제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

컴포넌트 기반 파이프라인 자바가상머신 인터프리터 설계

한상만, 이승룡
경희대학교 전자계산공학과
e-mail:{i30000, sylee}@oslab.kyunghee.ac.kr

Design of Pipelined Java Virtual Machine Interpreter Based on Component

Sang-Man Han, Sungyoung Lee
Dept. of Computer Engineering, Kyung Hee University

요 약

최근 자바가상머신 인터프리터의 성능 개선을 위하여 파이프라이닝 기법에 대한 연구가 활발히 진행중이다. 반면, 내장형 시스템 환경에서는 급변하는 시장의 적시성 요구(time-to-market)와 저렴한 비용으로 다양한 사용자 요구사항을 효율적으로 반영하기 위하여 재구성 가능한 컴포넌트기반 소프트웨어 개발 방법이 점점 주목받고 있다. 따라서, 본 논문에서는 자바환경을 지원하는 내장형 시스템에 적합한 가상머신 인터프리터를 개발하기 위해, 재구성과 융통성을 제공하기 위한 컴포넌트기반 소프트웨어 개발 방법과, 성능향상을 위하여 파이프라이닝 기법을 혼합한 새로운 설계 기법을 소개한다.

1. 서론

최근 Post PC 시대를 맞이하여 내장형 시스템에 대한 관심이 증대되고 있어 내장형 시스템 개발에 대한 연구가 많이 진행되고 있다. 내장형 시스템 소프트웨어 개발에는 이식성, 신뢰성, 재사용성에서 장점을 가지고 있는 자바가 개발언어로 급부상하고 있다.

자바환경을 지원하는 자바가상머신은 자바로 작성되어 컴파일된 바이트코드를 실행해주는 미들웨어로써, 클래스로더, 실행엔진, 메모리관리자, 쓰레드관리자 등으로 구성되어 있다. 여기서 실행엔진은 인터프리터와 JIT(Just-in-Time) 컴파일러로 구현이 되는데, JIT 컴파일러는 성능은 좋지만 이식성이 좋지 않다는 문제를 가지고 있으며, 인터프리터는 이식성 등은 뛰어나지만, 수행속도가 느리다는 문제점을 가지고 있다.

인터프리터의 수행속도를 향상시키기 위한 방법으로 파이프라이닝 기법이 있다. 이는 인터프리터의 작업수행시 파이프라이닝을 통해 동일한 시간에서도 보다 많은 명령을 수행할 수 있게 함으로써, 인터프리터의 성능을 향상시킬 수 있다.

한편, 내장형 시스템 환경에서는 시장의 적시성 요구(time-to-market)와 저렴한 개발비용으로 다양한 사용자 요구사항을 반영하기위하여 재사용과 재구성이 가능한 컴포넌트기반 소프트웨어 개발 방법이 점점 주목받고 있다.

따라서, 본 논문에서는 자바 가상머신의 인터프리터 설계 시 성능향상을 위한 파이프라이닝 기법과 재구성과 융통성을 제공할 수 있는 컴포넌트 기반 소프트웨어 기법 동시에 사용한 인터프리터 설계방법에 대하여 소개한다.

컴포넌트 개발 방법론에 의거한 파이프라이닝 기반 인터프리터 설계를 위해서 PBO(Port-Based Obejct)모델에서 제공하는 Framework를 사용한다. 단순히 컴포넌트 모델로서의 Framework는 컴포넌트화된 객체들에 대한 재구성 등의 제어를 하게 되지만, 본 논문에서는 Framework의 역할을 확장하여 컴포넌트 객체들에 대한 제어뿐만이 아니라 파이프라이닝을 위한 제어를 동시에 수행할 수 있게 함으로써 내장형 시스템에 적합한 인터프리터 구축이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서, 파이프라인 인터프리터와 컴포넌트 모델인 PBO모델에 대해 소개하고, 3장에서는 본 논문에서 제안하는 컴포넌트 기반 파이프라인 인터프리터 설계에 대해 설명을 하며, 4장에서는 결론을 맺으며 향후 과제에 대해 언급한다.

2. 관련연구

본 장에서는 기존의 파이프라인 인터프리터에 관한 연구와 컴포넌트 모델의 하나인 PBO(Port- Based Object) 모델에 대해 살펴보겠다.

