# 多进程提取数据

由 danielnzhou(周亚楠)创建 大约4小时以前

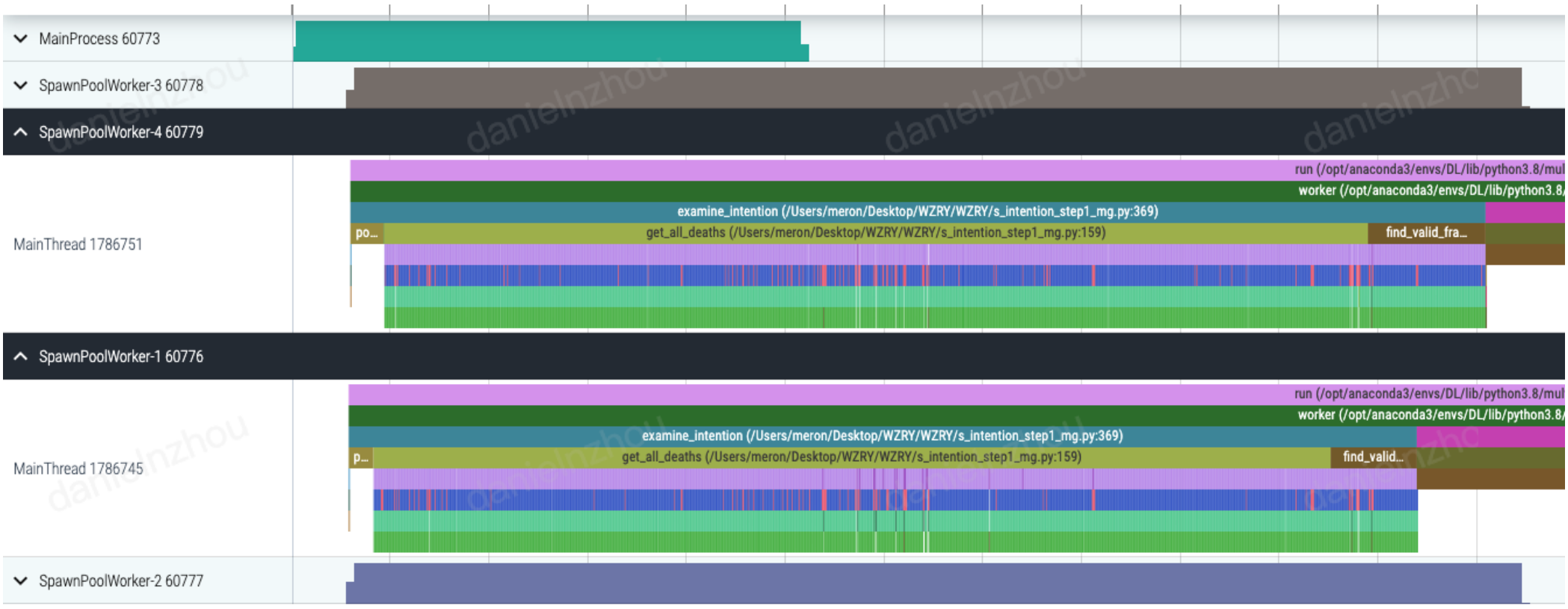
因为 extract data 需要较长的时间，所以采用 multiprocess 的方法进行数据并发

代码如下：

```
# 这里开始并行，并行的对象是每一局游戏
pool = Pool(processes=NUM_WORKER)
pool_res_list = []
for game_count, game_id in enumerate(game_ids):
    pool_res = pool.apply_async(examine_intention, (game_count, game_id, pjt_args, stat_flag, intention_cnt, ))
    pool_res_list.append(pool_res)

pool.close()
pool.join()
```

