LaTTe: 공개 소스 자바 가상 머신과 Just-in-Time 컴파일러

문 수 묵
서울 대학교 전기 공학부

최근 자바는 내장형 시스템에서 대형 서버에 이르는 넓은 응용범위를 가지는 주요 언어가 되어가고 있다. 자바언어로 짜여진 프로그램은 주로 바이트코드 (bytecode) 라 불리는 플랫폼에 독립적인 중간코드로 컴파일되고, 이 바이트코드는 자바 가상 기계 (Java Virtual Machine; JVM) 라 불리는 소프트웨어 계층을 통해 해석 (interpretation)에 의하여 수행 된다. 자바 가상 기계라는 중간 소프트웨어를 통해 프로그램이 수행되기 때문에, 소프트웨어 개발자에게는 플랫폼에 관계없이 한번 작성한 코드를 어느곳에서나 독립적으로 수행시킬 수 있다는 장점이 있으며, 최종 사용자에게는 보안성과 호환성을 보장 받을 수 있다는 장점이 있다.

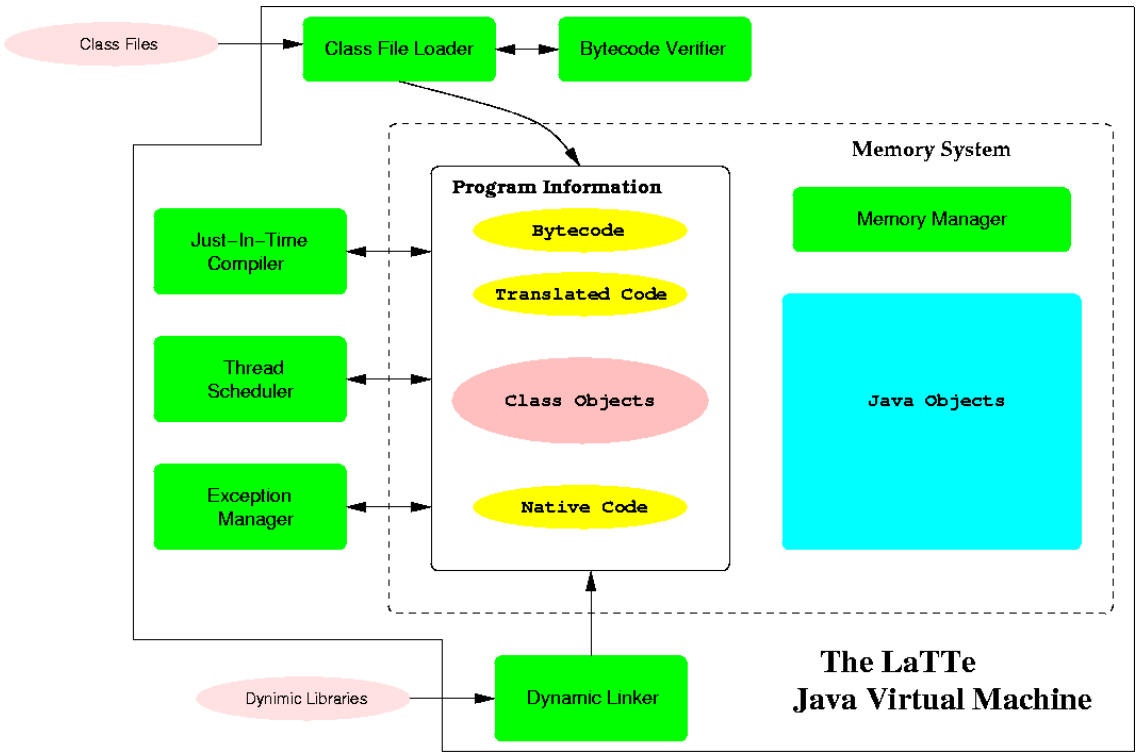
하지만, 프로그램의 수행이 자바 가상 기계라는 소프트웨어 계층을 통해 이루어지는 것은 필연적으로 수행속도의 저하를 수반한다. 이것은 크게 두 가지 측면으로 설명할 수 있다. 하나는 바이트코드를 실행시간에 해석해야 하는 것에서 오는 부담이고, 다른 하나는 자바의 언어적 특징을 지원하기 위해 필요한 쓰레기 처리, 쓰레드 관리, 예외 처리 등에서 오는 부담이다. 특별히 앞의 요인으로 인한 속도 저하를 줄이기 위해 전통적으로 바이트코드 수행을 위해 쓰여왔던 해석기 (interpreter) 방식 대신, 적시 (Just-in-Time; JIT) 컴파일 방식이 많이 사용되고 있다. JIT 컴파일은 자바의 바이트코드를 실행 중에 일정한 단위 (대개의 경우 메쏘드 단위) 씩 먼저 기계어 코드 (native code)로 컴파일하고, 이 컴파일된 코드를 실행 함으로써 실행할 때마다 해석하는 부담을 없애는 기법이다. 보통의 컴파일과 달리 JIT 컴파일은 컴파일에 걸리는 시간도 전체 수행 시간의 일부분이 되기 때문에, 컴파일에 걸리는 속도가 중요하다. 만약, 컴파일로 인한 이득이 컴파일에 걸리는 시간보다 크지 않다면, JIT 컴파일은 오히려 성능을 떨어뜨리는 요인이 된다. 따라서, 자바 가상 기계에서 JIT 컴파일러를 구현하는 데 있어서 많은 공학적 선택과 타협을 필요로한다.

