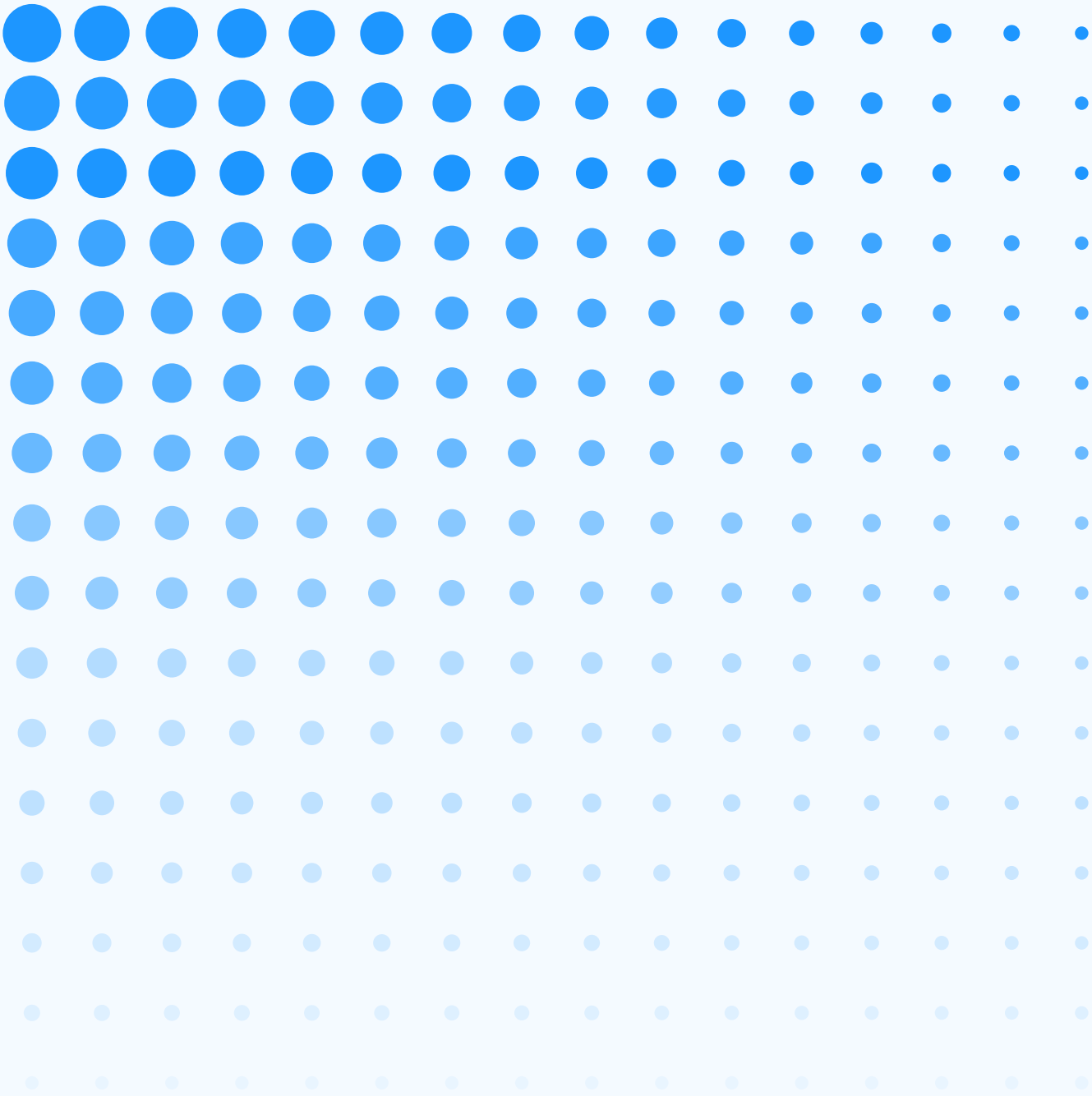
4 Simple Steps To Set-up Your WLM in Amazon Redshift For Better Workload Scalability



integrate.io

Table Of Contents

01

INTRODUCTION

One of the major propositions of Amazon Redshift is simplicity. It only takes minutes to spin up a cluster.

