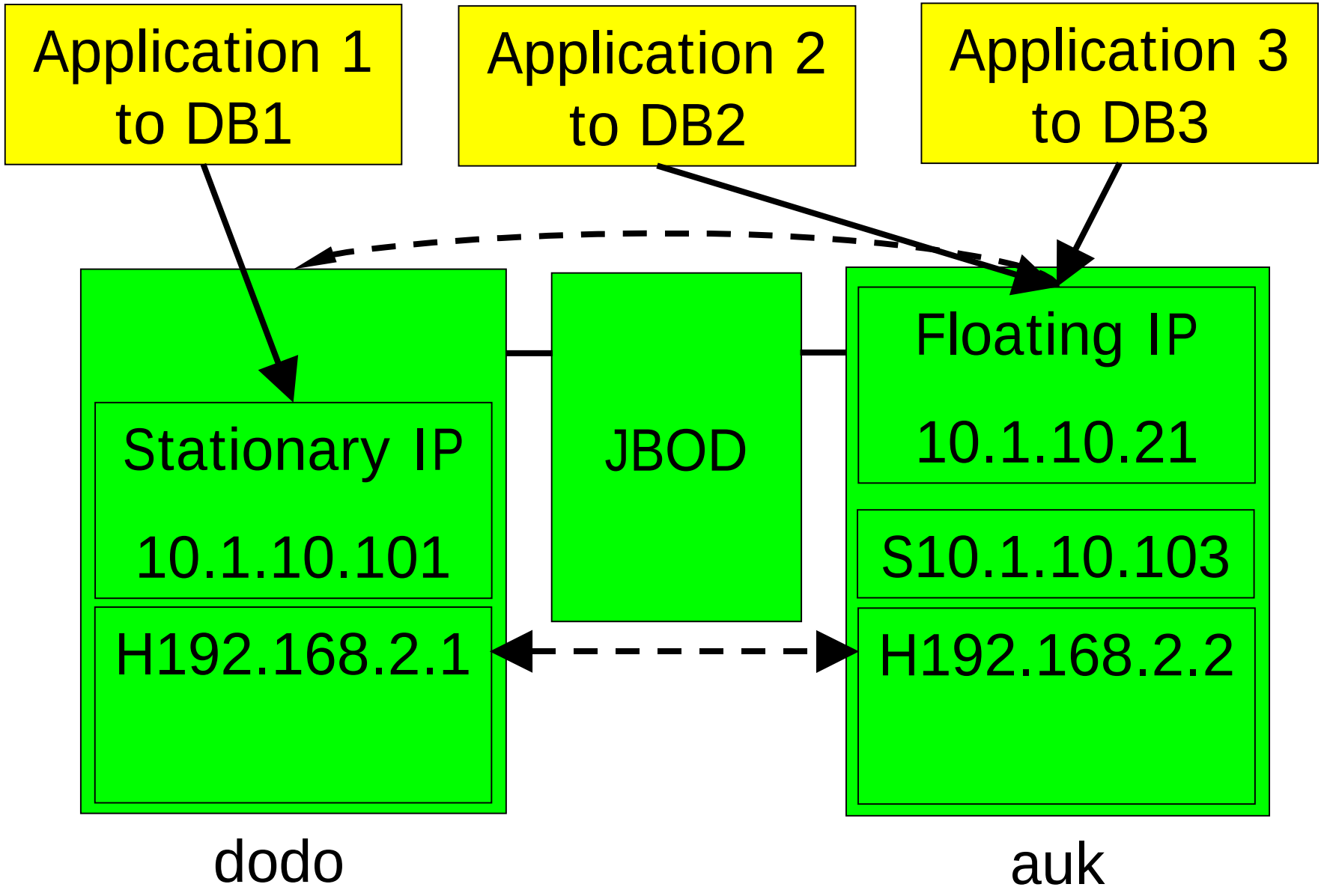
The Right Way In Pictures



The Wrong Way

- When the right way doesn't work, here are some bad ways to fix it
  - Point the application to the static IP on one server only
    - Loss of High Availability
  - Use a floating IP, but only configure one floating IP for multiple databases
    - Loss of Load Balancing

