# Greenplum Resource Management

Pivotal® Greenplum Database

Pivotal™

# Greenplum Workload Management

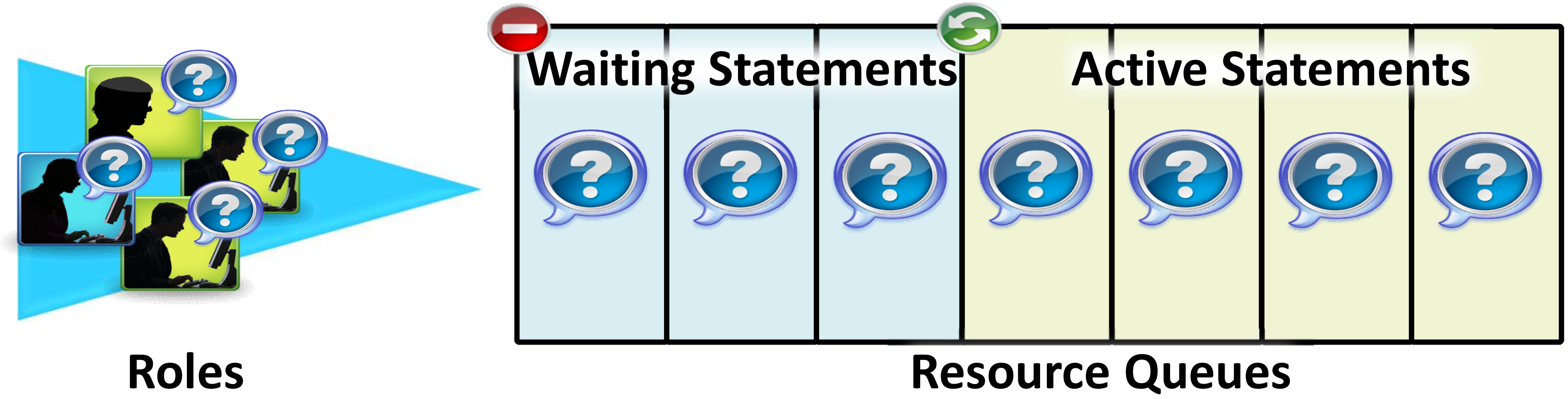
Let us examine the following:

- What is workload management?
- How do you create resource queues?
- How do you assign roles to resource queues?
- What is the runtime evaluation of resource queues?
- What are resource queue configuration parameters?
- How do you view the status of resource queues?

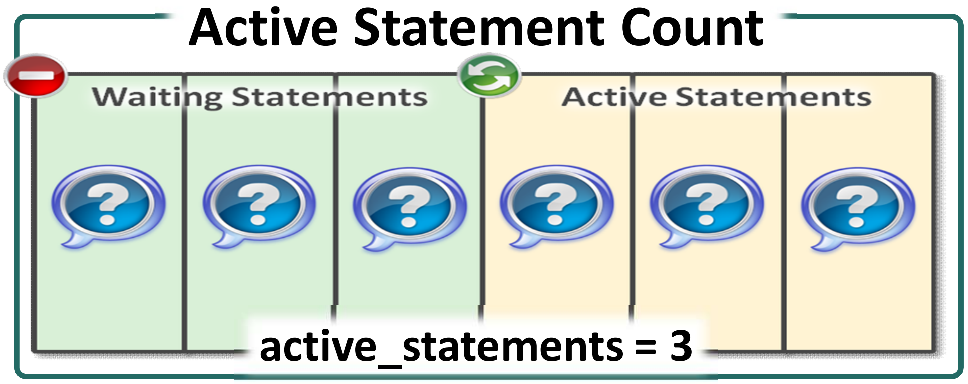
# Workload Management Overview

Workload management:

- Is used to limit the number of active queries
- Is meant to prevent overloading of system resources
- Looks at resource allocation from statement-level point-of-view

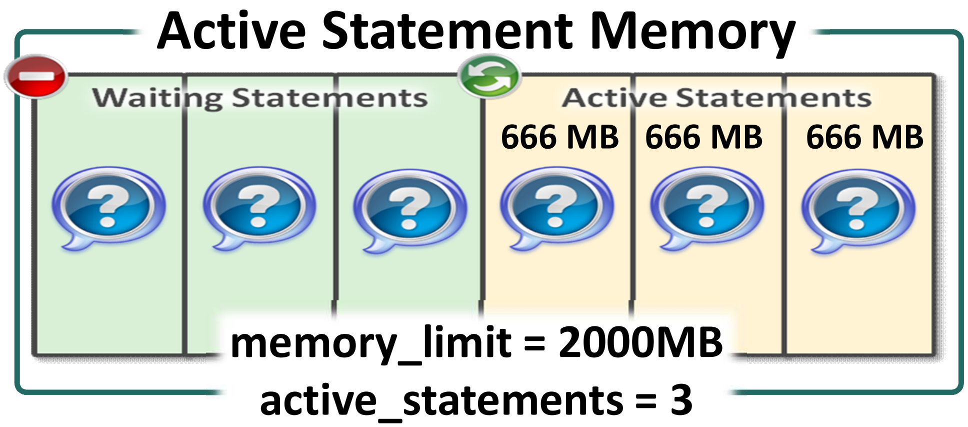


# Configurable Limits on a Queue



This parameter determines query <u>entry</u> into the system to actively run.

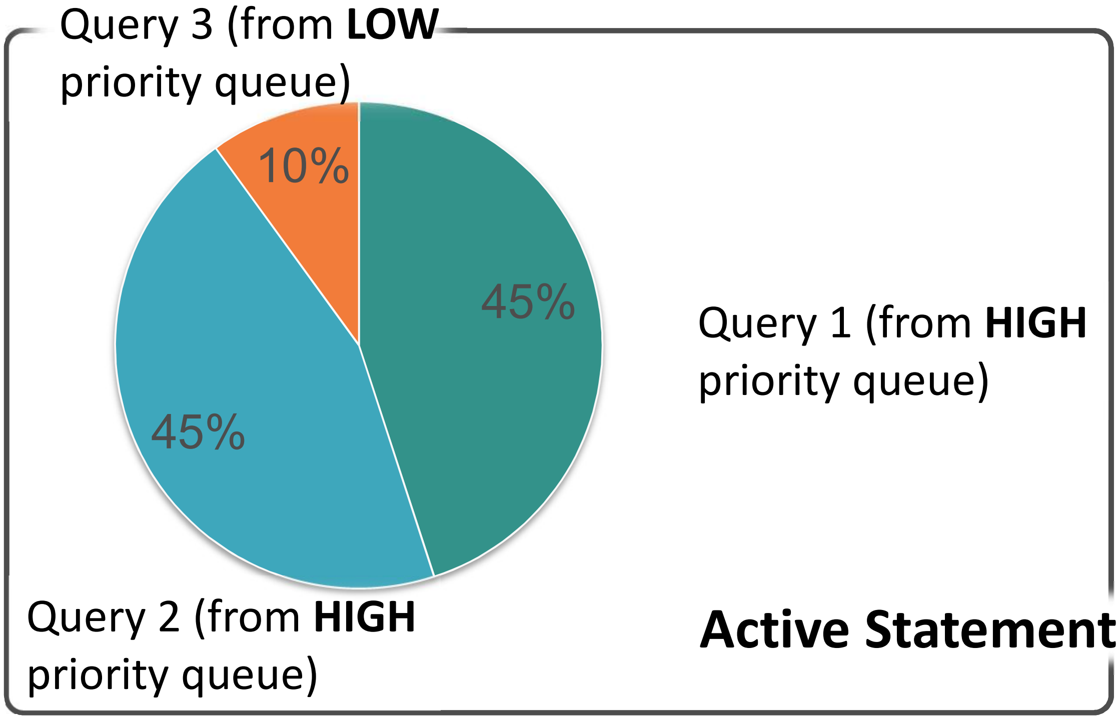
The maximum number of queries executing in the active queue is limited by the `active_statements` limit.



This parameter determines resources <u>allocated</u> to the query.

Each query submitted to the queue will be given a portion of the memory. This amount is based on the amount of queries that can run in the queue or the cost of the query.

