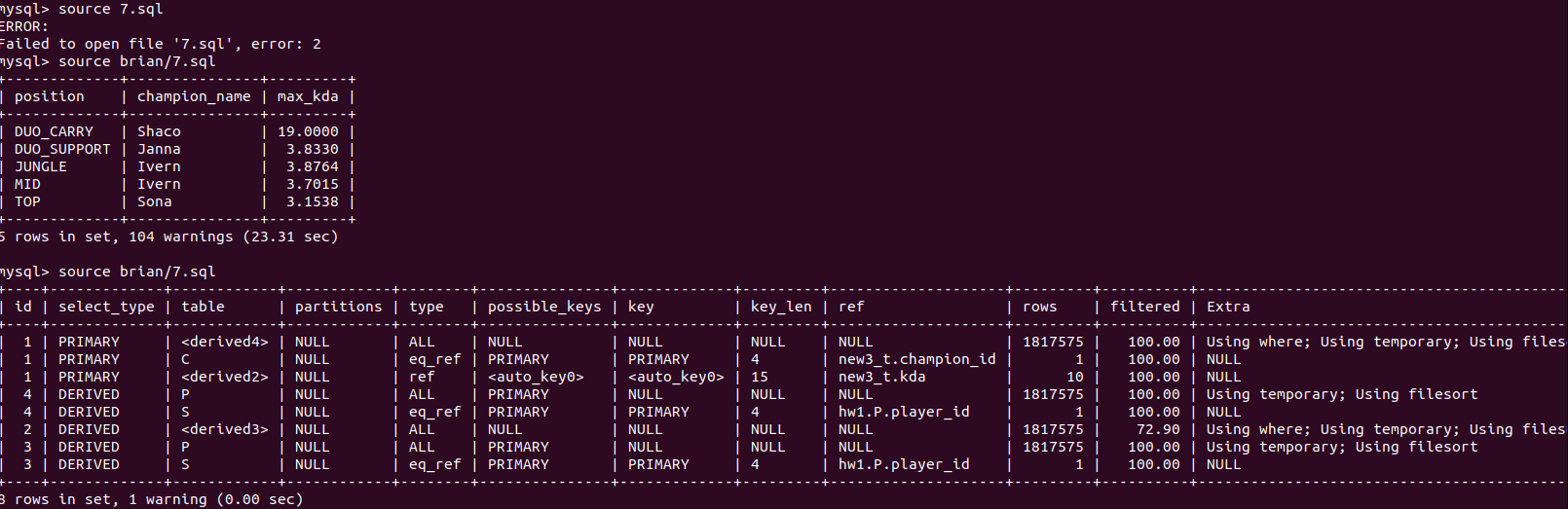
Database HW3 report

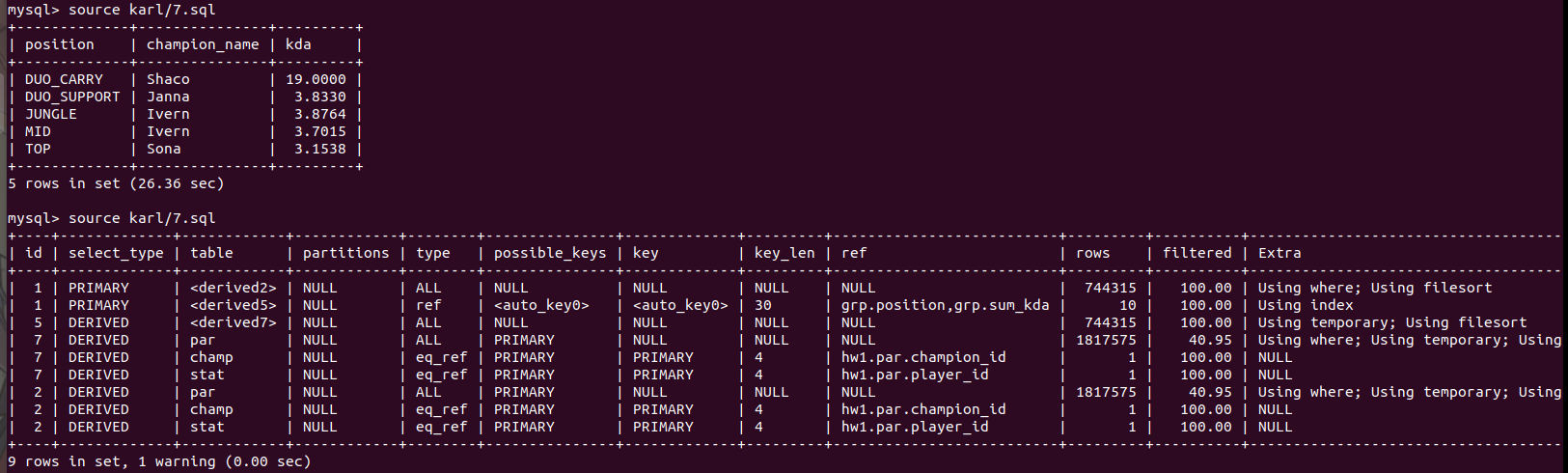
2.1.1

Reference : <https://segmentfault.com/a/1190000008131735>

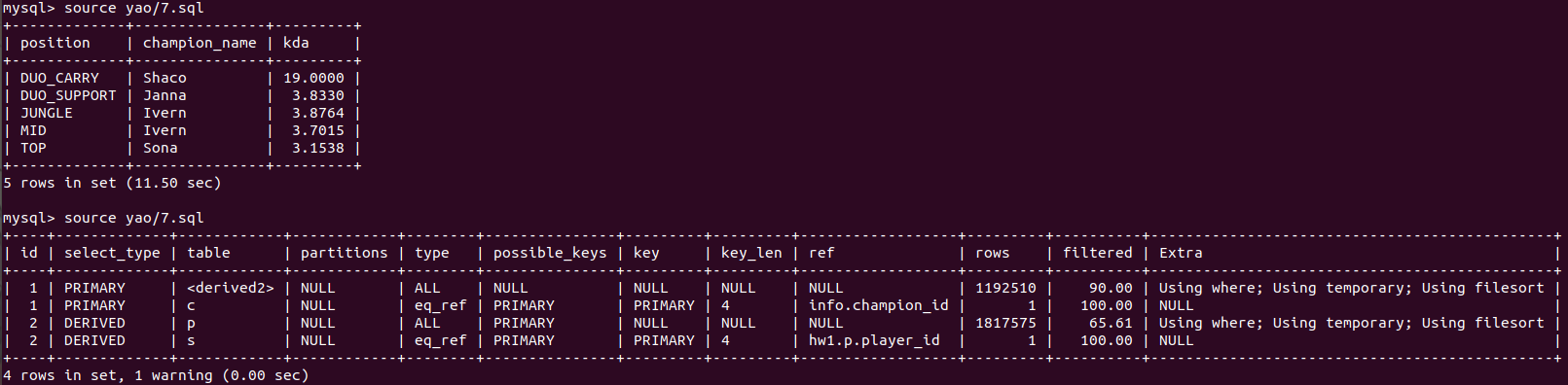
我的 7.sql



Karl 7.sql



Yao 7.sql



我的跟karl助教的explain差異不大 但我只用8個select而karl助教用9 select。

Yao助教則是只用了4個select完成。

Type性能比較:ALL < index < range ~ index\_merge < ref < eq\_ref < const < system

從上面可以知道，用越多ALL type會越慢。

並且在extra用到Using temporary,using filesort也很耗時

