Multithreading and Multiprocessing

Zoltan Mathe

Monday 29th May, 2017

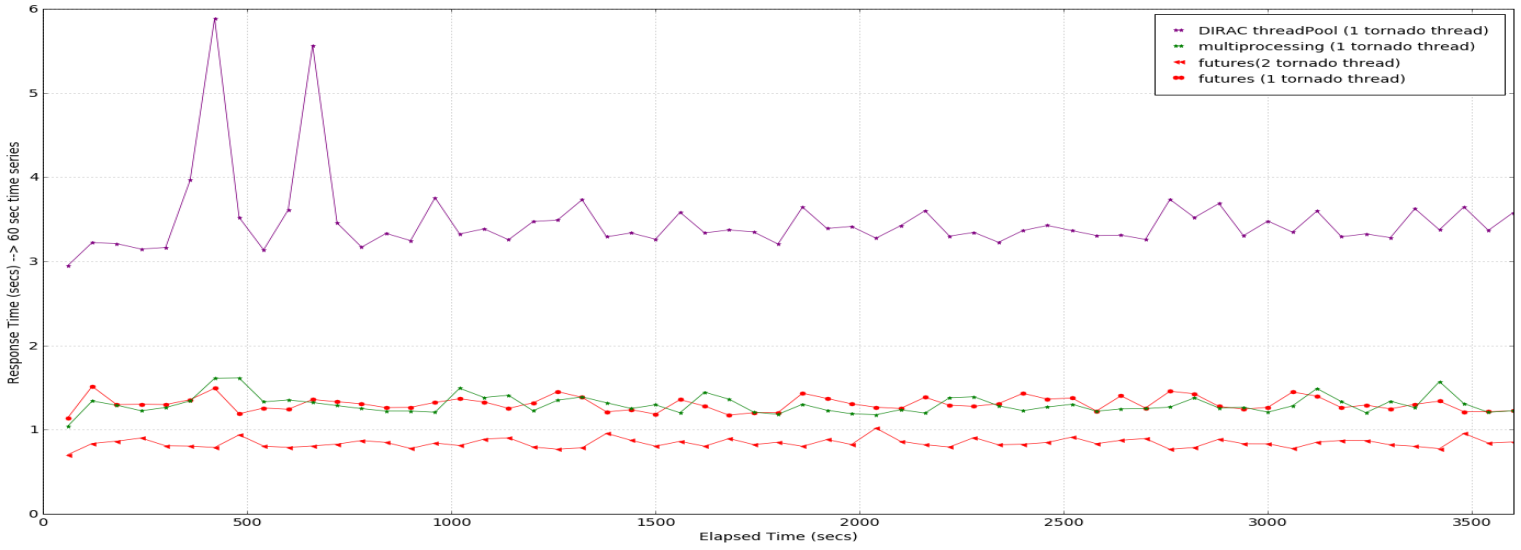
1. Multihreading and multiprocessing implementations
   - Multithreading
     - ThreadPools
     - The web server performance comparison
     - Performance tests: DIRAC ThreadPool, concurrent.futures and multiprocessing.ThreadPool
   - Multiprocessing
     - DIRAC ProcessPool
     - concurrent.future.ProcessPoolExecutor
     - multiprocessing ProcessPool
     - ProcessPool vs ProcessPoolExecutor
2. Enabling multiprocessing for DIRAC services