2.1 파이프라인 자바가상머신 인터프리터

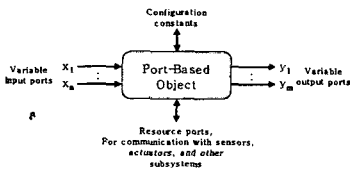
자바가상머신 인터프리터는 개발하는 시간이 빠르고, 유지비용이 적으며, 호환성이 좋다는 장점을 가지고 있다. 하지만, 이러한 장점에도 불구하고 인터프리터는 실행속도가 느리다는 문제점을 가지고 있다. 따라서, 이러한 인터프리터의 실행속도 문제를 개선하기 위한 연구가 진행되고 있다. 이러한 인터프리터 실행속도를 개선하기 위해 제시된 방법중의 하나가 바로 파이프라인 인터프리터이다. 여기서 파이프라인의 방법으로 작업을 수행시키기 위해서는 명확히 구분되어지는 작업의 단위가 필요로 하게 되는데, 이러한 단위는 인터프리터의 작업들 중 가장 중요한 부분들인 바이트코드 Fetch, 바이트코드 분석, 바이트코드 실행의 세부분으로 나누어지게 된다. 이렇게 나누어진 작업들은 파이프라이닝의 스케줄에 맞게 작업이 수행이 되며, 파이프라이닝에 따라 작업을 수행하게 된다. 때문에, 순차적으로 실행되는 바이트코드에 비해 동일한 시간에 더 많은 작업들을 수행할 수 있게 함으로써, 인터프리터의 실행속도를 개선시킬수 있게 하였다. 하지만, 파이프라이

※ 이 논문은 2001년도 한국과학재단 목적기초연구(과제번호: R01-000-00357-0)의 연구비에 의하여 연구되었음.

닝을 적용한 인터프리터의 경우, 인터프리터의 성능을 많이 개선시킬 수는 있지만, 인터프리터의 융통성은 제공해 줄 수가 없다. 즉, 다양한 플랫폼에 적용되었을 경우, 각 플랫폼에 적합한 인터프리터를 재구성하는 것이 어렵다는 것이다. 따라서, 인터프리터의 성능만을 고려하는 것이 아니라, 재구성이 가능한 인터프리터 역시 고려한 인터프리터에 대한 연구가 필요하다.

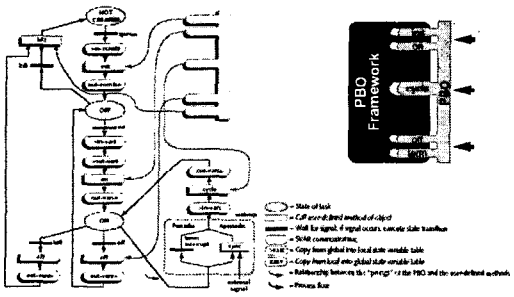
2.2 PBO(Port-Based Object) 모델

PBO[Stewart00,Stewart01,Stewart97]는 Carnegie Mellon Univ의 Advanced Manipulators Laboratory에서 개발하였으며, 도메인 컴포넌트의 개발에 기반하고 있다. PBO 모델의 기본은 독립적인 태스크이다. 이 태스크는 다른 컴포넌트와의 통신을 허용하지 않으며, 느슨하게 결합되어 재사용하기 쉽다. PBO의 설계 목적은 컴포넌트 사이의 동기화와 통신의 최소화이다. PBO의 데이터 흐름은 입출력 포트를 통하여 규정한다. 입력 포트를 통해 데이터 생산자에 대한 지식 없이도 가장 최근의 정보를 읽을 수 있으며, 다른 컴포넌트를 위한 유효한 정보를 만들기 원할 때 출력 포트에 저장할 수 있다. PBO에서는 컴포넌트를 더욱 유연성 있고 재사용할 수 있게 만들기 위해 매개변수 인터페이스(parameterization interface)가 제공되며, 이를 통해 서로 다른 행위가 단일 컴포넌트에 의해 구현이 가능하다. [그림 1]은 PBO를 나타낸다.



[그림 1] Port-Based Object

PBO로 정의된 객체들은 PBO Framework를 통해서 작업을 수행하게 된다. 즉, PBO Framework는 PBO 객체들의 작업수행을 제어하며 컴포넌트의 재구성을 위한 메카니즘을 제공한다. PBO Framework의 구조와 흐름은 [그림 2]와 같다.