The time-to-first-report, i.e. the time it takes to go from creating a cluster to seeing the results of their first query, can be less than 15 minutes. That's true even for petabyte-scale workloads.

Because it's so easy to set-up a cluster, it can also be easy to overlook a few housekeeping items when it comes to the set-up. That can cause problems with scaling workloads down the road. A general complain we often hear is "slow queries", or "slow dashboards".

A key configuration to use is the Amazon Redshift Workload Management (WLM). Without using WLM, each query gets equal priority. The result is that some workloads may end up using excessive cluster resources and block business-critical processes.

1. Loading data takes too long. Even with efficient copy operations from S3, it takes too long to import data at scale.
2. Queries overflow to disk and consume the entire SSD. Trying to avoid inefficient queries can seem impossible.
3. Huge strain and contention on a cluster when data loading and querying take place at the same time.

You can address these challenges with our top 14 performance tuning techniques for Amazon Redshift. However, odds are you'll be able to get some quick performance gains by adjusting your WLM.

And so in this post, we'll recommend a few simple best practices that will help you configure your WLM the right way and avoid these problems. Using workload management the right way has a lot of benefits. It's the single best way to achieve concurrency scaling for Amazon Redshift. Your users will be happy (fast queries), you can scale as your data volume grows, and you'll spend less time fighting fires.

