Greenplum Database Tuning

Pivotal® Greenplum Database

Pivotal™

Approaching a Performance Tuning Initiative

The following key points should be followed when tuning:

- Set performance expectations by defining goals
- Set benchmarks
- Know your baseline hardware performance for throughput and capacity
- Know your workload:
  - Heavy usage times
  - Resource contention
  - Data contention
- Focus your optimizations

This slide is intentionally left blank.

Common Causes of Performance Issues

The following are common causes of performance issues:

- Hardware issues / failed segments
- Resource allocation
- Contention between concurrent workloads
- Inaccurate database statistics
- Uneven data distribution
- SQL formulation
- Database design

Pivotal™

Common Causes of Performance Issues

The following are common causes of performance issues:

- Hardware issues / failed segments
- Resource allocation
- Contention between concurrent workloads
- Inaccurate database statistics
- Uneven data distribution
- SQL formulation
- Database design

Pivotal

Common Causes of Performance Issues

The following are common causes of performance issues:

- Hardware issues / failed segments
- Resource allocation
- Contention between concurrent workloads
- Inaccurate database statistics
- Uneven data distribution
- SQL formulation
- Database design

Hardware Issues

Common hardware failures include:

- Disk failures
- Host failures
- Network failures
- OS not tuned for Greenplum
- Disk Capacity:
  - 70% maximum recommended
  - VACUUM after updates, deletes and loads
- VACUUM configuration parameters

Pivotal™

Hardware Issues – `VACUUM` Configuration Parameters

Set the following `VACUUM` configuration parameters:

- `max_fsm_relations`:
  - This parameter should be set to *tables* + *indexes* + *system tables*
  - Sets the number of relations for which free space will be tracked in the memory free-space map
- `max_fsm_pages`:
  - This parameter is equal to *16* * *max_fsm_relations*
  - Sets the number of disk pages for which free space will be tracked

Resource Allocation and Contention

To work around resource allocation issues:

- Greenplum resource queues
  - Limit active queries in the system
  - Limit the size of a query a particular user can run
- Perform admin tasks at low usage times
  - Data loading, ETL
  - `VACUUM` and `ANALYZE`
  - Backups
- Design applications to prevent lock conflicts
  - Concurrent sessions not updating the same data at the same time
- Set resource-related configuration parameters

Setting Resource Related Configuration Parameters

Resource-related configuration parameters include:

- `work_mem = 32MB`
- `maintenance_work_mem = 64MB`
- `shared_buffers = 125MB`

Example: Set and reset a configuration parameter

```
=# SET work_mem TO ‘200MB’;
=# …SQL statements…;
=# RESET work_mem;
```

Example: Set a configuration parameter for a role

```
ALTER ROLE admin SET maintenance_work_mem = 100000;
```

This slide is intentionally left blank.

Setting Memory Management Parameters

Memory management parameters include:

- `statement_mem = 125 MB`
- `max_statement_mem = 2000 MB`
  `(segment_physical_memory/`
  `average_number_concurrent_queries)`
- `gp_vmem_protect_limit = 8192`
  `(X * physical_memory )/primary_segments`

Note: These parameters are used by Greenplum only when `gp_resqueue_memory_policy` is set to `eager_free` or `auto`.

Database Statistics – `ANALYZE`

Greenplum:

- Uses a statistics-based query planner
- Collects information such rows and range of values
- Uses `ANALYZE` to collect statistics. It should be run after:
  - Data loads
  - Restores from backups
  - Changes to schema
  - Inserts, updates, or deletes

Configuring Statistics Collection

Use the following to configure statistics collection:

- `default_statistics_target = 25`
- `gp_analyze_relative_error = .25`
- On specific table columns, run:

  ```
  ALTER TABLE name ALTER column SET STATISTICS #;
  ```

Greenplum Data Distribution

When working with data:

- Consider your table distribution key
- Check for data skew and avoid, if possible, unbalanced data
- Rebalancing a table if necessary

Greenplum Data Distribution – Consider the Table Distribution Key

When deciding on the table distribution key, look for:

- Even data distribution, where:
  - All segments should contain equal portions of data
  - The distribution key is unique for each record
- Local over distributed operations, where:
  - It is faster if the work can be performed at the segment level
  - A common distribution key improves joining or sorting
  - Local operations can be 5 times faster than distributed operations
- Even query processing, where:
  - All segments handle an equal amount of the query workload
  - Distribution policy and query predicates are well matched

This slide is intentionally left blank.