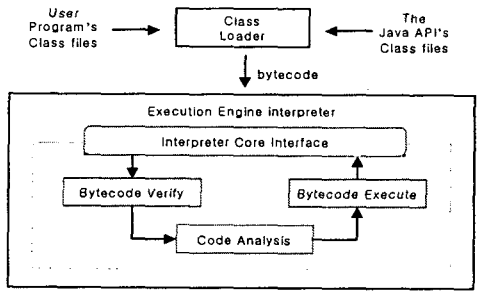
[그림 2] PBO Framework

이러한 PBO 모델을 적용하여 인터프리터를 설계할 경우, 인터프리터의 컴포넌트 화가 이루어지게 되어, 다양한 플랫폼에서 각 플랫폼에 적합한 인터프리터를 재구성하는 것이 용이해 진다. 하지만, 단순히 재구성을 위한 인터프리터를 구성하기 위해 컴포넌트화를 할 경우, 인터프리터의 융통성은 보장이 되지만, 인터프리터가 가지고 있는 문제점 즉, 실행속도 저하에 따른 문제를 개선해 줄 수가 없다. 따라서, 인터프리터의 재구성을 위한 컴포넌트화와 동시에, 인터프리터의 성능을 보완해 줄 수 있는 연구가 필요하다.

3. 컴포넌트 기반 파이프라인 자바가상머신 인터프리터

3.1. 자바가상머신 인터프리터

자바가상머신 인터프리터는 자바가상머신의 핵심 요소로서, 시스템이 초기화되고 사용자의 프로그램을 동작시킬 준비가 된 이후에 사용자 프로그램에서 정의한 main 메소드를 실행하는 역할을 한다. 또한, 사용자의 프로그램에서 요구하는 작업에 대한 바이트코드를 클래스 로더를 통해 전달받아 수행해주는 작업을 하게 된다. 이러한 작업을 위해 인터프리터는 [그림 3]과 같은 작업들을 반복적으로 수행하게 되며, 수행되는 작업들의 역할은 [표 1]과 같다.



[그림 3] 실행엔진 Interpreter의 작업 수행도

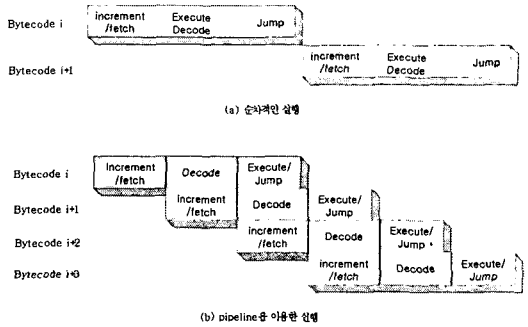
[표 1] 실행엔진 interpreter에서 수행되는 작업별 역할

작 업	역 할
Interpreter Core Interface	• 작업을 수행하는데 필요한 바이트코드, 메모리, 쓰레드, 락, 예외 등을 제공받기 위한, 다른 서브 시스템과의 연계를 위한 인터페이스 제공한다
Bytecode Verify	• Class Loader Subsystem으로부터 전달받은 바이트코드가 적합한 바이트 코드인지 검증하는 작업을 담당한다
Code Analysis	• 검증과정을 통과한 바이트코드에 대해 해당 바이트코드가 요구하는 명령이 무엇인지 분석하는 작업을 수행하는 것을 담당한다
Bytecode Execute	• 바이트코드를 실제 수행하는 작업 • 전달받은 바이트코드가 검증과정, 분석과정을 수행된 후 바이트코드에 해당하는 명령을 수행

3.2 파이프라이닝(pipelining) 인터프리터

자바가상머신 인터프리터는 개발시간, 유지, 호환성등이 좋다는 장점이 있지만, 실행속도가 느리다는 치명적인 단점을 가지고 있다. 게다가, 인터프리터에서 실시간과 컴포넌트를 고려하게 될 때, 그 성능은 일반 인터프리터보다 떨어지게 된다. 따라서 인터프리터에서 실시간과 컴포넌트를 고려해도 그 성능이 떨어지는 것을 최소화하기 위해서는 인터프리터의 성능향상이 요구되게 된다. 본 논문에서는 인터프리터의 성능향상 방법으로 파이프라인 인터프리터를 제공한다. 파이프라인을 고려하게 되었을 경우 인터프리터는 작업 수행 시간을 단축시킬 수 있기 때문에 실시간, 컴포넌트가 고려되어서 저하되는 성능을 보완시켜 줄 수 있다. 다음 [그림 4]는 파이프라인을 고려했을 경우

의 작업수행과 파이프라인을 고려하지 않은 경우의 작업 수행을 비교해 주는 것이다.



