電信4B 8614022 劉士皓

Time-Memory-Processor Trade-Offs

Hamid R. Amirazizi
&
Martin E. Hellman
Fellow, IEEE

IEEE Transactions on information theory, vol. 34 No. 3, May 1988

論文發表動機：

當時個人電腦正逐漸興起，
因價位較大型主機低廉許多，
故產生由多台小型機器取代大型主機的想法。
Hamid和Martin兩位學者以簡化的model
說明大型主機的cost effectiveness
是小型機器無法超越的優勢所在。

物換星移，
即使是在個人電腦空前盛行的今天，
大型主機的地位仍然無法被取代，
可見這篇論文觀點的正確性。

單一**processor**的年代：
TM：Time-Memory Trade-Offs

$$TM = N$$

T: time
M: memory
N possible sol.

相同的計算量
memory越多，花費時間越少

$$C_m = C_p + c_m M$$
$$\therefore C_m \sim M$$
$$C_T \sim C_m T$$
$$C_S \sim C_T / S$$

C_m: cost of machine
C_p: cost of processor
c_m: cost of memory
C_T: total cost
C_s: cost per sol.
$\bar{M}$: amount of memory
S: simultaneity of sol.

TM：砸金錢買memory換取時間
對整體的cost並沒有增加

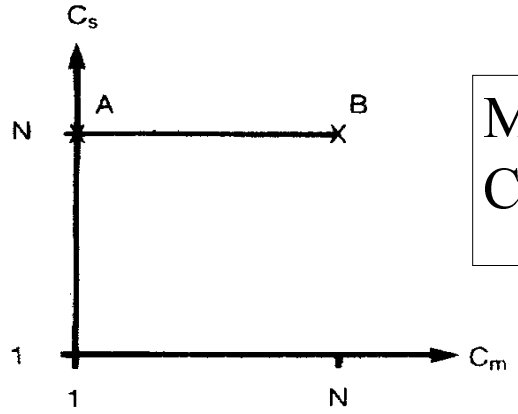

Fig. 1. ($C_m - C_s$) curve for usual TM trade-off.

MT=N

$C_S \sim C_m T/S$

因爲TM不能節省cost，才會有TMP的產生

TMP：

$C_m = C_p P + c_m M$

$\therefore C_m \sim (P,M)$

$C_T \sim C_m T$

$C_S \sim C_T/S$

C_m: cost of machine

C_p: cost of processor

c_m: cost of memory

C_T: total cost

C_s: cost per sol.

P: amount of processor

M: amount of memory

S: simultaneity of sol.

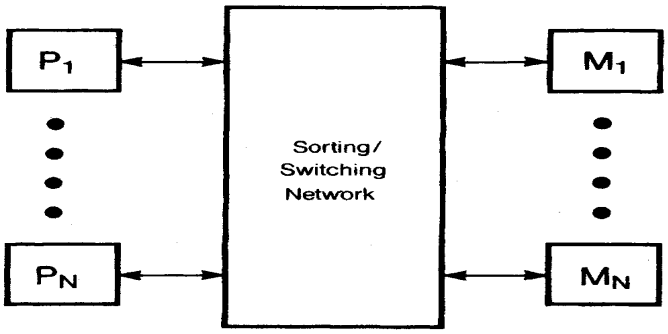

Fig. 2. Cost-effective table lookup machine architecture.

Sorting：將計算結果依大小存至記憶體中，每個processor存取固定位置的記憶體。

Switching：每個processor可要求存取資料所在的記憶體。

平行處理的威力！

Table Lookup

錢花的越多，processor與記憶體就越多
同時處理的工作就越多

S (Simultaneity sol.)

