

# OldFrontPage

here are the contents of the [FrontPage](#) as of 2006-09-28

- [Apache Harmony Proposal](#)
  - [Apache\\_Harmony\\_Proposal\\_Spanish](#) - Propuesta Inicial del proyecto Apache Harmony en castellano
  - [Apache\\_Harmony\\_Proposal\\_Portuguese](#) - Proposta Inicial do projeto Apache Harmony em português
  - [Apache\\_Harmony\\_Proposal\\_Italian](#) - Proposta iniziale del progetto Apache Harmony in italiano
  - [Apache\\_Harmony\\_Proposal\\_Turkish](#) - Teklifin Türkçe Versiyonu
  - [Apache\\_Harmony\\_Proposal\\_Czech](#) - Návrh v češtině
  - [Apache\\_Harmony\\_Proposal\\_German](#) - Deutsche Übersetzung des Antrags
  - [Apache\\_Harmony\\_Proposal\\_Chinese](#) - Apache Harmony Proposal in Chinese
  - [Initial FAQ accompanying the proposal](#)
  - [Initial\\_FAQ\\_Spanish](#) - Incomplete (FAQ in spanish - FAQ en castellano)
  - [Initial\\_FAQ\\_Portuguese](#) - Incomplete (FAQ in portuguese - FAQ em português)
  - [Initial\\_FAQ\\_Italian](#) - (FAQ in italian - FAQ in italiano)
  - [Terminology](#)
  - [TechnicalFAQ](#)
  - [HarmonyArchitecture](#) Proposed Harmony Architecture (Overview)
  - [HarmonyArchitectureItalian](#) Architettura Proposta Per Harmony (Overview) (incompleta)
  - [HarmonyArchitectureChinese](#) Proposed Harmony Architecture (Overview)
- 

## Resources

- [Building\\_instructions](#)
  - [SVN](#)
  - [JIRA](#)
  - [Email Archives](#)
  - IRC: #harmony on irc.freenode.net
  - Add yourself on our [People](#) page if you are interested in getting involved.
- 

## Mini Projects

The following have been suggested on the list and are either things that are needed, or good little experiments for interested people

- [CommandLineCompiler](#) Create a "javac" compatible command-line java compiler using the eclipse compiler
- 

## Motivations

- Focus on modular, interchangeable components
    - exploit existing compilers, memory managers etc
    - promote configurability (different components for different contexts)
    - allow diversity in development approaches
    - encourage large-scale contributions (here's a compiler)
  - Bootstrap the project rapidly
    - capture momentum
    - seed the project with an existing VM-core (or cores)
  - Design a clean core (or cores) from scratch
    - do this concurrently with work on components in existing core/s
    - the core should be lightweight
    - multiple cores may make sense
      - the needs of different contexts may require this
      - competing approaches may be healthy
- 

## Requirements

- Nice Clean Super-Fast VM
  - J2SE 5.0 Implementation
  - Performance - [Performance Comparison of Java/.NET Runtimes](#)
  - Debuggability
  - Platform Coverage (including Win32)
- 

## Existing work for pluggability

- [Hooks from Classpath to VM](#)

- [Hooks from VM to Classpath](#)
  - [Boehm GC](#). **NOTE:** Kaffe has two GC thingies [kaffe-gc](#) and [boehm-gc](#)
  - [MMTk from JikesRVM](#)
  - [SableJIT - a module of SableVM](#)
  - [JVM\\_Feature\\_Comparison](#)
  - [JVM\\_Implementation\\_Ideas](#)
- 

## Other usable components

- zlib
  - fdlibm
  - APR
- 

## Components

- VM
  - [ClassLibrary](#)
  - GC
    - kaffe-gc and Boehm GC in Kaffe
    - simple copying GC and generational copying GC in SableVM
  - JIT - [List of Java just-in-time \(JIT\) compilers](#)
  - Compiler
  - Bytecode Verifier (Note: BCEL, [libgcj](#), IKVM)
  - OS Layer (interfacing to files, sockets, threads)
  - GUI Layer (AWT/Swing) - [GUI Toolkit Diagram](#)
  - JNLP
  - Java plugin
  - [Debugging via JVMDI/JDWP in SableVM](#)
- 

