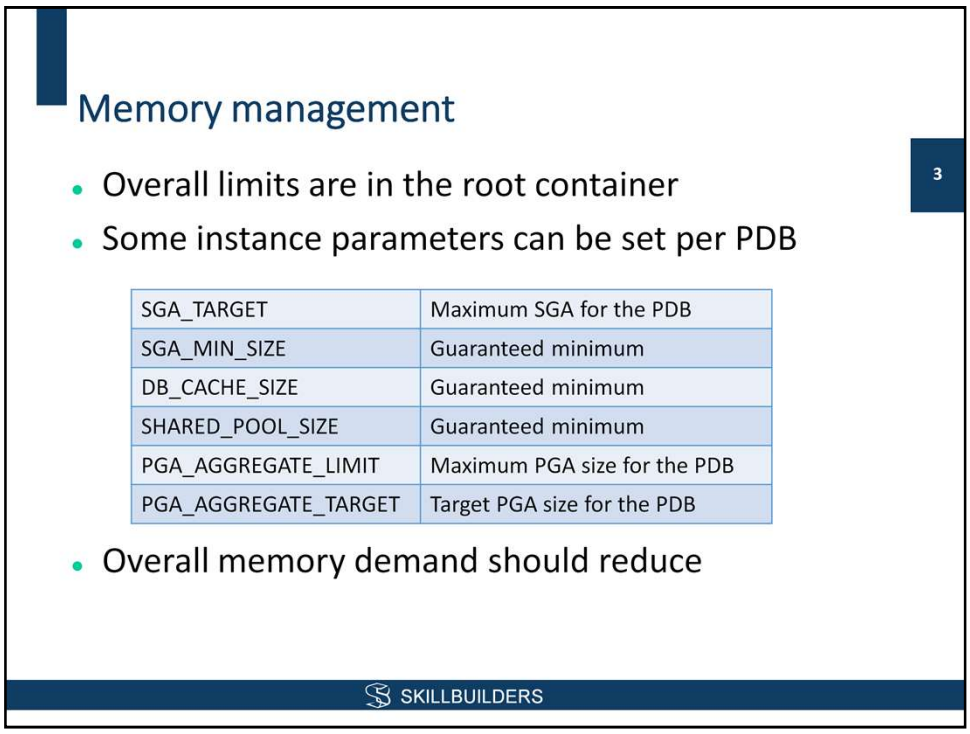
Lesson 6
Performance

Objectives

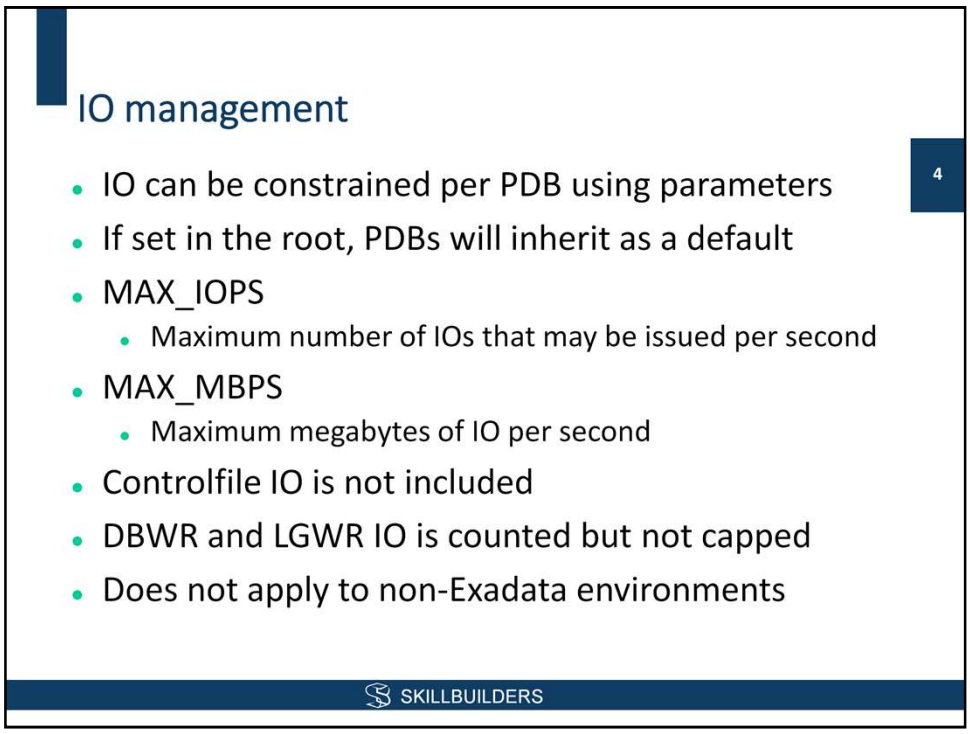
- Resource management
  - Memory and IO limits
  - Use of the Resource Manager
- Monitoring
- The AWR and Statspack



