

# Free Java Runtimes

自由且更適合研究與  
自訂化的實做

JavaTwo 2005

Jim Huang ( 黃敬群 / **jserv** )

<http://jserv.sayya.org/>



Powered By  
*jserv's lab*

# To be listed here...

- Stuart Ballard, Mark Benvenuto, Geoff Berry, James E. Blair, Eric Blake, Sascha Brawer, Nic Ferrier, Paul Fisher, Anthony Green, Jochen Hoenicke, Brian Jones, Roman Kennke, Michael Koch, John Keiser, John Leuner, Warren Levy, Bryce McKinlay, Audrius Meskauskas, Aaron M. Renn, Andrew Selkirk, Tom Tromeey, Ronald Veldema, Mark Wielaard, Jon A. Zeppieri, Per Bothner, Archie Cobbs, Andrew Haley, Alexandre Oliva, Casey Marshall, David Gilbert, Lillian Angel, Thomas Fitzsimmons, Jeroen Frijters, Guilhem Lavaux, Robert Schuster, Dalibor Topic, Chris Burdess, Audrius Meskauskas, Sven de Marothy, Kim Ho, ...
- Various Hackers !

# Agenda

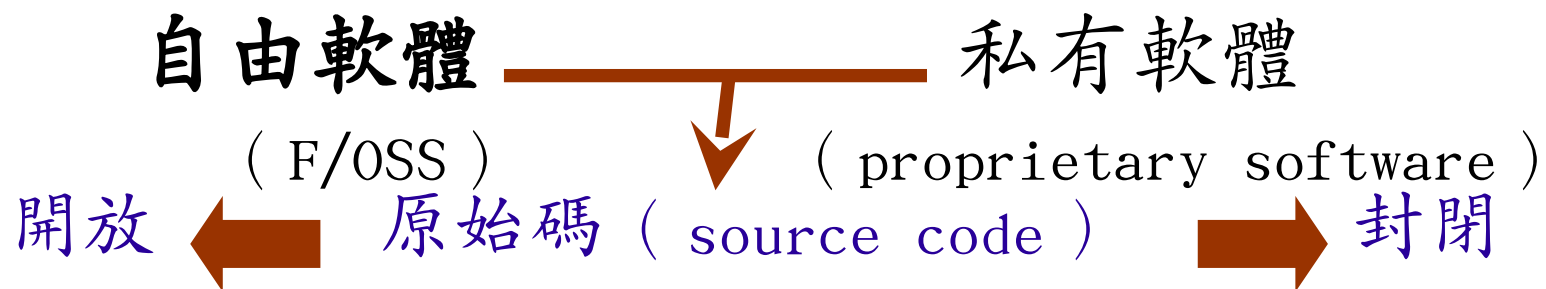
- Why Free Java?
- 目前 Free Java Runtimes 概況
- 如何建構 cleanroom JavaVM
- Case Study
- 展示

# Why Free Java?

- Why Free Java?
  - “Free” 的定義
  - Sun JDK 的授權疑慮
- 目前 Free Java Runtimes 概況
- 如何建構 cleanroom JavaVM ?
- Case Study
- 展示

# “Free” 的定義

- 法國革命家羅蘭夫人在十八世紀嘗言：
  - 「自由、自由，多少罪惡假汝之名以行。」
- 1985 年 Richard Stallman 成立 Free Software Foundation 貫徹四大自由：(copyleft)
  - 執行程式的自由
  - 研究程式的自由
  - 散佈程式的自由
  - 改良程式的自由



# Sun JDK 授權疑慮

- Sun JDK 的授權方式
  - Binary Code License
  - SCSL (Sun Common Source License)
  - JRL (Java Research License)
- 與 OSD (Open Source Definition) 的分野
- Sun Microsystems 逐步釋出善意
  - 2005 年的 **OpenSolaris**
  - Matt Thompson's Talk in Day1
    - Sun Microsystems 的 Open Source 策略
  - SCSL 到 JRL 的轉變



# OSD vs SCSL

	OSD	SCSL
<b>Free Redistribution</b>	Yes	No
<b>Source Code</b>	允許修改 / 再散布	不允許自由再散布
<b>Derived Work</b>	允許也認可 (原始程式碼的整合性)	不允許 (也不允許自由散布個別的修改)
<b>Enhancement</b>	鼓勵	有限度的研究、商業使用，或者個人名義使用
<b>Forking</b>	授權不得限制其他軟體，也不得有排他性	一旦同意 SCSL 授權，就不得散布不相容的軟體 (如果有違反 TCK 測試項目疑慮時，無法散布 Kaffe、GCJ，甚至是 Xerces 等軟體)

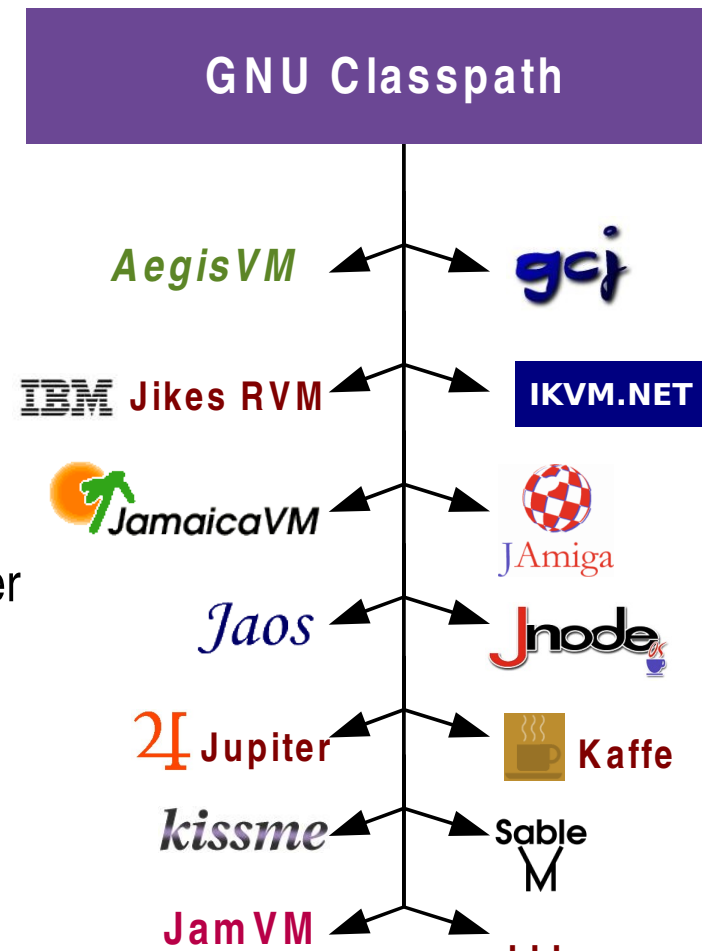
# 目前 Free Java Runtimes 概況

- Why Free Java?
- 目前 Free Java Runtimes 概況
  - GNU Classpath 簡介
  - 發展近況與相容度
  - 活躍專案介紹
- 如何建構 cleanroom JavaVM ?
- Case Study
- 展示



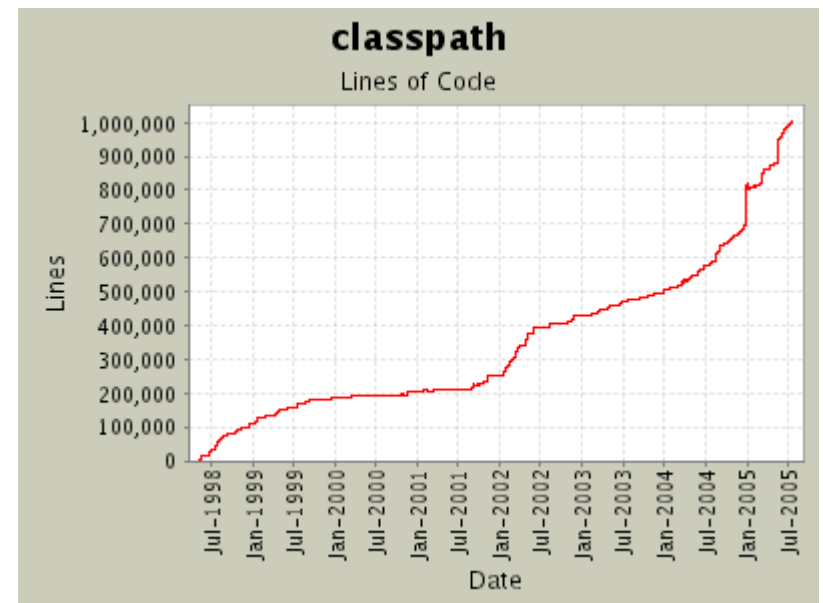
# GNU Classpath 簡介

- 以 GNU Classpath 為發展中樞
  - Java class library 的自由實做
  - 為其他 Free Java Runtimes 的基礎
  - 但不是
    - JDK replacement、virtual machine、compiler
  - 實作 java.\* javax.\* packages (core API and most extensions)
  - 目前實作 J2SE 1.3/1.4/1.5
    - 依據 Sun 的文件實作
    - 沒有 Sun TCK 認證



# GNU Classpath 簡介 (2)

- 1998 年計畫創立
- 全球約三十餘位活躍開發者
- 超過一百萬行程式碼 (Java + C)
- 每兩個月有新的釋出版本
  - 目前 0.17
- 許多合作的專案
- 授權方式
  - **GNU GPL with exceptions**



# Line of Code 的意義

- LoC (Line of Code) 在許多自由軟體專案用以描述軟體規模
- Edsger W. Dijkstra 在 〈 Two views of Programming 〉 提到：

When a very well-known and widely respected computer scientist recently used [lines of code as a measure of] programmer productivity in a lecture, the suggestion came from the audience that, instead of talking about “the lines of code produced” we should talk about “the lines of code used” and that, therefore, the speaker was booking them on the wrong side of the ledger. The speaker answered that he stuck to his productivity measure, because he did not know of any alternatives that allowed proper quantification!

\* 咱們看看 "the line of code used" in GNU Classpath

# Free Java Runtimes

*AegisVM*

*Jaos*

 Jupiter

*gcj*

**IBM** Jikes RVM

*kissme*

**JamVM**



JAmiga

 **JamaicaVM**



Kaffe

Sable  
M

**Jnode** 

 **MAUVE**

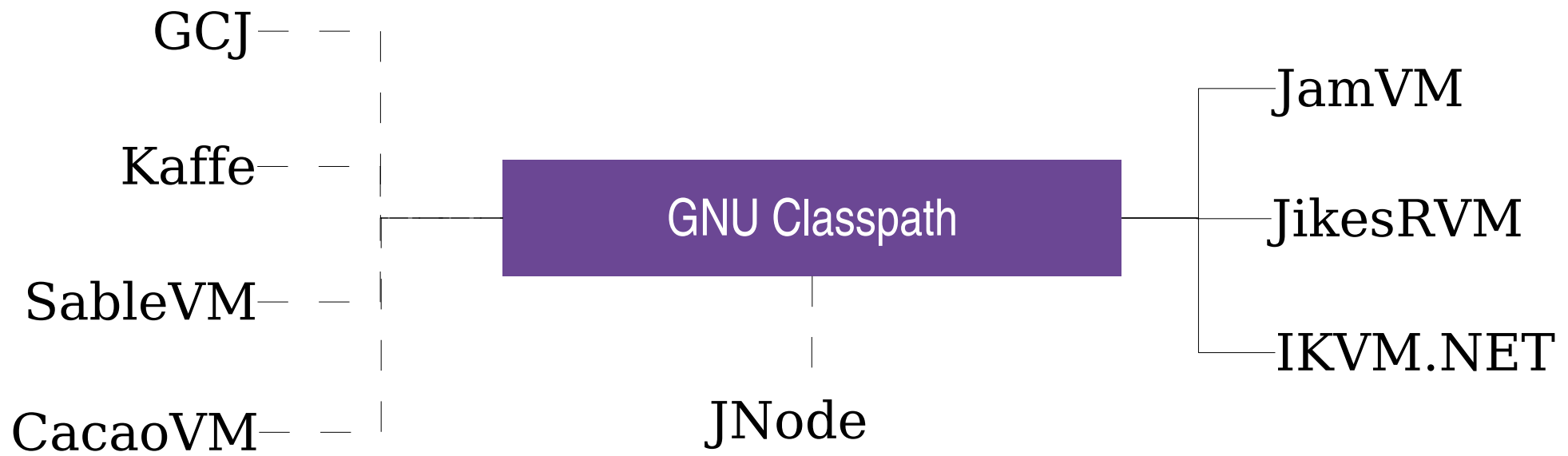
<http://sources.redhat.com/mauve/>

**IKVM.NET**

 **Apache Harmony**  
The Apache Software Foundation  
<http://www.apache.org/>

Apache  
**Gump** 

# GNU Classpath 發展想法



# 相容度

- 涵蓋
  - 78% 1.5 Java API
  - 87% 1.4/1.3 Java API
  - 89% 1.2 Java API
  - 99% 1.1 Java API
- 相當成熟的實做
  - java.{ lang | math | io | nio | rmi | security | util }
- 積極發展中
  - AWT peers (GTK+ / Qt4)
  - Swing
  - CORBA

# Java 5 support

- *generics-branch* 在 2004 年秋季建立
- 支援 JDK 1.5 所引入的新語言特徵
- new APIs
- 里程碑
  - 2004 年十二月雛型完成
  - 2005 年五月份首度運作於 JamVM 1.3.0
- compilers
  - ECJ (compiler of Eclipse 3.1)
  - GCJX (實驗性的 GCJ Frontend)

# 活躍專案介紹



Kaffe

gcj

IKVM.NET

Jnode 



# Kaffe VM



- 第一個 Free JVM (1996 年一月份) , 1997 年 Transvirtual Technologies 提供商業服務
- 能在五十個平台上運作 (“NetBSD of JVMs”)
- 自 2002 年以來就開始與 GNU Classpath 整合
- 豐富的 API 實做
  - JacORB, Tritonus, GNU Crypto, ...
- 可作為完整的 JDK 替換

# GCJ



- GNU Compiler for Java
  - part of the GCC
- compiler and runtime environment (GLJ)
- Ahead-Of-Time compiler
  - {source- and bytecode} -> machine language
- CNI (Compiled Native Interface)
  - C++  $\infty$  Java

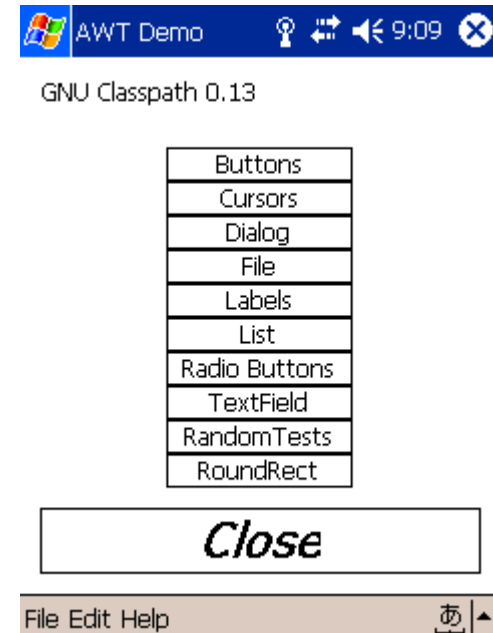
- 建構於 .Net 的 Java environment
  - 同時運作於 Mono 與 Microsoft .NET
- bytecode to CIL translation
  - Bytecode => parse trees => CIL instructions
  - dynamic interpretation at runtime
  - compiler (Jar -> .NET Assembly)



- 以 Java 撰寫的作業系統
- 只有一小部份的 x86/IA64 Assembly 與 Java ，並提供 Java 撰寫的 JIT-compiler
- 提供以 Java 撰寫的 Disk controller driver 、 Graphics driver 、 Keyboard driver 、 NIC driver 、 Input device driver ，以及 USB driver

# Mysaifu JVM

- 以 GNU Classpath 為基礎的 JVM ，運作於 Windows Mobile 2003 (PocketPC 2003)
- 作為一個 Java SE 的實現
- 支援 AWT/Swing



# 如何建構 cleanroom JavaVM ？

- Why Free Java?
- 目前 Free Java Runtimes 概況
- 如何建構 cleanroom JavaVM ？
  - "cleanroom" 定義
  - 如何進行相容性測試
  - VM 概況
- Case Study
- 展示

# "Cleanroom" 定義

## ■ Cleanroom

- 讓開發者只由公開文件或程式庫的文件中，獲悉規格，但過程中不得涉及標準實做的原始程式，之後再重新實做，這樣確保相容性與合法性。
- BSD 與 AT&T 的 UNIX 訴訟就是以 cleanroom 手法擺脫著作權

- 1996 年 Tim Wilkinson 以 BSD-like license 釋出 Kaffe ，這是第一個 Free Java / Cleanroom 實做。

# 如何進行相容性測試？

- Why not Sun TCK (Technology Compatibility Kit)?