The Wrong Way In Pictures



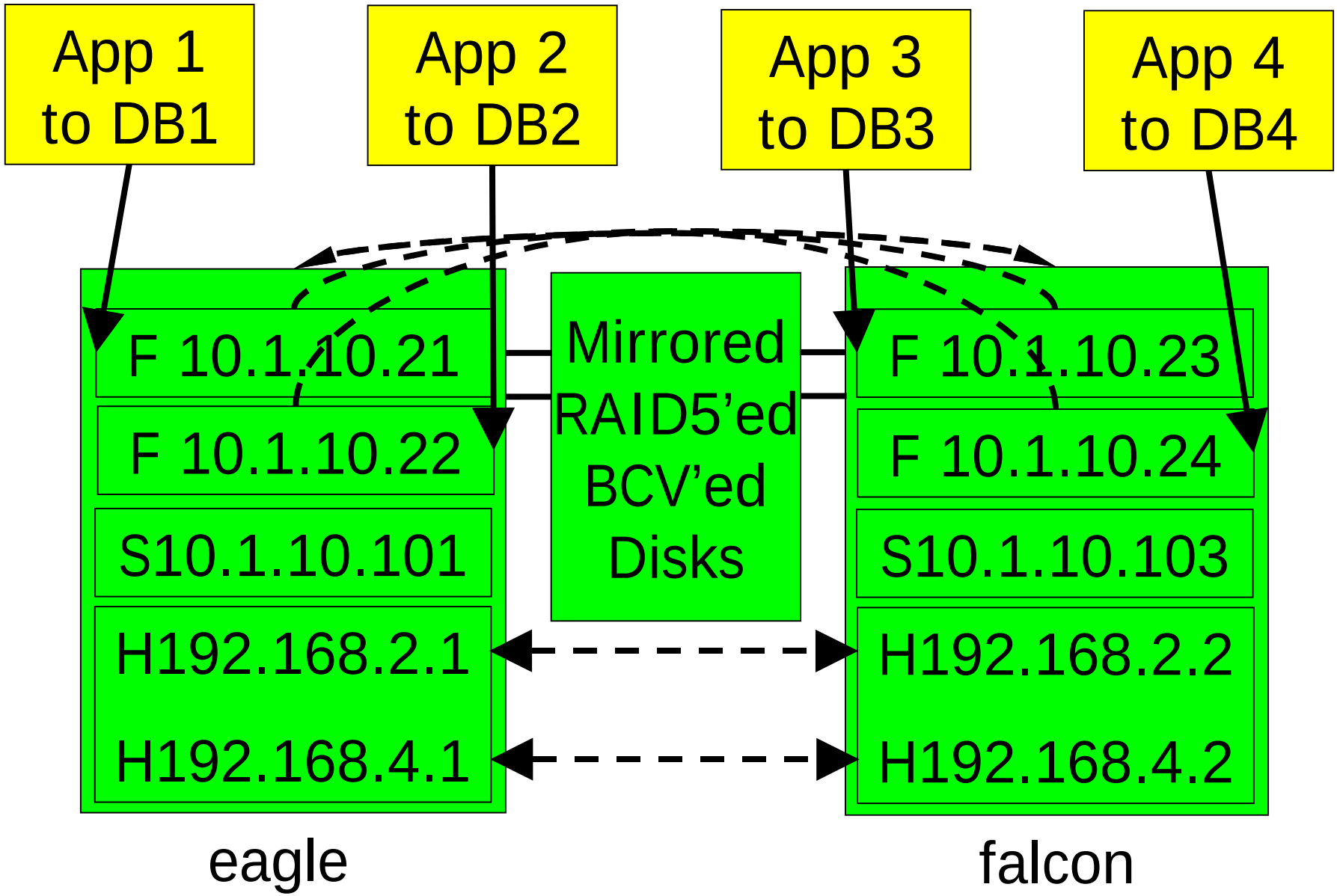
Why Bill's Way?

- When applications are not tuned for OPS, creating excessive locking overhead between OPS database instances
- When applications do not function well or cannot be made to work at all with multiple tnsnames IP addresses for connection to OPS databases

What Bill's Way Is

- Create a hybrid HA/OPS using virtual IP's balanced across systems
- This is accomplished with Floating IP/Listener Packages
  - A single virtual IP for access of each OPS database
  - Monitoring of the database instance and floating listener

Bill's Way In Pictures



Everyone Take Two Steps Back

- Before we jump into creating Floating IP/Listener Packages, let's cover the OPS-specific ServiceGuard configuration of:
  - The cluster
  - The database packages
- We'll look at the ASCII configuration files and scripts…


Configuring the Cluster

Editing cmclconfig.ascii

- Configuration parameters in cmclconfig.ascii specific to OPS
  - Some hints for Network Parameters
  - OPS Shared Mode Volume Groups
  - Distributed Lock Manager (DLM) or Group Membership Service (GMS) or “Nothing”

