# 1.(a)I/O Bound vs CPU bound process:

I/O Bound def: 指的是系統的CPU效能相對I/0(硬碟/記憶體)的效能要好很多，此類型工作需要大量I/O 動作或I/O request的頻率很高，但僅需少量的CPU 計算或CPU處理速度很快，其工作效能取決於I/O 存取的快慢，此時CPU Loading不高。

CPU bound def:指的是系統的I/O(硬碟/記憶體)效能相對CPU的效能要好很多，此類型工作需要大量計算或CPU 處理速度很長，此时系统運作CPU Loading 100%，但僅需少量的I/O request或I/O存取速度快，其工作效能取決於CPU速度，此時CPU Loading很高。

I/O Bound可能發生情況:

1. 資料量大I/O要讀取資料需花費一段時間。
2. 資料過於分散混雜，要找到需要資料需花費一番功夫。
3. 該process需要大量I/O request，而pipeline做得不是很好。
4. 硬碟存取速度過慢。

CPU Bound可能發生情況:

1. 需要作大量的計算或邏輯判斷，要花費一段時間才能完成。
2. 執行的程式效能不好，演算法不夠好，像是將數字從1,2,3,4,…,N累加起來，如果用迴圈計算，下列程式時間複雜度需要O(n)，

int sum(int n){

int sum = 0;

for(int i = 1;i <= n;++i)

sum += i;

return sum;

}

而從數學思考邏輯改為下列方式時間複雜度只需要O(1)，當數字越大相差的時間就越 多。

int sum(int n) {

return n \* (n + 1) / 2;

}

1. 資料處理完需要壓縮/解壓縮，造成CPU已經計算完資料還要花額外時間壓縮/解壓縮。

避免I/O Bound解決方式:

1. 提升資料儲存裝置存取速度，像原本硬碟為Hard Disk Drive可以換成SSD固態硬碟，提高存取效率，SSD就是將許多的隨身碟中的記憶體，通通拚在一起後組成的硬碟。由於 SSD 的儲存運作原理，並不需要轉動詞盤與移動讀取頭，因此省略了「找尋資料 → 移動磁盤與讀取頭 → 讀取」的時間，同時也可以更快、更有效率的傳輸資料，這造就了 SSD 在傳輸速度上大勝傳統硬碟(HDD)的結果，其效能有時甚至可差距到十倍以上
2. 提升記憶體規格或增加記憶體容量。
3. 將硬碟重組將分散資料集中加快存取速度。
4. 運用multiprogramming的方式，若該process需要大量I/O而CPU Loading不高，可以先執行別隻快速執行完的process，待原先的process的資料載入後再執行。

避免CPU Bound解決方式:

1. 提高CPU效能增加時脈頻率。
2. 改善執行程式效能。
3. 以空間換取時間增加存儲空間，減少CPU針對資料的壓縮/解壓縮的頻率。

I/O Bound vs CPU bound感想:

不管是I/O或CPU兩者都是會互影響的，有時候CPU執行過久的情況可能不單只是CPU的問題，也可能是硬碟空間不足而需要針對資料壓縮或資料整理排序而花費而外的時間，像早期硬碟空間少且成本很昂貴、記憶體容量提升需要花費的成本是幾倍成長，因次就需要針對資料儲存花費額外的功夫作處理，而現今硬碟、記憶體成本大幅度降低、大小容量也越來越高，且存取速度也越來越進步，因此CPU也只需要計算不需要額外的處理，但一切還需要一個厲害的OS來對兩者進行調和。

# 1.(b)Multiprogramming vs Time-sharing system

Multiprogramming def: