LCG Database Deployment &
Persistency Workshop

Database Availability

Impact on Applications and Higher Level Services

Dirk Duellmann, CERN IT

High Availability

- Many grid and experiment service need to be available *almost* all the time, because
  - Unavailability makes computing resources unusable
  - Recovery after service failure is costly
- Many services use a database as back-end to implement their service function
  - Availability expectations result in database availability requirements but numbers on different layers are not the same!
- Example:
  - If a service failure of 2 minutes results in 10k grid jobs of 8h length to abort then the loss of “grid computing time” is 4 hours (average)
  - Same is true between application and database availability

How to increase availability?

- We could use only perfect hardware and software
  - We haven't tried that yet :-)
- In the real world: Retry, Redundancy and Failover
  - Multiple components to implementing the same task/service
    - Eg multiple nodes in a RAC cluster, multiple mirrored disks, multiple network paths,
  - Retry failed operations (for a while…)
  - Failover to an alternate service (and back after recovery)
- Effort spent should scale with risk of unavailability
  - Risk = Probability of failure * Damage caused
- Applies to all layers in the system
  - h/w, network, db, higher level services, apps

Database Unavailability - Main Causes

- Planned interventions
  - Security patches - OS and Database s/w
    - Affect all boxes
    - Time constraint in some cases immediate -> quasi an unplanned intervention
  - "Normal" Software upgrades
    - As before but less time constraints
  - Hardware extension / replacement
- Unplanned interventions
  - Software failures including database overload
    - Cluster becomes unstable (eg because of timeouts)
  - Hardware failures
    - CPU/memory/etc, double disk failure
  - Human error

Planned Interventions - Status Today

- Increasing number of interventions can be done transparently
  - Thanks to new RAC and FibreChannel setup
- Many Oracle patches (including security ones) still require to bring down all cluster nodes
  - Oracle is aware of the problem and promises "rolling upgrade"
  - Db services - try to minimise the intervention time by
    - automising and testing patches (eg in the validation setup)
    - failing over to a DataGuard setup would be a possibility - but significant effort
- Can not expect to remove a service outages completely:
  - few minutes (failover to data guard and back to production)
  - Some 30 mins (prepared non-rolling upgrade)

Instabilities caused by Overload - Status Today

- Databases don't like overload
  - neither single servers nor database clusters can go beyond 100% CPU used
- Oracle cluster software detects node failures will issue node restart
  - Based on heartbeats / timeouts - works well outside overload conditions
- Need to leave sufficient (h/w) headroom to stay away from overload area
  - But Experiments/Grid s/w often do not control the database load caused by application running somewhere in the grid
  - But "culture" on physics side is: use all computing resources you can get
- Need to determine and agree on "standard" working conditions for key apps during the validation phase
  - length of sessions, number of sessions, max CPU use per application
- Introduce database "throttling" to avoid overload conditions
  - Normally: Queue db requests based on priorities (eg per application)
  - Exceptionally: Kill sessions which risk to destabilize the service affecting others

Impact on “Normal” Applications

- Limiting the damage - avoiding the “black hole” syndrome
  - Apps need to retry and wait at least on database connection attempt
    - This should allow to avoid draining grid or local job queues
  - CORAL release will include this functionality for LCG AA software
    - Based on experience with ATLAS connection library
- Failover if possible (eg read-only apps)
  - Logical database lookup (eg via CORAL)
    - Connect string is determined at runtime based on job location, replica locations and service availability
    - Avoids hard-coding of connection information we have today
  - Allows client to failover to other replicas or locally cached data
- Will have to accept that a few jobs will abort

Impact on Higher Level Services

- Central services - classified by risk
  - Grid services : Jamie's with availability targets
  - Experiment services: required by many grid jobs?
- Need to check if expected outages pose a problem to achieve their required availability
  - Developers & DBAs should validate if code handling of outages is working as expected
    - Does code re-issue failed queries? React connection failures? service failover?
  - Services often appear as a single user to the database
    - End-to-end logging is required to determine source of excessive use
- Need to schedule validation in time to avoid surprises during production
  - Use the validation service (either CERN T0 or 3D testbed)
  - Test plans should include schedule for these tests

Summary

- New RAC based service will provides much higher service availability than single box setups in the past
  - Adding DataGuard could increase availability further but would require significant additional effort
  - 100% availability not achievable
- Applications and services need implement connection retry and failover to take advantage of service redundancy and gracefully handle remaining short unavailability
  - POOL/CORAL implements this for user applications
  - Single job loss can not always be avoided.
  - Grid services may need more work to achieve their availability targets

Conclusions

- Security patches - will happen at least 4 times a year
  - Few minutes to one hour until DB rolling upgrades become reality
- Overload conditions - happen with similar frequency and duration today
  - Separate experiment/grid RACs and additional resources and flexibility will help
  - New applications / code changes / access pattern will make it worse
  - Database throttling based on results from validation is required to lower this risk
  - Service throttling and end-to-end monitoring in high level grid and experiment services will be essential to avoid/react on db overload
- Expected database outages need to be taken into account by the deployment plans of db users
  - Critical applications and services need validation!