Network Parameters

- Two dedicated heartbeats (Crossover)
- Standby NIC for the Data LAN

```
NODE_NAME                eagle
  NETWORK_INTERFACE      lan3
    HEARTBEAT_IP         192.168.2.1
  NETWORK_INTERFACE      lan2
    HEARTBEAT_IP         192.168.4.1
  NETWORK_INTERFACE      lan0
    STATIONARY_IP        10.1.10.101
  NETWORK_INTERFACE      lan1
```

OPS Volume Groups

- **Shared Volume Groups (vgchange -a s)**

OPS_VOLUME_GROUP	/dev/vgepp
OPS_VOLUME_GROUP	/dev/vgepp_t2
OPS_VOLUME_GROUP	/dev/vgedi
OPS_VOLUME_GROUP	/dev/vgedi_t2
OPS_VOLUME_GROUP	/dev/vgepi
OPS_VOLUME_GROUP	/dev/vgepi_t2
OPS_VOLUME_GROUP	/dev/vgautosys
OPS_VOLUME_GROUP	/dev/vgautosys_t2
OPS_VOLUME_GROUP	/dev/vg_lock

DLM for OPS Versions Prior to 8.0

- Change DLM_ENABLED to YES
- Leave GMS_ENABLED as NO

DLM_ENABLED	YES
DLM_CONNECT_TIMEOUT	30000000
DLM_PING_INTERVAL	20000000
DLM_PING_TIMEOUT	60000000
DLM_RECONFIG_TIMEOUT	300000000
DLM_COMMFAIL_TIMEOUT	270000000
DLM_HALT_TIMEOUT	240000000

GMS for OPS 8.0.x

- In OPS 8.0.x and later, DLM parallel cache management is an internal component of OPS
- Leave DLM_ENABLED as NO
- Change GMS_ENABLED to YES
- Set Oracle GMS_LOCATION

```
GMS_ENABLED                    YES
GMS_CONNECT_TIMEOUT            30000000
GMS_LOCATION   /opt/oracle/product/8.0.5/bin/ogms
```

“Nothing” for OPS 8.1.x

- In OPS 8.1.x, the GMS daemon is an HP component known as cmgmsd
- Leave DLM_ENABLED as NO
- Leave GMS_ENABLED as NO
  - Note: With OPS 8.0.x *and* OPS 8.1.x in the same cluster, change GMS_ENABLED to YES


The Database Package

Database Package Structure

- Use one package on each node to start all the OPS DB instances on that node
- Name the package *nodename*_dbs
- Name the Package Configuration File *nodename*_dbs.ascii
- *nodename*_dbs.ascii calls the Database Package Control Script control.sh
- control.sh calls the custom Oracle Script *nodename*_oracle.sh

Editing *nodename*_dbs.ascii

- Configuration parameters in *nodename*_dbs.ascii specific to OPS
  - Only *nodename* can run the package
  - Run Script / Halt Script is control.sh
  - Package Switching is Disabled
    - Don't want a potentially damaged OPS instance to start on reboot
- We will use eagle as the *nodename* in the following file examples

eagle_dbs.ascii

```
PACKAGE_NAME                    eagle_dbs

NODE_NAME                       eagle

RUN_SCRIPT      /etc/cmcluster/eagle_dbs/control.sh
RUN_SCRIPT_TIMEOUT              NO_TIMEOUT
HALT_SCRIPT     /etc/cmcluster/eagle_dbs/control.sh
HALT_SCRIPT_TIMEOUT             NO_TIMEOUT

PKG_SWITCHING_ENABLED           NO
```

Database control.sh

```
VGCHANGE="vgchange -a s"

function customer_defined_run_cmds
{
/etc/cmcluster/eagle_dbs/eagle_oracle.sh start
        …

function customer_defined_halt_cmds
{
/etc/cmcluster/eagle_dbs/eagle_oracle.sh stop
        …
```

eagle_oracle.sh (Written for 8.0.5)

- See Handout A or Website for full script
- The only lines you need to change for each cluster, and perhaps each system, are:

```
export ORACLE_BASE=/opt/oracle
set -A ORACLE_SIDS eppprod epiprod ediprod autosys
```

- If you do not call your local listener lsn_local, you will have to edit that as well