*ATTACHMENT E TECHNOLOGY COMPATIBILITY KIT, 1. TCK License.:*

*b) TCK Use Restrictions. You are not authorized to create derivative works of the TCK or use the TCK to test any implementation of the Specification that is not Covered Code. **You may not publish your test results** or make claims of comparative compatibility with respect to other implementations of the Specification. In consideration for the license grant in Section 1.a above **you agree not to develop your own tests** which are intended to validate conformation with the Specification.*

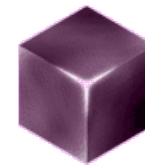
- 一旦同意 TCK License...
  - 不得公開測試結果
  - 不得發展額外的 test cases
  - 不得發展造成與規格不相容的 code / extensions



# 如何進行相容性測試？ (2)

## ■ Mauve

- free test suite (GNU GPL)
- 超過 175000 個測試案例
- 已可與 IBM Jikes 計畫的 Jacks compiler suite 交互運作

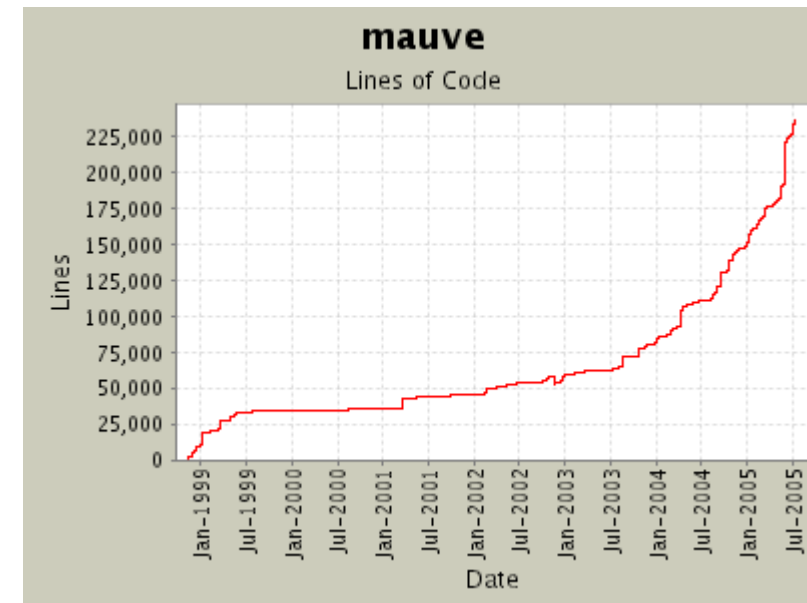


MAUVE

<http://sources.redhat.com/mauve/>

## ■ Apache Gump

- 整合性測試
- 可自動載入軟體測試項目、編譯，然後進行測試
- 約有 800 個套件 (Jakarta 計畫)



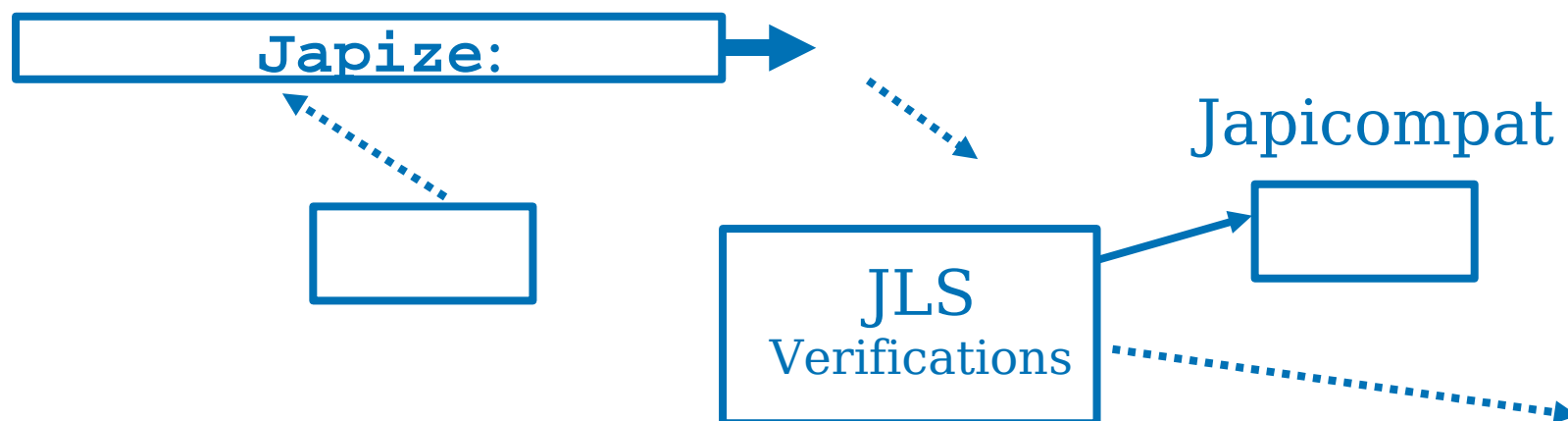
# 如何進行相容性測試？ (3)

## ■ japitools

- 驗證 binary compatibility

- 運作模式

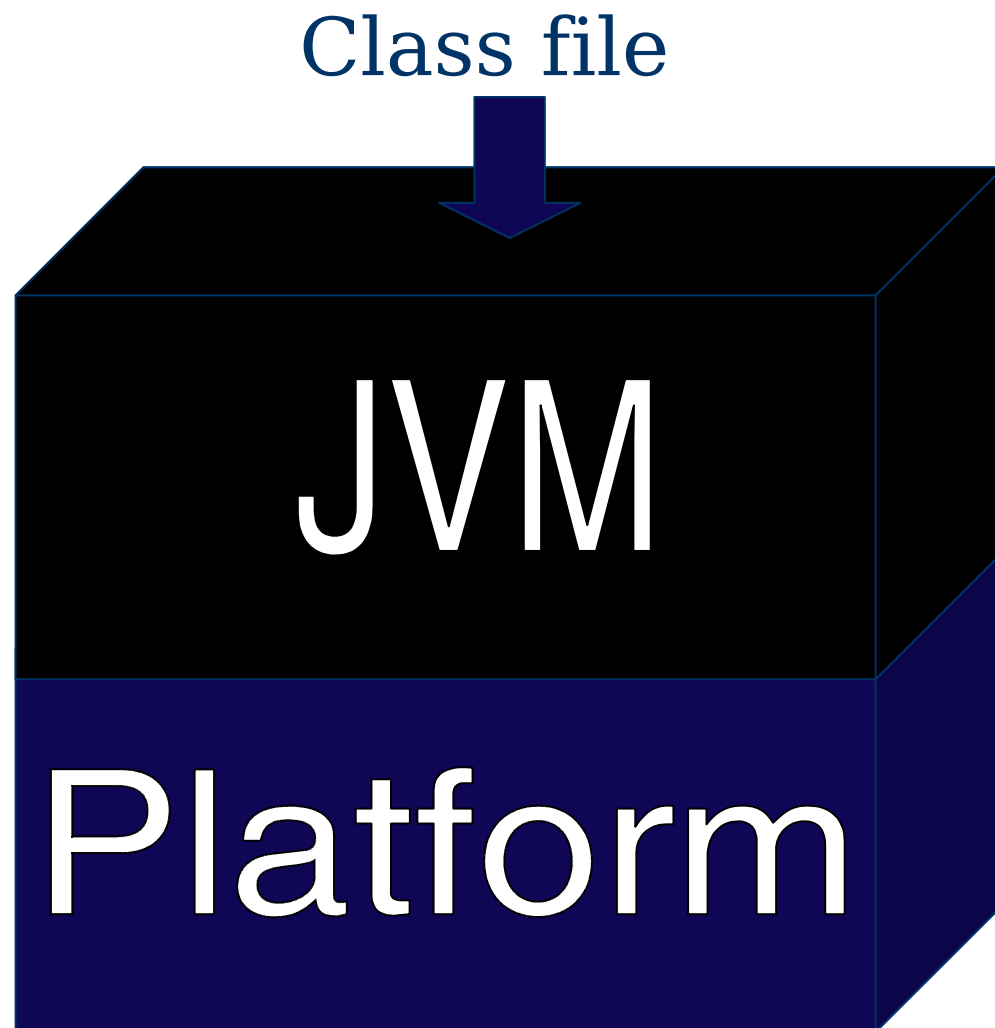
- 以 Java 撰寫的 Japize 發出一系列 API Listing 的要求
- 另一隻程式 Japicompat 稍後以 JLS (Java Language Specification) 規範的行為與該回應比較



# JavaVM 概況

- JVM 架構
  - 外在觀點
  - 內在觀點
  - 動態觀點
- Execution Engine
- Interpreted code & JIT compiler

# JVM 架構：外在觀點



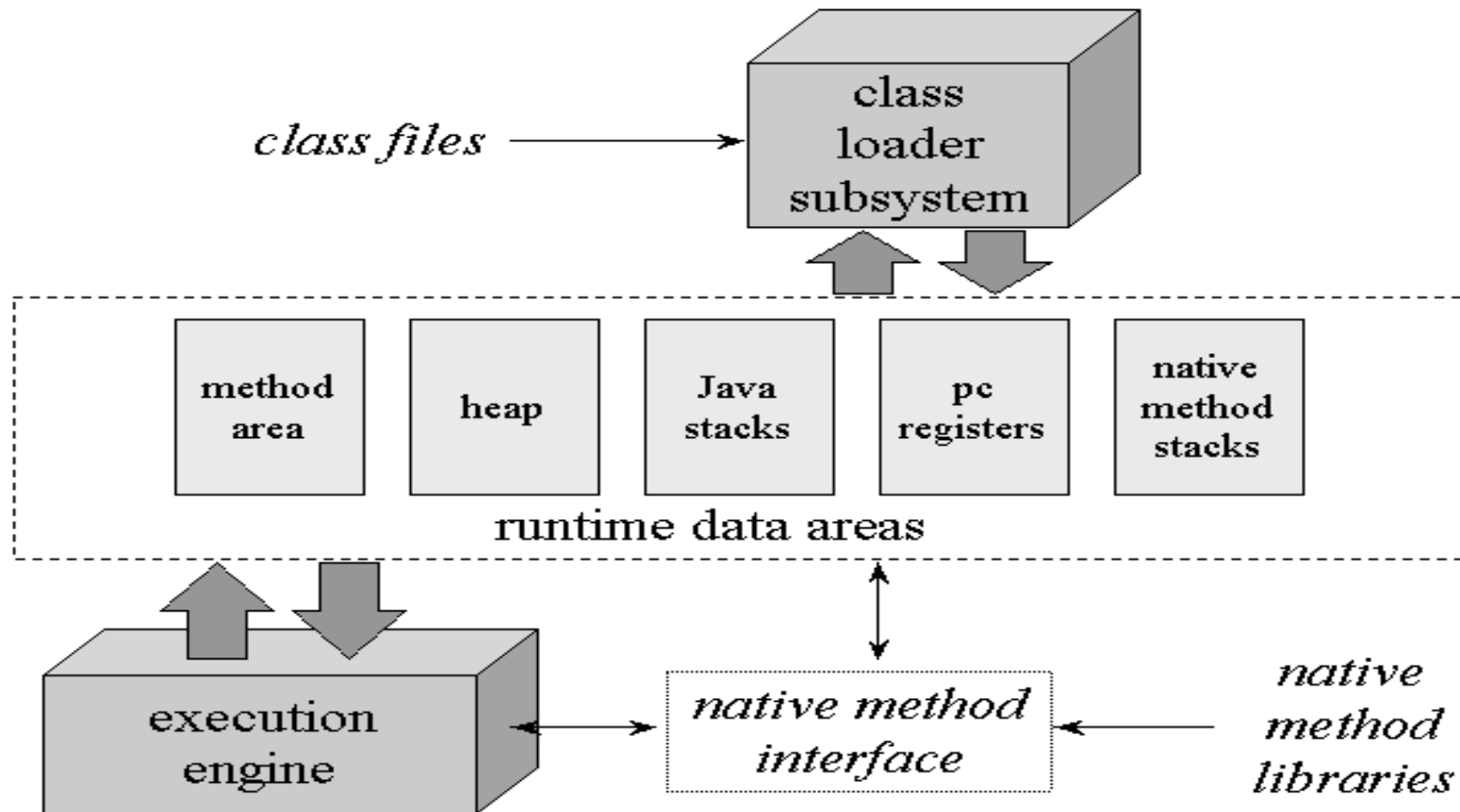
```
$ javac Greetings.java  
產生 Greetings.class 檔
```

```
$ java Greetings "Hello Class"  
呼喚 JVM 實做
```

Hello Class

```
$  
作業系統顯示訊息到終端機
```

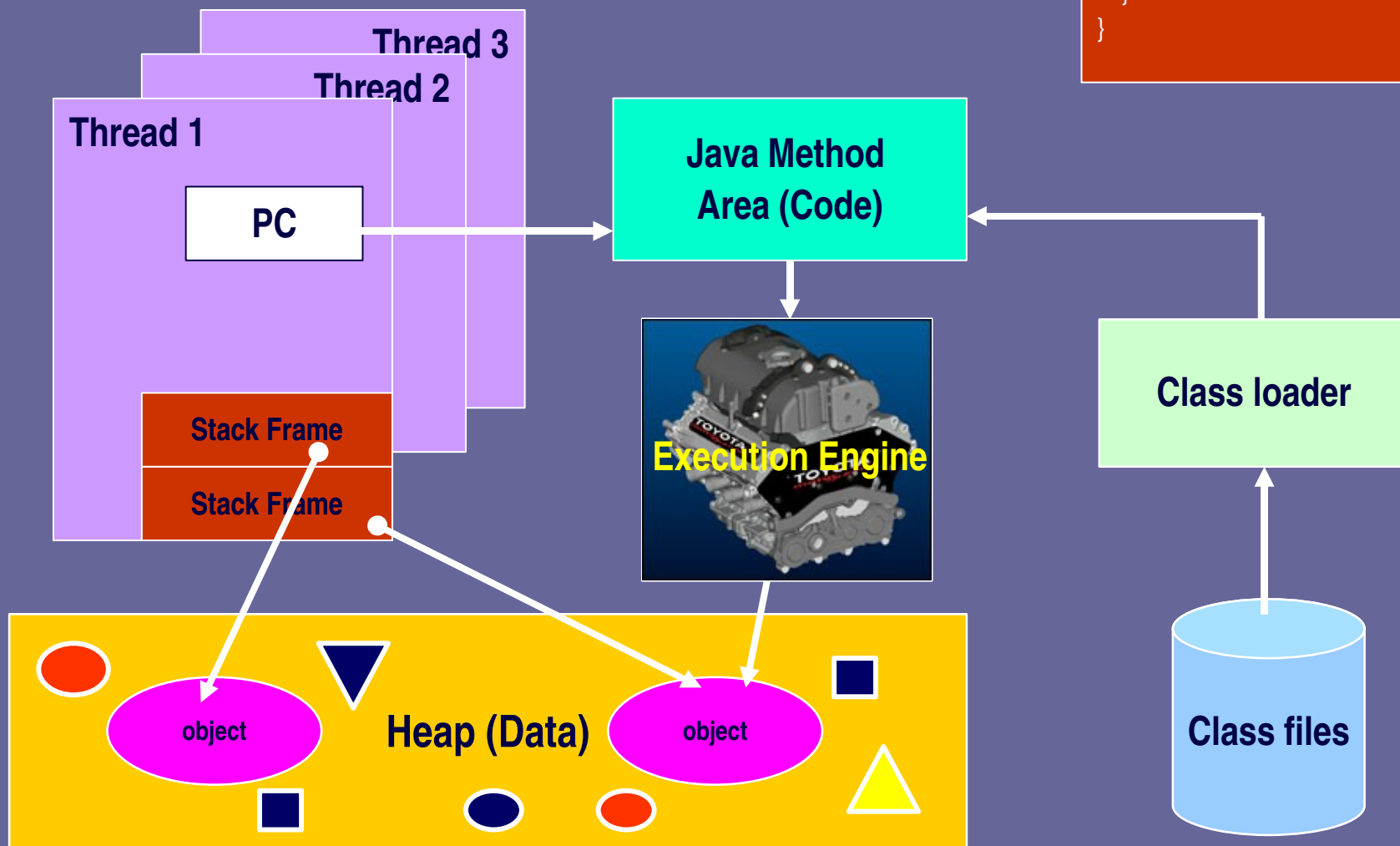
# JVM 架構：內在結構



# JVM 架構：動態觀點

```
A Multithreaded Java Program
public class ProducerConsumerTest {
    public static void main(String[] args) {
        CubbyHole c = new CubbyHole();
        Producer p1 = new Producer(c, 1);
        Consumer c1 = new Consumer(c, 1);

        p1.start();
        c1.start();
    }
}
```



# 執行引擎

- 執行引擎就是 Java VM 的核心
- 邏輯上切割“frame”以保存 *local variables* 與 *operand stack* (編譯時期就決定型態)
- 指令集
  - **Load/store** between locals and operand stack
  - **Arithmetic** on operand stack
  - **Object creation** and **method invocation**
  - **Array/field** accesses
  - Control transfers and exceptions
- 執行引擎可被視為以下三種層面：
  - Abstract specification
  - Concrete implementation
  - Run-time instance