Generally speaking, applications in a CDB should perform best when Oracle has complete freedom to allocate resources. A basic consolidation proposition is that the more workloads you consolidate, the less likely it becomes that they'll all reach peak demand at the same time. In turn this tends to smooth out the net demand for any resource that can be elastically allocated. However, in some consolidation cases it will be necessary to guarantee certain resource availability and to ensure that activity in one container will not impact on another.

The minimum memory allocations are restricted: the total may not be more than half of that allocated for the instance in the root container.

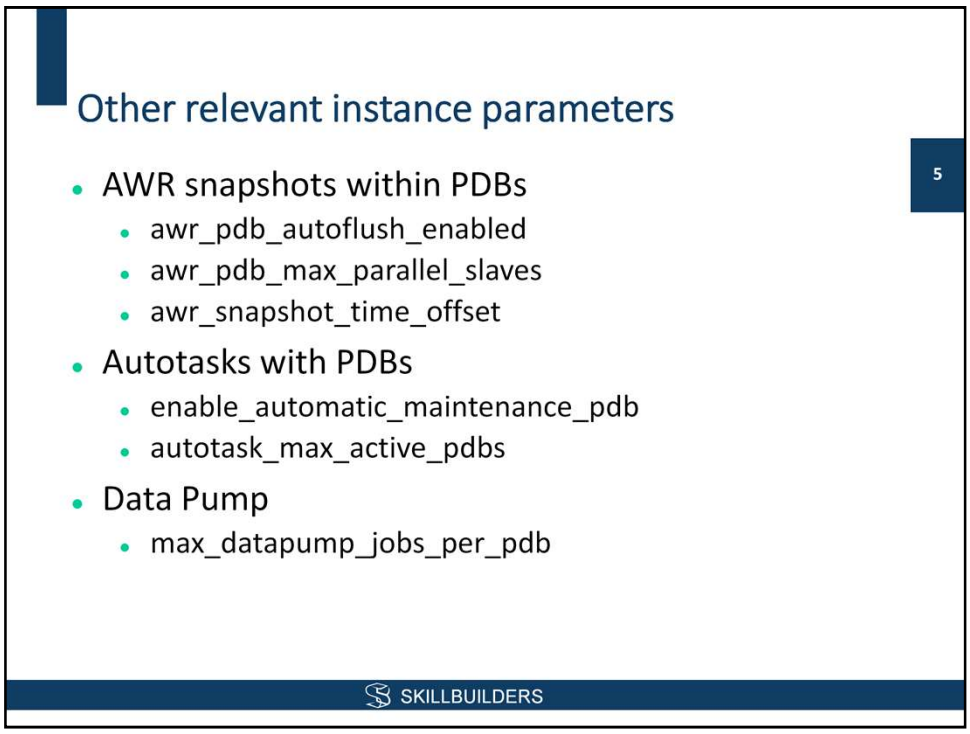
As a general rule, when several databases are consolidated into a CDB the overall memory demand should reduce considerably. The SGA will usually not need to be anything like the combined SGAs of the unconsolidated databases. PGA requirement may also reduce, if only because of the reduction in the number of background processes.



The IO limits are MAX_IOPS (maximum input/output operations per second) and MAX_MBPS (maximum megabytes per second). These can be set within each PDB, with default values being inherited from any settings in the root container. If set to zero (the default) there is no restriction.

