Beyond Jobs: Kauffman Challenge Deliverables

1b The timezone issue in complex grid displays

Table of Contents

Background: migration to Amazon Web Services

Database security

In order to protect data we have ip whitelisted all the databases, meaning that only certain ips can access the databases. Also, by utilising AWS security groups, access to databases has been restricted to certain servers.

For example, the database for demo-uflexi can only be accessed by the demo-uflexi servers and is not accessible to test-uflexi or dev-uflexi servers. The security groups are easy to maintain and change if required.

Image security

Uflexi uses seller images and we wanted to ensure these were protected. Images are stored in an Amazon S3 Bucket and can only be accessed by a valid user logged in to Uflexi. Direct access to images has been prevented by applying AWS security groups to the S3 bucket.

Performance

Uflexi has many data rich reports and database performance is likely to be an issue. So we've used read replica databases for any read only functionality such as reports. We have replicas of all the databases created from snapshots and these are updated whenever there is a change to the source database. In this way we reduce load on the main database.

Read replicas are designed for security; Amazon RDS sets up a secure channel between source and replica and adds the security group entries required to maintain a secure channel of communication.
RDS Read Replica databases - https://aws.amazon.com/rds/details/read-replicas/

Database reliability

Backup snapshots

We take a snaphot of every database daily; these can be used for re-creating a database should the main database ever fall into a critical state.

Multi-AZ failover

Although not currently enabled (because it is so costly) , there is the option to enable Multi A-Z failover: https://aws.amazon.com/rds/details/multi-az/

A key feature is that a replica database is maintained in a different availability zone should access to a particular zone ever be lost. This provides similar functionality to that we already gain by utilising replica databases.

Caching

Ensuring the cached data was up-to-date when moving to Hibernate for quicker/easier storage/access to data caused issues. Due to hibernate only being used in specific areas and jdbc access in other areas, retrieving the data by two separate methods caused stale data. Setting the annotation (for our internal NEMsCacheGroup) in both files accessing the data, solved the issue. This was for access to any new Java class using Hibernate.

Timezone handling

Throughout most software systems, dates and times will be used and stored. The Postgres database stores in UTC (Coordinated Universal Time) which is the base for all timezones. Moving to Pacific Time caused issues due to legacy code being written predominantly for the UK market.

In particular, the seller's 'Availability Grid' and the Agency/Buyer 'Make a Booking – Aggregated Availability Grid both need the correct timezone to ensure the data is displayed correctly. Due to legacy code however, updates to move from UK to Pacific Time meant that adjustments for time were hard-coded in the system, thus resulting in the application only working at optimum when the timezone for the server and the user's computer are in the same timezone (specifically Pacific Time). Note. allowances are made for daylight savings, so this is not an issue.

Moving to the Amazon AWS servers however caused a further issue, setting the server timezone could easily be achieved, but due to auto-scaling (where new application instances are created to avoid over-loading one system), the new instances were being created in the wrong timezone.

An eventual fix for this was to set the JVM options as part of the configuration to the appropriate timezone.