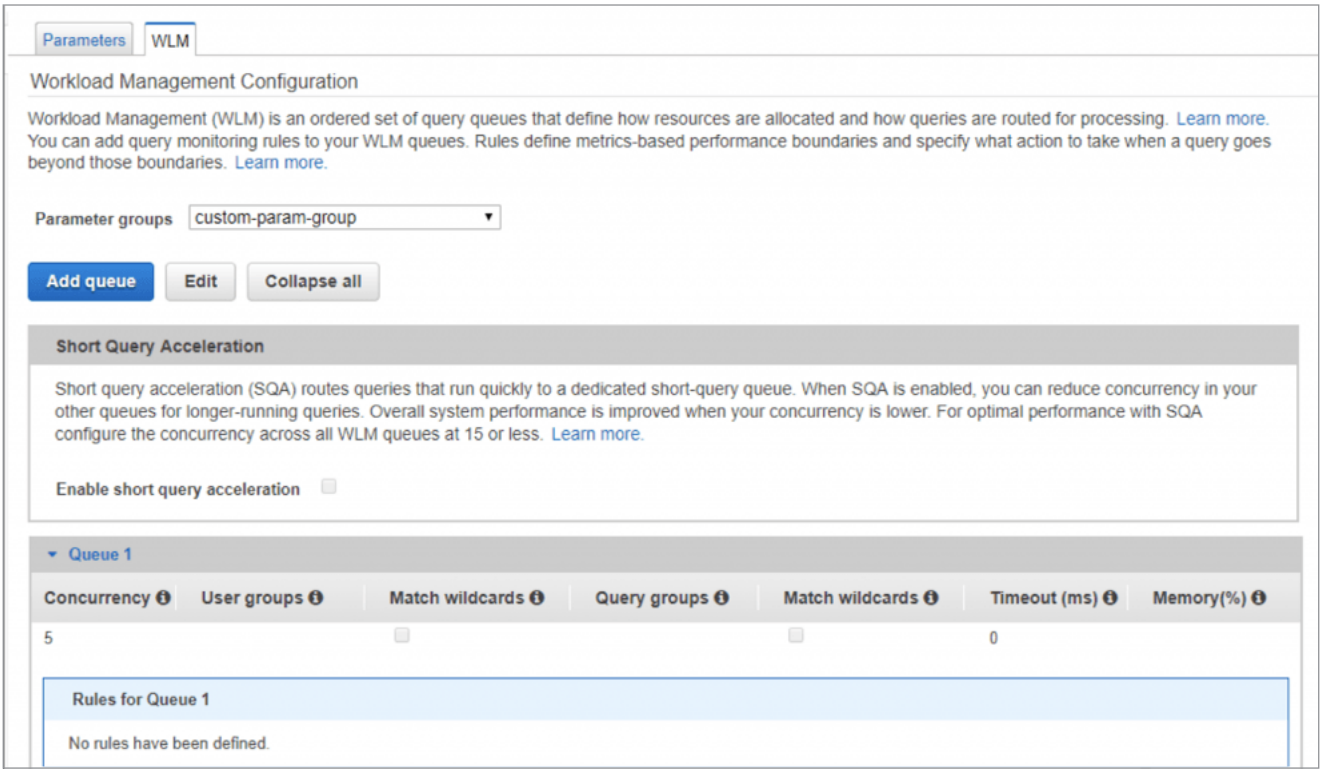


Figure 1: The WLM tab in the Amazon Redshift console

02

UNDERSTANDING AMAZON REDSHIFT WORKLOAD MANAGEMENT: QUEUES, CONCURRENCY AND MEMORY

Amazon Redshift operates in a queueing model.

First step is to define queues for your different workloads. Next you need to assign a specific concurrency / memory configuration for each queue.

Amazon Redshift allows defining up to 8 queues with a total of up to 50 slots. In the Amazon Redshift docs you'll read to not go above 15 slots. By using the techniques in this post though you'll be able to use all 50 available slots. With clear visibility when and how you need to fine-tune your settings.

The default configuration for Redshift is one queue with a concurrency of 5. If you run more than 5 concurrent queries, then your queries wait in the queue. That's when the "takes too long" goes into effect.

The available amount of memory is distributed evenly across each concurrency slot. Say that you have a total of 1GB, then with a default configuration, each of the 5 concurrency slot gets 200MB memory.

If you run a query that needs more than 200MB, then it falls back to disk. That means it takes longer to execute. Disk-based queries also consume a lot of I/O. That slows down the entire cluster, not just queries in a specific queue.

Users then try to scale their way out of contention by adding more nodes. That can become an expensive proposition. The performance increase is also non-linear as you add more nodes.

You can achieve a much better return on your Amazon Redshift investment by fine-tuning your WLM. You can fix slow and disk-based queries by configuring Redshift specific to your workloads. Because odds are the default WLM configuration of 5 slots will not work for you. That includes using the option of Short Query Acceleration.

03

FOUR STEPS TO SET UP YOUR WORKLOAD MANAGEMENT

When the user runs a query, WLM assigns the query to the first matching queue and executes rules based on the WLM configuration.

And so the key concept for using the WLM is to isolate your workload patterns from each other. You can then create independent queues, and each queue supports a different business process, e.g. data loads or dashboard queries. With separate queues, you can assign the right slot count and memory percentage.

And so let's look at the four steps in detail.

1 SET-UP INDIVIDUAL USERS

Invididual logins / user provide more control and better visibility

The first step is to create individual logins for each user. A user can be a person, an app or a process. Anything that can run a query.

Separating users may seem obvious, but a lot of times logins get shared. The problem then is that you can't tell who is driving which workloads. Sure, with a few users that may be possible. But as your organization grows, there will be a lot of guessing involved.