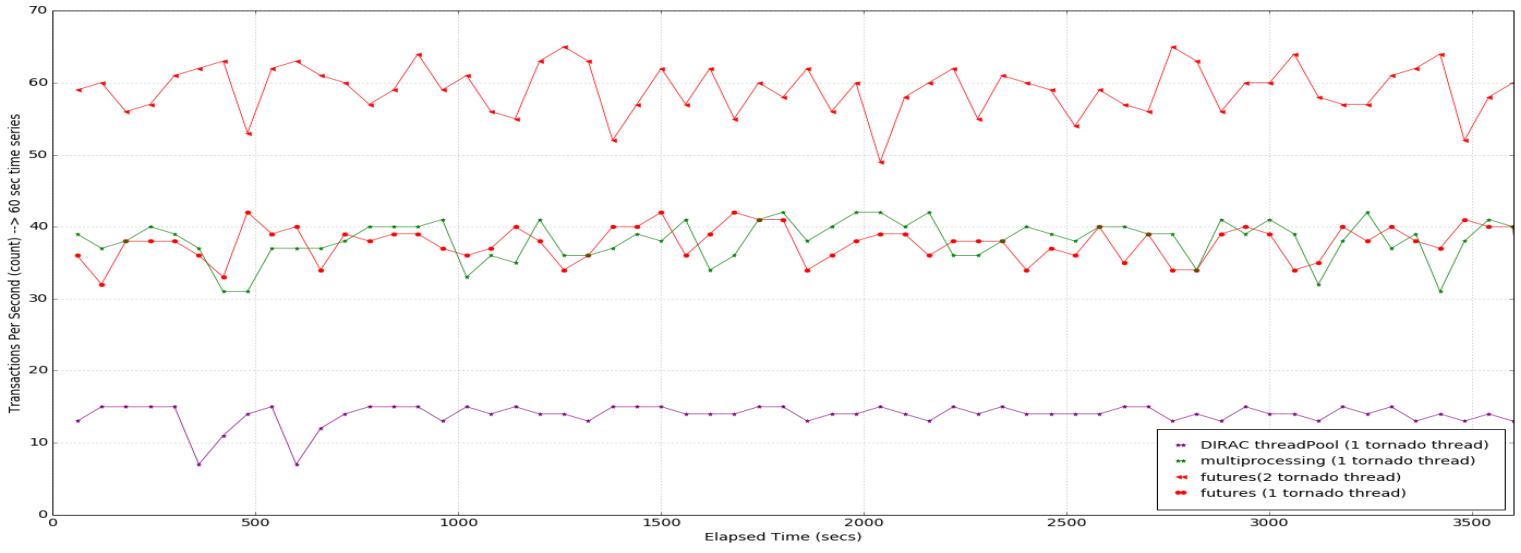
- The DIRAC ThreadPool implemented to process a queue of tasks in parallel.
- Advantages:
  - optimises the usage of the computing resources like CPU
- Disadvantages:
  - required maintenance
  - only DIRAC community is using it
- Usage:
  - all services are based on the DIRAC ThreadPool
  - most of the agents are using it

- Do we want to continue maintaining it?
- Will it work in python3?
- Are they other solutions?

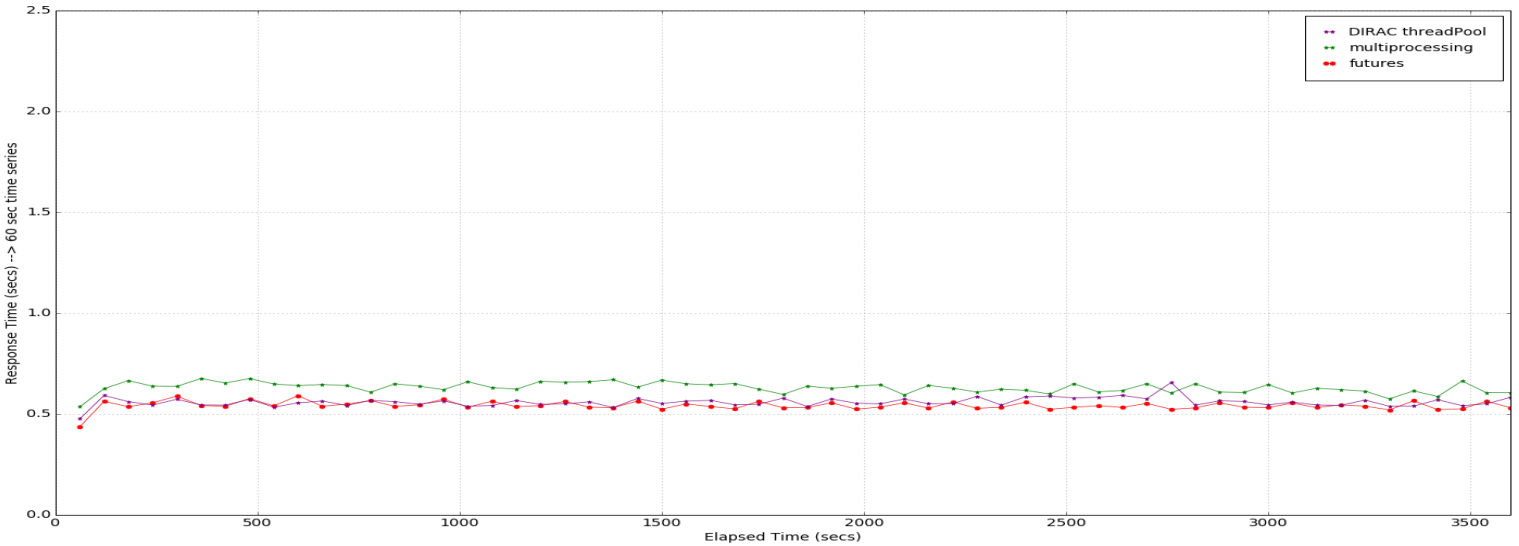
- multiprocessing ThreadPool:
  - provides the required functionalities
  - does not work with Tornado (DIRAC web framework)
- concurrent.futures.ThreadPoolExecutor:
  - it works with Tornado
  - provides what we need
  - it is not optimised for resource usage (we can hit the max threads limit of the OS)
  - it is already used by the web server