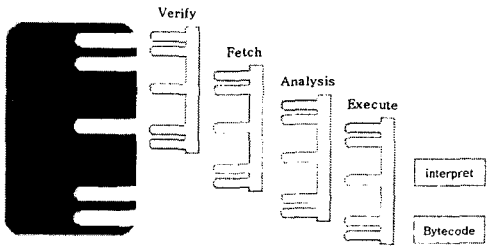
[그림 4] Pipeline의 적용 유무에 따른 작업수행도

기존의 자바가상머신 인터프리터는 [그림 4]의 (a)와 같은 순차적인 실행을 하고 있다. 즉, 하나의 바이트코드를 실행하게 되면, 그 작업의 수행이 끝나기 전까지는 다른 어떤 바이트코드도 수행하지 못하고 대기하고 있게된다. 즉, 한번에 하나의 작업만을 수행할 수가 있다. 반면, 파이프라인을 이용한 인터프리터는 [그림 4]의 (b)과 같은 작업을 수행하게 된다. [그림 4]를 예로 들 때, 순차적인 실행을 할 때 두 개의 작업을 수행하는 시간동안 파이프라인을 고려한 작업 수행에서는 4개의 작업을 수행할 수 있게 된다. 즉, 그만큼의 작업 수행에 있어서 시간을 단축할 수가 있는 것이다. 따라서, 기존 자바가상머신의 인터프리터와 같이 순차적인 방법으로 수행할 경우 실시간, 컴포넌트 등을 고려할 때 저하되는 성능에 대한 부분을 파이프라이닝을 제공하여 보완할 수가 있다.

3.3 컴포넌트를 고려한 인터프리터

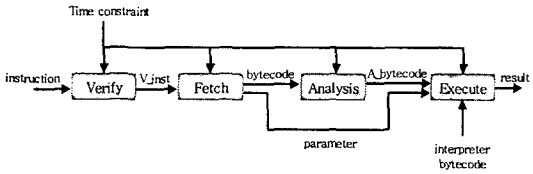
인터프리터의 컴포넌트화를 위하여 본 논문에서는 PBO(Port-Based Object) 모델을 적용하여 인터프리터를 설계하였다. 인터프리터를 구성하는 각 모듈들을 역할에 따라 PBO 모델에 맞는 각각의 PBO를 구상하여 인터프리터를 설계를 한다. PBO 모델을 적용하여 설계를 할 때는 다음과 같은 사항을 고려하여야 한다.

- 인터프리터의 수행순서에 맞게 각각의 포트들이 수행될 수 있도록 설계한다.
- 각각의 포트들에 대한 입력값과 결과값이 무엇인지 정의한다.
- 포트를 수행하면서 요구되는 제약조건에 대해 정의한다.
- 포트를 수행하면서 요구되는 리소스들에 대해 정의한다.



[그림 5] PBO 모델을 적용한 인터프리터

[그림 5]는 PBO 모델을 적용한 인터프리터를 나타낸 것이다. 인터프리터를 구성하는 요소들 중 Main Module에 해당하는 모듈에서 인터프리터의 가장 주된 작업들인 *Verity, Fetch, Analysis, Execute*의 모듈들을 PBO로 구성을 하고, 그 외의 인터프리터 스케쥴러, 메모리 영역과 다른 서브 시스템과의 인터페이스 부분은 PBO 프레임워크(framework)에서 수행되도록 설계하였다. 그리고, Main Module의 모듈 중 PBO로 구성하지 않은 *interpret*, Bytecode 모듈들은 각 모듈의 작업들이 독립적인 작업들이 아니라 Execute 모듈을 수행하면서 수행되는 작업들로 하나의 PBO를 구성하기에는 부적합하여 일반적인 모듈의 형태로 설계 하였다.

