# My Review: Fixing the ID Resolution Pipeline

I took a close look at our ID Resolution pipeline recently because it’s been taking way too long to run – in some cases more than 20 hours, which just isn’t workable.  
Our goal is to get this down to under an hour so that we can move faster on customer updates, especially for things like Adobe AJO marketing.  
Here's a breakdown of what I found, what I changed, and a few ideas for what we could do next.

## 1. What I Noticed

Right away, I noticed that batches were being processed one after another, with no parallelism at all.  
That meant even if we had resources available, we weren’t using them.  
Also, the Spark job configs were pretty basic – no adaptive execution, no optimized partitioning, nothing tuned for performance.  
On top of that, the memory wasn’t being cleaned up between batches, so each one got slower than the last.

## 2. What I Fixed

To get things moving faster, I made a few solid changes:

• Added multi-batch support using ThreadPoolExecutor – now we can run up to 8 batches at once.

• Tuned the Spark configs: more shuffle partitions, adaptive query execution turned on, and Arrow for PySpark enabled.

• Cleaned up memory after every batch using a cached DataFrame remover we already had.

• Handled override paths properly so we don’t waste time when that shortcut is available.

• Put in a smart timeout to shut the job down after 22 hours if anything goes wrong.

• Improved the logs so now we can see runtimes, batch results, and failures all in one place.

## 3. What I Think We Can Do Next

The job is already a lot faster now, but I think we can push it even further. Here are a few next steps I’d recommend:

• Break batches into smaller chunks and run them using AWS Step Functions for better orchestration.

• Add CloudWatch or Datadog monitoring so we can keep an eye on batch durations and failures in real-time.

• Consider testing this as a Ray job in Glue 4.0 to see if we can get even better concurrency.

• Add a simple retry loop for failed batches instead of just logging them and stopping.

• Build a small dashboard (even just basic S3 logging or Athena queries) to keep track of batch health and job stats.

After these updates, the job is already running close to our 1-hour target depending on data volume.  
It’s faster, cleaner, and way easier to debug.  
Let’s keep pushing from here.