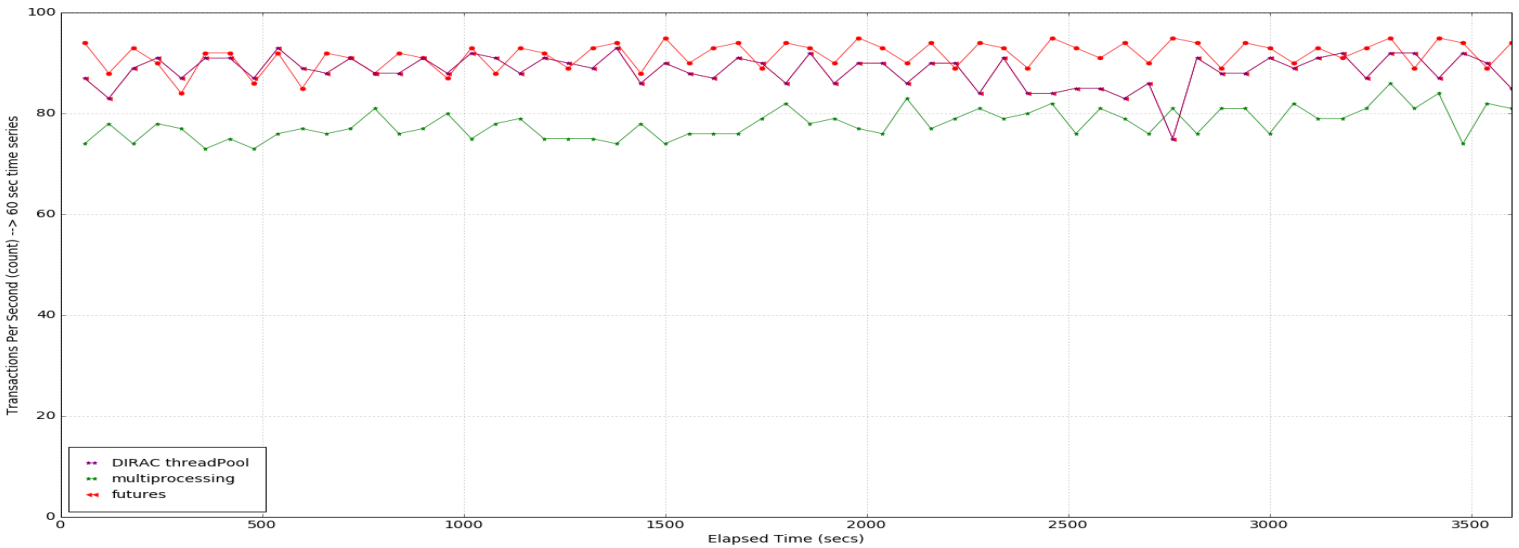
The web server performance comparison





Performance tests: DIRAC ThreadPool, concurrent.futures and multiprocessing.Pool





DIRAC ProcessPool

- It is based on multiprocessing package
- It works, but ...

concurrent.future

- it uses the multiprocessing module
- it provides the same interface than ThreadPoolExecutor
- when a task is submitted to the ProcessPoolExecutor then a Future is returned.
- Future encapsulates the asynchronous execution of a callable.