# Configurable Limits on a Queue (Cont)

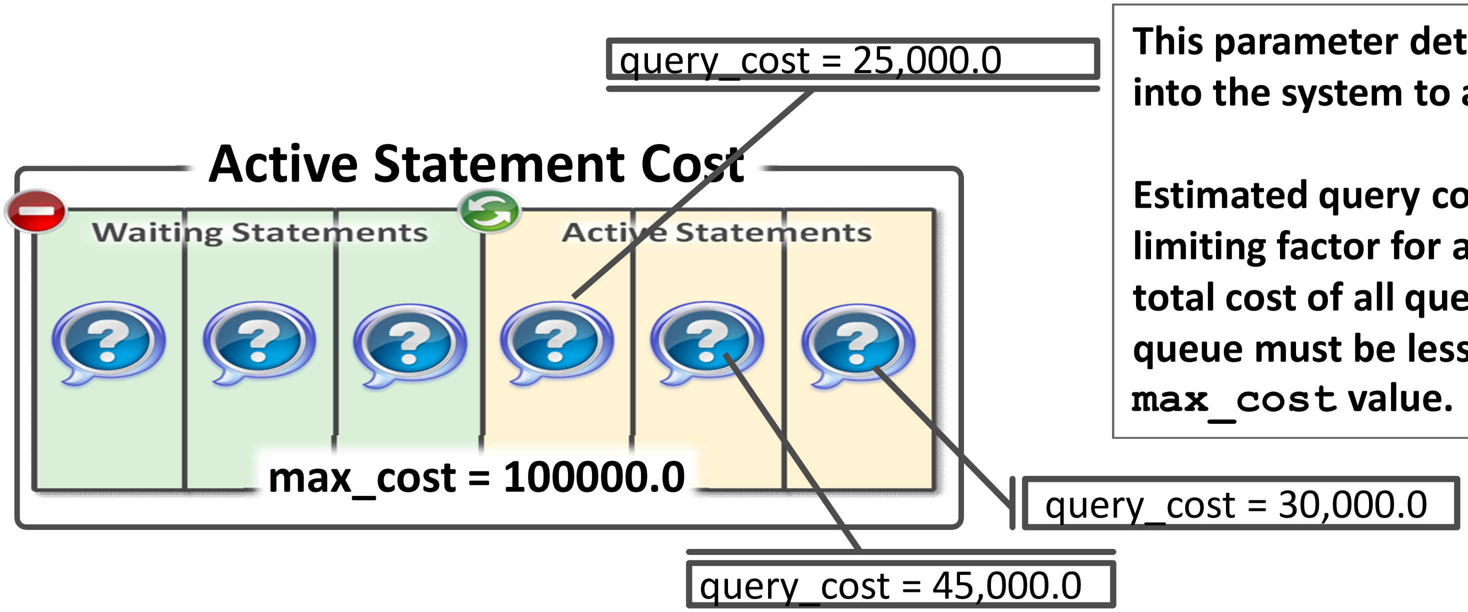


**Active Statement Priority**

**This parameter determines resources allocated to the query.**

**Queries with higher priorities consume more CPU resources over queries with lower priorities. Queries with the same priorities will evenly share CPU resources.**

# Configurable Limits on a Queue (Cont)



**This parameter determines query <u>entry</u> into the system to actively run.**

**Estimated query costs can be used as a limiting factor for a resource queue. The total cost of all queries running in the queue must be less than or equal to the `max_cost` value.**

# Creating Resource Queues

To create resource queues and manage thresholds, use the following commands:

| Action | SQL Syntax |
|---|---|
| Create or alter a resource queue | CREATE\|ALTER RESOURCE QUEUE name WITH (MAX_COST=n \| **ACTIVE_STATEMENTS=n** [,COST_OVERCOMMIT] [,MIN_COST=n] [,**MEMORY_LIMIT=‘nM\|GB’] [,PRIORITY=level]**) |
| Drop a resource queue | DROP RESOURCE QUEUE name |

# Managing CPU Utilization with Priorities

Priorities:

- Control a resource queue’s consumption of resources
- Are managed with the following SQL commands:

| Action | SQL Syntax |
|---|---|
| Create a resource with a specific priority | `CREATE RESOURCE QUEUE` *`name`* `WITH (PRIORITY=`*`level`*`)` |
| Alter a resource queue to have a specific priority | `ALTER RESOURCE QUEUE` *`name`* `WITH (PRIORITY=`*`level`*`)` |

- Can be one of the following:
  - `MIN`
  - `LOW`
  - `MEDIUM` (default setting if not specified)
  - `HIGH`
  - `MAX`

# Managing Memory Allocation with Memory Limits

Query memory allocation is controlled with:

- MEMORY_LIMIT parameter for the resource queue:

| Action | SQL Syntax | Memory Allocation per Query |
|---|---|---|
| Create or alter a resource with a specific memory limit and maximum number of statements | CREATE\|ALTER RESOURCE QUEUE *name* WITH (ACTIVE_STATEMENTS = x, MEMORY_LIMIT=‘xxxM\|GB’) | MEMORY_LIMIT/ ACTIVE_STATEMENTS |
| Create or alter a resource queue to have a specific memory limit with MAX_COST defined | CREATE\|ALTER RESOURCE QUEUE *name* WITH (MAX_COST = x.x, MEMORY_LIMIT=‘xxxM\|GB’) | MEMORY_LIMIT * (query_cost / MAX_COST) |

