Use CCM and QPM to optimize recovery performance and execution plans in Amazon Aurora PostgreSQL

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Use CCM and QPM to optimize recovery performance and execution plans in Amazon Aurora PostgreSQL

Table of Contents

Use CCM and QPM to optimize recovery performance and execution plans in Amazon Aurora PostgreSQL

Raunak Rishabh, Rohit Kapoor, and Sujitha Sasikumaran, Amazon Web Services

As businesses expand, they use more and more data to make critical decisions. With increasing amounts of data, it is important to optimize database performance and stabilize it during system changes. Highly transactional workloads, such as those involving financial transactions or customer orders, require stable, consistent, and fast performance because poor performance can affect customer satisfaction and business revenue. For databases that handle these highly transactional workloads, such as Amazon Aurora PostgreSQL-Compatible Edition database instances, it is critical that you understand and implement the available performance-optimization features.

Amazon Aurora PostgreSQL-Compatible is a fully managed relational database engine that helps you set up, operate, and scale PostgreSQL deployments. It is a widely used database engine because of its self-sustaining storage architecture and its features, which help you optimize performance on real-life workload scenarios with minimal maintenance overhead.

Two of these features are cluster cache management (CCM) and query plan management (QPM). CCM helps recover application and database performance in the event of a failover, and QPM helps you manage the query execution plans generated by the optimizer for your SQL applications. Both of these features can help optimize the performance of SQL queries by providing more control over the database. This guide is intended to help managers, product owners, and database architects (DBAs) understand the benefits and potential business outcomes of implementing CCM and QPM.

Intended audience

The intended audience for this guide is business stakeholders who want to understand the available features for optimizing the performance of Amazon Aurora PostgreSQL-Compatible database instances and understand the use cases for those features.

Targeted business outcomes

You can use this guide to achieve the following business outcomes with cluster cache management (CCM):

- In the event of a failover, recover quickly to help maintain stable and optimal workload performance.
- Reduce business losses caused by poor workload performance after a failover.
- Help prevent unnecessary I/O costs after a failover.

You can use this guide to achieve the following business outcomes with query plan management (QPM):

- Improve plan stability by forcing the optimizer to choose from a small number of approved plans. This prevents the optimizer from selecting a sub-optimal plan for a given SQL statement after system or database changes.
- Optimize plans centrally and then distribute them globally.
- Determine which indexes aren't being used, and evaluate the effects of adding or removing an index.
- Automatically recognize any new minimum-cost plans that the optimizer identifies.
- Test new optimizer features with minimal risk because you can decide to accept only the plan changes that improve performance.

Cluster cache management

Caching is one of the most important features of any database (DB) because it helps reduce the disk I/O. The most frequently accessed data is stored in a memory area called the *buffer cache*. When a query runs frequently, it retrieves the data directly from the cache instead of the disk. This is faster and provides better scalability and application performance. You configure PostgreSQL cache size by using the `shared_buffers` parameter. For more information, see Memory (PostgreSQL documentation).

After a failover, cluster cache management (CCM) in Amazon Aurora PostgreSQL-Compatible Edition is designed to improve application and database recovery performance. In a typical failover situation without CCM, you might see a temporary but significant performance degradation. This occurs because when the failover DB instance starts, the buffer cache is empty. An empty cache is also known as a *cold cache*. The DB instance must read from the disk, which is slower than reading from the cache.

When you implement CCM, you choose a preferred *reader* DB instance, and CCM continuously synchronizes its cache memory with that of the primary, or *writer*, DB instance. If a failover occurs, the preferred reader DB instance is promoted to the new writer DB instance. Because it already has a cache memory, known as a *warm cache*, this minimizes the impact of the failover on the application performance.

How does cluster cache management work?