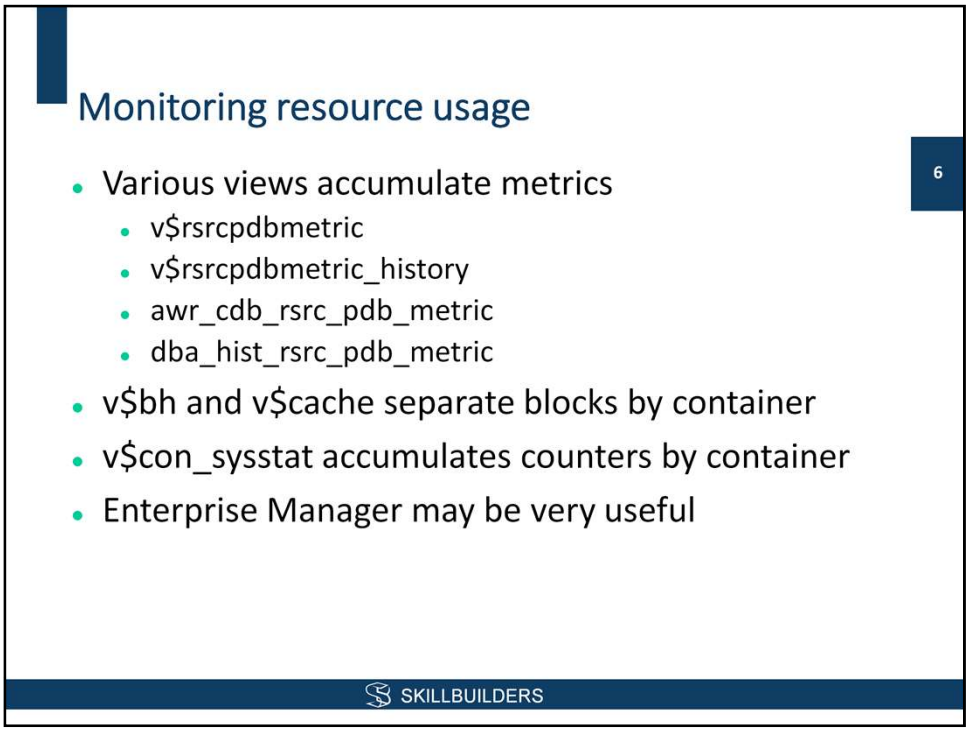
When selecting values, it may help to look at the figures exposed in (for example) AWR reports and to the operating system and hardware capabilities.

If a session is throttled back due to the PDB having reached its limit, it will wait on the event "resmgr: I/O rate limit".



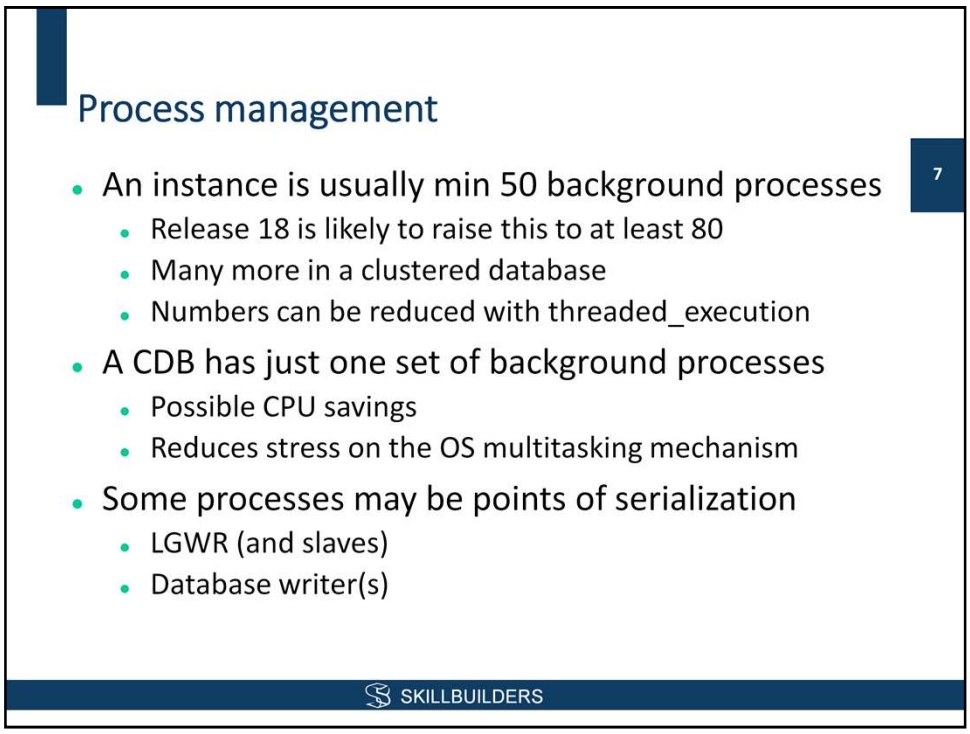
By default, full AWR snapshots are not enabled in PDBs. For this reason, AWR reports are incomplete and it is necessary to run the reports in the root container to gain the full picture. Enabling them may put a considerable strain on the system if a large number of PDBs attempt to gather their snapshots at the same time. This can be avoided by reducing the number of processes used or by giving each PDB a different offset from the hour.

The autotasks may also stress the system if they were all to run concurrently, as could a large number of concurrent Data Pump jobs.