# Segment

javaframe



**optop**

**method**

**Class fields**

**pc**

others

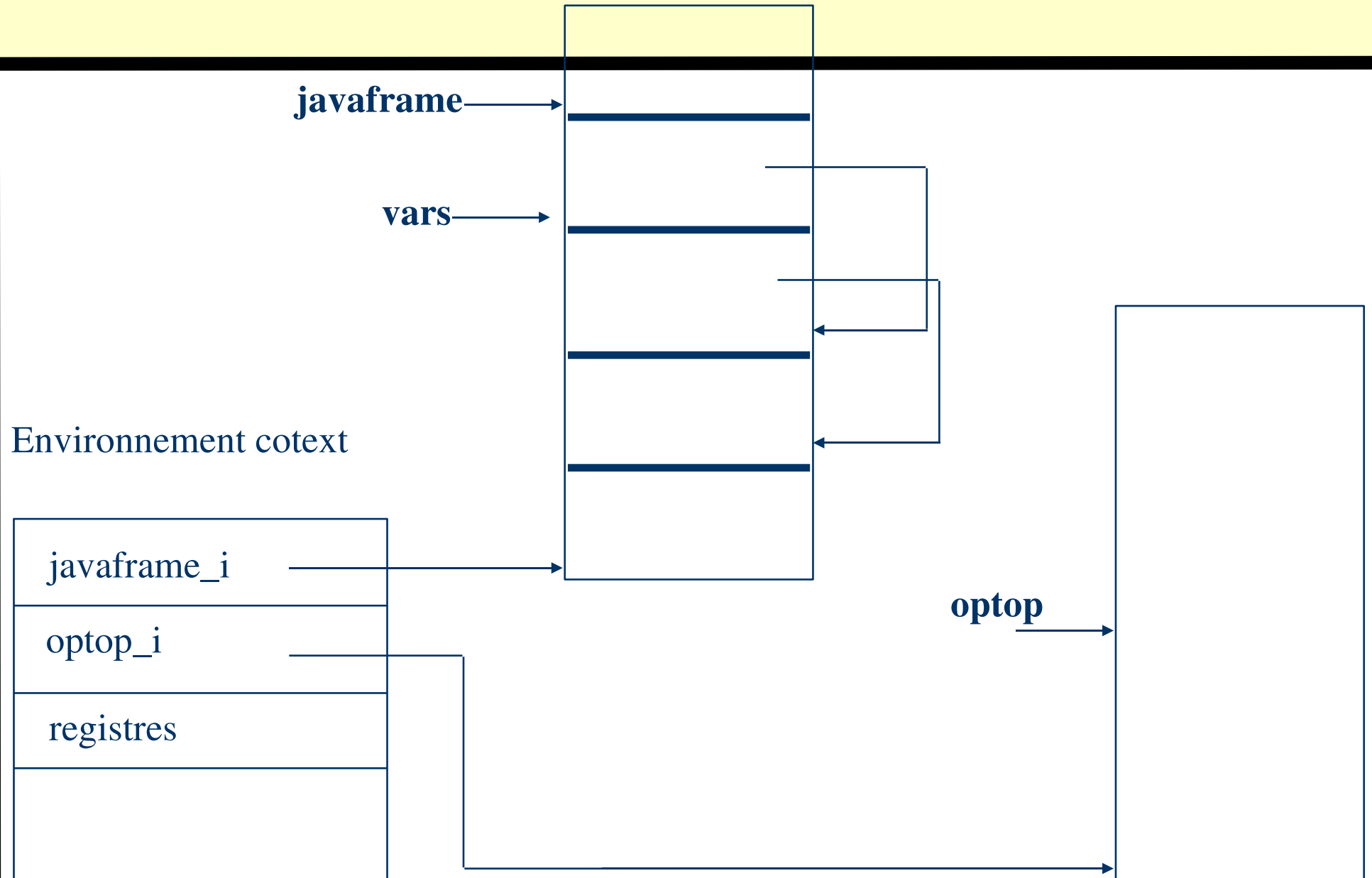
{ *Variables locales* }

vars



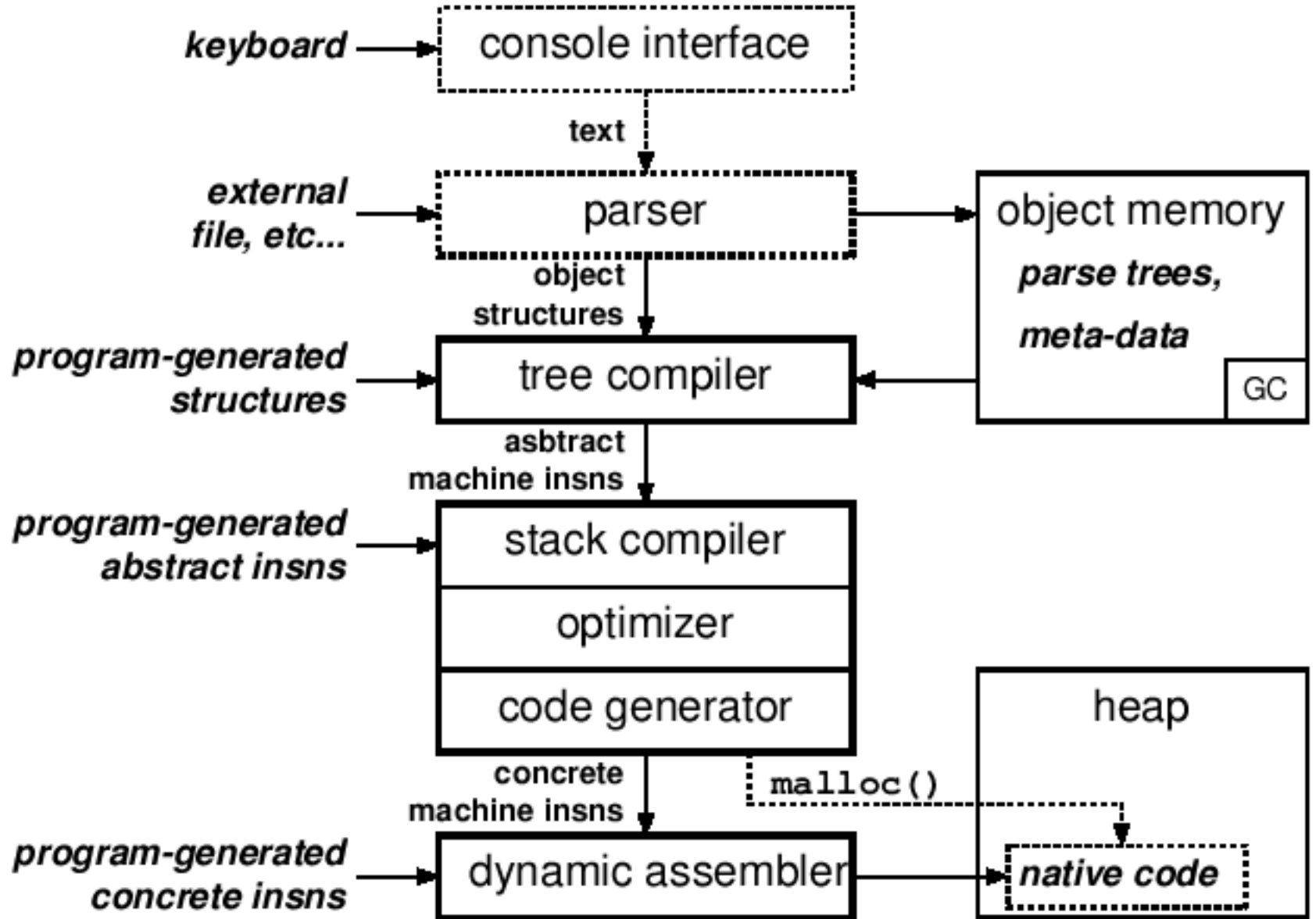


# Environnement Execution



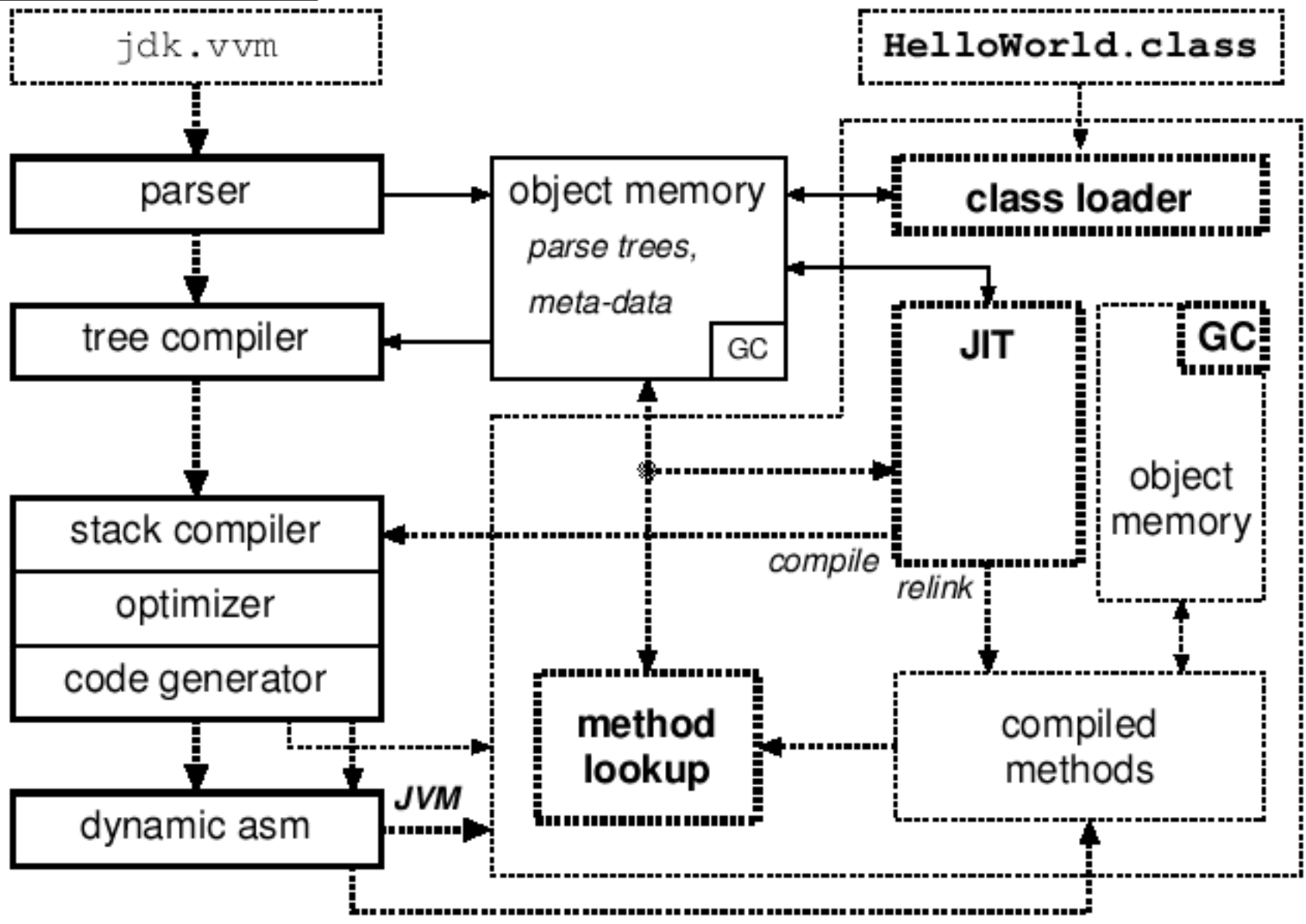
# 一般化的 VM / Dynamic compiler 設計

## architecture



# JVM 特化的結構，強化 ..

- .ClassLoader
- JIT feedback
- Method Lookup

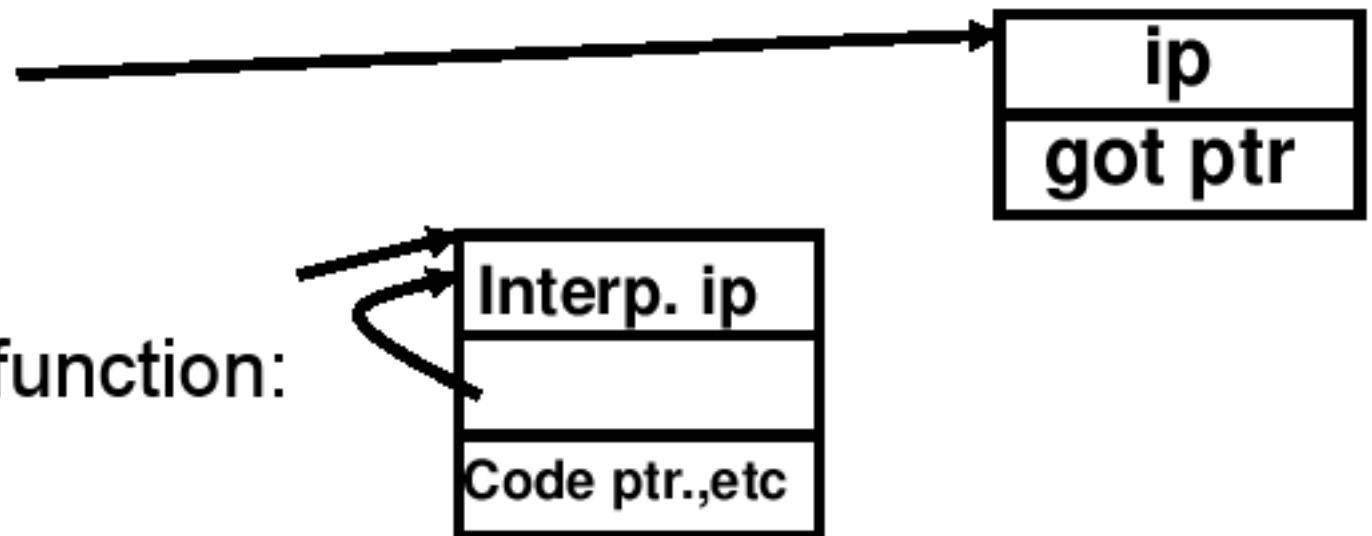


# Interpreted Code 展現

- 考慮到 JIT(Just-In-Time) Compiler 整合的議題
- GCJ 與 Kaffe-GCJ bridge 將已經 interpreted code 視同 compiled C functions，並透過 libffi (GNU GCC 的一部分) 處理平台相關的 calling conventions。
- 邏輯上來看：

– C function:

– Interpreted function:



# Case Study

- Why Free Java?
- 目前 Free Java Runtimes 概況
- 如何建構 cleanroom JavaVM ?
- Case Study
  - Kaffe
  - GCJ (GNU Compiler for Java)
- 展示

# Kaffe 概況

- GNU GPL 授權，為許多研究計畫採用
- 眾多平台移植
- 眾多分支
  - KaffeOS
  - JESSICA2
  - LaTTe
- 多種 AWT backend
- 成功案例
- KOE (Kaffe-based Operating Environment)

# Kaffe 多平台移植

## Processors

- X86
  - IA32
  - IA64
- ARM
  - StrongARM
  - Xscale
- MIPS
- PowerPC

## Operating Systems **GNU/Linux, GNU/Hurd** BSDs

**FreeBSD, NetBSD, OpenBSD, Darwin/Mac OS X**  
Proprietary Unixes  
**NeXTStep, Solaris, SunOS, IRIX, AIX, etc.**  
Microsoft OSes  
**Windows/Cygwin, Windows CE (KaffeCE), DOS**  
Embedded OSes / RTOSes:  
**eCOS, VxWorks, pSOS, ThreadX, SMX, Nucleus, RTEMS**  
Research OSes: **KaffeOS / OSKit / L4**  
Other OSes  
**AtheOS, BeOS, AmigaOS, MiNT, Plan9**

## Processors

- Sparc
- m68k
- S390
- Alpha
- Pseudo Platform

# Kaffe 支援多種 AWT backends

- 主要的 AWT backends ( 可動態切換 )
  - Xlib
    - I18n Ready
  - Qt
    - Qt/X11 2.x/3.x
    - Qt/Embedded 2.x/3.x 與 Qtopia
    - Qt4
  - GTK+ 2 based
  - Nano-X / MicroWindows
- Win32
- BeOS (BeKaffe)
- DirectFB
- VNC/SDL/pure X11/Framebuffer (Odonata)

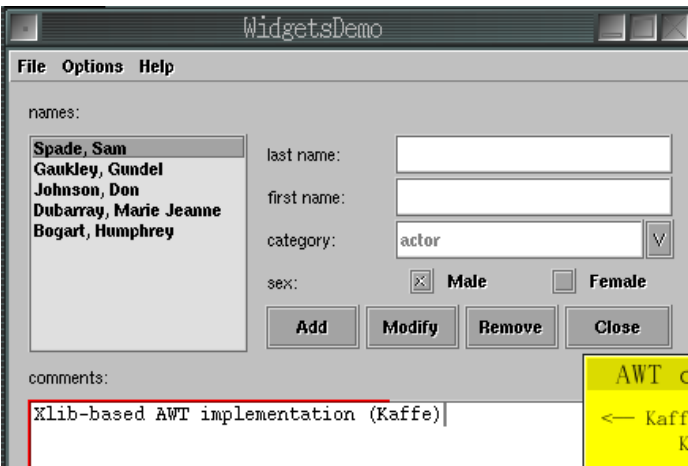


# Kaffe 的 peerful & peerless AWT

- Peerful
  - GTK+ 2.x (GNU Classpath)
  - Qt4 (發展中)
- Peerless
  - 取自 Swing 的 “pluggable L&F” 的概念
  - 支援 PersonalJava
  - 直接在 peerless AWT 實作
    - 高效率
    - 節省資源
    - 眾多變化



# Kaffe 主要的 AWT 實作



AWT comparison

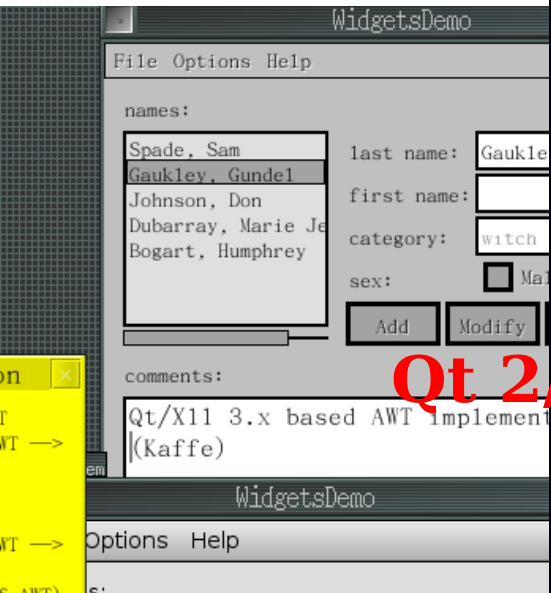
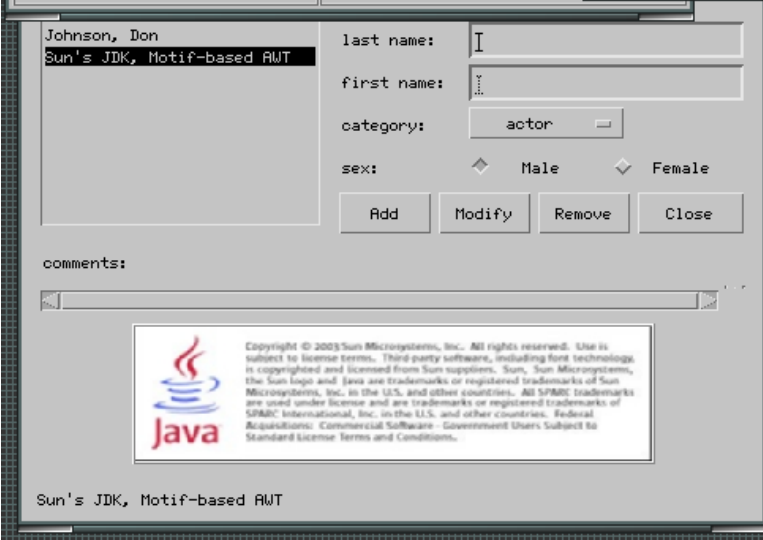
← Kaffe's Xlib AWT  
Kaffe's Qt AWT →

GTK+ 2.x based AWT →

└─ Sun JDK (Motif AWT)

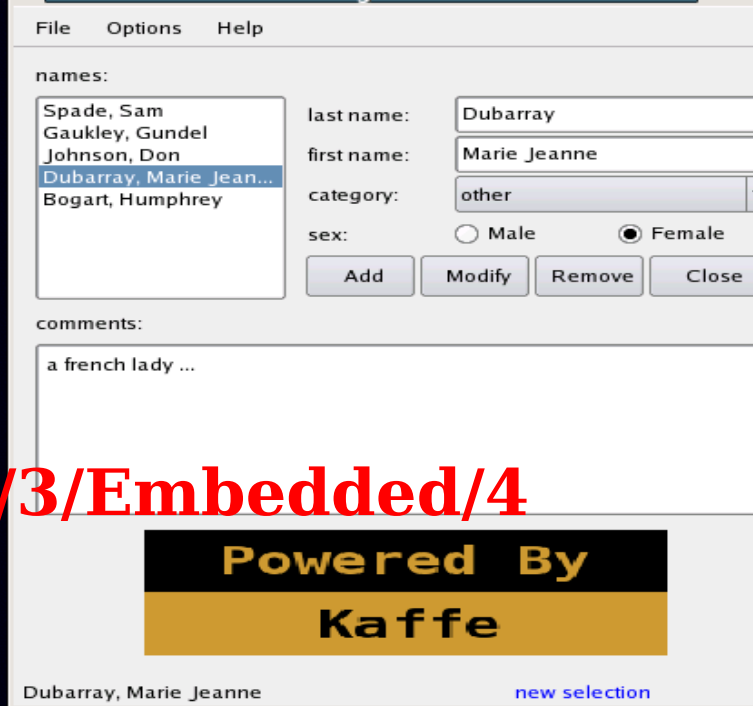
Powered By  
Kaffe

Xlib



Qt 2/3/Embedded/4

Powered By  
Kaffe



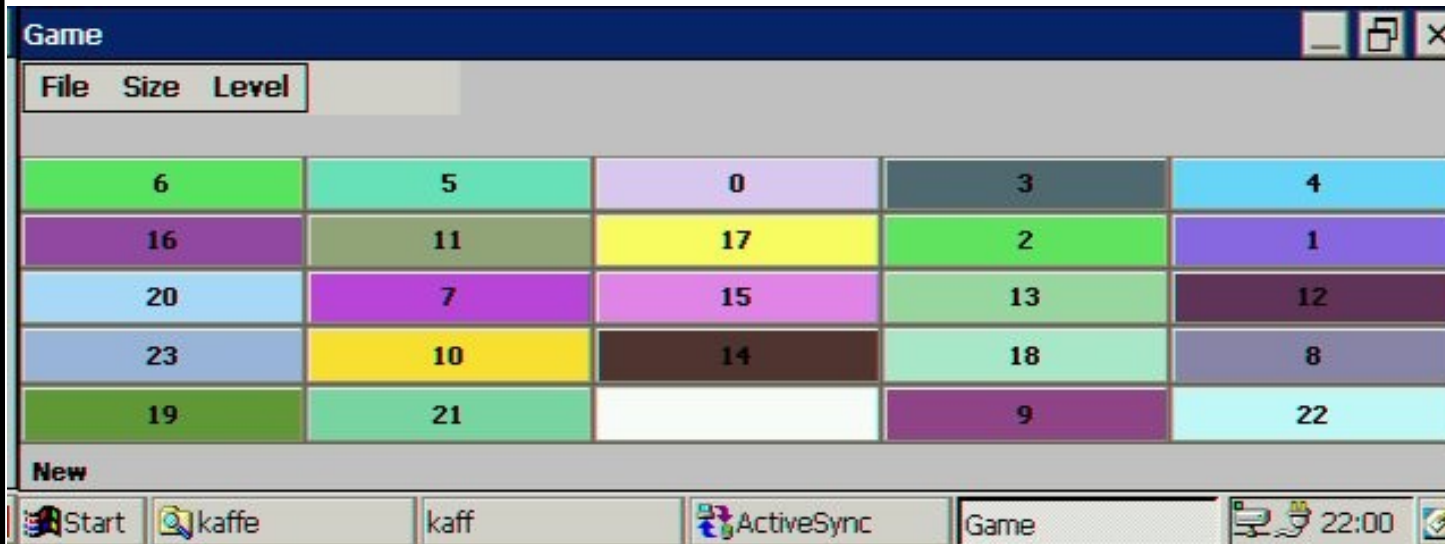
GTK+



Microwindows  
/Nano-X

# KaffeCE

- KaffeCE 以 Kaffe 1.0.5 為基礎
  - 支援 Win32 與 WinCE
  - PersonalJava



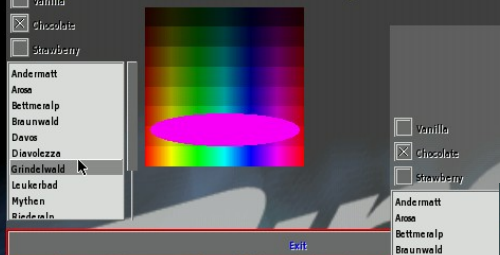
# kawt : DirectFB AWT backend

Move the mouse over a window to activate it.