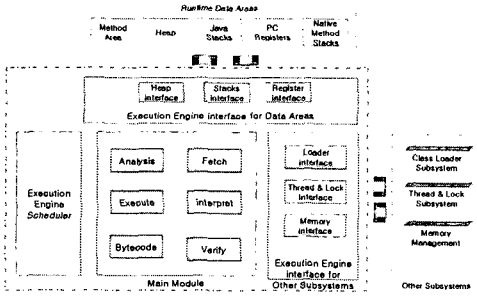


[그림 6] PBO 모델을 적용한 인터프리터 수행도

[그림 6]은 PBO 모델을 적용한 인터프리터의 수행도를 나타낸 것이다. 인터프리터가 수행되면서 처음에 클래스로더로부터 instruction을 전달받아 Verify, Fetch, Analysis, Execute의 순서로 작업을 수행한다. Execute는 작업수행을 위한 리소스로 interpreter, bytecode 모듈을 받아서 수행이 되며, Execute의 작업이 종료되면 수행하고자 했던 instruction이 최종적으로 수행되며 그에 대한 결과가 나오게 된다. 모든 작업들은 수행되면서 제약조건으로 시간제약을 갖게 된다. 수행시간이 무시되면, 하나의 작업이 종료될 때까지 기다려야 하기 때문에 오류등으로 인해 하나의 작업이 종료가 되지 않고 계속 수행이 되는 경우, 다른 작업들을 수행하지 못하는 결과가 발생한다. 따라서, 실시간을 고려하여 설계된 자바가상머신의 스케줄러로부터 특정 시간을 받으면, 그 시간 안에 모든 작업이 수행되어야 하기 때문에, 인터프리터는 주어진 시간 안에 작업을 수행할 수 있도록 시간에 대한 제약을 주어야 한다.

3.4. 컴포넌트 기반 파이프라이닝 인터프리터

[그림 7은]은 본 논문에서 설계하고자 하는 컴포넌트 기반 파이프라인 인터프리터의 전체구조이다.



[그림 7] 컴포넌트 지향 실시간 자바가상머신 인터프리터 구조도

3.4.1 Main Module

Main Module은 interpreter에서 수행되는 요소(코드분석, 코드검증, 코드실행, 코드정의 등)들로 이루어져 있다. 이러한 요소들은 interpreter에서 수행되는 작업들

로 구성되며, 각 요소들을 정리하면 다음 [표 2]와 같다.

[표 2] 실행엔진 interpreter Main Module의 요소별 역할

모 듈	역 할
Verify	• 바이트코드의 검증을 수행하는 부분
Fetch	• 전달받은 instruction을 바이트코드로 fetch하는 작업을 수행
Analysis	• 검증된 바이트코드를 분석하는 부분 • 현재 수행될 바이트코드가 어떠한 명령을 수행해야하는지 분석한다
Bytecode	• 200여개의 바이트코드들에 대한 정의 - 바이트코드가 수행해야 하는 작업 정의
interpret	• 바이트코드 interpreter에 의해 사용되는 일반적인 루틴들에 대한 정의 • 데이터 정렬을 위한 매크로 정의 • 바이트코드 interpreter operationts 정의
Execute	• Java interpreter 실행 루프를 위한 매크로 정의

3.4.2 Execution Engine Interface

인터페이스를 하나로 통합할 경우 메모리 영역에 대한 인터페이스와 다른 서브시스템과의 인터페이스들이 한데 묶여 인터페이스 부분이 복잡해질 가능성이 있다. 따라서 인터페이스 영역을 둘로 나누어 구성을 하였으며 각 영역과 그 역할은 다음 [표 3]와 같다.

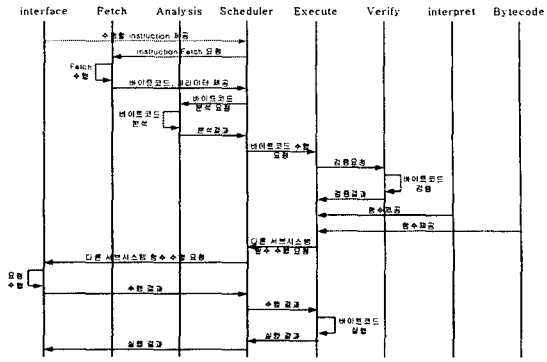
[표 3] 실행엔진 interpreter Main Module의 요소별 역할