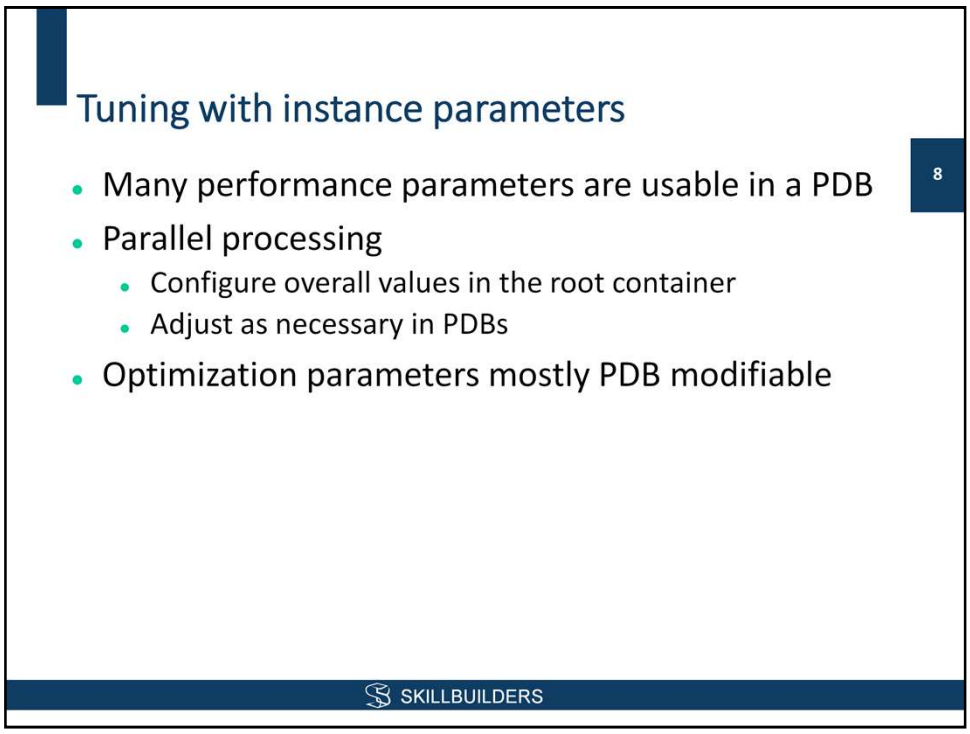
An example query (that would require the diagnostics pack):

```
pdby1> select con_id,max(iops),max(iombps),max(sga_bytes),max(pga_bytes)
  2  from cdb_hist_rsrc_pdb_metric group by con_id order by 1;

    CON_ID  MAX(IOPS) MAX(IOMBPS) MAX(SGA_BYTES) MAX(PGA_BYTES)
---------- ---------- ----------- -------------- --------------
         1 2682.67803  97.9240975      699767466              0
         2  .11335254  .006667796       28731813          17631
         3 385.346734   10.046446      287742681       39903161
         4 4.45521859  .102511675      161056752       43225152
         5 .052743348  .001387983       13892898        4999636

5 rows selected.

pdby1>
```

