**專家心得報告之二**

106級甲班 F74024070 蔡侑軒

今日講師：Luba Tang (唐文力) 老師

上課時間：105年4月29日下午兩點到五點

簡介：

清華大學碩士班畢業

致力於

**electronic system level(ESL) design**：electronic system level simulation, electronic system level design

**Compiler**：iterative compiler, ahead-of-time compiler, link-time optimization

近期成果: dynamic liking and link-time optimization

名稱由來：阿魯巴 => 欠阿的luba

經歷：

晨星半導體

Marvell

聯發科(做出MC Linker)

現任Skymizer的CEO

◎Linker

* 前言：目前困境
* 硬體膨脹的極限：3 wall
  + Power wall：電腦會越來越燙
  + Memory wall：記憶體的頻寬限制
  + ILP(Instruction Level Parallelism) wall：指令平行化困難
    - 摩爾定律在2015年消失！
* 應用軟體佔運算時間幾乎不到10%
  + - 單純加速應用軟體效果不彰
* 折舊資金比例高
* 現今重loding的工作大部分都用硬體做掉
* 加速有極限(因ILP wall)
  + - 平行化主程式

🡺 平行程式？但人類的思維很難平行！

※Thread parallelism、Data parallelism

🡺 結論：Compiler！減輕硬體loding，可以延緩折舊

* 評量CPU表現指標
* Dhrystone
* CoreMark
* Compiler窘境
* GCC雖有最佳化，但其幾乎都會被linker破壞(因relocation)
* Compiler看得不夠大，無法最佳化
* 但Compiler看大不一定能解決問題
* Compiler會看到不應該看到的東西
* Linker現況
* Linker太複雜！有太多的歷史痕跡！導致難以修改
* Linker的演化
  + Gun id linker：Interpreter(一行一行看)
  + Google gold linker：Two stage
    - Stage 1：做出中間表示法
    - Stage 2：平行化處理(假平行！？)

🡺 MC Linker 很棒！

* 語意
* 由前到後的：INPUT(File1, File2)
* Group：Group(archive1, archive2)
  + Group是檔案集，若第一次找不到檔案，就在找一次，到找到為止
* Linker command line
* 是一種程式語言
* AST的點：object files、archives、linker、script entrances of group
  + Archive包含很多.o檔，順序重要
* AST的線：positional edges、inclusive edges
  + Positional edge：A->B 表A在B前面
* MC Linker
* 核心概念：把困難的Link問題轉成離散數學問題
* 共有四階段：Normalization、resolution、layout、emission
  + Normalization
    - Input：object file，archive，linker script
    - Step：
      * 輸入command line option
      * 分辨archives and linker scripts
      * 將兩者轉換成subtrees
      * 合併整合subtrees
* Weak symbol：若printf在兩個檔案出現時應該要先挑哪個，first come first serve (有順序性)
* Comdat：first come first serve (有順序性)
* 建構樹：input tree
* DL heal
* Link symbol：一定要DFS
* Link section：其實方法不限，在這裡選用BFS
* 程式三部分：指令、資料、bss(放0區域)
* Fragment：function、label、data
* Defined symbol(define use關係)
* Symbol resolution =>topology 拓譜排序問題=> Fragment reference graph
* Fragment reference graph 的最佳化
* Fragment stripping：如果unreachable(未使用)，就刪除它
* Branch optimization：用low cost branch取代 high cost branch、改變relocation type
* Apply relocation：S-p+a
* layout 的最佳化
* Dynamic prelinking：runtime時linking
* Static prelinking：一開始就linking
* Symbol stripping
* Block reordering：重建順序，把類似的東西放一起，增加hit rate；自己寫的code通常很常用，所以會放在前面優先搜尋，增加hit rate

課程心得：

聽台上的Luba又是講述著他的MC Linker如何屌打全世界的人，又是談著他如何開公司，資本額幾年內就翻破十倍……一個想法突然從我的腦中呼嘯而過，現在台上的這個人，毫無疑問的是成功者，佇立於世界頂端的佼佼者，但對我而言，僅僅是坐在台下聽他講課，真的就能引領我走向成功的道路嘛？雖然他在Linker上貢獻十分卓越，但現在的我對Linker沒有任何興趣，也不會因為聽了這堂課就突然跑去寫Linker，即便未來我真的會去寫Linker，但到那時候今天所講的一字一句早已消失在茫茫腦海中，最後也還是得Google資料從0開始，那今天這堂課給我的東西究竟是…？上了大三之後，為了自己有興趣的領域開始奮鬥，漸漸感受到與大一大二截然不同的時間壓迫感，總是覺得沒有時間來讓自己的東西變得更加完美……而上一次的專題meeting，教授就直問我，數學跟程式都還不錯吧？程式因為有持續在寫所以覺得罩得住、有自信，但數學我卻難以啟齒，並非我當初學得不好不認真還是恰巧低空飛過，而是時過兩年，要再次提起筆快速解出微積分，老實說我真的沒把握…於是現在若真必要用到，也是只能乖乖開始從以前的課本、Google上從頭學起，既然如此，我大一大二所花在這些課上的時間究竟是甚麼！？如果這些時間能空出來，用來精進我現在有興趣的東西上，讓其更加完美，等到我真需要去使用到微積分工程數學時，再開始學，學完馬上應用，學習效率不是應該更好嘛？

在這些思緒如夢魘般不斷摧殘我的思緒時，一段從沉默台下發起的提問聲打斷了這一切，是Jserv教授在幫大家努力的提問題……聽著Jserv的聲音，我想起了他的經歷：成功大學肄業，在大學三年級時毅然決然地離開學校……這是盧文祥教授最喜歡在他上課講的故事之一。或許Jserv是發現了現在這個學制的異狀，所以才離開這裡的？ 或許他是發現了現在的通才/全人教育，如同想要伸手緊握智慧之砂，但卻往往讓人一無所獲……

想到這裡，我忽有這股衝動想要立刻起身離開教室，但箝制住我的，既非鐵牢也非手銬，而是一張輕如鴻毛的文憑，為了它，靜靜地坐在這裡，修完每一堂課是必要的……我想我所缺乏的，正是衝破這紙所需要的勇氣，以及力量吧？

於是，我開始深思這堂課所能帶給我的究竟是甚麼？我想不是MC Linker如何能超越遠古時代的Linker，而是概念，引領Luba邁向成功的概念。第一點就是將你無法解決的難題，轉化成你所熟悉的問題。Luba在上課時提到，他們一開始在研究Linker時是一個頭兩個大，因為即便找了一個新演算法，也很難證明其的正確與否。於是他們開始把這複雜難懂的問題，先轉換為大家所能接受的離散問題，再用離散數學的概念去解題，再去證明！第二點是，冷靜的思考才是致勝方針，而且要做就要一次做到最好！並非只有Luba他們在搶改善Linker這個大碗，像是Google這種大公司也是對其躍躍欲試，但Google他們，或許是汲汲於要做出績效給上面的人看吧？所以總是在改善一些小地方的效能，而沒有坐下來冷靜思考整個Linker的全貌，才會在一開始領先Luba團隊，但最後卻被Luba狠狠甩在腦後吧？而有趣的是，不久前莊坤達教授也同樣說出了概念類似的話！不謀而合呢！