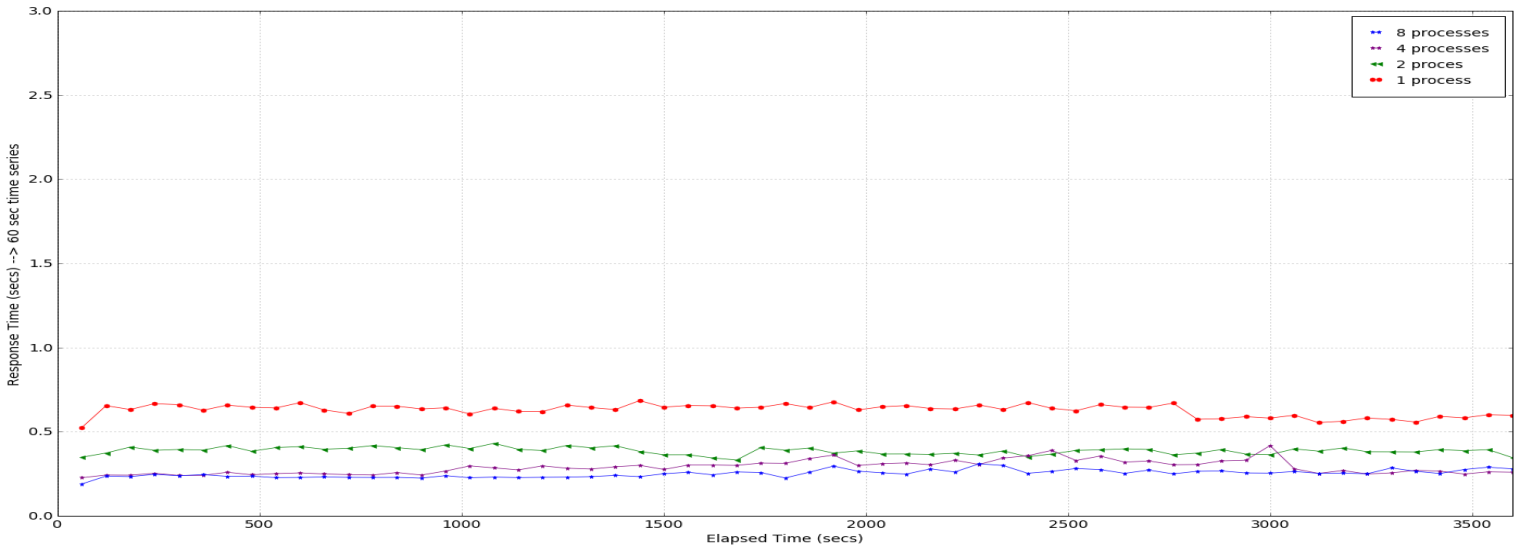
ProcessPool (multiprocessing)

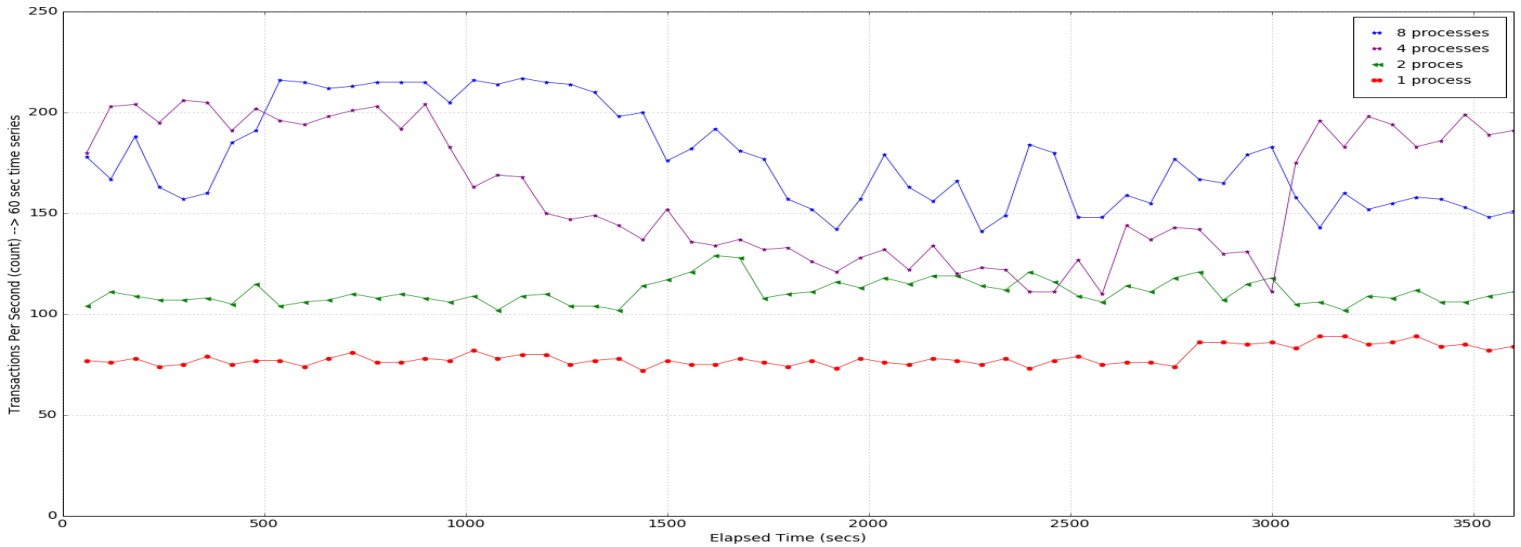
- it provides all functionalities what we need
- optimised for submitting big batches tasks
- the callback is added when the task is submitted

ProcessPool vs ProcessPoolExecutor

- ProcessPool for big batches of tasks
- ProcessPoolExecutor for long running tasks, it very well notify the user by callback, easily can check the status, and can cancel individually.

- the purpose is to run as many services as we can





Thank you! Questions?