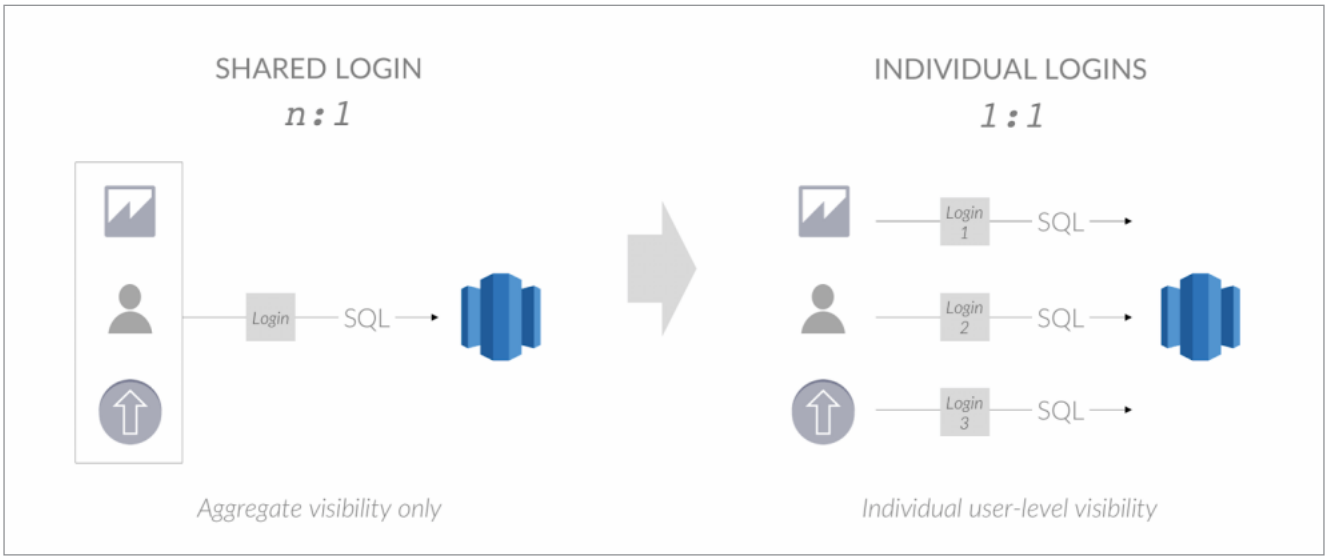


Figure 2: The WLM tab in the Amazon Redshift console

Also, do not use the default Redshift user for queries. For one, because it has admin privileges. But consider it as your lifeline when you run into serious contention issues – you will still be able to run queries with the default user.

If your cluster is already up and running with a few users, we recommend doing a reset. Delete the old users and assign everybody new logins.

2 DEFINE YOUR WORKLOADS

Define each login / user by their type of workload: load, transform, or ad-hoc queries

The next step is to categorize all user by their workload type. There are three generic types of workloads:

1. **Loads**: Jobs that load data into the cluster. These are COPY and UNLOAD statements
2. **Transforms**: Batch jobs and scheduled transformations. INSERT, UPDATE and DELETE transactions
3. **Ad-hoc**: These are queries by analysts / dashboards. SELECT statements

Figure 3: Define workload types

Defining users by workload type will allow to both group and separate them from each other. What you'll find is that workload of the same type share similar usage patterns.

3 GROUP USERS BY WORKLOAD TYPE

Create on user group per workload type

We can use the similarity in workload patterns to our advantage. By grouping them, we'll have groups of queries that tend to require similar cluster resources. For example, loads are often low memory and high frequency. Ad-hoc queries on the other hand run less frequent, but can be memory-intensive.