遞增

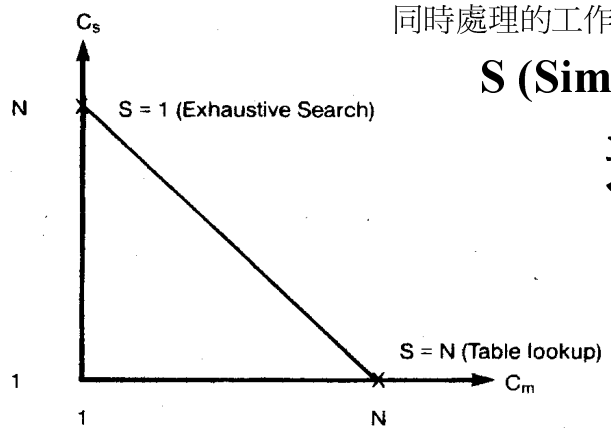

Fig. 3. TMP trade-off for searching (N possible solutions).

S變大不是好現象！
→會有delay產生

Ex:(a) $C_m \sim N^{1/2}$, $T \sim 1$, $S=1$ $\therefore C_s \sim N^{1/2}$

(b) $C_m \sim N^{1/2}$, $T \sim N^{1/2}$, $S= N^{1/2}$ $\therefore C_s \sim N^{1/2}$

(a) can imitate the performance of (b) by solving $N^{1/2}$ problems in $N^{1/2}$ units of time "without" any time delay.

更好的兩種TMP範例：（Binary Knapsacks）

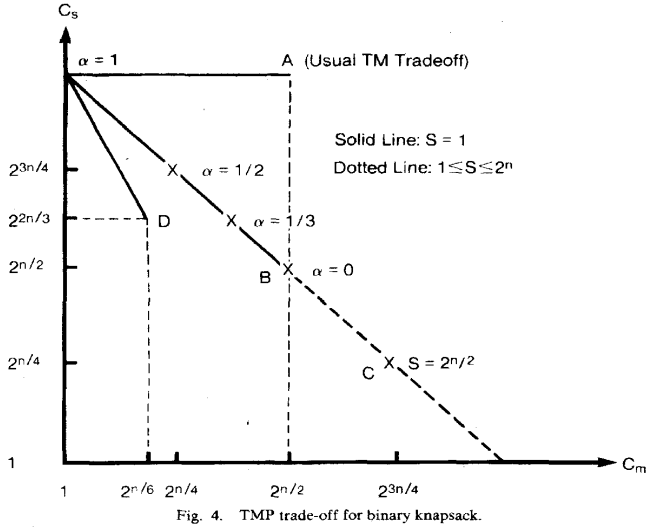

Fig. 4. TMP trade-off for binary knapsack.

更好的兩種TMP範例：（Double Encryption）

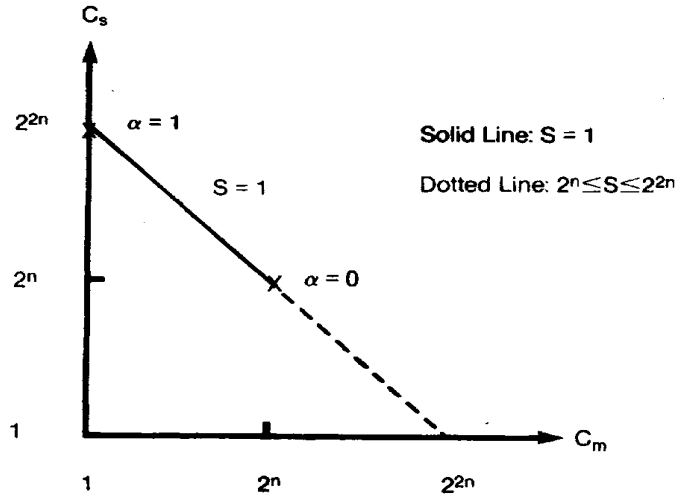

Fig. 5. TMP trade-off for double encryption.

結論：

1. Big is better!

只要計算量大到一定程度，多處理器的大型主機一定比多部小型機器經濟。

2. Simultaneity越大，time delay越大，cost越大

推廣與應用：

1. C＋＋物件導向的程式開發工具（ex:VC++, BCB）
   使得程式開發更爲迅速方便
   但是在各函式之間則多出不少redundancy
   特別撰寫的程式執行效率應該會較好

2. 一人或多人同時間處理多份工作
   效率不若
   同時間處理一份工作，處理完再處理下一個