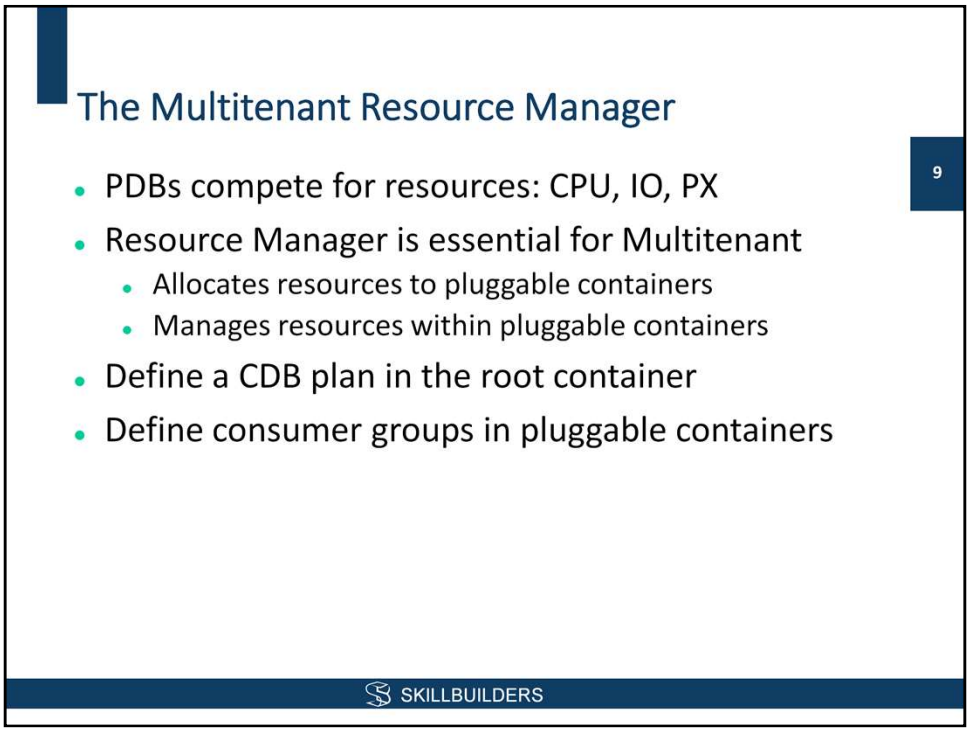


The number of processes in an Oracle instance is increasing with every release (and indeed with every patchset). One significant benefit of Multitenant is reducing the number of background process as compared with running a multi-instance environment. This immediately saves the PGA needed by many processes, and more importantly reduces the stress on the operating system's multi processing algorithm. Context switches as processes are brought on and off CPU are an expensive operation, and reducing the number of instances an operating system image must support can result in significant CPU savings.

However, there is only one instance (assuming RAC is not in use) and this can result in bottlenecks in demand for some processes. The must obvious is redo generation, which must be monitored.

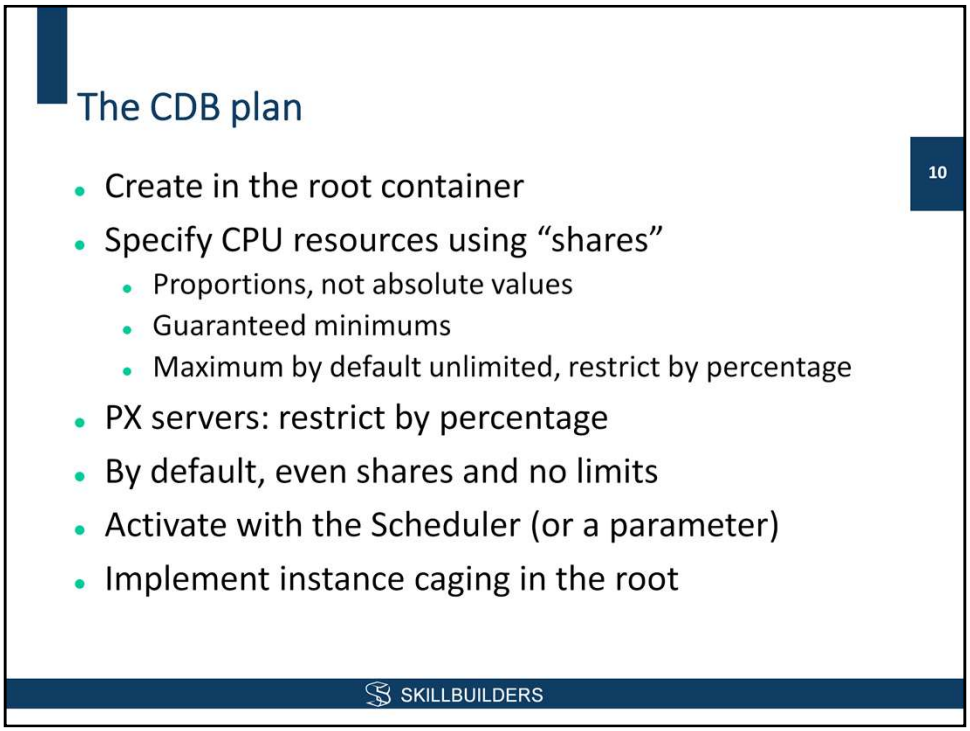


Many instance parameters are PDB modifiable, including most of those relevant to performance. The optimizer can be configured differently in each container, making it possible to consolidate databases that have very different workload characteristics.



The default Resource Manager scheduling algorithm is to treat all sessions equally. In a shared environment this makes it possible for sessions in one container to impact on sessions in other containers. If it is necessary to ensure that PDBs do get a certain level of resources (even if this means that sessions in other containers may experience degraded performance) the Resource Manager can do this.

A Resource Manager plan defined in the root container will allocate resources to pluggable containers, which can then manage them in the usual way.