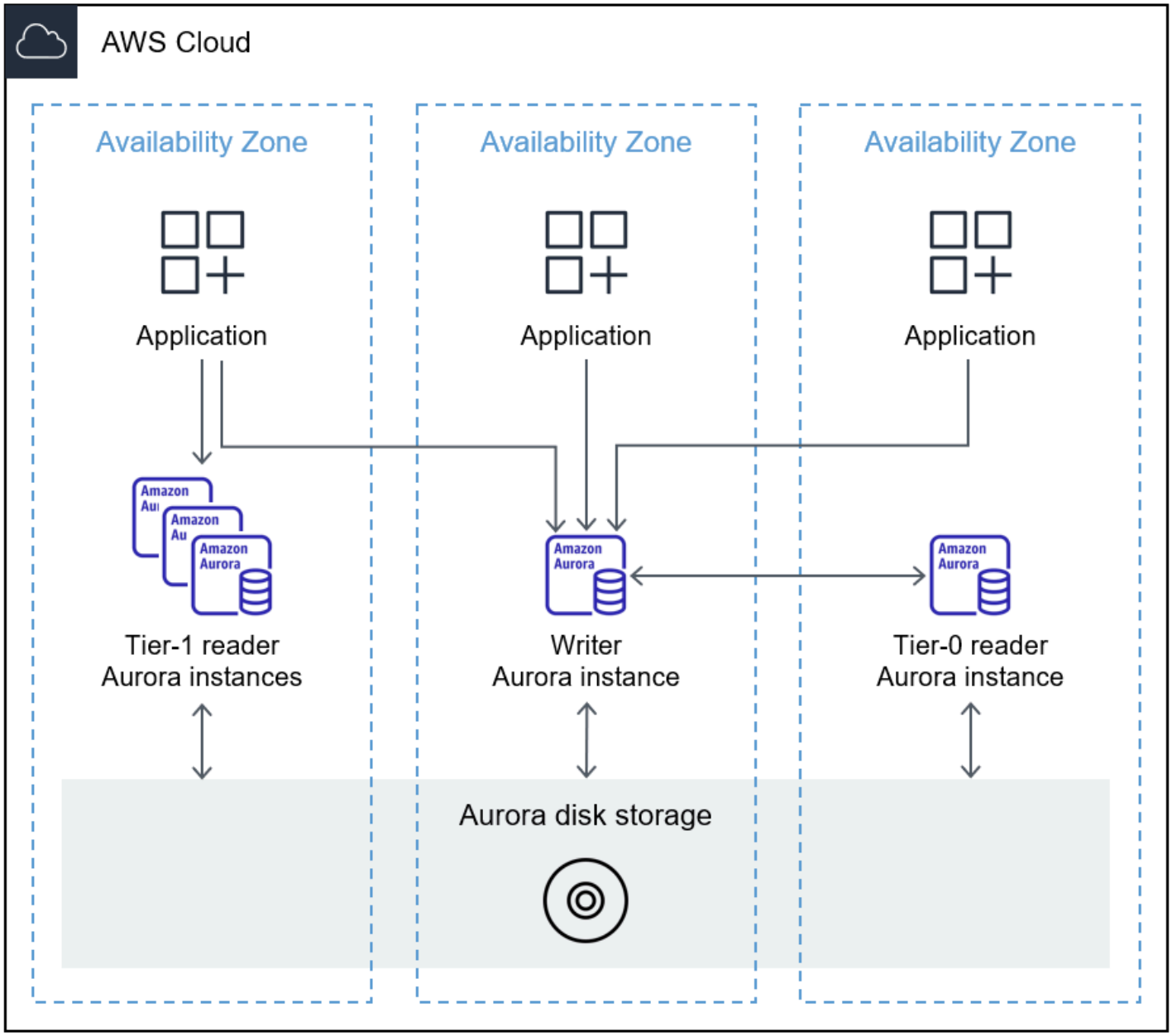
Failover DB instances are located in different availability zones from the primary, writer DB instance. The preferred reader DB instance is the priority failover target, which is specified by assigning it the **tier-0** priority level.

> Note: The promotion tier priority is a value that specifies the order in which an Aurora reader is promoted to the writer DB instance after a failure. Valid values are 0–15, where 0 is the first priority and 15 is the last priority. For more information about the promotion tier, see Fault tolerance for an Aurora DB cluster. Modifying the promotion tier doesn't cause an outage.

CCM synchronizes the cache from the writer DB instance to the preferred reader DB instance. The reader DB instance sends the set of buffer addresses that are currently cached to the writer DB

instance as a bloom filter. A *bloom filter* is a probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set. Using a bloom filter prevents the reader DB instance from sending the same buffer addresses to the writer DB instance repeatedly. When the writer DB instance receives the bloom filter, it compares the blocks in its buffer cache and sends frequently used buffers to the reader DB instance. By default, a buffer is considered frequently used if it has a usage count greater than three.

The following diagram shows how CCM synchronizes the buffer cache of the writer DB instance with the preferred reader DB instance.



For more information about CCM, see Fast recovery after failover with cluster cache management for Aurora PostgreSQL (Aurora documentation) and Introduction to Aurora PostgreSQL cluster cache management (AWS blog post). For instructions about how to configure CCM, see Configuring cluster cache management (Aurora documentation).

Limitations

The CCM feature has the following limitations:

- The reader DB instance must have the same DB instance class type and size as the writer DB instance, such as `r5.2xlarge` or `db.r5.xlarge`.
- CCM is not supported for Aurora PostgreSQL DB clusters that are part of Aurora global databases.

Use cases for CCM

For some industries, such as retail, banking, and finance, delays of only a few milliseconds can cause application performance issues and result in a significant loss of business. Because CCM helps recover application and database performance by continuously synchronizing the buffer cache of the primary database instance to the preferred backup instance, it can help prevent businesses losses associated with failovers.