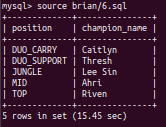
還有如果filter比例越低 這個select規模越小越快。

以上決定了yao的query效率較高。

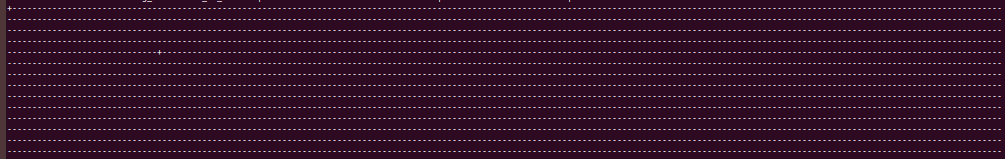
2.1.2

我的optimizer\_trace:

首先先看時間



然後把optimizier\_trace的json打開發現有點長，所以複製到mac OS存成json檔再用VScode開啟，然後因為要符合直立式螢幕，所以有些行我手動縮排。



無意義的行數我先不複製

這些都弄好後，json如下









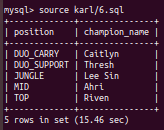


可以看到一個select下面還有subselect，並且select #4,5,8,9各重複兩次，這就是造成時間為17sec的原因

karl:

助教的json看起來很短

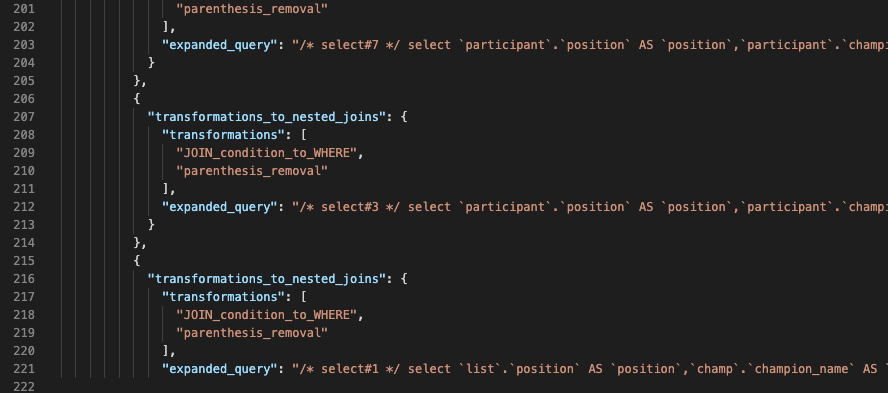
先來看看執行時間



Json:



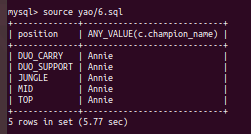




可以看到其實select #4,5,8,9也都重複兩次 寫法跟我的很像

所以得出來的時間才會差不多

Yao:

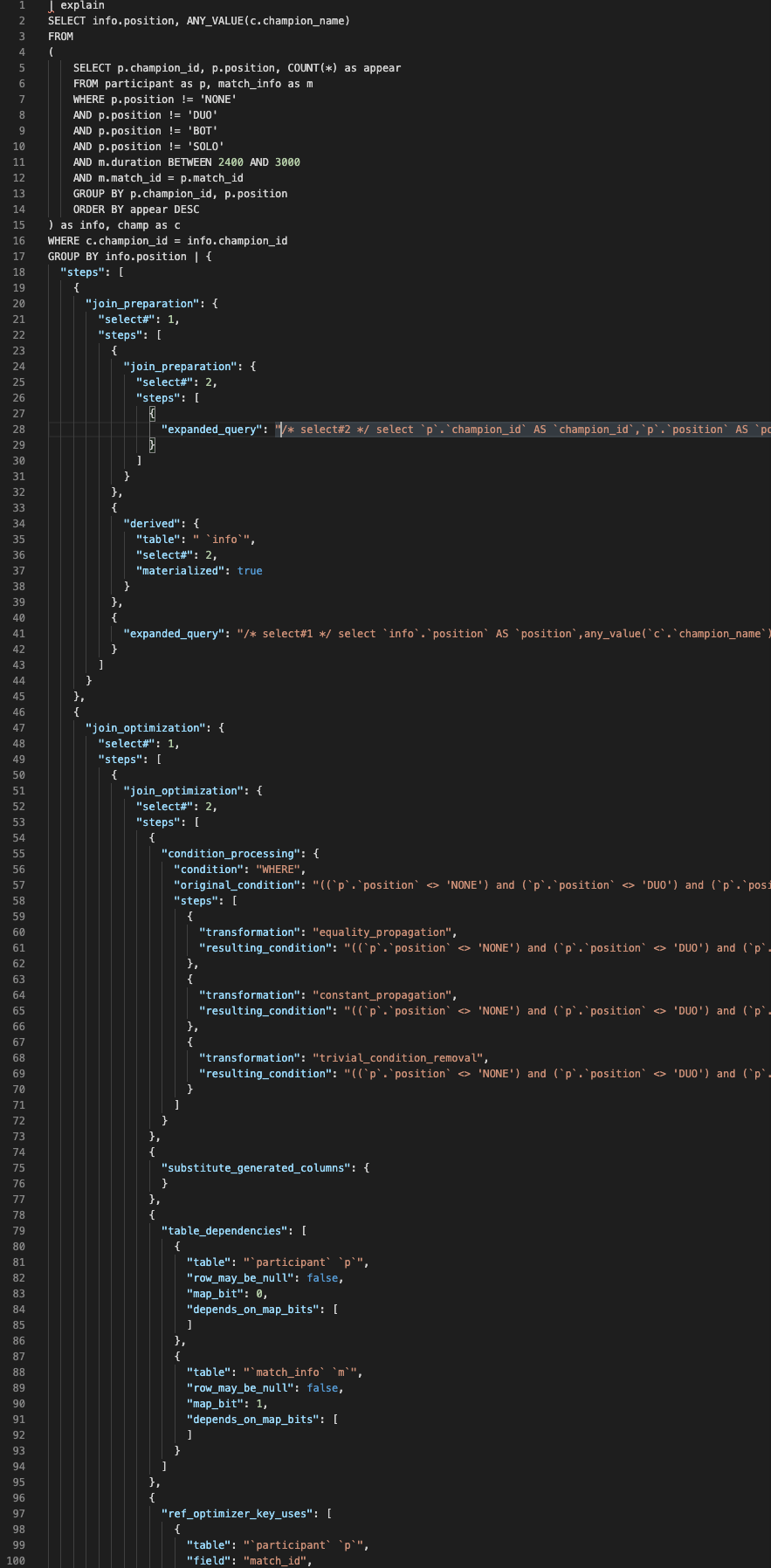


時間只需1/3

但答案在﻿Ver 14.14 Distrib 5.7.30, for Linux (x86\_64) using是錯的

應該是any\_value()的問題

Query也不能在Ver 14.14 Distrib 5.6.21, for osx10.8 (x86\_64)環境下跑













Phase 1 growing phase在22~71行

Phase 2 shrinking phase在72~81行

我的code是用 VXmutex (type為mutex)與 VSmutex(type為mutex)