Greenplum Data Distribution – Check for Data Skew

Check for data skew using:

- `gp_toolkit.gp_skew_coefficients`
- `gp_toolkit.gp_skew_idle_fractions`
- System tools using `gpssh` to run them on multiple systems:
  - `top`
  - `iostat`

Greenplum Data Distribution – Rebalancing a Table

Rebalancing a table can be performed with the following:

- Change the distribution policy to a different column and redistribute the table and child tables:

```
ALTER TABLE sales SET DISTRIBUTED BY (customer_id);
```

- Redistribute table data to correct data skew:

```
ALTER TABLE sales SET WITH (REORGANIZE=TRUE);
```

SQL Formulation – General Considerations

When creating your queries:

- Know your data
- Minimize returned rows
- Avoid unnecessary columns in the result set
- Avoid unnecessary tables
- Avoid sorts of large result sets
- Match data types in predicates

SQL Formulation – Greenplum Specific Considerations

Greenplum-specific guidelines for creating queries include:

- Use common distribution keys:
  - For joins and aggregations
  - So most of the work is performed at the segment level
- Consider the table data distribution policy and query predicates:
  - To have segments handle an equal amount of work
  - To provide the best possible performance

Database Design

When considering the database design:

- Select appropriate data types
- Use a denormalized model
- Consider table partitioning
- Reconsider the use of indexes

Database Design – Selecting Appropriate Data Types

When selecting data types, choose a data type:

- That uses the least possible space
- That best constrains the data:
  - Use character data types for strings
  - Use date or timestamp data types for dates
  - Use numeric data types for numbers
  - Use `TEXT` or `VARCHAR` for character data
- Use identical data types for columns used in cross-table joins

This slide is intentionally left blank.

Database Design – Denormalization

Normalization:

- Is the process of eliminating redundancy and improving data organization
- Is used by online transaction processing (OLTP) databases

Denormalization:

- Is used by online analytical processing (OLAP) databases
- Translates into redundant data
- May facilitate ease of use and performance
- Is used by the star schema, where:
  - Data is stored in a central fact table
  - Dimension tables are denormalized
  - Complexity of queries is reduced
  - ETL processing may be required

Database Design – Table Partitioning

Table partitioning:

- Addresses the problem of supporting very large tables
- Divides large tables into smaller, manageable pieces
- Can improve query performance
- Lets the query planner scan only relevant data
- Should be used to help selectively scan data based on query predicates

Database Design – Indexes

If you are considering indexes, use the following guidelines:

- Use sparingly in Greenplum Database
- Test the query workload without indexes
- Ensure any indexes added are used by the query workload
- Verify that indexes improve query performance
- Indexes can improve performance of OLTP type workloads

Database Design – Index Considerations

When incorporating indexes, use the following guidelines:

- Avoid using indexes on frequently updated columns
- Avoid overlapping indexes
- Use bitmap indexes where applicable instead of B-tree
- Drop indexes for loads
- Consider a clustered index
- Configuring index usage with the following:
  `enable_indexscan = on | off`
- Compressed append-optimized tables may benefit from indexes
- If indexing partitioned tables, index columns should not be the same as partition columns

This slide is intentionally left blank.

Tracking Performance Issues

Performance management steps taken:

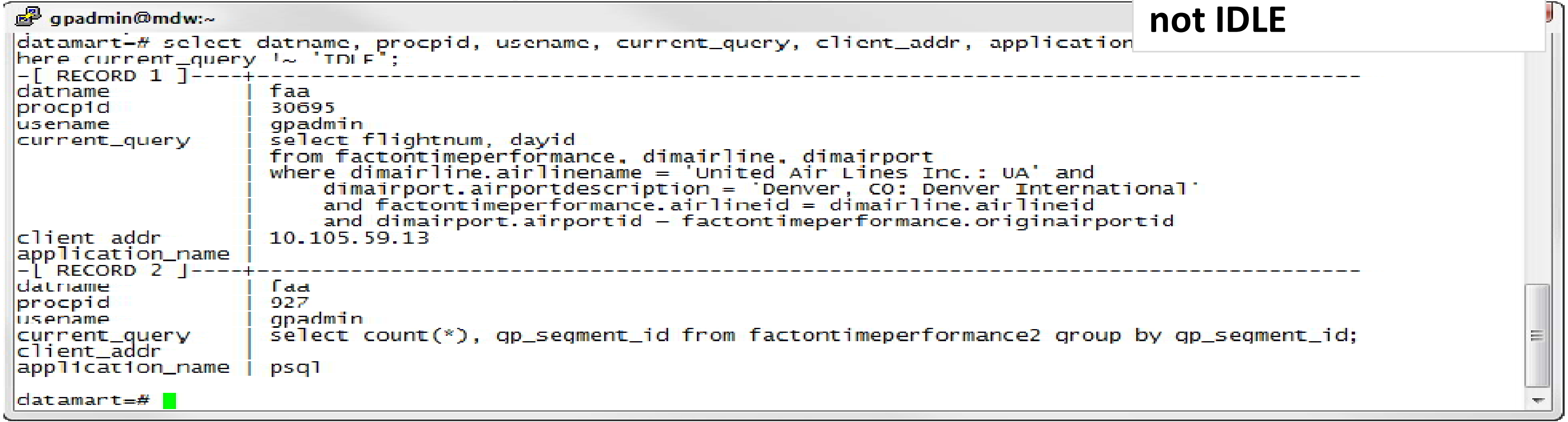
- Are often reactive
- Can focus efforts on tuning specific workloads
- Can be caused by:
  - Hardware problems
  - System failures
  - Resource contention
- Can be tracked with:
  - `pg_stat_activity`
  - `pg_locks` or `pg_class`
  - Database logs
  - UNIX system utilities

Tracking Performance Issues – `pg_stat_activity` System Catalog View

The `pg_stat_activity` view:

- Is a system catalog view
- Shows one row per server process

All processes that are not IDLE



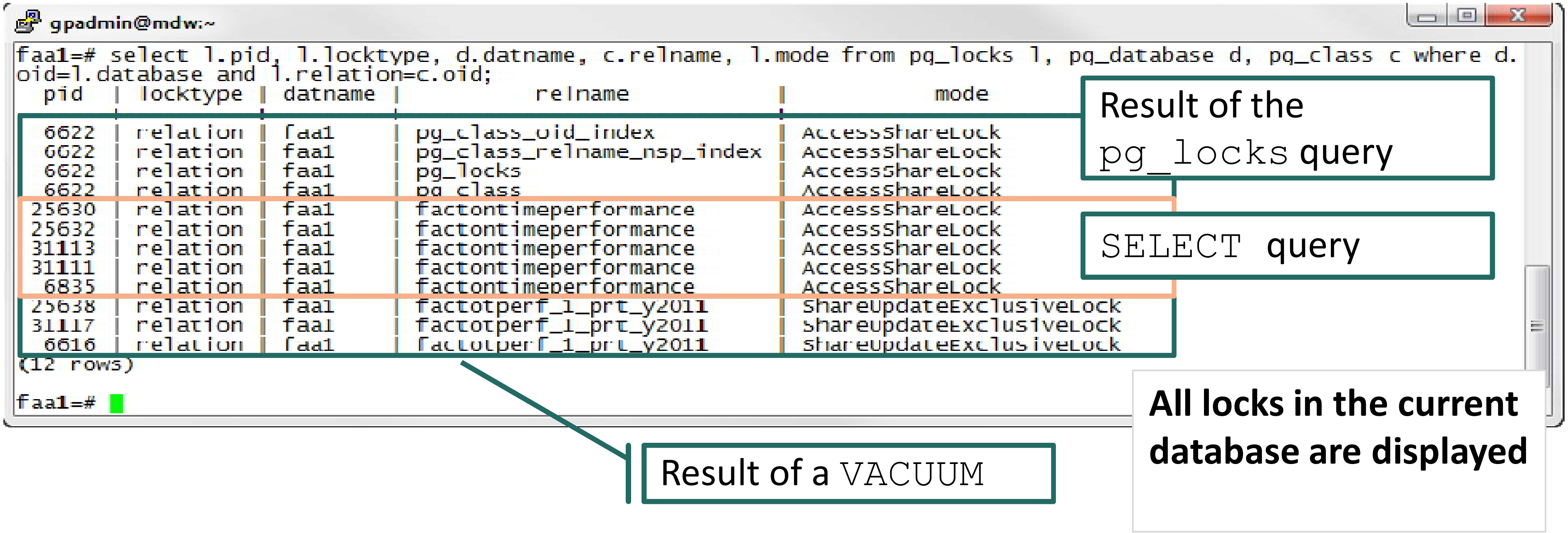
```
datamart=# select datname, procpid, usename, current_query, client_addr, application
here current_query !~ 'IDLE';
-[ RECORD 1 ]----+---------------------------------------------------------------------------------
datname          | faa
procpid          | 30695
usename          | gpadmin
current_query    | select flightnum, dayid
                 | from factontimeperformance, dimairline, dimairport
                 | where dimairline.airlinename = 'United Air Lines Inc.: UA' and
                 |     dimairport.airportdescription = 'Denver, CO: Denver International'
                 |     and factontimeperformance.airlineid = dimairline.airlineid
                 |     and dimairport.airportid = factontimeperformance.originairportid
client_addr      | 10.105.59.13
application_name |
-[ RECORD 2 ]----+---------------------------------------------------------------------------------
datname          | faa
procpid          | 927
usename          | gpadmin
current_query    | select count(*), gp_segment_id from factontimeperformance2 group by gp_segment_id;
client_addr      |
application_name | psql

datamart=#
```