Press left mouse button and drag to move the window.

Press middle mouse button to raise/lower the window.

Press right mouse button when you are done.



extensions for transparency and other completely implemented and the core We dropped its development after starting from scratch with DirectFB and MHP in mind from The kAWT, however, needs some code cleanup, es Toolkit stuff which abstracted the d X11, Win32 etc.

If anybody is interested in testing, cleaning optimizing, feel free to do so.

You need 'jikes' and 'ant' to build it.

After installation you can test it by starting

```
kaffe -classpath 'kawt-classpath':/usr
rdparty/log4j.jar test.awt.AwtTest
dok@skunk:~/cvs/directfb/kawt$ kaffe -classpat
e/kaffe/Klassen.jar:thirdparty/log4j.jar test.
log4j:ERROR No appenders could be found for ca
log4j:ERROR Please initialize the log4j system
(*) parsing config file '/etc/directfbrc'.
(*) parsing config file '/home/dok/.directfbrc
```

----- DirectFB v0.9.14  
(c) 2000-2002 convergence integr  
(c) 2002 convergence GmbH

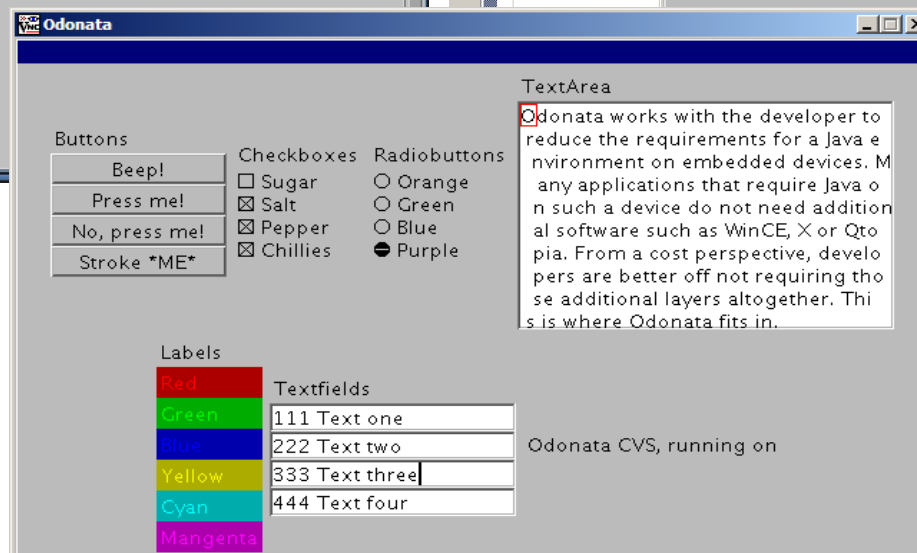
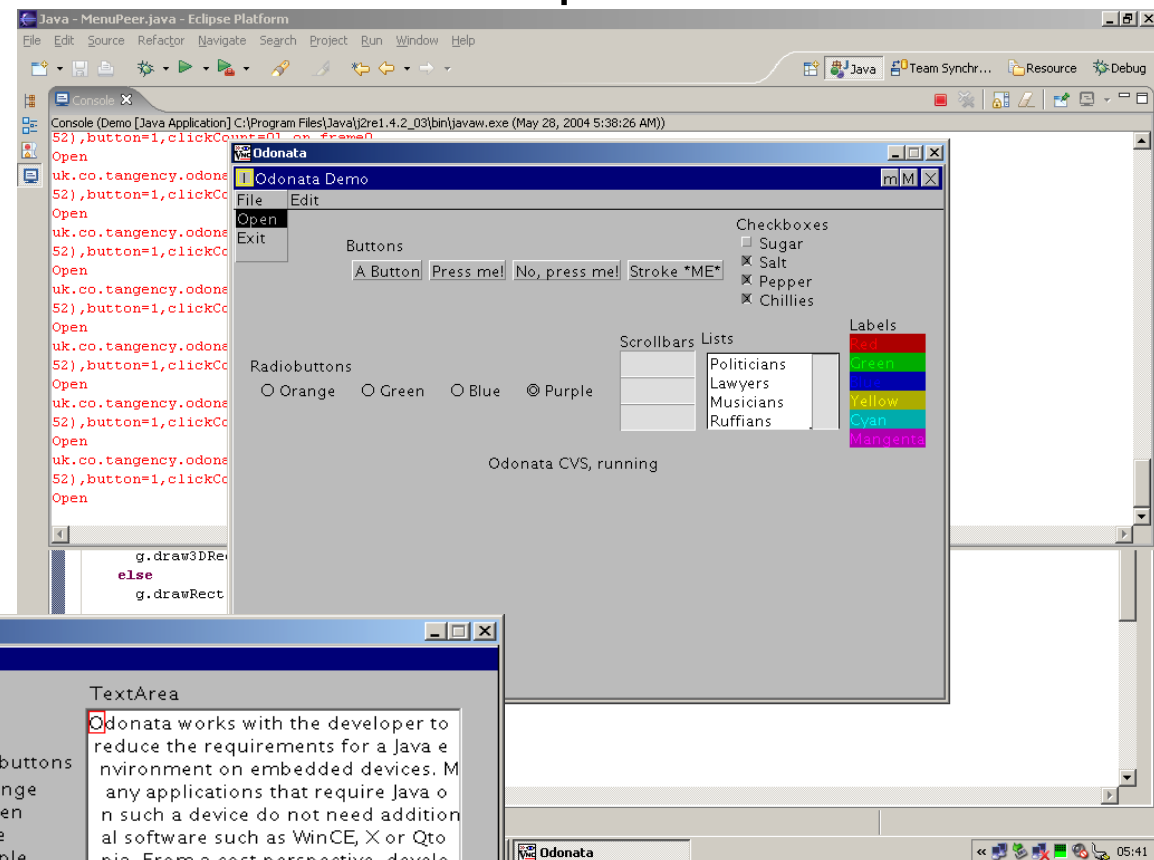
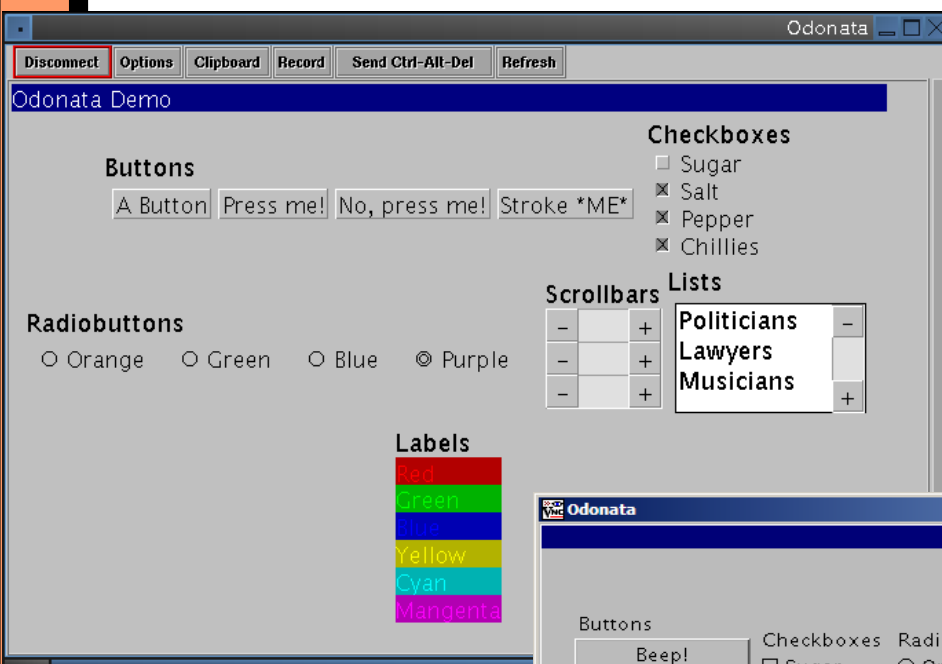
(\*) Multi Application Core. (with MMX support)  
(\*) DirectFB/misc/memcpy: using MMXEXT optimiz  
(\*) DirectFB/Interface: Loaded 'FT2' implement





# Odonata

- Lightweight AWT Toolkit provider with VNC/SDL/pure X11/Framebuffer supports

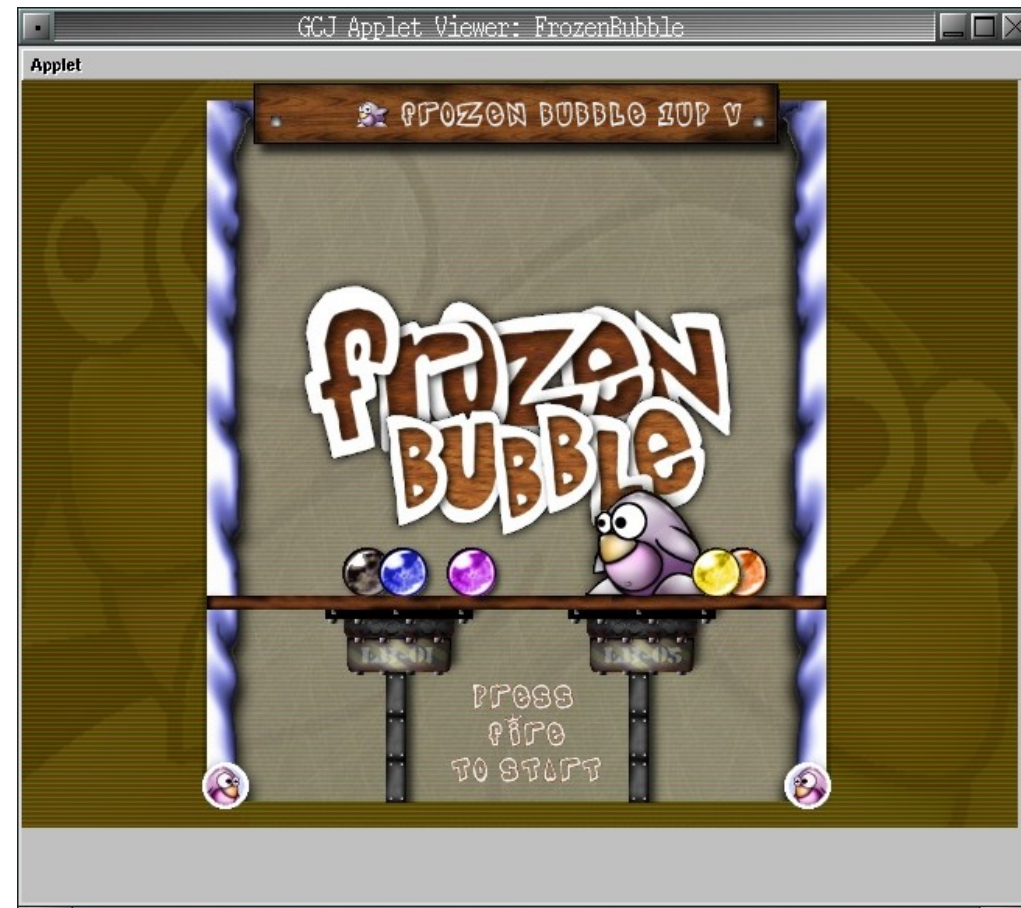


# Kaffe 成功案例

- GCJWebPlugin
- Tomcat/JBoss
- Eclipse
- SwingWT
- OpenGL/SDL
- SMIL/JavaTV
- JMP
- Escher (pure Java X11 Implementation)

# GCJWebPlugin

- GCJWebPlugin 目標是建立 Free/Open-Source Java Browser Plugin 解決方案
- 一開始只支援 GCJ，0.3.0 版之後加入對其他 GNU Classpath 為基礎 JavaVM 實作的支援
  - Kaffe
  - JamVM



# Tomcat 5 + Kaffe

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload  Browse...

Deploy

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/5.0.27	null	Kaffe.org project	Linux	2.6.6-1-686	i686

Copyright © 1999-2003, Apache Software Foundation

Done 1 Adblock



# Eclipse on Kaffe

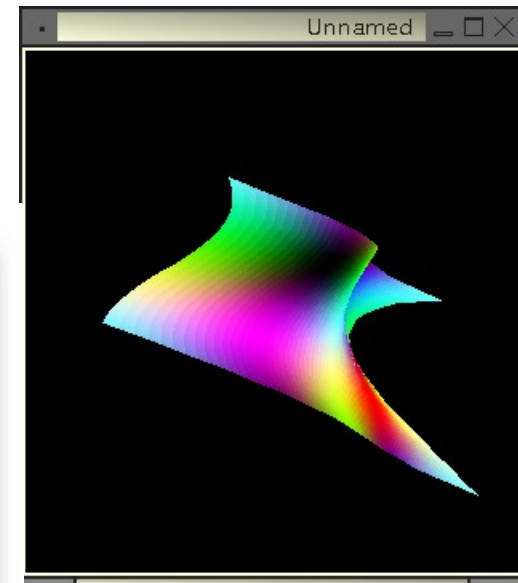
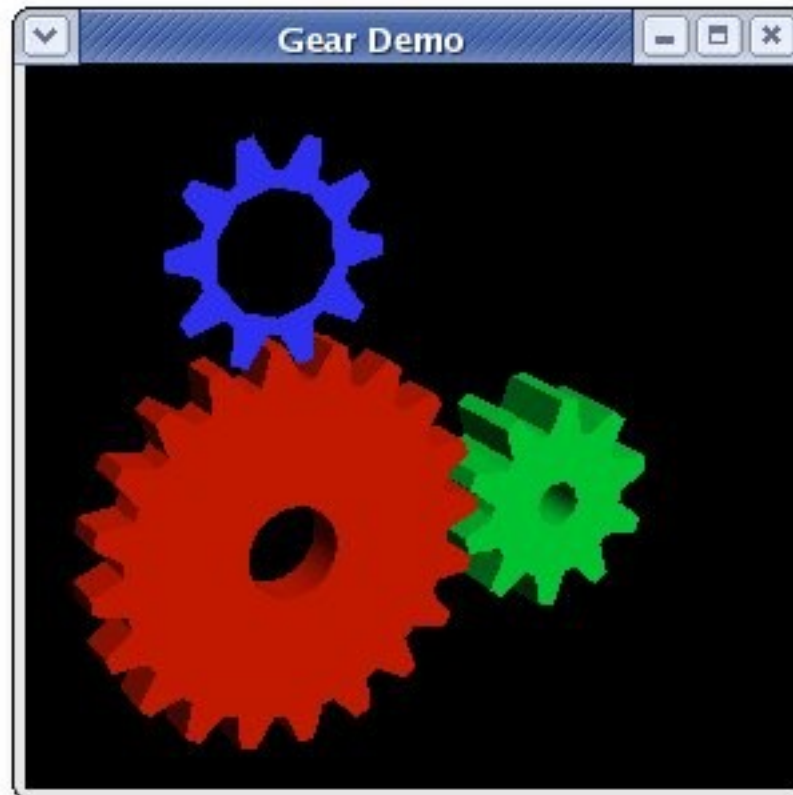
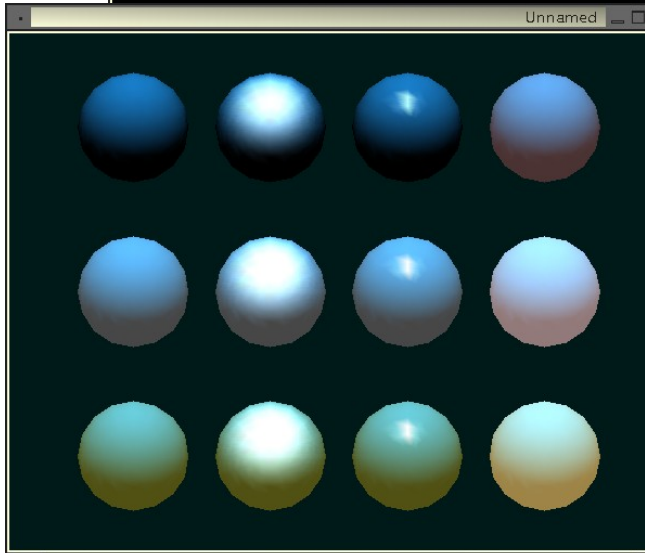
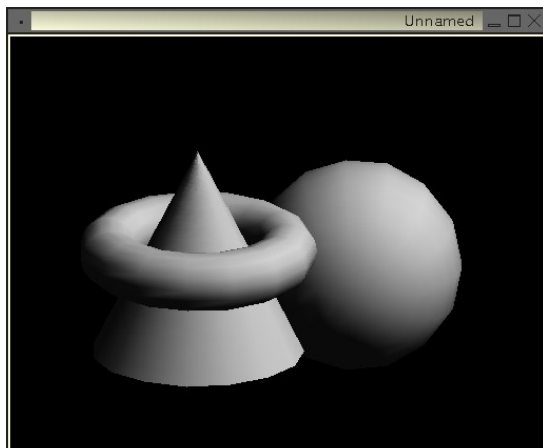
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'Hello' with a package 'org.wildebeest.gcj' containing a class 'World' with a method 'main(String[])'. The main editor displays the source code for 'World.java', which includes a package declaration and a main method that prints the system property 'java.vm.name'. The console at the bottom shows the output 'Hello Kaffe!'. The Preferences dialog is open, showing the 'Installed Java Runtime Environments' section. A table lists two JREs: 'Standard VM gcc34' and 'Standard VM kaffe'. The 'gcc34' entry is selected. The 'kaffe' entry is highlighted with a red box.

```
/*  
 * Created on 3-Nov-03  
 * To change the template for this generated file go to  
 * Window>Preferences>java>Code Generation>Code and Comments  
 */  
package org.wildebeest.gcj;  
  
/**  
 * @author mark  
 *  
 * To change the template for this generated type comment go to  
 * Window>Preferences>java>Code Generation>Code and Comments  
 */  
public class World {  
  
    public static void main(String[] args) {  
        System.out.println("Hello " + System.getProperty("java.vm.name") + "!");  
    }  
}
```