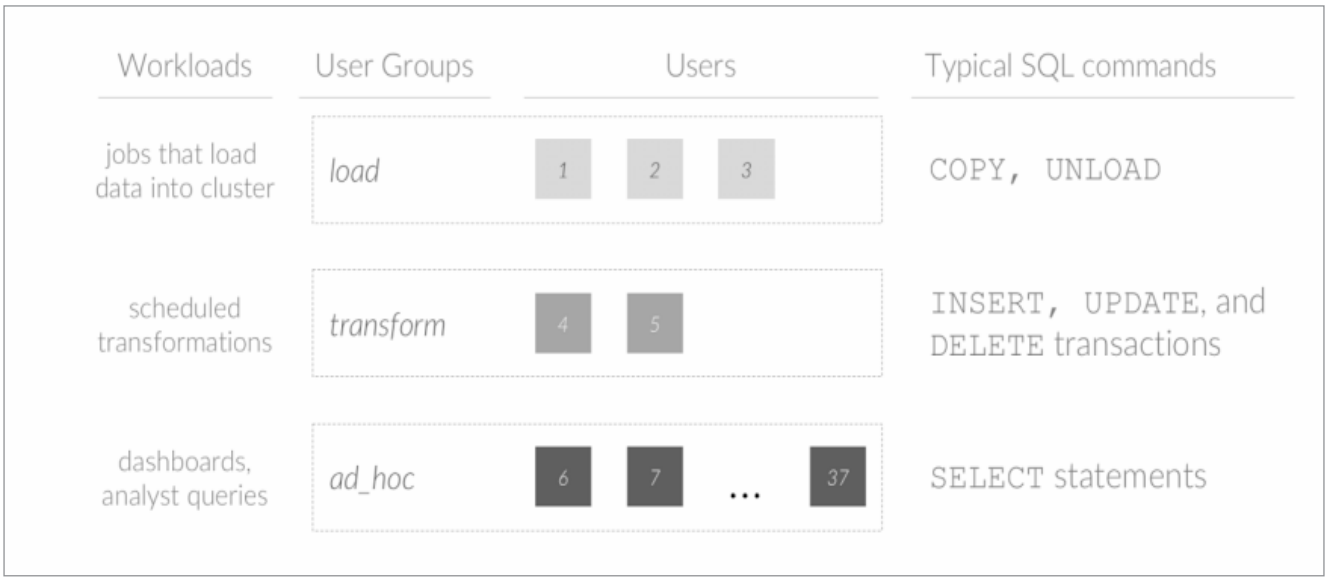


Figure 4: User groups in Amazon Redshift types

Use the CREATE GROUP command for creating the three groups 'load', 'transform' and 'ad_hoc'. As you can see, they match the workload types we defined for our users. Use ALTER GROUP to add the users we defined in step #2 to their corresponding group.

You can of course create more granular sub-groups, e.g. for sales, marketing or finance. That way you can give the users in each group the appropriate access to the data they require. But stay within the logic of workload patterns and don't mix different workload groups.

4 DEFINE SLOT COUNT & MEMORY PERCENTAGE

Create a new parameter group within the Redshift WLM console

In the final step, we determine what slot count we give each queue, and the memory we allocate to each slot.

We keep the default queue reserved for the default user, and set it to a concurrency of 1 with a memory percentage of 1%. The default queue is your insurance in case something goes wrong.