在每局游戏的维度上进行并行，每个子进程会提取一个游戏的数据，返回本局游戏中的关键信息，之后进行汇总。




对于 s_intention_step1 这个脚本，采用了 multiprocess 多进程并行的方式，

在本机上采用 4 个 worker，在服务器上采用 16 个 worker

脚本中 examine_intention() 函数会被加载进进程池，假设这里有4个worker，则主进程会分出4个子进程，分别从四局游戏中抓取数据。

4个子进程如上图 【60776, 60777, 60778, 60779】，主进程是 【60773】

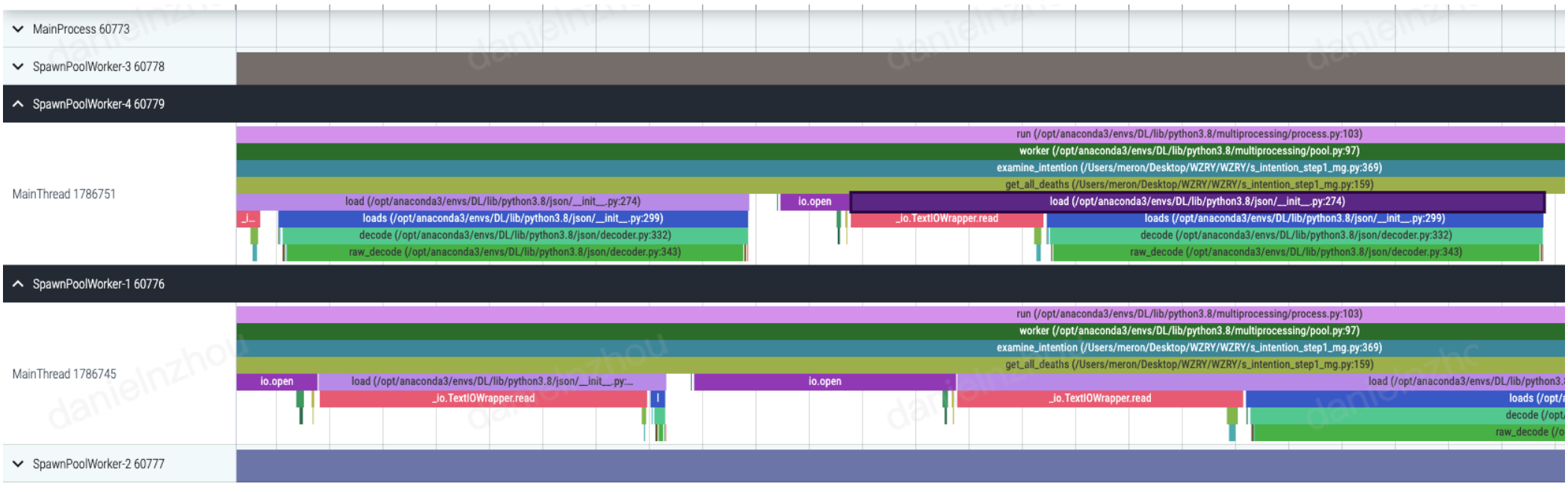
可以从上图直观地看出来，examine_intention() 函数中的时间主要分为两部分，

- get_all_deaths()　这个函数用于提取一局游戏中所有死亡的信息；
- find_valid_frame_hero_death()　因为并不是所有的英雄死亡事件都和意图有关，所以需要提取出有效的英雄死亡的帧数，用于构建正样本和负样本。

但是 get_all_deaths() 函数所用时间太长了，需要检索每一局 battle_obj_frame.json 来判断此帧是否有英雄死亡，大部分时间都卡在数据的 IO 上，

最好能有一种更好的方法直接获取英雄死亡的帧数，这样可以减少很多 data extraction 的时间。

从下图可以看出 json.load() 会占用绝大部分的时间。


MainProcess 60773
SpawnPoolWorker-3 60778
SpawnPoolWorker-4 60779
MainThread 1786751
run (/opt/anaconda3/envs/DL/lib/python3.8/multiprocessing/process.py:103)
worker (/opt/anaconda3/envs/DL/lib/python3.8/multiprocessing/pool.py:97)
examine_intention (/Users/meron/Desktop/WZRY/WZRY/s_intention_step1_mg.py:369)
get_all_deaths (/Users/meron/Desktop/WZRY/WZRY/s_intention_step1_mg.py:159)
load (/opt/anaconda3/envs/DL/lib/python3.8/json/__init__.py:274)
loads (/opt/anaconda3/envs/DL/lib/python3.8/json/__init__.py:299)
decode (/opt/anaconda3/envs/DL/lib/python3.8/json/decoder.py:332)
raw_decode (/opt/anaconda3/envs/DL/lib/python3.8/json/decoder.py:343)
io.open
_io.TextIOWrapper.read
SpawnPoolWorker-1 60776
MainThread 1786745
SpawnPoolWorker-2 60777