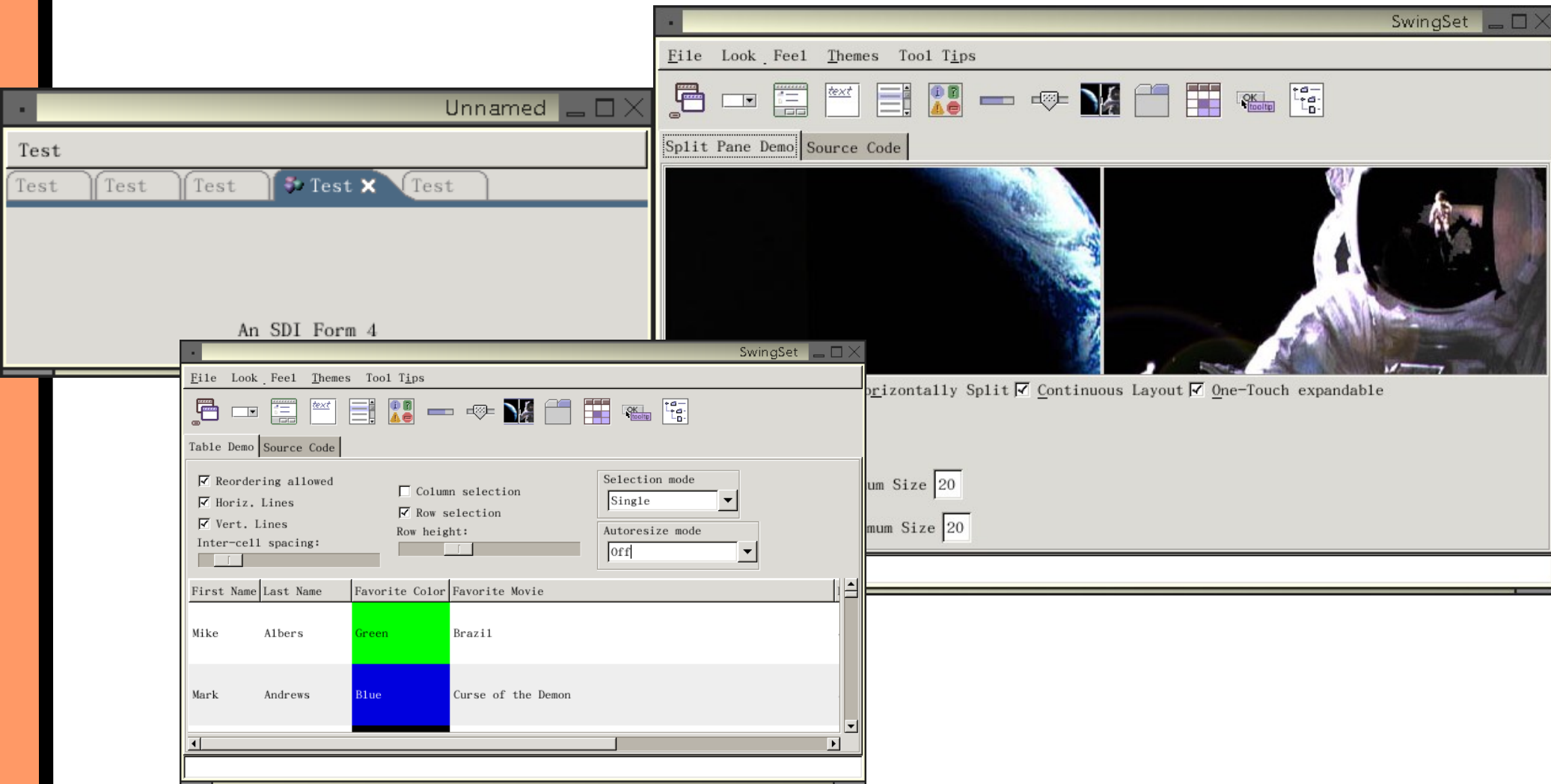
JRE Type	Name	Location	
<input checked="" type="checkbox"/> Standard VM	gcc34	/usr/local/gcc34	Add...
<input type="checkbox"/> Standard VM	kaffe	/usr/local/kaffe	Edit... Remove Search...

# Kaffe 的 OpenGL/SDL 支援

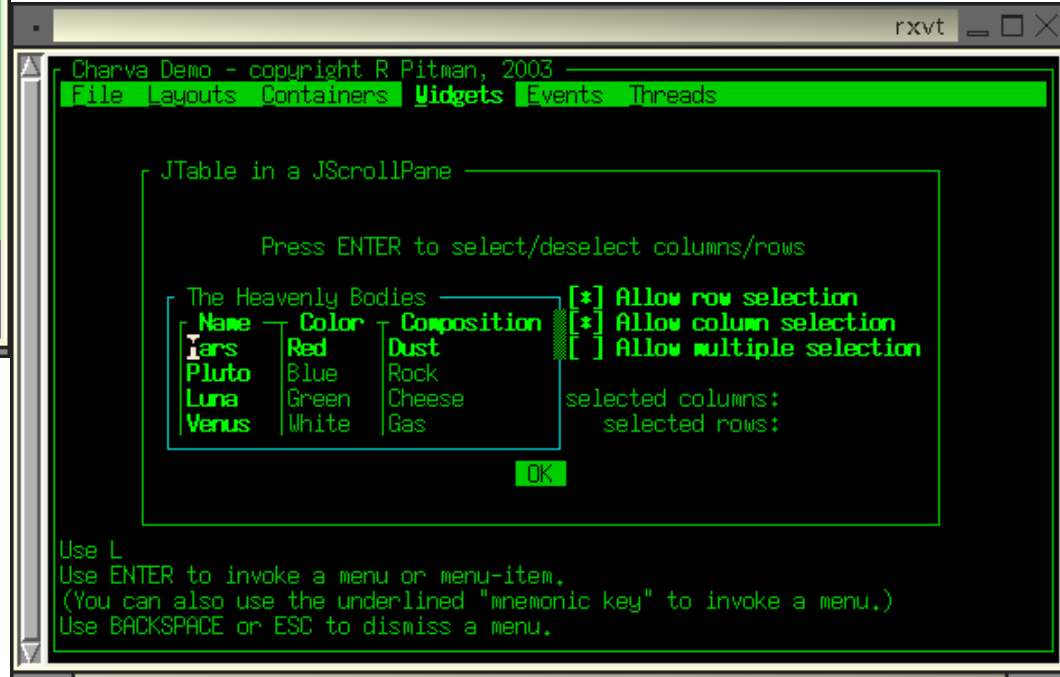
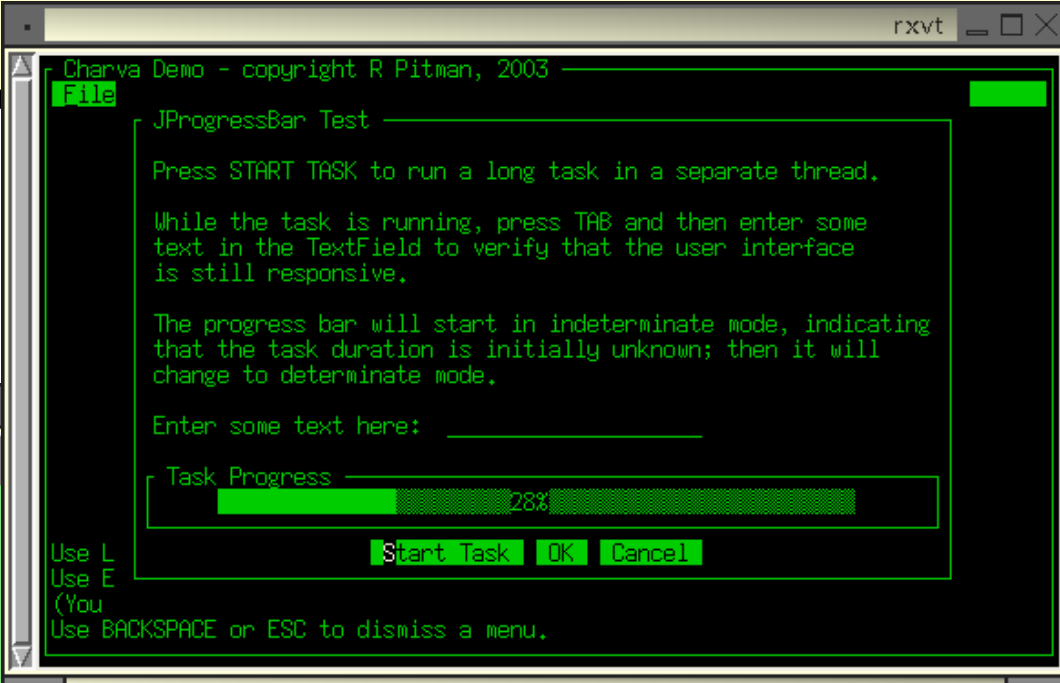
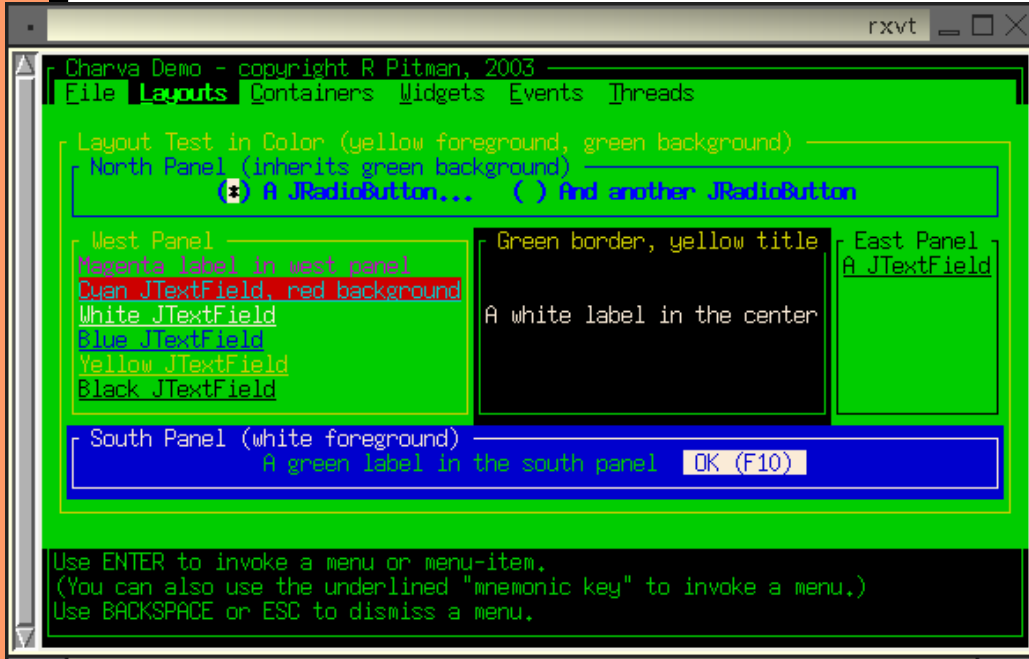
- 2005 年二月份開始支援 jawt (Java Native Interface) ，開始能運作 JoGL (Java on GL) 與 SDLJava 兩項專案



# Kaffe + SwingWT (Swing over SWT)



# Kaffe + Charva





# Kaffe-based MHP (Multimedia Home Platform)



Mon 06.23.03 9:18PM

**GIST** ALL CHANNELS

**4 WNBC** 8:00PM-8:30PM

**FRIENDS**

Ross sets up a blind date for Phoebe, Joey's new girlfriend seems familiar, and Monica worries about Chandler's interest in "shark porn."

Courtney Cox Arquette, Jennifer Aniston, Lisa Kudrow, Matt LeBlanc, David Schwimmer



- Channels
  - Programs
  - Services
  - Configuration
  - Help
- Navigator 1.0

**Program Guide** pe 18.02 03:44:33

**city City-tv**

**YLE24** 7.10.1999 Thu

07.30	South Park: Spontaneous Combustion
08.00	Kummisetä
10.50	Hänen majesteettinsa rouva Brown
12.30	Grand Hotel, Tukholma
13.25	Enter the Eagles
15.05	Hit and Run
16.35	The House of Yes

3:30PM	9:00PM
	CSI: Crime Scene Investigation
bs	Will & Grace
	Law & Order
ny Wife & Kids	According To Jim
the Parkers	One On One
	Global Extremes: Mount Everest
	Half & Half

EMIND ME CLOSE

Welcome to YLE Super TextTV Service ma 28.08 04:53:36

**Russia remembers lost crew** Page 1 of 6

President Putin meets a dead sailor's relative

As Russia observes a national day of mourning for the 118 sailors who died in the Kursk submarine, President Vladimir Putin has unexpectedly left the vessel's home port and returned to Moscow.

On the eve of the commemoration, President Putin received a hostile reception from bereaved families at the northern naval base of Vidyayevo.

Man lowers flag

He had been expected to visit the scene of the accident on Wednesday to lay a wreath on the waves of the Barents Sea where the nuclear submarine sank on 12 August.

The service has reportedly been cancelled at the request of the relatives.

Page 1 of 6

ola. N: Marlon Brando, Al Pacino, päätyntyt gangsteriklasikko to Corleonen väkivaltaisesta (Brando) ja käsikirjoitus (Coppola

net4tv.com

**AOLTV**

[23] Friends - Threes Friends 3

Anj33: hated the bears since Dikka...lol

DexDavenport: anyone heard of AOLTV?

Anj33: yes dex

OnlineHost: FLE123 has left the room.

OnlineHost: Mosiea has entered the room.

Char80465: I think we are bigger woobie

DexDavenport: do you have it?

Wooblescoo: woobie is boring woobie to death

Sweetfemale54711: lala!alalala

IM Get Profile Ignore

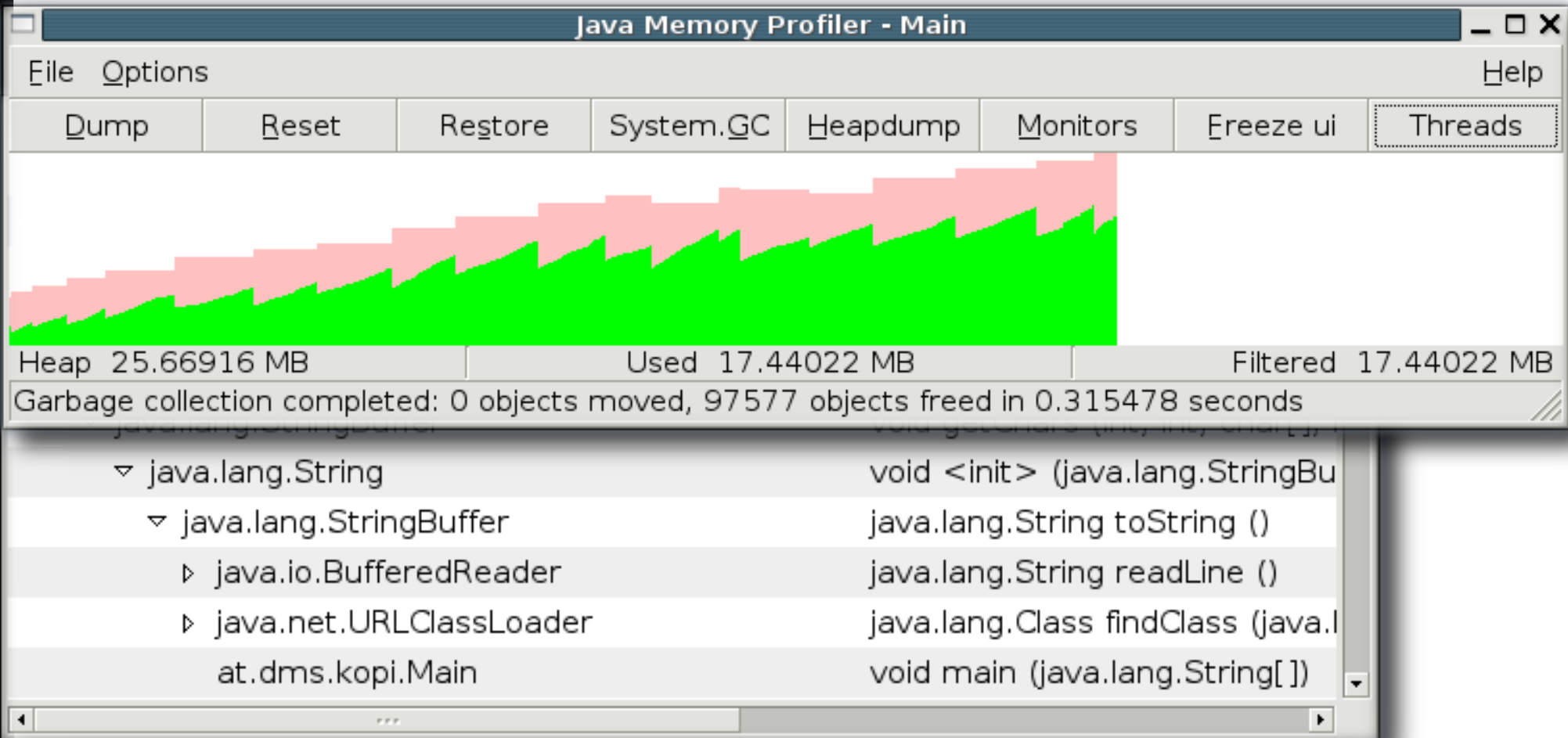
DRDABBYGIRL Sweetfemale54711 R burg9027 Mosiea

Send

Buddy List Find A Chat Exit Room

# JMP

- JMP 使用 JVMPI (Java VM Profiler Interface) 來追蹤 Object、Method，以及 Call-Graph 使用量與狀態，並提供 GTK+ 輸出的介面

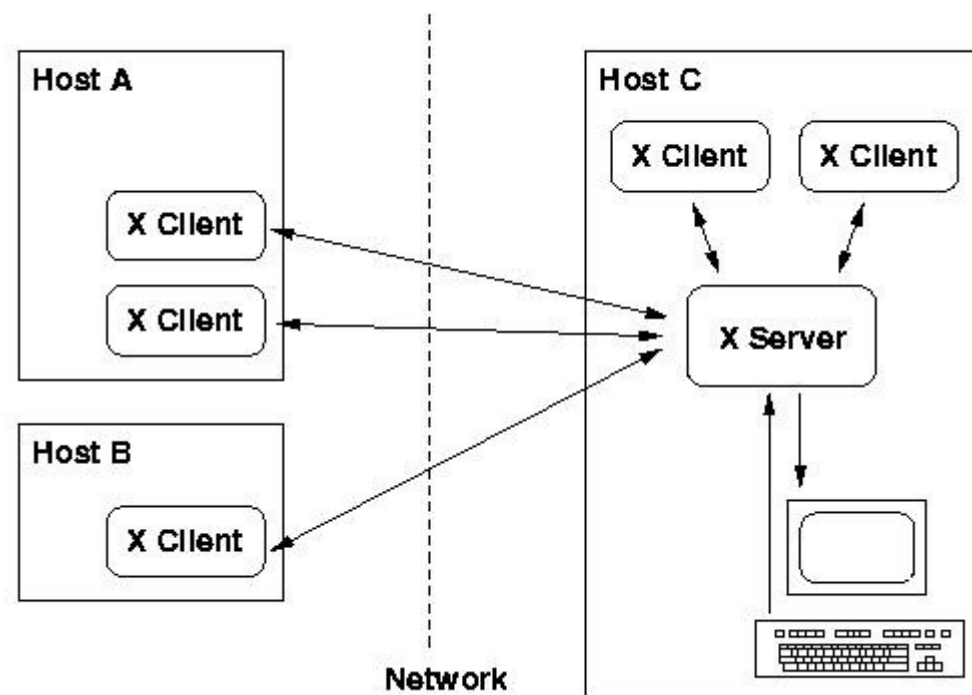
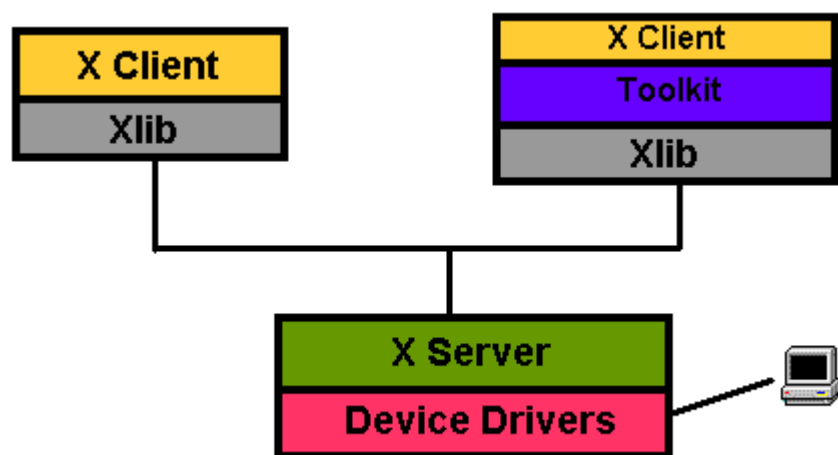