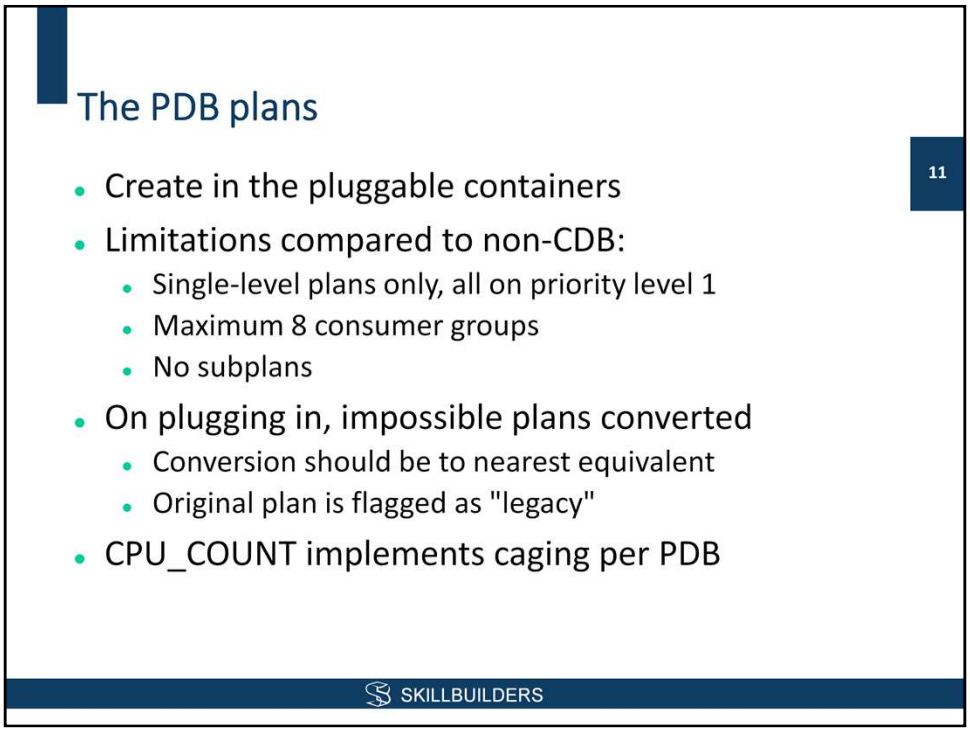


Resources a passed out to pluggable containers in the form of "shares". This is a proportional mechanism: different containers can be assigned a certain ratio of system resources, rather than an absolute value. Maximum values can be imposed, which will ensure that containers' resource usage will be capped even if there is no activity in other containers.

It is not possible to create a normal Resource Manager plan or any consumer groups in the root container. Attempting this will throw an error. The usual process of creating, validating, and saving a pending rea does apply.

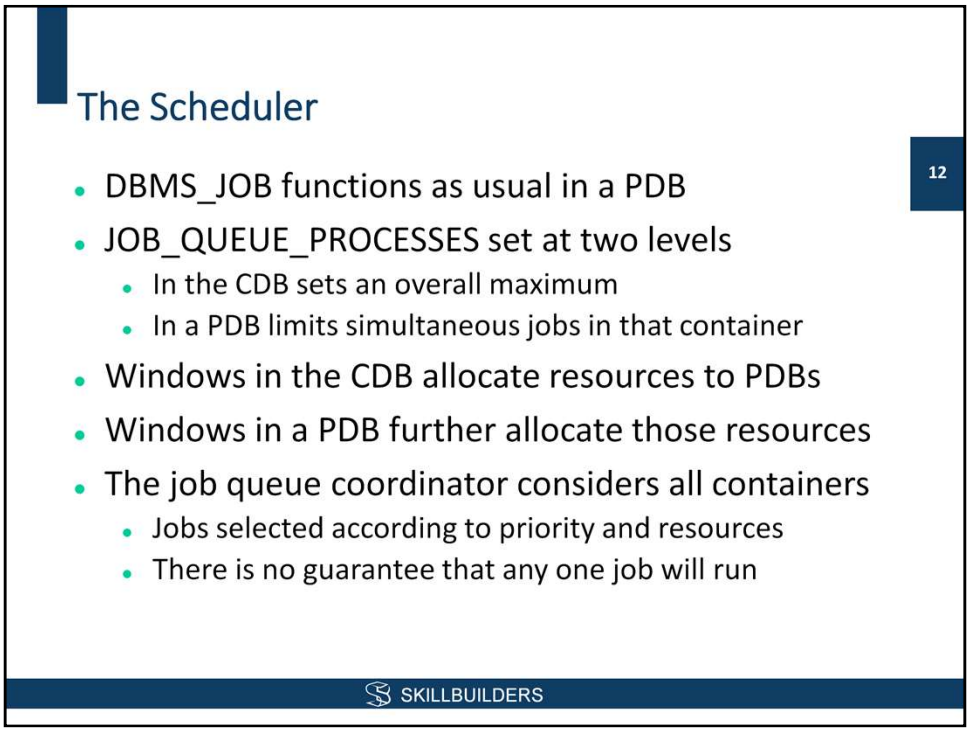
Setting the CDB plan can be accomplished with the Scheduler, or for testing purposes by adjusting the RESOURCE_MANAGER_PLAN parameter. The DEFAULT_PLAN and the DEFAULT_MAINTENANCE_PLAN will be used if nothing else has ben specified.