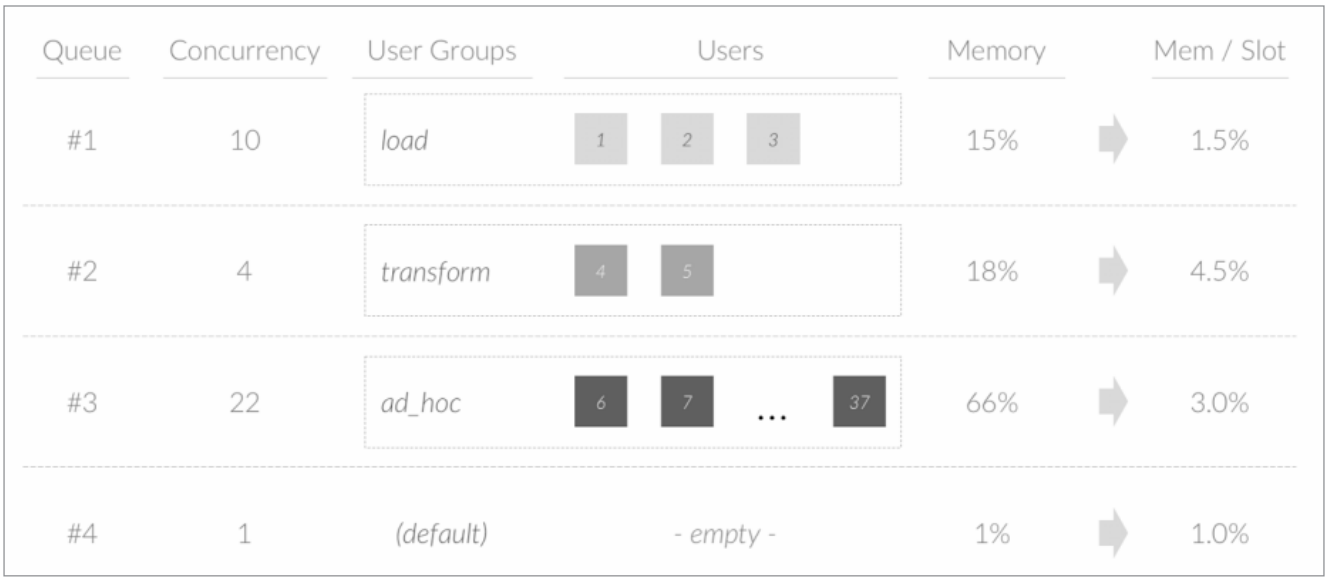


Figure 4: Defining WLM slot count and memory percentage in Amazon Redshift

For the other queues, slot count and memory will determine if each query has:

- a slot at run time
- enough memory to execute in-memory
-

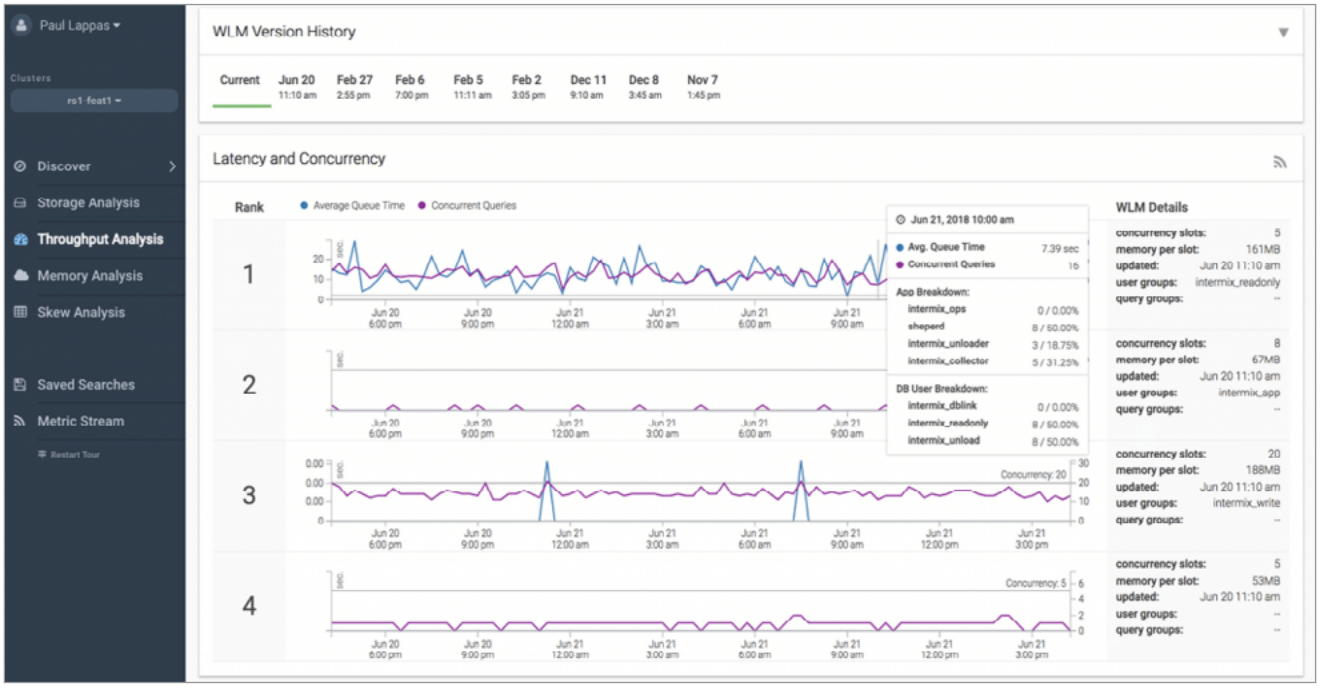
If both is true, that's when you get blazing fast queries and throughput. To apply the new settings, you need to create a new parameter group with the Redshift console. But finding the right slot count and memory is a little bit like trying to look into a black box. How to find the right slot count and memory percentage for your WLM queues.

AWS provides a repository of utilities and scripts. They involve querying the system tables (STL Tables and STV Tables). The scripts help you to find out e.g. what the concurrency high-water mark is in a queue. Or which queries fall back to disk.