# Escher

X Protocol  
Device Independent Layer

X Protocol  
Device Dependent Layer

- Sun Looking Glass 計畫使用修改過的 Xserver 搭配 Escher (pure Java Xlib 實做) ，並透過 Java 3D 建構 3D Living World
- X Window System 的精髓在於 X Protocol ，以 Message-based 驅動 client-server 架構





# Direct Access X Protocol via Java/Kaffe

Java Application

Widget (double click for demo)

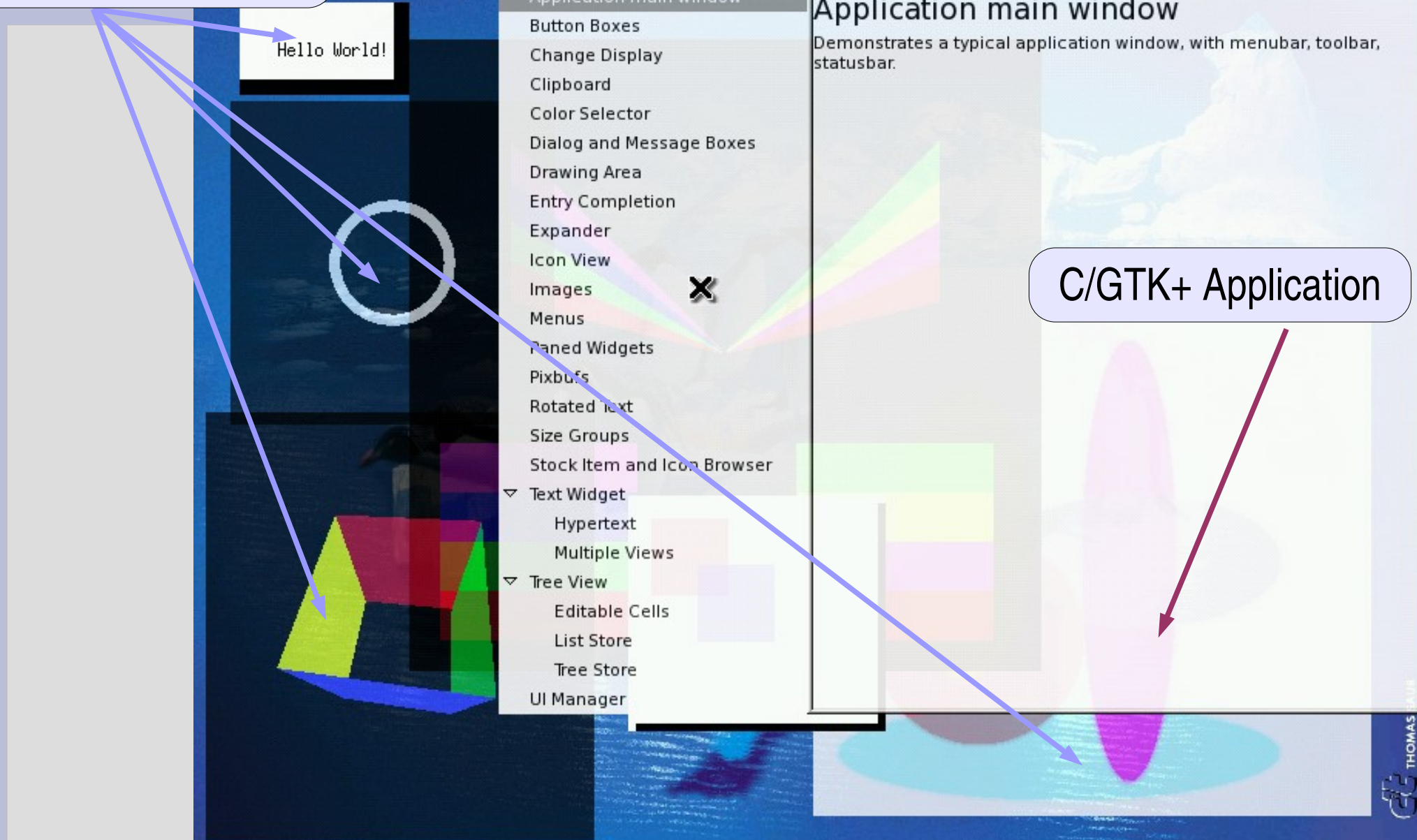
Info Source

- Application main window
- Button Boxes
- Change Display
- Clipboard
- Color Selector
- Dialog and Message Boxes
- Drawing Area
- Entry Completion
- Expander
- Icon View
- Images **X**
- Menus
- Paned Widgets
- Pixbufs
- Rotated text
- Size Groups
- Stock Item and Icon Browser
- Text Widget
  - Hypertext
  - Multiple Views
- Tree View
  - Editable Cells
  - List Store
  - Tree Store
- UI Manager

Application main window

Demonstrates a typical application window, with menubar, toolbar, statusbar.

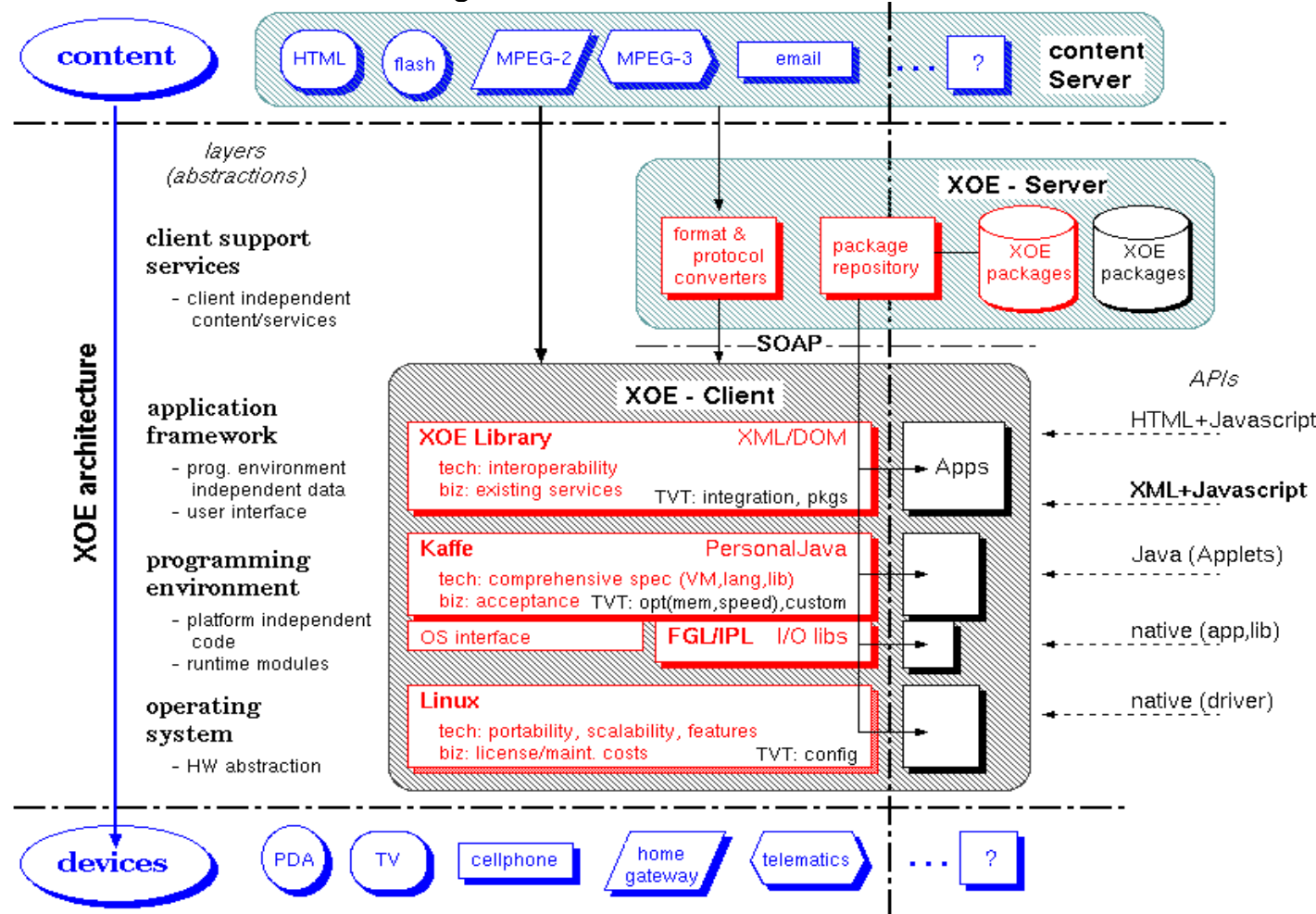
C/GTK+ Application















# KOE (Kaffe-based Operating Environment)

- 源自 Transvirtual Technologies 的 **PocketLinux (XOE)** 產品
  - 2002 年 Transvirtual 結束營運
  - 2004 年重新啓動，並整合到 Kaffe.org





 Settings	 Video Player	 User Friendly	 Themes	 Slashdot
 News	 Memo Pad	 Music Player	 Layout Test	 Flash Player
				



[Patches](#) | [Mailing Lists](#) | [Tasks](#) | [News](#)  
[Patches](#) | [Mailing Lists](#) | [Tasks](#) | [News](#)

ect to create free core class libraries for use with language.

Classpath is still a work in progress. The first public release will be version 1.0. There have been no public releases; however, pre-release source code is available via GNU's anonymous [CVS server](#), and snapshots of the Classpath tree have been released and are available from <ftp://alpha.gnu.org/gnu/classpath/> (older releases can be found in the [archive](#)).

**Documentation**

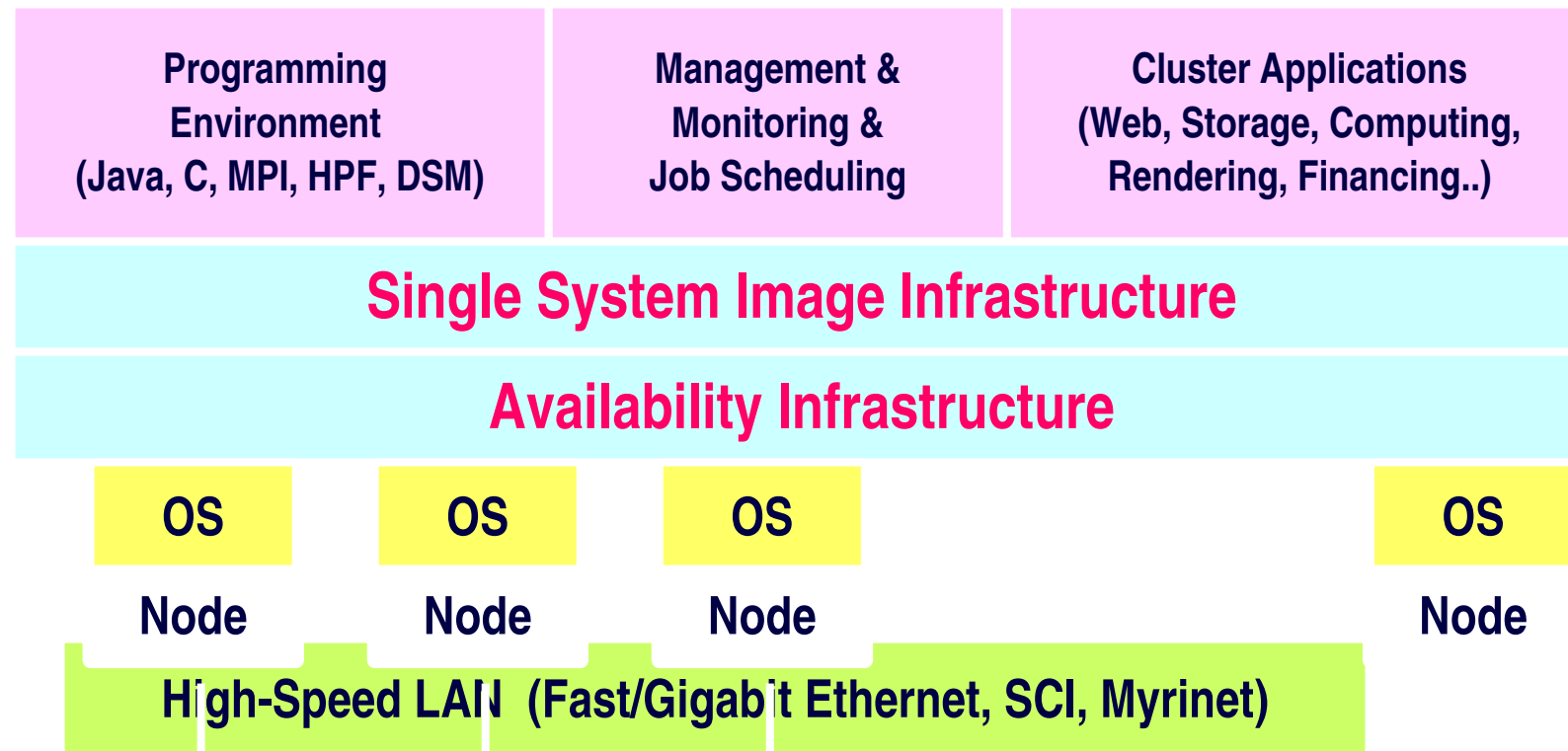
- [Current Documentation](#)
- [Old News](#)

**Status**

GNU Classpath 1.0 will be fully compatible with the 1.1 and largely compliant with the 1.2 API specification and

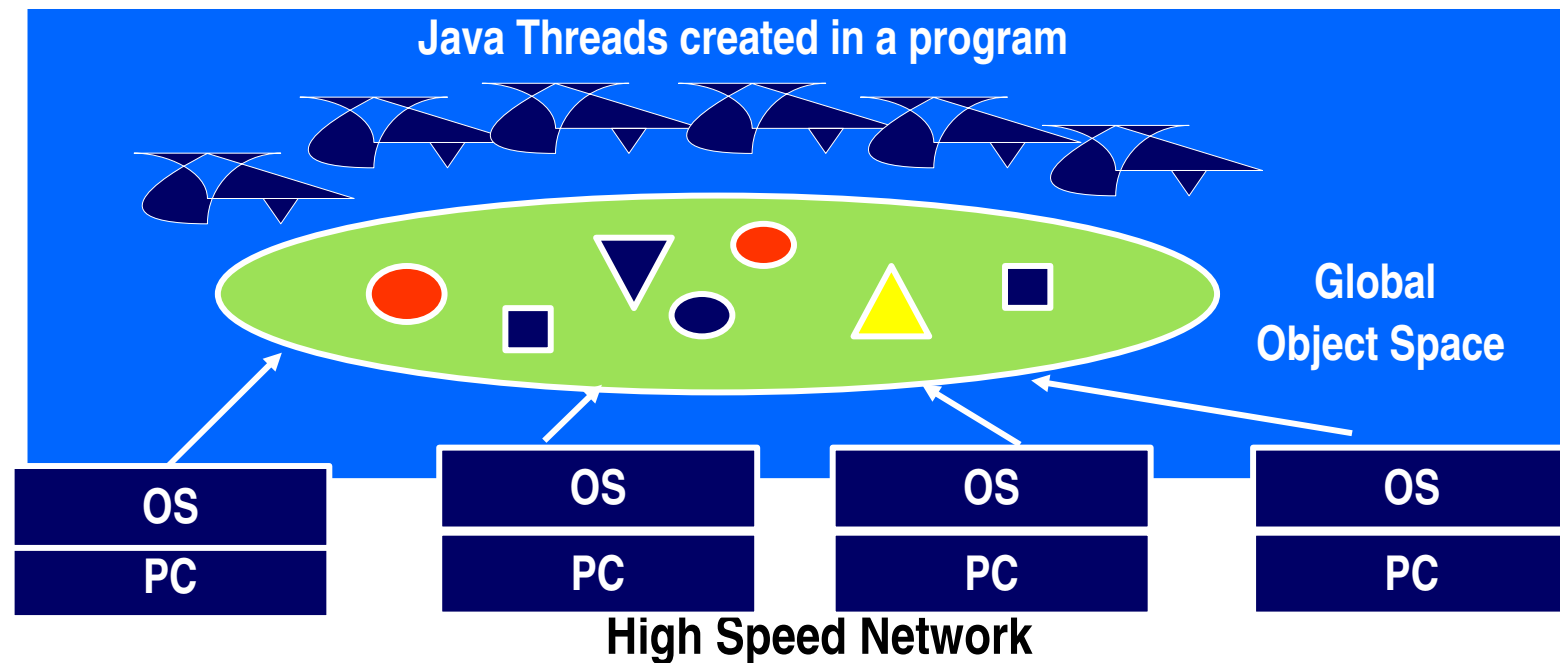
# JESSICA2

- JESSICA – Java-Enabled Single-System-Image Computing Architecture ， 香港大學
- 修改 Kaffe (Version-1: Interpreter only, Version-2: modified JIT) ， 著眼於 Cluster computing 環境中， 建立 SSI 的 middleware ， 以便能夠透明的作平行 multithreaded Java 運算



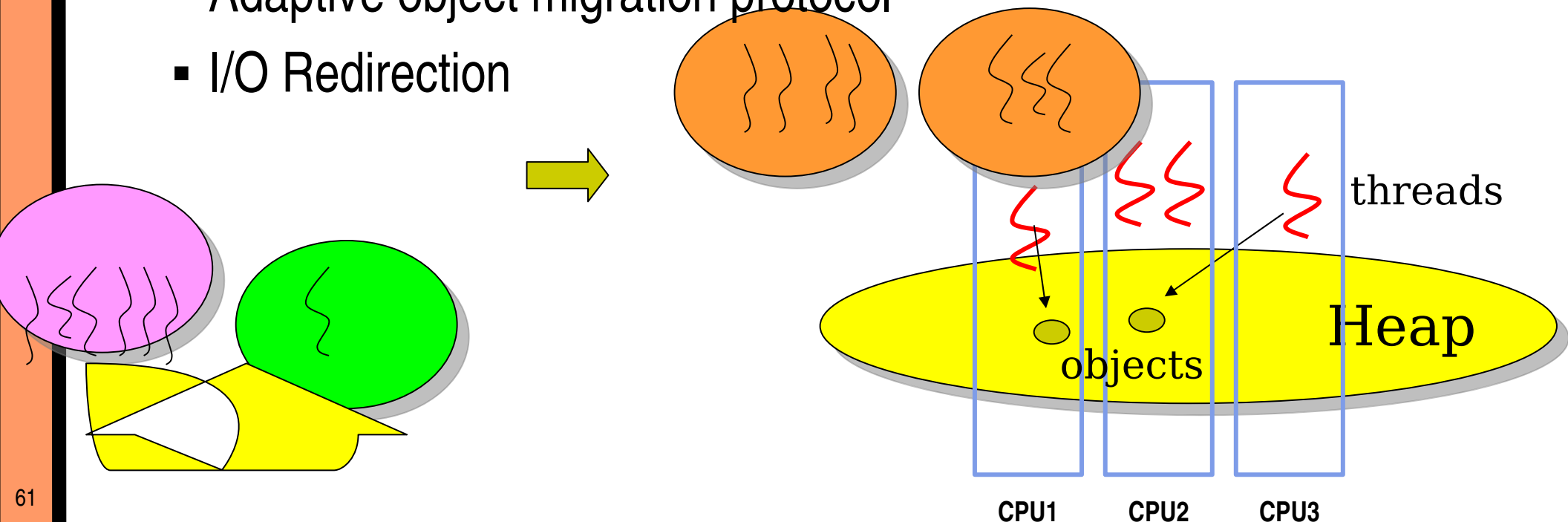
# JESSICA2 巨觀

- 採用 Java 在 Clustering computing 的考量
  - Code mobility (Dynamic Class Loading)
  - Data mobility (Object Serialization)
  - 在語言層面支援 Multithread
- 重要的設計考量
  - Preemptive thread migration