The IP/Listener Package

IP/Listener Package Structure

- Each DB has a Floating IP/Listener package
- Package monitors DB instance and listener
- Name the package *database*_ip
- Name the Pkg. Conf. File *database*_ip.ascii
- *database*_ip.ascii calls the IP/Listener Package Control Script control.sh
- control.sh calls the custom DB Instance and Listener Monitor Script monitor.sh, and the custom Listener Script listener.sh

Editing *database*_ip.ascii

- Configuration parameters in *database*_ip.ascii specific to OPS
  - All nodes can run the package
  - Run Script / Halt Script is control.sh
  - Package Switching is Disabled
    - Don't want the package to start monitoring before the DBs start
- We will use eppprod as the *database* in the following file examples

eppprod_ip.ascii

```
PACKAGE_NAME                    eppprod_ip

NODE_NAME                       eagle
NODE_NAME                       falcon

RUN_SCRIPT      /etc/cmcluster/eppprod_ip/control.sh
RUN_SCRIPT_TIMEOUT              NO_TIMEOUT
HALT_SCRIPT     /etc/cmcluster/eppprod_ip/control.sh
HALT_SCRIPT_TIMEOUT             15

PKG_SWITCHING_ENABLED           NO
```

eppprod_ip.ascii Service

- We also need to configure a Service since we will be monitoring the OPS DB "eppprod" Instance on the Active Node

SERVICE_NAME	eppprod
SERVICE_FAIL_FAST_ENABLED	NO
SERVICE_HALT_TIMEOUT	10

IP/Listener control.sh

- **The Floating IP**

  IP[0]="10.1.10.21"

  SUBNET[0]="10.1.10.0“

- **The OPS Database Instance Monitor**

  SERVICE_NAME[0]=eppprod

  SERVICE_CMD[0]="/etc/cmcluster/oracle_monitor/
  monitor.sh eppprod"

  SERVICE_RESTART[0]=""

Note: The SERVICE_CMD line has wrapped here.

IP/Listener control.sh (con't)

- **The Floating Listener**

```
function customer_defined_run_cmds
{
/etc/cmcluster/oracle_listener/listener.sh start eppprod
        …

function customer_defined_halt_cmds
{
/etc/cmcluster/oracle_listener/listener.sh stop eppprod
        …
```

monitor.sh

- See Handout B or Website for full script
- Checks for ora_pmon, ora_smon, ora_lgwr, ora_dbw0, and ora_arch for given Instance

```
export ORACLE_SID=$1
set -A MONITOR_PROCESSES ora_pmon_${ORACLE_SID}…
for i in ${MONITOR_PROCESSES[@]}
do
    ps -ef | grep ${i} | grep -v grep > /dev/null
    if [[ $? != 0 ]]
        … exit 1 …
```

monitor.sh (con't)

- **Also checks for the Listener for this Database, and attempts one restart**

```
… lsnrctl status lsn_${ORACLE_SID} …
if [[ $? != 0 ]]
        then
                … lsnrctl start lsn_${ORACLE_SID} …
                … lsnrctl status lsn_${ORACLE_SID} …
                if [[ $? != 0 ]]
                        then
                                … exit 1 …
```

listener.sh

- See Handout C or Website for full script
- Starts / Stops Floating Listener on the Active Node

```
export ORACLE_SID=$2
if [ $1 = "start" ]
then
        … lsnrctl start lsn_${ORACLE_SID}
        …
elif [ $1 = "stop" ]
then
        … lsnrctl stop lsn_${ORACLE_SID}
```