	역 할
메모리영역과의 인터페이스	• Heap 영역과의 연계를 위한 Heap interface • Stack과의 연계를 위한 Stack interface • Register와의 연계를 위한 Register interface
다른 서브시스템과의 인터페이스	• 클래스로더와의 연계를 위한 Loader interface • 쓰레드, 락과의 연계를 위한 Thread&Lock interface • Memory Management와의 연계를 위한 Memory interface

3.4.3 Execution Engine Scheduler

실행엔진 스케줄러는 실행엔진이 수행되는 작업을 제어하기 위한 부분으로, Main Module, interface 등에 대한 제어를 한다. 본 논문에서 제안하는 실행엔진 스케줄러는 컴포넌트화를 위해 PBO 모델의 PBO Framework의 역할을 하게 되며, 동시에 파이프라인 인터프리터를 위한 파이프라이닝에 대한 스케줄링을 제공하게 된다. PBO Framework의 역할을 제공하면서, 파이프라이닝을 동시에 고려함으로써, 인터프리터의 컴포넌트화와 성능개선을 동시에 제공할 수 있지만, 파이프라이닝과 PBO를 동시에 제어하면서 데이터의 무결성이나 스케줄링의 충돌 등이 발생할 수가 있다. 데이터의 무결성을 제공하기 위해서는 파이프라이닝에 대한 정책 개선이 필요하며, 파이프라이닝과 PBO를 제어하면서 일어나는 충돌현상은 이것들을 제어하는 PBO Framework 내부의 최적화된 정책이 필요하게 된다.

[그림 8]은 자바가상머신이 실행된 후, 인터프리터가 클래스로더로부터 하나의 instruction을 전달받아 실행하는 과정을 나타낸다.



[그림 8] 인터프리터 수행과정에 대한 시퀀스 다이어그램

6. 결론 및 향후연구

본 논문에서는 컴포넌트 기반 파이프라인 자바가상머신 인터프리터의 설계에 대해 소개하였다. 제안된 인터프리터는 컴포넌트 모델인 PBO (Port-Based Object) 모델을 적용하여 인터프리터의 컴포넌트화를 제공할 수 있도록 하였다. 그리고, 자바가상머신에서의 컴포넌트, 실시간 제공 등으로 인한 인터프리터의 성능저하를 보완하기 위해 파이프라이닝을 지원하는 인터프리터를 제공함으로서, 인터프리터의 융통성과, 성능개선 방법을 제공하여, 내장형 시스템에서 동작하는 자바가상머신에서 보다 효율적인 인터프리터를 제공할 수 있도록 하였다.

본 논문에서 제시한 인터프리터는 융합된 컴포넌트 모델과 파이프라이닝을 보다 효과적으로 수행하기 위한 최적화 방안에 대한 추가 연구가 필요하며, 인터프리터의 보다 개선된 성능을 위한 파이프라이닝의 정책에 대한 연구가 필요하다. 더불어, 인터프리터에서의 실시간성을 지원하기 위한 연구도 필요하다.

참고문헌

[1] Jan Hoogerbrugge, Lex Augustejin. Pipelined Java Virtual Machine Interpreters. *In Proceedings of the 9th International Conference on Compiler Construction.* Springer LNCS, 2000.

[2] David B. Stewart. Software Components for Real Time. Embedded Systems Programmig, December, 2000.

[3] David B. Stewart. Designing Software Component for Real-Time Applications. *2001 Embedded Systems Conference San Francisco, CA*, April 2001.

[4] David B.Stewart, Richard A.Volpe, and Pradeep K.Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. *IEEE Transactions on Software Engineering, VOL. 23, No. 12*, pages 759-776, December 1997.

[5] Vicky H.Allan, Reese B.Jones, Randall M.Lee, and Stephan J.Allan. Software Pipelining. *ACM Computing Surveys*, 27(3), September 1995.

[6] David Gregg, M.Anton Ertl, and Andreas Krall. Implementing an Efficient Java Interpreter. *In High-Performance Computing and Networking (HPCN Eurpoe 2001)*, pages 613-620. Springer LNCS 2110, 2001.

[7] David Gregg, M.Anton Ertl, and Andreas Krall. A Fast Java Interpreter. In Uwe Assmann, editor, *JOSES WORKSHOP at ETAPS'01*, 2001.