# JESSICA2 & Java Threads

- Java Threads 本質上來看 ...
  - Lightweight 計算單元
  - 具備 Java 語言層面的支援，並使用 shared memory
- 引入”Global Object Storage”的概念來解決
  - 共享的 Global Heap
  - Adaptive object migration protocol
  - I/O Redirection



# Thread Migration

Program:

```
1: i=1;  
2: j=2;  
3: k=i+j;
```

- 思考 ...
  - 至少要紀錄程式碼與執行狀態
- 執行狀態
  - PC (Program Counter)
  - 變數的直接表示狀態
- 考慮 JIT compiler 時 ...
  - native code 無法 migrate!
  - 機器可能不一致

**Node 1**

i=1;  
j=2;

**Node 2**

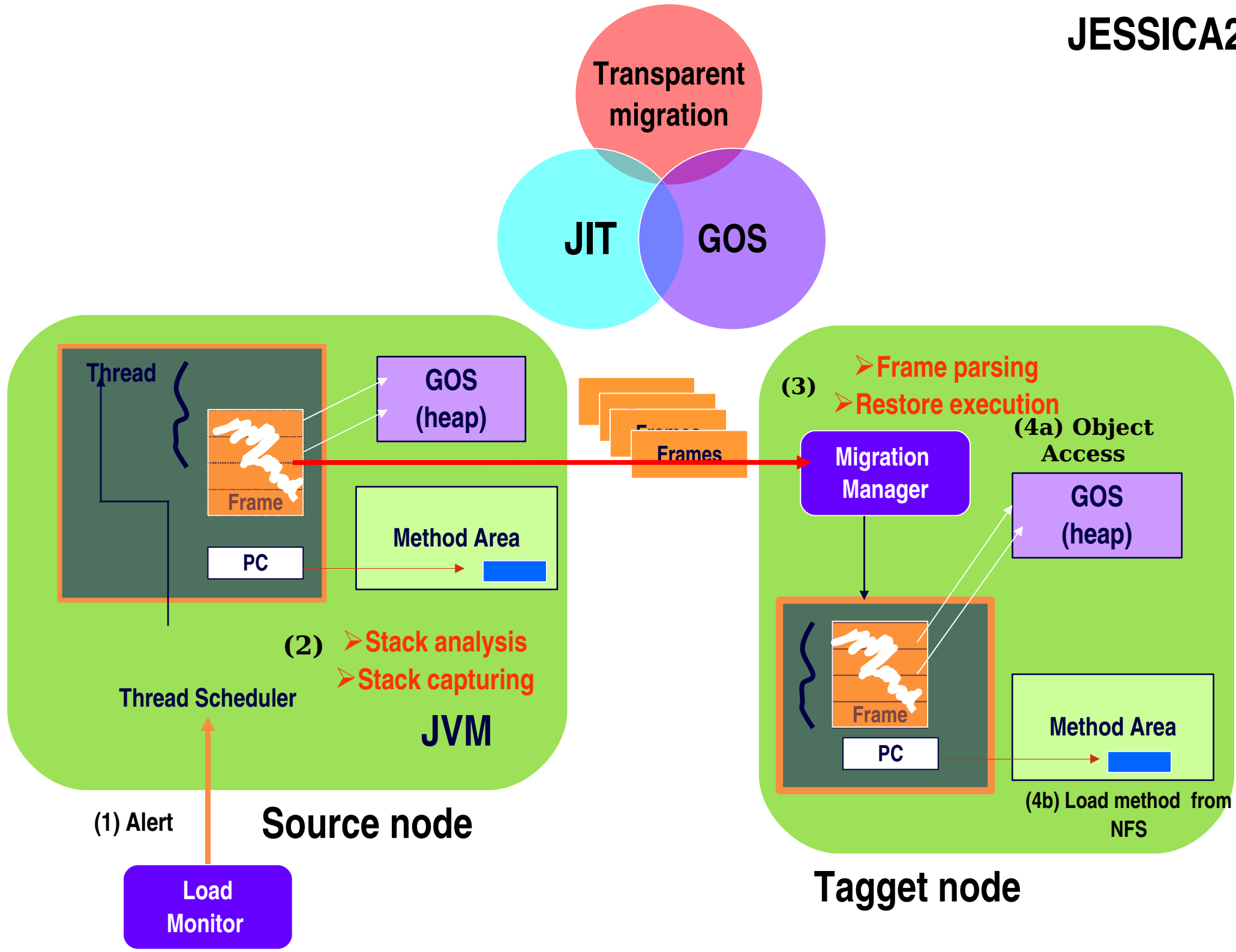
k=i+j;

執行中的  
Java  
bytecode

JIT Compiled



Native code



# JESSICA2 的解決方案

自原始 RTC (Raw Thread Context) 規範可攜性的 BTC  
(Bytecode-oriented Thread Context) **Source node**

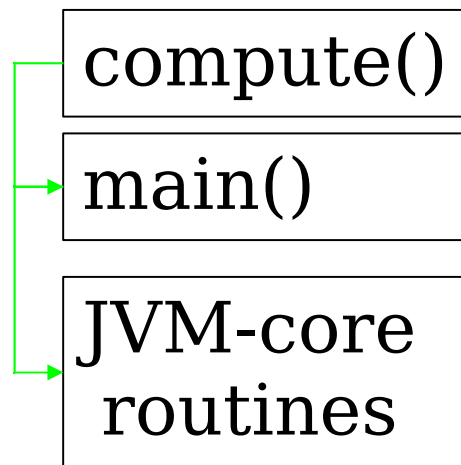
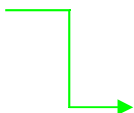
Migration 後，BTC 還原回 RTC

**Target node**  
對兩邊的 thread context 執行 JIT re-compilation





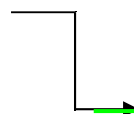
# Stack walk



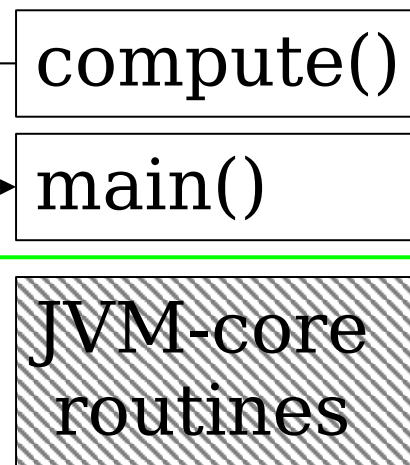
# Stack walk



# Frame segmentation



欲 migrate 的  
Java frames



Stack walk

Frame segmentation

**Bytecode PC  
positioning**

- Branching instructions
  - if\_icmplt (Bytecode)
  - cmpl (x86)
- 具體意義：改變 Execution state / Flow control / Thread Context

compute()

**incl %ebx**

**cmpl \$0x1e, %ebx**

**jl 0x82512432**

iinc 1, 1

iload\_1

bipush 30

if\_icmplt 5

Mapping

main()

Stack walk

Frame segmentation

Bytecode PC positioning

**Breakpoint  
selection**

compute()

```
incl %ebx  
cmpl $0x1e, %ebx  
jl 0x82512432
```

```
iinc 1, 1  
iload_1  
bipush 30  
if_icmplt 5
```

**breakpoint**

```
getstatic  
#2
```

```
return
```

Stack walk

Frame segmentation

Bytecode PC positioning

Breakpoint selection

**Type derivation**

compute()

**local 0: java.lang.thread**  
**local 1: int**  
**stack 0: float**  
**stack 1: int**

main()

**local 0: java.lang.thread**  
**stack 0: double**  
**stack 2: double**

- 依據 Breakpoint 重新產生 native code
- 在 Translation 階段已準備 Migration

Stack walk

Frame segmentation

Bytecode PC positioning

Breakpoint selection

Type derivation

**Translation**

compute()

```
inc %ebx  
cmpl $1e, %ebx  
jl 0x82512432  
0x82512432:  
... capture frames...  
--- save local var/type  
return
```

main()

```
...other native code...  
jmp start_migration
```

getstatic #2  
lload\_1



- 將原本儲存的 return address 替換為剛剛產生的 native code

Stack walk

Frame segmentation

Bytecode PC positioning

Breakpoint selection

Type derivation

Translation

**Native code  
patching**

compute()

```
inc %ebx  
cml $1e, %ebx  
jl 0x82512432  
  
0x82512432:  
... capture frames...  
--- save local var's  
and their type  
return
```

**saved return address**

activation record of  
compute()

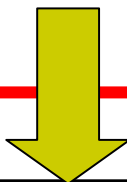
main()

```
...other native code...  
jmp start_migration
```

# JESSICA2 的解決方案

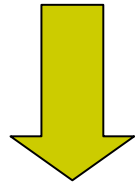
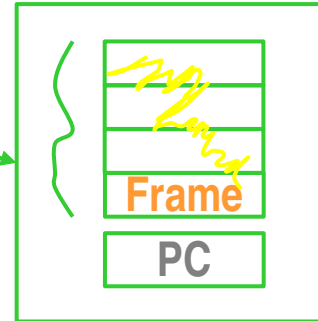
自原始 RTC (Raw Thread Context) 規範可攜性的 BTC  
(Bytecode-oriented Thread Context) **Source node**

Migration 後，BTC 還原回 RTC



**Target node**  
對兩邊的 thread context 執行 JIT re-compilation

**Thread creation**



Thread creation

**Dynamic register patching**

compute()

Register recovering stub:  
movl %0x1234, %eax  
movl %0x5678, %ebx

- 透過 JIT compiler 處理 register
- 建立還原點的 code stub

main()

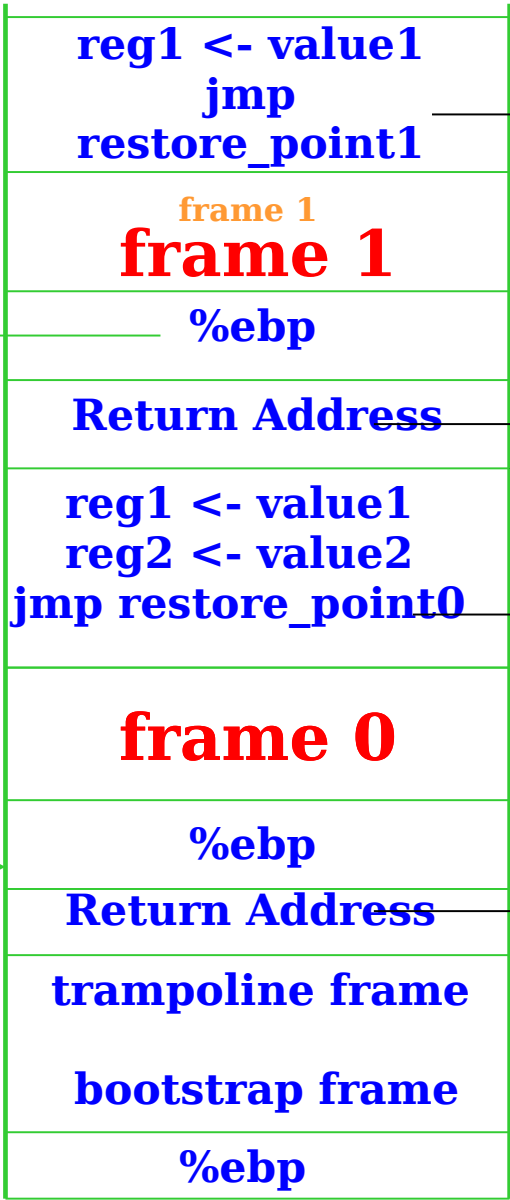
Register recovering stub:  
movl %0x1111, %eax  
...



Thread creation  
Dynamic register patching

Stack rebuilding

# Linking code stub and native frames



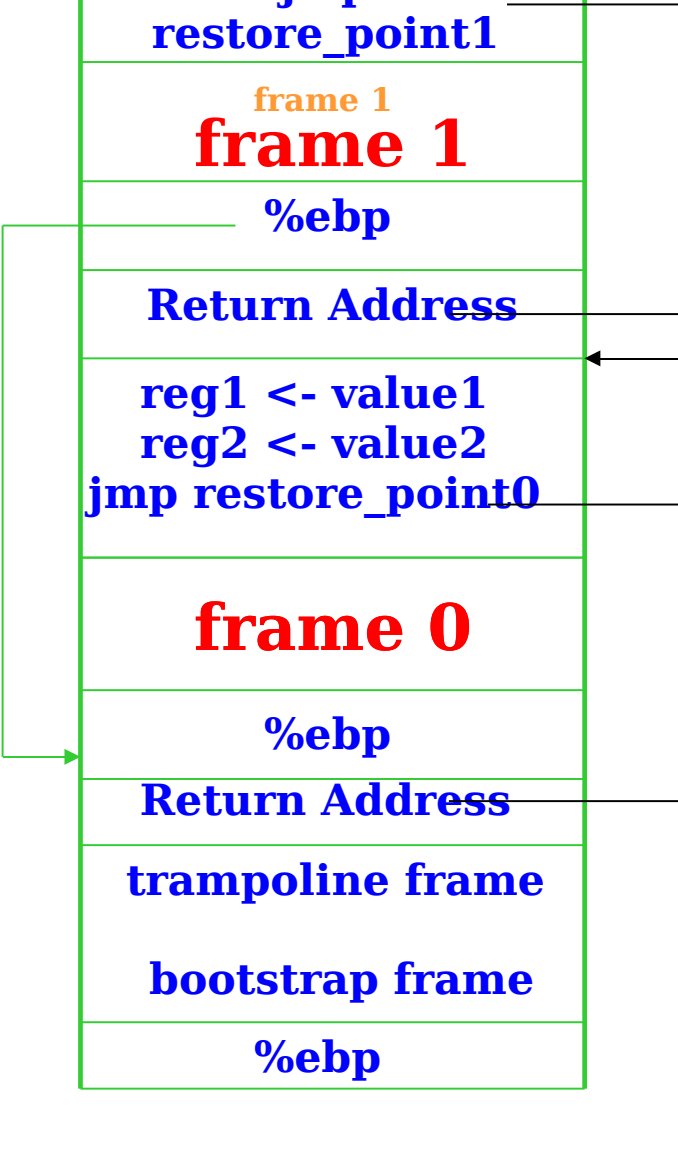
## Compiled methods:

```
compute(){  
...  
restore_point1:  
}
```

```
main(){  
...  
restore_point10:  
}
```

```
trampoline
```

```
bootstrap(){  
  trampoline();  
  closing handler();  
}
```



# Parallel Ray Tracing on JESSICA2

(Running at 64-node Gideon 300 cluster)

stack 3:int,0;  
stack 4:RayTracer;225,0x8495e80;  
}  
  
currentJThread=0x0x8203010, ge  
stack=0x0x81c1840  
Thread 0x8203010 restore execution a  
224708, sp=820a428,bp=820a4b8  
Finish migration journey  
□

GD280B  
GD245B  
GD246B  
GD247B  
GD248B  
GD244B  
GD250B  
GD251B  
GD253B  
GD249B  
GD254B  
GD252B  
GD257B  
GD255B  
GD265B  
GD277B  
GD269B  
GD256B  
GD273B  
GD259B  
stack 3:int,0;  
stack 4:RayTra  
}  
  
current  
stack=0x0x81c1  
Thread 0x8203  
224708, sp=820  
Finish migrati  
□

File

rxvt  
processing i/o job 0x8446e18: id=54(net\_PlainSocketImpl  
socketWrite), fd=190  
processing i/o job 0x8446e18: id=54(net\_PlainSocketImpl  
socketWrite), fd=190  
processing i/o job 0x8446e18: id=54(net\_PlainSocketImpl  
socketWrite), fd=86  
processing i/o job 0x8446e18: id=54(net\_PlainSocketImpl  
socketWrite), fd=193  
processing i/o job 0x8446e18: id=54(net\_PlainSocketImpl  
socketWrite), fd=193  
processing i/o job 0x8446fd8: id=50(net\_PlainSocketImpl  
socketClose), fd=190  
processing i/o job 0x8446fd8: id=50(net\_PlainSocketImpl  
socketClose), fd=193  
Migration complete for thread 0x873d010 aggregate msg=10  
0242  
Migration complete for thread 0x87c4010 aggregate msg=10  
0247

**Linux Kernel 2.4.18-3**  
**64 nodes: 108 seconds**  
**1 node: 3430 seconds**  
**(~ 1 hour)**  
**Speedup = 4402/108=40.75**

total time: 94s

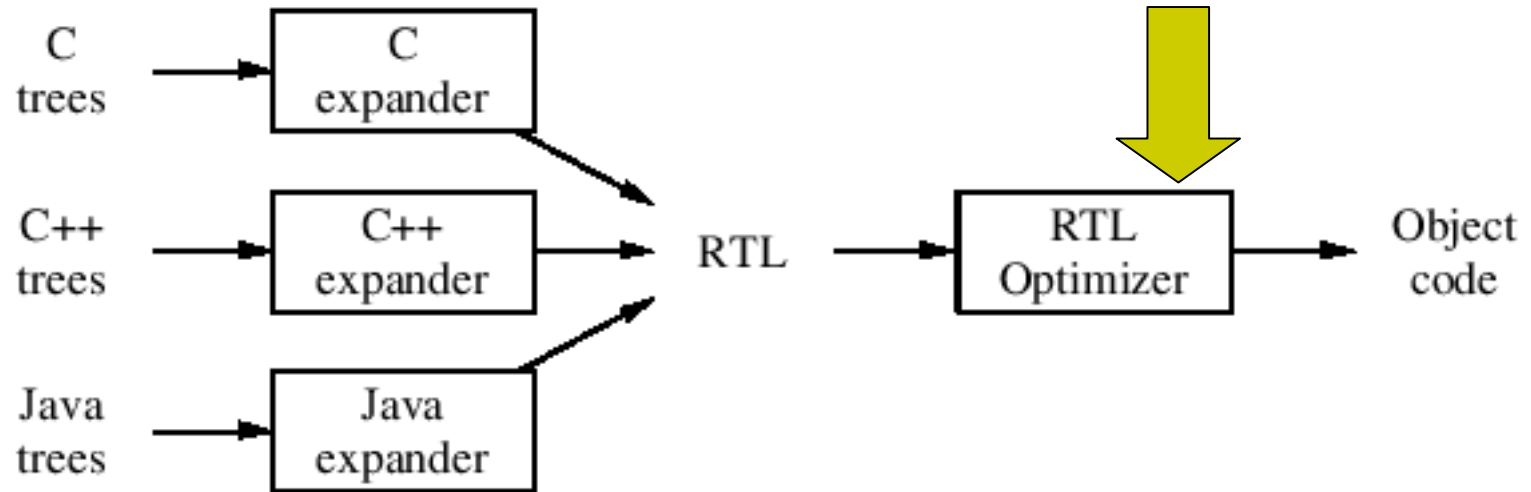
11:27 AM

# GCJ 簡介

- GNU Compiler for Java
  - not Java !
  - but it does run a great many applications written in Java !
- 里程碑
  - 1996 年十月份由 Per Bothner 開始 ( 與 Transvirtual 合作 )
  - 1999 年八月第一個釋出版本現身
  - 2000 年三月份展開與 GNU Classpath 的整合工作
  - 2005 年七月完成與 GNU Classpath 的整合
- GCC4 後引入新機制

