行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　※
※　　　　　JAVA 程式原始碼最佳化工具　　　　　※
※　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：個別型計畫
計畫編號：NSC89－2213－E－009－146－
執行期間：89 **年** 8 **月** 1 **日至** 90 **年** 7 **月** 31 **日**

計畫主持人：楊武
共同主持人：

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊科學學系

中　華　民　國 90 **年** 10 **月** 12 **日**

行政院國家科學委員會專題研究計畫成果報告
國科會專題研究計畫成果報告撰寫格式說明
Preparation of NSC Project Reports

計畫編號：NSC 89-2213-E-009-146
執行期限：89 年 8 月 1 日至 90 年 7 月 31 日
主持人：楊武　國立交通大學資訊科學學系
計畫參與人員：陳建財 黃經緯　交通大學資訊科學學系

一、中文摘要

使用應用軟體伺服器可以大幅節省軟體安裝及修補的成本，一部應用軟體伺服器可以供多少客戶端使用，是由網路頻寬及程式大小決定。加大網路頻寬是提昇伺服器利用率的方法之一，但寬頻網路需要花費大成本。另一方法是利用軟體技術縮減程式大小，便宜又省事。

Java 是低成本、高存取的 Internet 和 intranet 應用系統理想的程式語言，由於 Java 的可攜性、網路集中性，許多企業亦將其視為未來運算環境的主要開發平台。同時 Java 程式碼體積小、跨平台的特性，更有助於應用軟體伺服器與簡易型客戶端(thin-client)的施行，大幅節省軟體安裝及修補的成本。

本計畫的目的是要利用程式切片的技術開發出一套縮減 Java 程式碼大小的工具，其方向有二，一是刪除用不到的 class、函式、和指令，二是將 class 檔案的目錄結構扁平化及名稱簡潔化，縮減 class 檔案中 string pool 的大小。

此工具可以大幅縮減 Java class 檔案的大小，除了加快程式載入執行的速度外，更可以使每部應用軟體伺服器所能支應的客戶端變多，使企業更願意採用 Java 開發應用程式。

關鍵詞：程式最佳化、程式切片、Java

Abstract

Using an application server can reduce the cost of software installation and patching. How many clients the server can support is decided by the bandwidth of network and program size. Extending the bandwidth is one way to increase the utilization of application server. But this way needs a lot of money. Another way is using software technique to reduce the size of a program. This way will be more cheap and easy than the previous one.

Java is a low cost, high accessing programming language. It is very suitable to be used to develop application systems of Internet and intranet. Many enterprises treat Java as the main development platform of computing in the future because Java is portable and network-concentrated. Furthermore, the features of small object file and cross-platform are very helpful to implement thin-client structure.

The purpose of this project is to develop a Java program source code optimizer based on program slicing technique. There are two directions to go. First, there are some redundant classes, methods, and statements in a program. We can eliminate those redundant code to optimize program. Second, package names and class name can be abbreviated and the directory structure can be flatten such that the size of string pool in the class file can be reduced significantly.

This tool can reduce the size of a class file significantly. The time of loading program will be shorter. Furthermore, the number of client that an application can support will be increased. The benefit is very important to an enterprise.

Keywords: program optimization, program slicing, Java

二、緣由與目的

在早先大、中型主機當道時期，由於電腦主機製造費用昂貴，因此採用便宜的終端機與主機連線，讓多人可以同時使用，提昇其利用率。這時候的終端機並無運算功能，所有的軟體及運算都在主機上進行。爾後，由於硬體技術進步快速，個人電腦普及，其計算能力與之前的大型主機有過之而無不及，此時的軟體及運算都變成在個人電腦上進行。

以企業的觀點來看，個人電腦價格便宜，計算能力又強，是企業裡不可或缺的工具。但要管理數以千計萬計的電腦卻是非常困難的工作，假設某一企業擁有一千二百台個人電腦，安裝或修補（patch）一套軟體需十分鐘，則總共需要二百小時才能完成，若以每個工作天為八小時計算，則需二十五天才能完工。這還是在理想狀態下的數據，若要再考慮昇級前後資料格式同步的問題，一家企業是不可能停工二百個小時來等軟體昇級或修補的。

因此就有一些解決方案提出，例如發展出一些自動昇級的程式及協定，讓個人電腦自動連線下載及更新軟體。另一解決方案是應用程式伺服器，把應用軟體安裝在伺服器上，提供給個人電腦在需要時連線下載使用，若軟體需要昇級或修補，只需針對伺服器即可。

應用程式伺服器是個很好的解決方案，但目前的瓶頸是由於軟體龐大，網路頻寬有限，造成每台應用程式伺服器所能支應的 client 端非常少，軟體下載啟動耗時，效果不彰。另外再考慮到作業系統平台的問題，每台應用程式伺服器只能提供給同質（homogeneous）的平台使用，無法達到跨平台的便利性。

最近由於 Java 的推出，使得 Internet 可發揮分散式主從運算的最大功效，讓電腦程式設計人員可以寫出複雜而又強壯的客戶端應用程式，可以運用客戶端 CPU 的運算和處理能力，作出一些像是試算表、工程計算、卡通動畫、電腦遊戲和交談式應用程式之類的複雜電腦應用。

Java 打破傳統的籓蘺，平台作業系統不再重要，因為 Java 就是一種標準的平台。從其問世以來，Java 即對軟體開發業界造成了巨大的衝擊。而這自有其很好的理由。Java 是低成本、高存取的 Internet 和 intranet 應用系統最理想的程式語言。

由於 Java 的可攜性、網路集中性，許多企業亦將其視為未來運算環境的主要開發平台；同時 Java 亦有可能改變現有的軟體銷售方式，呈現新的軟體租售方式。我們可以預見的是：