兩種鎖的vector進行管理

因為研究了很久還是不會用ReadWritelock，也沒有用semaphore

若要write則鎖住兩種鎖，若read則緊鎖著VSmutex。

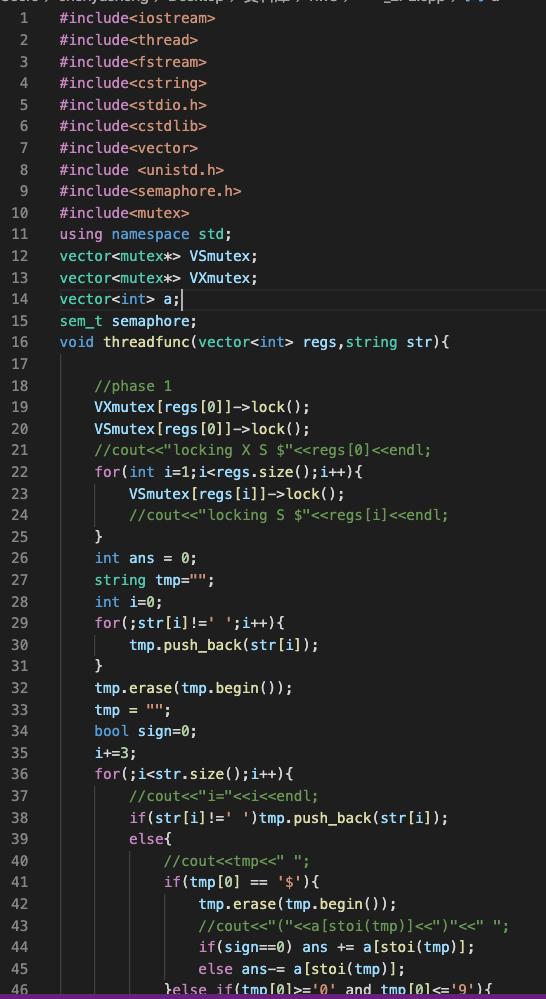
確保資料平行的同時也不會搶順序。

讀算式是一行一行讀進來，第一個token為write的對象，去掉$後即為index，而等號後面帶$的都是read的對象，我先把這些對象紀錄成vector<int> regs，並把regs與整行line全部傳入運算function(我命名為threadfunc)

接者看到function前面幾行，就是先把該上鎖的lock起來，避免其他thread動用到。然後就是詳細的運算，並把結果丟到要write的對象。等全部算完後再unlock。

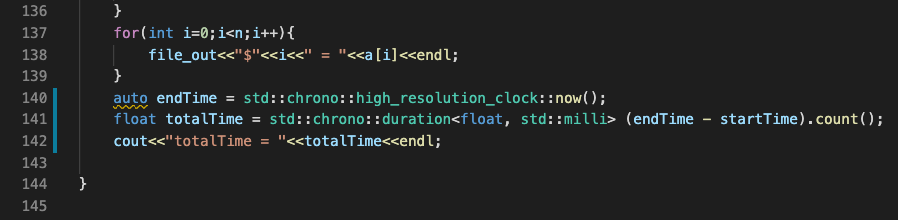
整個程式用chrono函式庫計算時間。

Output出來的答案跟範例答案no difference。

­­­ 

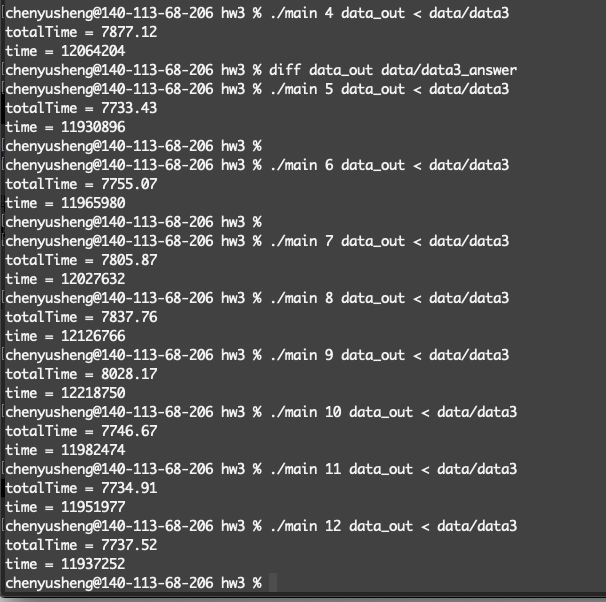






2-2-2

不同的Thread數時間都差不多



2-2-3

Data2在任何thread數量下都正確