Managing the Cluster

So You're in Production Now…

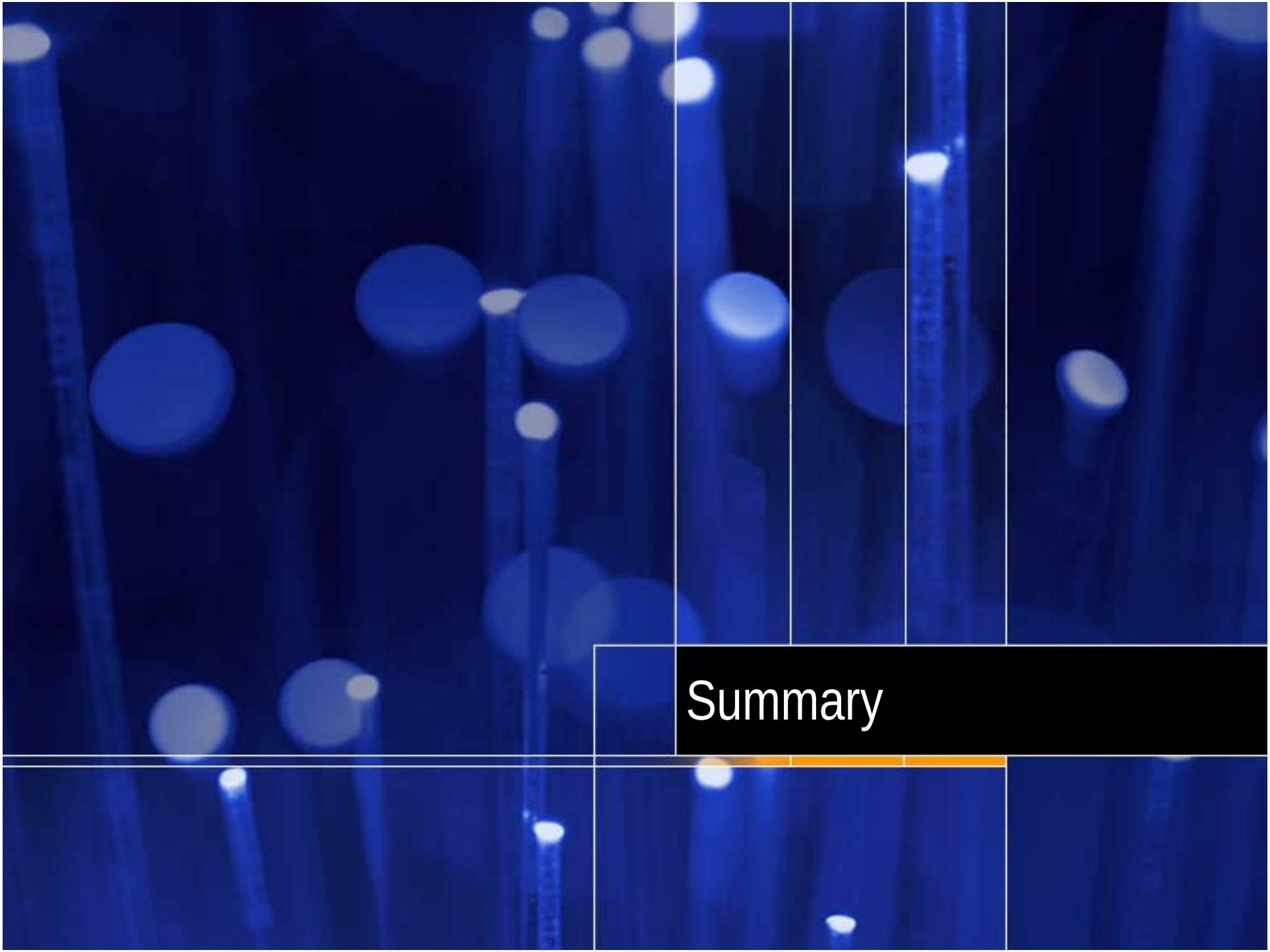
- Control your cluster, cluster nodes, and databases and listeners properly
  - Create a procedure document for cluster, cluster node, and database instance startup and shutdown
  - Communicate with your DBA's!
- Monitor your cluster, cluster nodes, and packages

Startup / Shutdown Document

- See Website for Sample document
- How to shutdown the cluster
- How to shutdown a node while continuing to run the *Oracle Package* and all the *IP/Listener Packages* on remaining node(s)
- How to shutdown individual DB instances on one or more nodes with the *Oracle Package* still running on all nodes, moving or stopping *IP/Listener Packages* as needed

Monitoring

- Monitor syslog for cluster messages
- Monitor package control logs
- Add monitoring agents, such as IT/O, to check processes and log files
- ServiceGuard SNMP --> NNM, ClusterView
- Custom alert script: e-mail, logger, SNMP trap, or integrated with monitoring agent
  - See Website for Example script framework using cmviewcl and STDOUT


Summary

What Did We (I Hope) Learn?

- Why / When to use Floating IP/Listeners
- The Cluster and Database Package Configuration Parameters specific to OPS
- Configure Packages and Create Scripts to start and stop database instances
- Configure Packages and Create Scripts to use a single virtual IP for access of each OPS DB, and to monitor DB and Listener
- Importance of Procedures and Monitoring

More Information

- The Presentation, Handouts, Sample shutdown / startup procedure document, Example monitor script framework, and All cluster and package configuration files and scripts are available on the Web at:

zorbanet.net/HP/HA-OPS

- Mirror site: dc.net/zorba/HP/HA-OPS