- Java 運算改變了企業的桌上型系統
- Java 運算改變了企業運算的系統管理模式
- Java 運算概念改變了軟體開發的方式
- Java 運算環境改變了既有網路的角色

Java 採用虛擬機器的方式來達到跨平台的目的，其編譯所產生的 class 檔案使用 bytecode，程式比一般的 Windows 程式小許多，然而放在應用程式伺服器上供人下載使用時，網路頻寬亦會成為其限制。Sun Microsystems 公司為 Java 開發出 JAR 技術的目的，就是希望能利用資料壓縮的方法，把 class 檔案和目錄結構一起壓縮到一個檔案裡，以節省下載時的網路頻寬。

根據使用經驗，利用 JAR 的確能大量減少下載的資料量。本計畫的目的，則是希望能從根本做起，先把要傳送的 class 檔案變小，再利用 JAR 技術壓縮，達到比原先單純使用 JAR 技術更少的資料傳輸量。

我們從以前的程式切片計畫中，發現物件導向的 class library 在連結後，其實有許多部份是可以省略的，並且不會影響程式的功能。我們希望能利用程式切片的技術，從根本的原始碼著手，對 Java 程式做最佳化，縮減其編譯後的體積，以大幅提昇應用程式伺服器的效能。

本計畫對於推廣 Java 有相當大的助益，如果 Java 程式可以大幅縮減體積，使得單一應用程式伺服器可提供更多的客戶端連接使用，除了明顯的經濟效益之外，更可提昇企業界採用 Java 做為設計應用程式的標準語言的意願。

三、結果與討論

本計畫在初期針對單一原始程式分析時，尚稱順利，功能亦符合既定的要求。但在實行後期，預將分析範圍擴展至整個應用程式（包括多個 packages 和多個程式）時，遭遇到一重大問題，在構想此計畫時，

並未想到 callback 的問題, callback 被大量運用在 Java 的 event model 裡。但 callback 的使用範圍不僅限於此，廣泛來說，任何可接受 object 為參數的 method 都有可能會使用 callback。如果無法分析這類會 callback 的 methods（caller）的原始程式，就無法確切得知那些 methods（callee）必須保留在最佳化過後的程式裡。若以安全的做法，保留所有可疑的 methods，由於 Java 大量使用 callback，最後會使得最佳化變得作用很小，甚至無作用，那就抹滅了本計畫的原意。

比較好的作法是能夠直接分析那些 callers, 但是如果 callers 並不是自己寫的, 而是存放於其它的程式庫, 如 Java JDK 裡或其它商業性的程式庫，就很難取得其原始程式。我們想到的變通方法是自己寫一個 decompiler，Java class 檔案保存有豐富的資訊，而且 Java 的程式是結構化的，其 decompiler 應該不難完成，decompiler 只須建立與現有系統相同之 abstract syntax tree，就可以與現有系統相結合，做精確的分析。

雖然目前已有許多現成的 decompiler，但不易與我們系統結合。我們預計在明年的計畫完成一 Java decompiler, 使 Java 程式最佳化系統更完美，做更精確的分析。

四、成果自評

目前本計畫的執行成果並不完美，因為在分析程式部份缺了一項重要工具－decompiler，因此最佳化的效果並不明顯，我們希望在未來完成 decompiler 之後，能提昇本系統的最佳化效果。

五、參考文獻

[1] Allen, J.R., K. Kennedy, C. Porterfield, and J. Warren, "Conversion of control dependence to data dependence," in Conference Record of the Tenth ACM Symposium on Principles of Programming Languages, (Austin, TX, Jan. 24-26, 1983), pp. 177-189, ACM, New York, January 1983.

[2] Ferrante, J. and K.J. Ottenstein, "A program form based on data dependency in predicate regions," in Conference Record of the Tenth ACM Symposium on Principles of Programming Languages, (Austin, TX, Jan. 24-26, 1983), pp. 217-231, ACM, New York, January 1983.

[3] Larsen, L.D. and M.J. Harrold , "Slicing Object-Oriented Software," TR 95-114, Clemson Univ., Computer Science Dept., No vember 1995.

[4] Horwitz, S., J. Prins, and T. Reps, "Integrating non-interfering versions of programs," in Conference Record of the Fifteenth ACM Symposium on Principles of Programming Languages, (San Diego, CA, Jan. 13-15, 1988), pp. 133-145, ACM, New York, January 1988.

[5] Horwitz, S., T. Reps, and D. Binkley, "Interprocedural slicing using dependence graphs," Proceedings of the ACM SIGPLAN 88 Conference on Programming Language Design and Implementation, (Atlanta, GA, June 22-24, 1988), ACM SIGPLAN Notices, vol. 23, no. 7, pp. 35-46, June 1988.

[6] Reps, T. and W. Yang, "The semantics of program slicing," TR-777, Computer Sciences Dept., Univ. of Wisconsin, Madison, WI, June 1988.

[7] Reps, T. and W. Yang, "The semantics of program slicing and program integration," Proceedings of the International Joint Conference on Theory and Practice of Software Development (Colloquium on Current Issues in Programming Languages), (Barcelona, Spain, March 13-17, 1989), Lecture Notes in Computer Science, vol. 352, pp. 360-374, Springer-Verlag, New York, 1989.

[8] Tip, F., "A Survey of Program Slicing Techniques ," TR 756, Computer Sciences Department, Univ. of Wisconsin-Madison, March 1988.

[9] Weiser, M., "Programmers use slices when debugging," Comm. ACM, vol. 25, no. 7, pp. 446-452, July 1982.

[10] Weiser, M., “Program slicing,” IEEE Trans. Software Engineering, vol. SE-10, no. 4, pp. 352-357, July 1984.