硬體膨脹極限 3 wall

摩爾定律有可能消失

=> 靠軟體加速 compiler

原因: 應用軟體佔運算時間幾乎不到 10%

折舊問題(折舊比例高 compiler可以延長壽命)

現今的重loading大部分都用硬體做掉

換硬體只有辦法解決重loading 無法解決

以手機加速有極限(ILP wall)

解決: 平行程式(平行化主程式)

Compiler!!!

CPU的綜合表現

Dhrystone

Coremark

GCC 的最佳化我們通常少用

且幾乎都會被linker破壞(relocation問題)

因為compiler看的不夠大 但即便看了也無法最佳化

Compiler 有時候會看到不應該看到的東西

人類的系統軟體 assembly linker.....

Linker 太複雜 有很多歷史痕跡 不好改

MC linker Good!!!

一開始的linker:

Gun id linker: interpreter(一行一行看)

Google gold linker: 有two stage 中間碼

人類思維很難平行!!

MC Linker

四階段

Normalization, resolution, layout, emission

Weak symbol:

Comdat: 都是first come first serve (順序性)

建構樹 Input tree

DL heal

Link symbol: 一定要dfs

Link section:自由

程式三部分 指令 資料 bss (放0)

Fragment: function, label, data...

Defined symbol (define use 關係)
Symbol (topology拓撲問題)

Fragment reference graph

最佳化
Fragment stripping(unreachable, GG他)

Apply relocation

S-p+a 找到位置 直接貼上!?

Dynamic prelinking: runtime

Static: 一開始就連好

Symbol:

Reorder: 重建順序 增加hit rate