Producer主要負責將Input queue傳到worker queue。首先要在main.cpp中根據spec要求的宣告4個producer，接著call各自的start ( )。

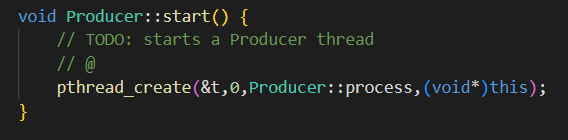
一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

而producer : : start ( ) 這個method主要就是call pthread內建的create，並用接下來實作的process當作執行的function。



Producer : : process ( ) 要先宣告一個producer\* type的變數去接進來的argument

也就是\*this這個本身thread。

一張含有 文字 的圖片

自動產生的描述

接著就進入無限迴圈，只要input queue裡面還有content，producer就要繼續dequeue並搬到worker queue。首先將從input queue dequeue出的item利用已經給的transformer.cpp中的producer\_transform將item的value update，其中要將item的兩個property opcode及val丟進去計算。然後下一步再call item的constructor去創一個新的item，最後enqueue進worker queue。

一張含有 文字, 螢幕擷取畫面, 螢幕 的圖片

自動產生的描述

Consumer controller一樣也是一開始在main.cpp中宣告並直接start。在call其constructor時，將定義好的constant check\_period、high、lowthreshold傳進。

一張含有 文字 的圖片

自動產生的描述



Consumer controller的start相同，一樣將實作的process當作執行的function。

一張含有 文字 的圖片

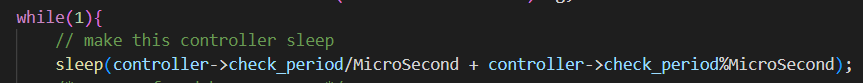
自動產生的描述

Process的起手式也一樣，用一個變數去記\*this。

一張含有 文字 的圖片

自動產生的描述

接著進入無限迴圈。首先要先檢查現在的時間，要符合spec所說microsecond再更新，這裡利用pthread內建的sleep去完成，Sleep ( ) function會將calling thread被suspend from execution。



接著就look into worker queue的size是否有超過high threshold或是低於low threshold的情況，有的話則要分別增加和刪減一個consumer thread。首先檢查size是否高過high threshold，如果有的話就new一個consumer thread，並將worker queue、writer queue、transformer傳入，因為consumer負責的就是利用consumer\_transform將worker queue的傳到writer queue。然後新增的這個thread也要call start讓他開始做事，最後再將它push\_back到consumers ( 定義在consumer controller private中 ) 這個紀錄所有consumer thread的vector。此外，這裡也要印出spec要求的Scaling up的message。

一張含有 文字, 螢幕擷取畫面, 螢幕 的圖片

自動產生的描述

再來檢查low threshold部分，首先要確保至少還有一個consumer thread，否則就continue。這邊一樣要印出spec要求的Scaling down，並從consumers這個vector中cancel掉newest的thread，也就是最晚push進來的back ( )。因此consumers也要做pop\_back ( )。

一張含有 文字, 螢幕擷取畫面, 螢幕 的圖片

自動產生的描述

Consumer部分，一開始只有在consumer controller創新的consumer thread時才會出現，並且在創完後call start。

一張含有 文字 的圖片

自動產生的描述

比較不一樣的是consumer有cancel這個method，在consumer controller要cancel consumer thread時便會call這個method。要在裡面將is\_cancel這個變數改為true，因為接下來process時會用到。最後再用pthread.h裡的cancel將calling thread cancel掉。

一張含有 文字 的圖片

自動產生的描述

接著進入process，同樣用一個變數去記\*this。下面call pthread\_setcanceltype的用意是這個function改變的只會是current thread的cancel state，不會改到其他thread的，而且不會馬上cancel，只有在check is\_cancel時才會做cancel。

一張含有 文字 的圖片

自動產生的描述

再來就進入迴圈了，只有在is\_cancel == false時會持續執行。如果calling thread的cancel state是PTHREAD\_CANCEL\_DISABLE，取消的請求會被放入隊列。直到calling thread的cancel state變爲PTHREAD\_CANCEL\_ENABLE，取消的請求才會從隊列裏取出，發到calling thread。接著look into woker queue，如果裡面還有content就dequeue出來。這邊跟producer很像，要用transformer.cpp中的consumer\_transform update新的item value，並重新做出一個item最後enqueue進writer queue給writer做事。

一張含有 文字 的圖片

自動產生的描述