## Decisions

- Object Layout
    - synchronization
    - getting the class pointer
    - does the object point to the vtable?
    - how does the GC know about the layout of the object?
    - plain old ordering and packing of the fields
  - Method Dispatch
    - C++-style vtables vs constant-time dispatch tables
  - Exception Handling
    - biased for speed
- 

## One Concerte Option

- Use two VMs as seeds
  - Jikes RVM is a possible candidate
    - Focus energy on cleaning it up and modularizing it. This is something we've already begun (see earlier post), but will take a lot of work.
      - Get a very good optimizing compiler
      - Get an efficient and modular memory management toolkit (MMTk)
      - Need to deal with licensing issues and gain the consent of the community (not insurmountable)
      - Need hard work to achieve modularity goal for whole VM
  - Another very different VM (Kaffe?)
    - amenable to modularization
    - amenable to other components (drop in MMTk?)
- Leverage extensive experience to build new core/s
  - Start with a clean slate
  - Leverage all of our diverse experience (gcj, Kaffe, ovm, joqe,

jnode,...)

- Work concurrently with above work on components in old core/s, minimize loss of momentum, try to really think it through carefully.
- May be sensible to develop more than one core

- Develop new components
  - Extract components from existing work, apply to new VM/s
  - Develop new components from scratch
  - Encourage porting of existing compilers etc into this framework

## Issues

- What do we do about com.sun.\* internal classes?
  - Process for getting bugs fixed in Classpath?
  - Use java for bulk of the code (like JikesRVM?)
  - modularity is compile-time?
  - handling memory fragmentation in a long running process?
- 

## Volunteers with contributions

- [Archie Cobbs - JCVm](#)
- 

## Random Thoughts

- write a new jikesrvm back end?
  - write a new vm around llvm?
  - m4 to speedup things?
  - writing an excellent jit is not easy, reuse is perhaps more efficient, eg use LLVM for that chunk?
- 

## Bookmarks

- <http://www.shudo.net/jit/perf/>
  - [Collection of Papers on JRE Issues, incl. JVM, JIT, GC, Emulators, etc.](#)
  - [A Collection of JVM Options](#)
  - [MIT Online Courseware: Computer Language Engineering](#)
  - [USF Programming Languages Course Lecture Notes/Audio](#)
- 

## Papers

- [A PORTABLE RESEARCH FRAMEWORK FOR THE EXECUTION OF JAVA BYTECODE](#)
  - [A Modular and Extensible JVM Infrastructure](#)
  - [The Jalapeño Virtual Machine\(Jikes RVM\)](#)
  - [JMTk: A portable memory management toolkit.](#)
  - [Combining Stack Caching with Dynamic Superinstructions](#)
  - [GC Points in a Threaded Environment – Sun Microsystems](#)
  - [Cost-Effective Compilation Techniques for Java Just-in-Time Compilers](#)
- 

## Articles

- [Java Implementations](#)
- 

## Tutorials

- [Dynamic Compilation and Adaptive Optimization in Virtual Machines](#)
  - [The Design and Implementation of the Jikes RVM Optimizing Compiler](#)
  - [The Design and Implementation of the Jalapeño Research VM for Java](#)
  - [Software Optimization and Virtual Machines](#)
- 

## Related Pages

- [JVM Feature Comparison](#)
- [Kaffe-Gump](#)
- [Nick Lothian's Unofficial Apache Harmony Blog](#)
- [Existing JVM Benchmarks](#)
- [Native code and assembly resources](#)
- [JVM In Java](#)
- [Unofficial Apache Harmony Blog in Spanish - Blog no oficial en castellano](#)

---

## IRC channels on irc.freenode.net

- [#harmony](#)
- [#kaffe](#)
- [#classpath](#)
- [#sablevm](#)

## What to do if you want to help NOW!

- [Contribute to GNU Classpath](#)
- [Contribute to JikesRVM](#)
- [Contribute to kaffe](#)
- [Contribute to SableVM](#)