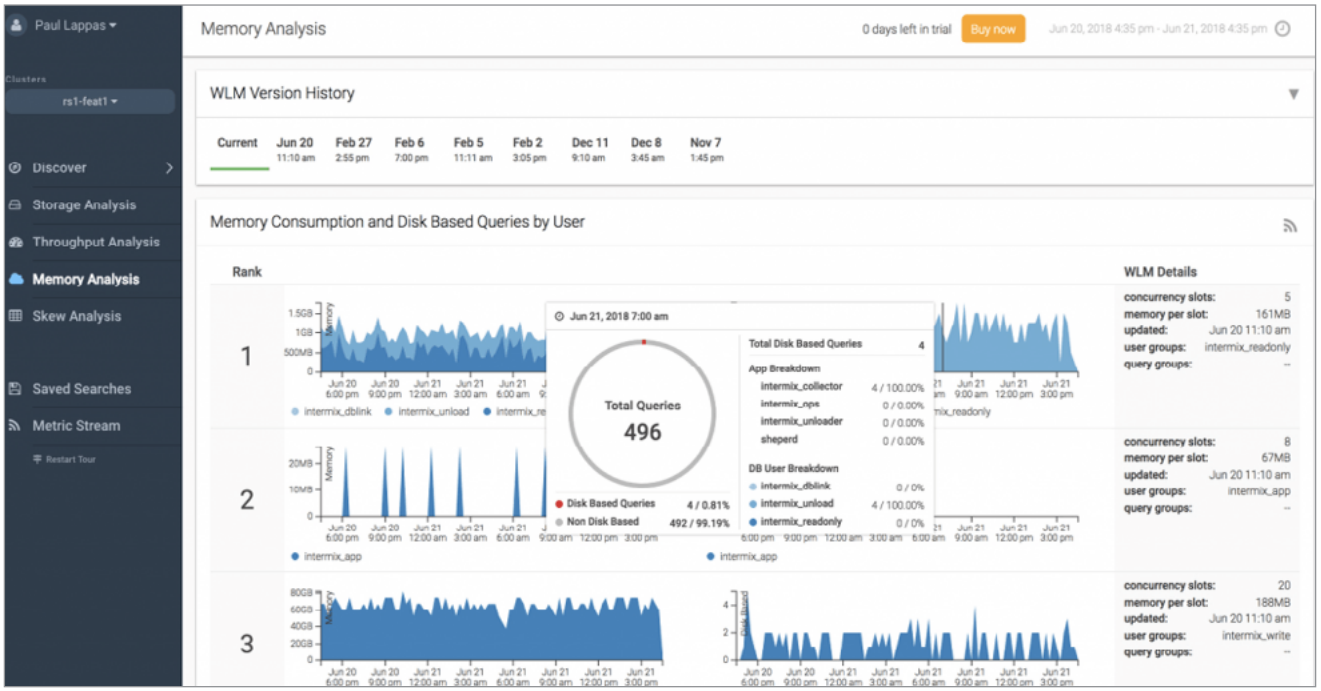
There are three potential challenges though with scripts:

1. Scripts can be incomplete: Some of the information is ephemeral as Redshift deletes logs on a rolling basis. If you don't run the script at the right time, the information is gone.
2. Scripts increase cluster load: Because you're querying the system tables, you're putting more load on the system. The exact opposite of what you want to do when you're experiencing contention.
3. Scripts require maintenance: Scripts need to run and store the results. It's almost like building another application. That's money. But also use of your most valuable resource, engineering hours.

With our Throughput and Memory Analysis, we make finding the right slot count and memory percentage easy. You can see the relevant metrics in an intuitive, time-serie dashboard.



Our Throughput Analysis shows you if your queues have the right slot count, or if queries are stuck in the queue. When queries get stuck, that's when your users are waiting for their data.



With our Memory Analysis then you can see the volume of disk-based queries. Some queries will always fall back to disk, due to their size or type. But we recommend keeping the share of disk-based queries below 10% of total query volume per queue.