Design of Multi User Multiprocessing Fitting System

Paul Kienzle and Omer Eisenberg

Fitting job may take a few hours to complete. One way of shortening processing time is to use multi-processing system. The purpose of the described system it to enable multiple users to run bumps fitting algorithms simultaneously on a multiprocessors computer.

Basic Idea

Users will submit fitting job request from client machine to the system server.

A Queue Manager (QM), running on the server, first verifies correct request structures. If valid, it will add the new job to the end of a First In First Out (FIFO) queue. That queue will be implemented in a relational database.

A Multiprocessing Interface (MI), also running on the server, will send jobs, from the top of the queue, to a distributed tasks queue system, such as Slurm or Celery. When the job is completed, the MI will save the result in the database and delete them from the disk, remove the job from the Slurm/Celery and add the next job.

During the job run time, status updates are being sent to the user from the processing nodes, through the MI and the QM.

When the job is completed, the MI removes saves the job results to the database, removes them from the disk, remove the job from Slurm/Celery, and submits the next job from the queue. Also, the MI sends notification to the appropriate client.

The system components and interconnection are described in the following figure.



Notes:

Time is a combination of date and day time.

The database row, displayed at the bottom, may be more complex than one line.

Results will be saved as BLOBs.

basic requirements

The system will be self maintained. That is, it may function without admin intervention.

Users may install the system independently from NCNR.

The system will include Amazon Web Services (AWS) interface. That interface will not include financial aspect.

Scalability - the system will be able to operate on a single machine, meaning all the system's part will run on a host machine. At the same time, the system will be able to operate so that the client, server, database, and computing nodes each implemented on a different machine.

The system will not maintain users.

Technology

The user interface will be based on Brian Maranville's reflectivity calculator.

The database will support SQL. It will enable automatic backup and simple restore.

Client Server communication will be implemented with Socket.IO, which is WebSocket wrapper.

Computing engine will be bumps.

Software

Client

The client will be based on the reflectivity calculator. The following additions are required.

Generate automatic tag for job identification.

Note: User tags will be of the format *random english* *word*\_n. For example: *FMradio\_*17. The client will generate a default word, but the user will be able to overwrite that default.

Handle communication with the server.

Implement bumps-like charts

Save/Load results to/from disk file.

Slurm/Celery interface. Slurm interface let the user select the number of processors, GPUs, memory, etc.

Display several fitting terms and results

Send fitting job to server.

Receive and display fitting job results from the server.

Receive and display fitting job log update.

Queue Manager (QM)

Receive fitting job from clients.

Validate job structure. Reply with error description if not valid request.

Assign unique ID to new job. Note: Client IP address and user supplied tag will be used for job identification.

Save job to database.

Receive log update from MI.

Send log updates and results to the designated client.

Automatically delete old jobs from database.

Commands: Add (new job), Status (job), ListMyJobs(), Kill (job), GetUserTags().

Note: Commands List and GetUserTags will send the IP address to the server without user intervention.

Multiprocessing Interface (MI)

Read job from database, send it to be performed on distributed system: Slurm or Celery, update job status in the database.

Poll job log, and send updates to QM.

When job complete: update job status, end time in the database, send notification to QM, remove job from distributed system, read next job, as described above.

Database

The database will be implemented on an SQL based system.

All database related statements will be with standard SQL, so that migrating from one RDBMS to another will not require software update.

Schema (later)

Development

later

Use Cases

later