- At the database level with statement_mem and max_statement_mem

# Creating Resource Queues – Examples

**Example: Create a resource queue with up to 3 active queries**

```
CREATE RESOURCE QUEUE adhoc with
   (ACTIVE_STATEMENTS=3);
```

**Example: Create a resource queue with a planner cost limit**

```
CREATE RESOURCE QUEUE webuser WITH
   (MAX _COST=100000.0);
```

**Example: Create a resource queue with a minimum cost limit**

```
CREATE RESOURCE QUEUE adhoc WITH
   (ACTIVE_STATEMENTS=10, MIN_COST=100.0);
```

# Assigning Roles to Resource Queries

Resource queues:

- Must be assigned at the user-level (group-level ignored)
- Do not affect superuser roles

| Action | SQL Syntax |
|---|---|
| Add a resource queue to a role | `ALTER ROLE role_name RESOURCE QUEUE queue_name;` |
| Create a role and assign it to a resource queue | `CREATE ROLE role_name WITH LOGIN RESOURCE QUEUE queue_name;` |

**Note:** All users are assigned to a resource queue, even if one is not specified. The default resource queue is `pg_default`.

# Runtime Evaluation of Resource Queues

For runtime evaluation:

- Resource queues are evaluated independently of each other
- Superusers (and unassigned roles) are exempt
- Queries are evaluated on first-in, first-out basis

If a query causes the queue to exceed its limits:

- Query must wait until queue resources are free
- Query must wait until queue is idle (`COST_OVERCOMMIT=TRUE`)
- Query will never run (`COST_OVERCOMMIT=FALSE`)

Evaluated SQL statements include:

- `SELECT`, `SELECT INTO`, `CREATE TABLE` AS `SELECT`, `DECLARE CURSOR`
- (optional) `INSERT`, `UPDATE`, `DELETE`

# Resource Queue Configuration Parameters

The following parameters are used to configure resource queues:

| Configuration Parameter | Default Value | Description |
|---|---|---|
| max_resource_queues | 9 | The maximum number of resource queues in the system |
| max_resource_portals_per_transaction | 64 | The maximum number of open cursors per transaction |
| resource_select_only | on | Determines which queries are managed by resource queues |
| stats_queue_level | off | Enables the collection of statistics on resource queue usage |
| resource_cleanup_gangs_on_wait | on | Clean up idle segment worker processes |

# Memory Utilization System Parameters

| Configuration Parameter | Default Value | Description |
|---|---|---|
| gp_resqueue_memory_policy | eager_free | The query plan is divided into stages and Greenplum Database eagerly frees memory allocated to a previous stage at the end of that stage's execution, then allocates the eagerly freed memory to the new stage |
| statement_mem<br>max_statement_mem | 125 (MB)<br>2000 (MB) | statement_mem: Allocates segment host memory per query<br>max_statement_mem: Sets the maximum memory limit for a query. (seghost_physical_memory) / (average_number_concurrent_queries) |
| gp_vmem_protect_limit | 8192 (MB) | Upper memory boundary that all query processes can consume on a segment host |
| gp_vmem_idle_resource_timeout | 18000 (Millseconds) | If database session is idle for longer than the time specified, the session will free system resources (such as shared memory), but remain connected to the database |
| gp_vmem_protect_segworker_cache_limit | 500(MB) | If a query executor process consumes more than this configured amount, then the process will not be cached for use in subsequent queries after the process completes |

# Viewing Status of Resource Queues

The following query system tables provide insight into resource queues:

| System Table | Description |
|---|---|
| `pg_resqueue` | Resource queues and attributes |
| `gp_toolkit.gp_resq_role`<br>`pg_roles` | Role to resource queue assignments |
| `pg_locks` | Queues that have waiting statements |
| `pg_stat_activity` | Process information about active and waiting queries |

To view queue status and statistics, access the following tables:

| System Table | Description |
|---|---|
| `gp_resqueue_status`<br>`pg_resqueue_status` | Queue limits, number of active and waiting queries |
| `pg_stat_resqueues` | Queue statistics and performance over time |

# Wrapping Up

- Roles and privileges
- Role assignment to object privileges
- Security issues that can affect your database and corresponding data
- Creating roles with specific privileges to control access
- Isolating users at the physical and functional layers
- Workload management process