Query plan management

Changes to statistics, constraints, environment settings, query parameter bindings, and upgrades to the PostgreSQL database engine can all cause query plan regression. *Query plan regression* is when the optimizer chooses a less optimal plan than it did before a given change to the database environment.

In Amazon Aurora PostgreSQL-Compatible Edition, the query plan management (QPM) feature is designed to ensure plan adaptability and stability, regardless of database environment changes that might cause query plan regression. QPM provides some control over the optimizer. Using QPM, you can manage the query execution plan generated by the optimizer for your SQL queries. The query execution plan forces the optimizer to choose from your approved plans for critical queries, to optimize their performance.

Enterprises commonly deploy applications and databases globally or maintain several environments for each application database, such as development, QA, staging, preproduction, testing, and production. Maintaining the query execution plans for each database, in each environment, and across all AWS Regions can be complex and time-consuming. QPM can export and import Amazon Aurora PostgreSQL-Compatible managed plans from one database to another. This helps you manage the query execution plan centrally and deploy databases globally. You can use this feature to investigate a set of plans in a preproduction database, verify that they perform well, and then load them into production environment.

QPM also provides several other benefits. For example, you can use QPM to improve execution plans that can't be changed in applications or when hints can't be added to the statement. QPM also automatically detects new, minimum-cost plans that the optimizer discovers, so you can continue to optimize costs in addition to performance.

We recommend that you enable QPM. When QPM is enabled, the optimizer uses the minimum-cost plan that you have approved. This helps prevent regression and reduces the time required to manage and fix suboptimal plans.

There are two different approaches for using the QPM feature: proactive and reactive. The proactive approach is designed to help prevent performance regression from occurring, and the reactive approach is designed to detect and repair performance regressions after they occur. You can select your approach on a per-query basis. For complex queries that might be prone to regression or for business-critical queries, you can use a proactive approach and approve the

optimal plans for those queries. If other queries experience query plan regression during runtime, you can use a reactive approach. When you detect the regression, change the status of that plan to `rejected` so that the optimizer chooses a different, approved plan. For more information, see Best practices for Aurora PostgreSQL query plan management (Aurora documentation).

How does query plan management work?

Plans are assigned one of the following statuses: `approved`, `unapproved`, `preferred`, or `rejected`. The optimizer sets the first generated plan for each managed statement to `approved` and then sets the status of additional plans to `unapproved`. Later, you can assess the `unapproved` plans and change their status to `approved`, `preferred`, or `rejected`. For more information, see Understanding Aurora PostgreSQL query plan management (Aurora documentation).

Managed plans can be captured either manually or automatically. The most common approach is to automatically capture plans for all statements that run two or more times. However, you can also manually capture plans for a specific set of statements. For more information, see Capturing Aurora PostgreSQL execution plans (Aurora documentation).

After you have set up a managed plan, the optimizer uses the minimum-cost `preferred` or `approved` plan that is valid and enabled for each managed statement. For detailed information, see How the optimizer chooses which plan to run (Aurora documentation).

For instructions about configure the QPM feature in Amazon Aurora PostgreSQL-Compatible, see Managing query execution plans for Aurora PostgreSQL (Aurora documentation).

Limitations

To use QPM, you must make sure that you meet the requirements for supported SQL statements, your statements don't reference system relations, and your DB instance class has sufficient vCPUs. For more information, see Supported SQL statements and Query plan management limitations (Aurora documentation).

Use cases for QPM

- **Preventing query plan regression** – Keeping your database version up to date provides many benefits, such as improved performance and security, access to new features, fixes to known issues, and compliance with regulatory requirements. However, there is a risk that database

updates can cause some queries to experience performance regression. This risk is higher with major version upgrades because they can contain changes that may not be backward-compatible with existing application queries. Implementing QPM can help prevent regression and stabilize performance during system changes. If you refresh statistics, add an index, change parameters, or upgrade to a new version of Amazon Aurora PostgreSQL-Compatible, QPM detects a new plan but continues to use the approved plan, thus maintaining plan stability.

- **Testing features** – You can view the plan history for all managed SQL statements and assess whether new PostgreSQL features or plan changes are improving performance. You can then decide whether to implement those features or new plans. For more information, see Examining Aurora PostgreSQL query plans in the dba_plans view (Aurora documentation).
- **Improving a plan** – In some cases, you might prefer to fix a suboptimal plan rather than reject, disable, or delete it. For more information, see Fixing plans using pg_hint_plan (Aurora documentation).

Resources

AWS documentation

- Cluster cache management (CCM)
- Query plan management (QPM)

AWS blog posts

- Introduction to Aurora PostgreSQL CCM
- Introduction to Aurora PostgreSQL QPM

AWS workshops

- Amazon Aurora Labs for PostgreSQL: CCM
- Amazon Aurora Labs for PostgreSQL: QPM

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Initial publication	—	January 20, 2023