Instance caging with the CPU_COUNT parameter will restrict the CPU used by the entire CDB.



Within a PDB, create plans as usual, including use of the pending area.

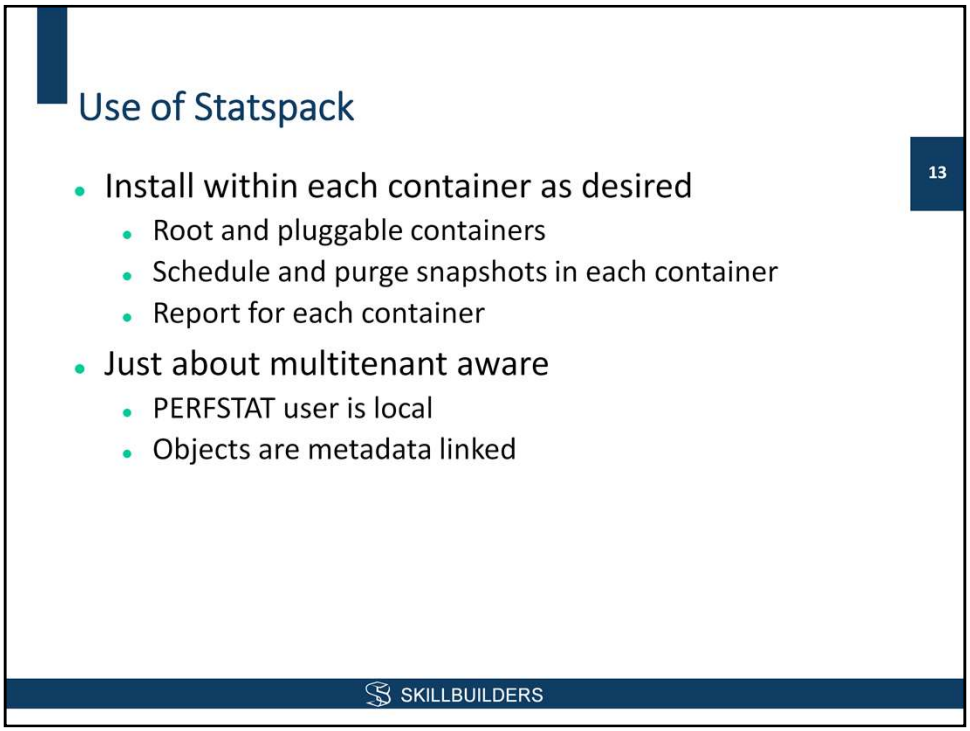
The limitations should not be too onerous. Eight consumer groups is probably plenty, and subplans are not widely used. The limitation of using priority level 1 only could be an issue.



The DBMS_JOB package is supported in multitenant environment, but has been deprecated for many years.

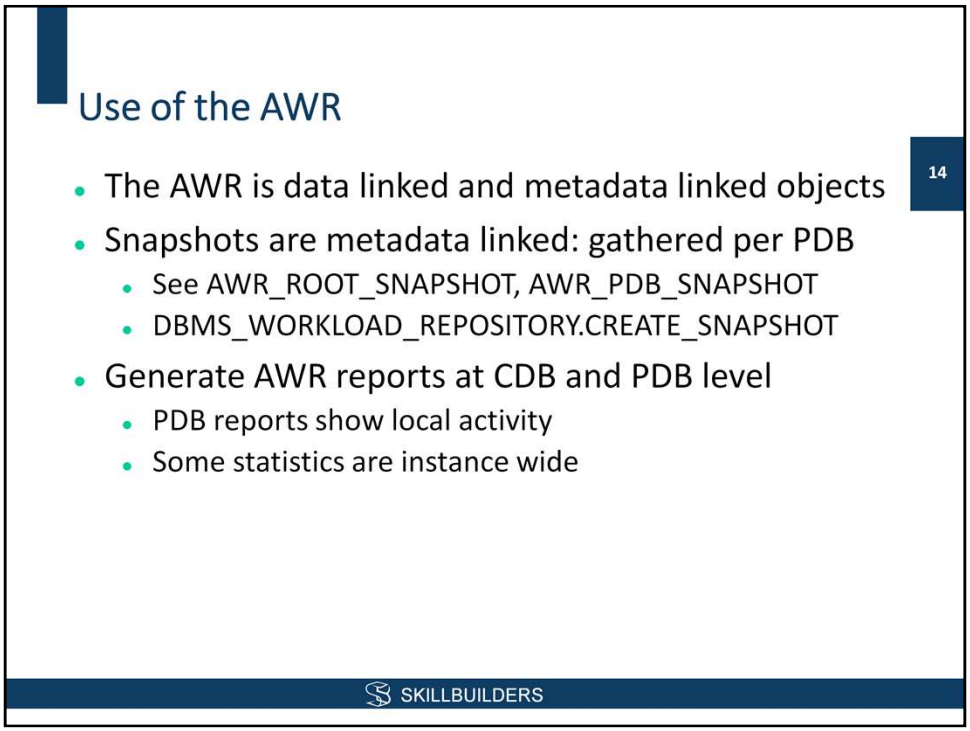
A pool of job queue processes is created in the root container as necessary up to the limit specified by the job_queue_processes parameter. The Job Queue Coordinator process then passes out jobs to the job queue processes according to the priority of the job's class and the job's scheduled start time, and the resources available to each container. The job queue process switches to the container for the duration of the job.