Tracking Performance Issues – `pg_locks` System Activity View

The `pg_locks` view:

- Is a system catalog view
- Lets you view information on outstanding locks
- Can help identify contention between sessions
- Provides a global view of all locks in the database system
- Can be joined to `pg_class.oid` for relations in the current database
- Can have the `pid` column joined to `pg_stat_activity.procid` for more session information

Tracking Performance Issues – `pg_locks` System Activity View (Cont)



```
faa1=# select l.pid, l.locktype, d.datname, c.relname, l.mode from pg_locks l, pg_database d, pg_class c where d.
oid=l.database and l.relation=c.oid;
  pid  | locktype | datname |           relname            |            mode
-------+----------+---------+------------------------------+----------------------------
  6622 | relation | faa1    | pg_class_oid_index           | AccessShareLock
  6622 | relation | faa1    | pg_class_relname_nsp_index   | AccessShareLock
  6622 | relation | faa1    | pg_locks                     | AccessShareLock
  6622 | relation | faa1    | pg_class                     | AccessShareLock
 25630 | relation | faa1    | factontimeperformance        | AccessShareLock
 25632 | relation | faa1    | factontimeperformance        | AccessShareLock
 31113 | relation | faa1    | factontimeperformance        | AccessShareLock
 31111 | relation | faa1    | factontimeperformance        | AccessShareLock
  6835 | relation | faa1    | factontimeperformance        | AccessShareLock
 25638 | relation | faa1    | factotperf_1_prt_y2011       | ShareUpdateExclusiveLock
 31117 | relation | faa1    | factotperf_1_prt_y2011       | ShareUpdateExclusiveLock
  6616 | relation | faa1    | factotperf_1_prt_y2011       | ShareUpdateExclusiveLock
(12 rows)

faa1=#
```

Result of the `pg_locks` query

`SELECT` query

Result of a `VACUUM`

All locks in the current database are displayed

Tracking Performance Issues – Greenplum Database Log Files

Log files:

- Can be found for the master and segments
- Is located in the data directory location of the instance
- Can be accessed with:
  - `gpstate -l` to get the location of log files
  - `gpstate -e` to list the last lines of the log files

Tracking Performance Issues – UNIX System Utilities

System monitoring tools:

- Include:
  - `ps`
  - `top`
  - `iostat`
  - `vmstat`
  - `netstat`
- Help to:
  - Identify processes running on the system
  - Identify the most resource intensive tasks
- Can help identify queries overloading system resources
- Can be run on several hosts at once using `gpssh`

Wrapping Up

In this module we covered:

- Key steps in approaching performance tuning
- Common factors that can affect performance
- Best practices, commands and tools to help tune the Greenplum Database system

Pivotal

A NEW PLATFORM FOR A NEW ERA