b07209019 尤紹宇

1. 設計

使用了4個CPU,
其中
TIMECPU專門計時,
FORKCPU專門fork child process,
CHILDCPU專門跑child processes,
AAAAACPU排程。
main一開始就先fork，然後用sched_setaffinity指定TIMECPU給parent就開始計時，一直到一個shared variable改變表示所有process都處理完了。
使用了mmap來弄shared memory讓各個process都可以讀取時間。
指定AAAAACPU給child然後開始讀輸入，讀完之後又fork一次。
其中一個process被指定FORKCPU，然後一直讀取時間，ready time到了就fork child process，fork成功後就指定CHILDCPU給child processes。
fork出來的child processes會先用sched_setscheduler設定SCHED_IDLE,讓它們看起來好像沒在跑。
另一個process負責排程。根據policy還有一直讀取時間看輪到哪個child process執行時，就用sched_setscheduler設定SCHED_OTHER，變成高的priority。換另一個執行時要休息的process就設定SCHED_IDLE或是剛好結束了就沒了。

2. 核心版本

Linux 4.14.25

3. 比較實際結果與理論結果，並解釋造成差異的原因

雖然不應該執行的process有被設定成SCHED_IDLE，但其實只是prioriy比較低而已，可能還是有在跑只是得到的CPU資源很少，這些有可能會造成誤差。

像是execution time太短的時候，有時候會馬上結束，儘管被設定成SCHED_IDLE，還沒輪到它。
還有context switch時判斷下個程序也會花費一些時間，讓context switch不會瞬間發生，時間就會延後。
還有各個CPU除了這個程式可能也有其他東西在爭奪CPU資源，所以結果很容易浮動。