Setting job_queue_process=0 in the root container prevents jobs from running in the entire database.



In release 12cR1, Statspack was not truly compatible with a Multitenant environment. It can be installed, but following upgrade to release 2 it is probably best to remove it completely and re-install.

The PERFSTAT schema, if installed in the root container, is created as a local user without the C## prefix. This would be impossible, except that the Statspack creation script sets "_oracle_script"=true in order to bypass the usual rules. The schema objects are created as metadata linked objects, so that when installing in pluggable containers only the segments are created: the object definitions are maintained in the root container.



Each container has its own SYSAUX tablespace with an AWR. The tables and views within it are either data linked (structure and data are shared) or metadata linked: data is different per container. This means that each container can accumulate and report on its own activity, and from the root container one can report on the whole CDB. If a PDB is moved from one CDB to another, its AWR will move with it.

To gain the full functionality of the AWR reports in PDBs, snapshots must be enabled with the awr_pdb_autoflush_enabled parameter.

Performance management: summary

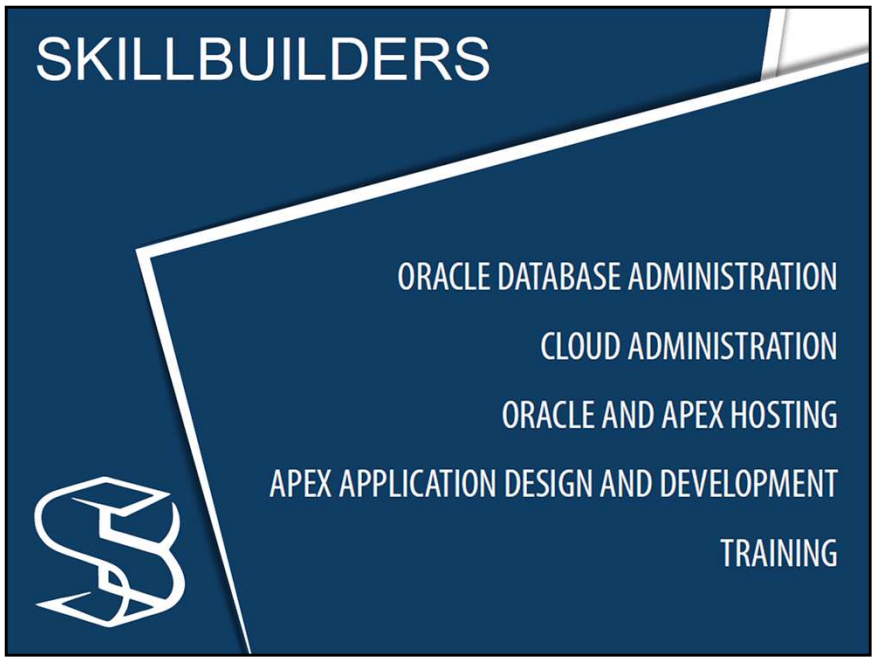
- Resource usage can be managed per container
- Use of the Resource Manager is essential
- Cloud control may be the best monitoring tool
- The AWR and Statspack are multitenant aware

Exercise

- Use the Multitenant Resource Manager
- Monitor performance

16

SKILLBUILDERS


SKILLBUILDERS
ORACLE DATABASE ADMINISTRATION
CLOUD ADMINISTRATION
ORACLE AND APEX HOSTING
APEX APPLICATION DESIGN AND DEVELOPMENT
TRAINING