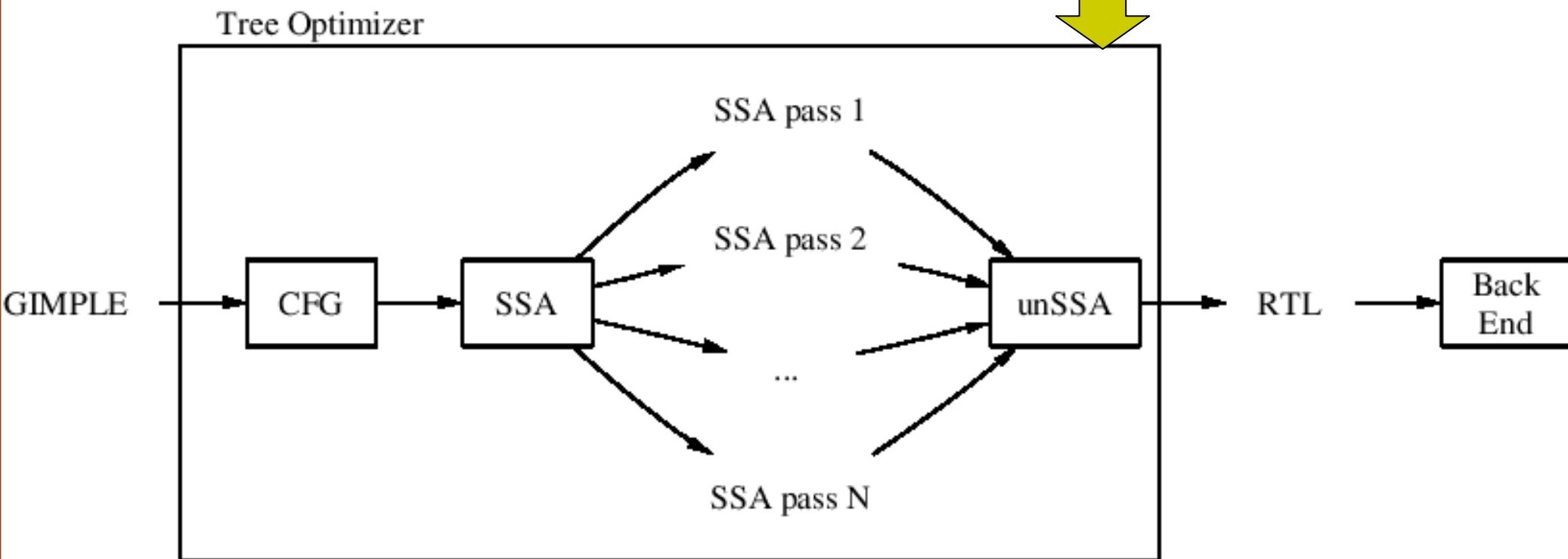
# 反省

## RTL based optimizers



- GNU GCC 一直以眾多硬體平台支援為訴求
  - RTL-based optimizer
- 盲點
  - RTL 不適合高階轉換
  - 必須遷就特定硬體平台而改變整體設計
  - RTL 喪失原始資料型態與控制結構的資訊
  - Addressing model 被一系列的變數指標所替換

# Tree-SSA Optimizer in GCC 4



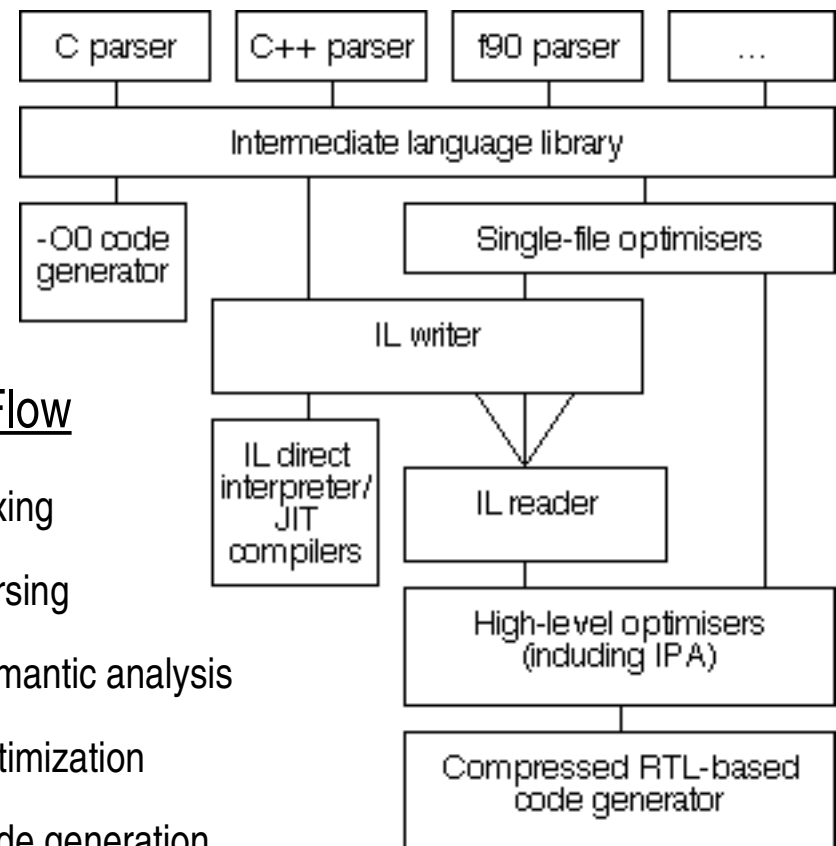
- GIMPLE tree 是語言 / 硬體平台無關的設計
- 所有的型別資料都被保存

# GCC4 新革命

- 過去 GCC 一直採用 GENERIC 途徑
- parser -> IL Library -> middle-end
- IL Library 透過 SSA(Static Single Assignment) 建立內部的最佳化表示法
- 可施行更多最佳化的途徑
  - CSE / SSA
- 允許 JIT 機制的引入

## Compiler Flow

- Lexing
- Parsing
- Semantic analysis
- Optimization
- Code generation



# 破除迷思

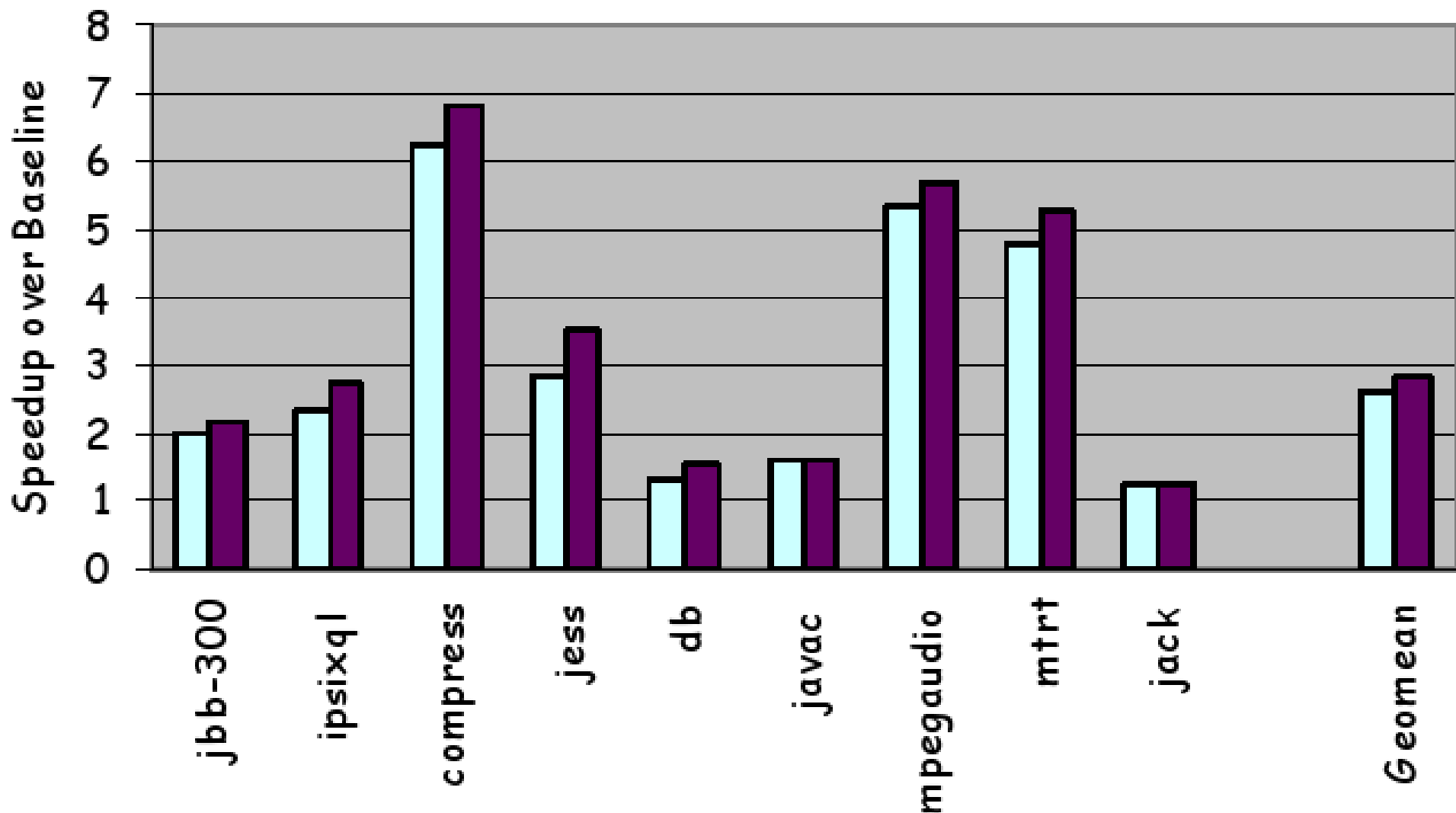
- 誤解：Static compiler 因為少去分析的時間，所以效能絕對比 Dynamic compiler 來得好
- 真相：
  - 良好設計的 JIT compiler 在許多層面可以實做出 Static compiler 所不能達到的最佳化設計
  - FDO (Feedback-Directed Optimization) 能夠比沒有 profiling 資訊的 IPA 來得有效率
  - 傳統 C compiler backends 喪失 JIT 能夠最佳化的類型與語言層次的資訊
  - AOT compilation 缺乏快速的啓啓動時間與 small footprint

# FDO (Feedback-Directed Optimization)

- 簡單來說：利用執行時期 (Runtime) 所獲取的資訊作進一步的最佳化執行
  - "selective optimization": WHAT to optimize.
  - "FDO": HOW to optimize.
- FDO 的特性
  - 可獲取 Static compilation 所未知的 Runtime info
  - 系統可以更動或反悔原先的 policy ，如果情況改變的話 (如：CPU bound vs. I/O bound)
  - 允許 Runtime 建構為更有彈性的設計



Default Default+FDO



- JikesRVM 的測試報告顯示 ...
  - FDO 可增進 long execution process 的效能

# 在 GCJ4 之前 ...

- 過去以 C++ 類似的方式建構 ABI (Application Binary Interface)
  - 亦即 DWARF2 Exception Handling
  - Static field/method references resolved by linker
  - [instance\_field:instance\_method] -> [offset:vtable]
- 面臨的問題
  - user-defined ClassLoaders 無法正確運作，更無法 precompiled
  - 無法允許 precompiled classes 繼承自 interpreted classes
  - ELF linkage model 無法確保 Java 語言層面的 namespace 議題
  - C++ linkage model 不適合 Dynamic Languages

# GCJ4 新特徵

- New ABI
  - 所有的 field offset 採用 indirect access
  - 所有的 method table 都在執行時期產生
- User-defined class loaders
  - gcc/libjava/java/lang/ClassLoader.java
- defineClass(byte[])
  - 產生 byte 的 MD5 檢查碼
  - 在系統資料庫查詢 symbol entry
  - 找到的話，載入適當的 shared library
  - 符合 JIT compiler 的行為

# GCJ4 新特徵 (cont)

## ■ New ABI

- 可在不需重新編譯應用程式的前提下，升級 GCJ Library
- Class <--> Shared\_Library Map
  - 完全隱蔽 Class 生成資訊
  - Old-ABI code call virtual method
    - ((vtable \*) obj)[index] (obj, ...)
  - New-ABI
    - (((vtable \*) obj)[otable[index]] (obj, ...)

## ■ 範例

- gcj foo.jar -shared -fPIC -o foo.jar.so
- gcj-dbtool -a system-java-symbols.db foo.jar.so

# GCJX : GCJ 實驗性計畫

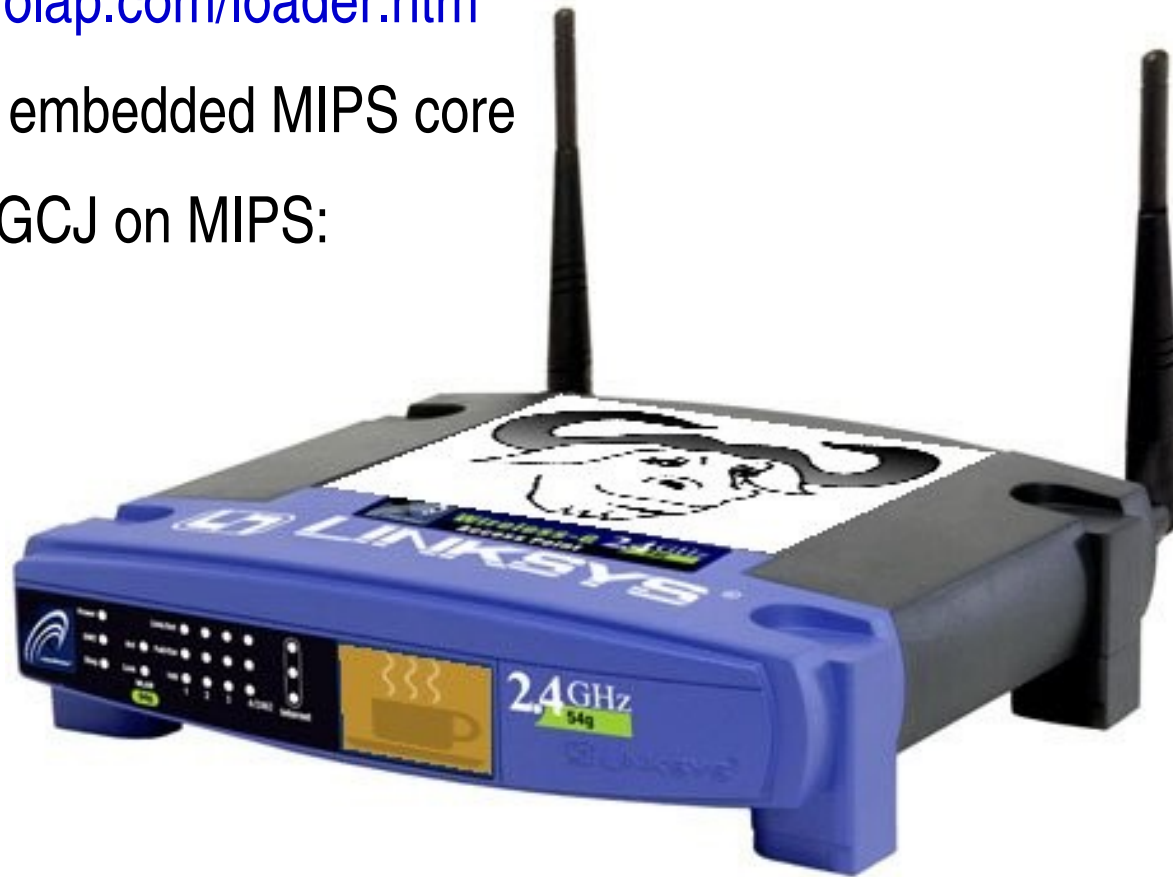
- 以 C++ 撰寫的新 GCC Frontend
- GCC 內部設計
  - RTL
  - Tree
    - Language-specific
    - GENERIC
    - GIMPLE
  - High-level Optimizations
- 著眼點
  - Language-specific Model
  - Static typing
  - Lower model to tree
  - Multiple backends

# 展示

- Why Free Java?
- 目前 Free Java Runtimes 概況
- 如何建構 cleanroom JavaVM ?
- Case Study
- 展示

# In Actions !

- platform support
  - <http://controlap.com/loader.htm>
  - Broadcom embedded MIPS core
  - Kaffe and GCJ on MIPS:



# 結論

- GNU Classpath 匯集許多專案，眾多開發者得以累積成果並相互受益
- 基礎建設逐漸完備，欲擴展或研究者可以專注於 VM 設計本身
- Mauve 是相當好的相容性驗證工具
- Free Java Hackers around the World !
- 開放原始碼創造無限可能，而整體來說，更是一種新契機



# 參考資源

- GNU Classpath
  - [www.classpath.org](http://www.classpath.org)
- news
  - [planet.classpath.org](http://planet.classpath.org)
- Kaffe
  - [www.kaffe.org](http://www.kaffe.org)
  - <http://latte.snu.ac.kr/>
  - <http://i.cs.hku.hk/~wzzhu/j2/>
- GCJ
  - <http://gcc.gnu.org/java/>
- Any help is greatly appreciated
  - Test Java programs on free Virtual Machines
  - Write test cases for Mauve
  - Write intellegible documentation
  - Implement library classes
  - Fund development
- How to proceed
  - Look at the task list in GNU Classpath
  - Copyrights must get assigned to FSF

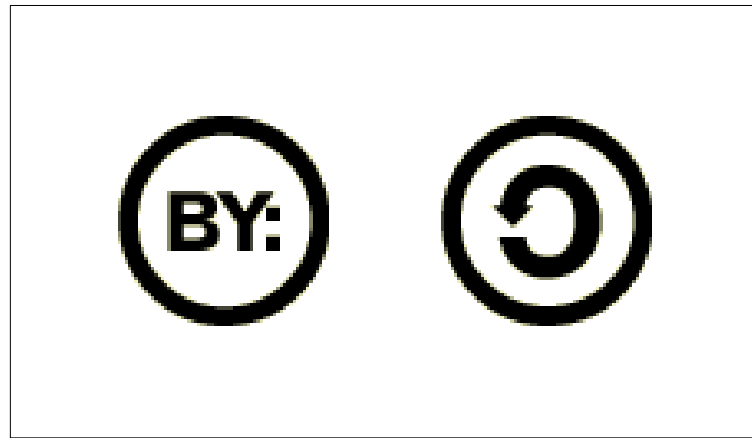
# Thanks!

Jim Huang ( 黃敬群 / **jserv** )

Website: <http://jserv.sayya.org/>

Blog: <http://blog.linux.org.tw/jserv/>

# Some rights reserved



This work is licensed under the Creative Commons Attribution-ShareAlike License.

To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/de/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.