LaTTe 는 본 연구실에서 IBM 왓슨 연구소의 지원을 받아 개발한 적시 컴파일러를 포함하는 자바 가상 기계로서, LaTTe 에 포함된 적시 컴파일러는 RISC 머신, 그 중에서도 UltraSparc 을 목표로 하고 있다. 자바의 바이트코드는 기본적으로 스택 (stack) 머신을 가정하고 있고 RISC 머신은 대부분이 레지스터 기반의 구조를 가지고 있다. 따라서, RISC 머신을 목표로

한 적시 컴파일러에서 중요한 것은 자바 가상 기계 스택에 있는 연산 스택 (operand stack) 과 지역 변수 (local variable) 을 얼마나 효율적으로 레지스터에 mapping 하느냐는 것이다. LaTTe 에서는 바이트코드를 일단 기호 (symbolic) 레지스터를 쓰는 중간코드로 변환한 후, 여기에 레지스터 할당을 행함으로써 최종코드를 얻어낸다. 이 때, 실행되는 바이트코드의 상당 부분을 차지하는 지역변수와 연산스택 사이의 이동 명령들이 복사자 융합 (copy coalescing) 을 통해 사라지게 됨으로써 효율적인 코드 생성이 가능해진다. LaTTe 에서 사용된 레지스터 할당 기법은 기존의 컴파일러에서 많이 사용되는 그래프 컬러링 (graph coloring) 과는 달리 선형시간에 이루어지므로 속도가 매우 빨라 실시간 컴파일러에 적합하다. 또한 이러한 레지스터 할당기 외에도 기존의 컴파일러에서 사용되던 전통적인 코드 최적화 기법을 제한적으로 적용함으로써 빠르게 최적화된 코드를 생성할 수 있다. 전체적인 프로그램의 수행시간이 40-70 초 정도일 때 실시간 컴파일러에서 사용하는 시간은 약 1-2 초 정도로 충분히 빠르다고 할 수 있다. 또한, LaTTe 에서는 자바 가상 기계의 다른 실행 부분에 해당하는 쓰레드 동기화, 예외처리, 쓰레기 처리 등도 최적화하였다. <그림 1> 은 LaTTe 자바 가상 기계의 전체적인 구성을 보여준다. 결과적으로, LaTTe 의 성능은 SUN 의 JDK 1.2 production release 나 JDK 1.3 release candidate client/server VM (HotSpot)등의 상용 자바 가상 머신과 비교하여 손색이 없다. 현재 LaTTe 0.9.1 버전은 최근의 SUN JVM 에 비하여 (elapsed time 기준으로) SPECjvm98 에 대해 평균 25% 그리고 Java Grande 에 대해 평균 30% 정도 좋은 성능을 보여주고 있다.

현재 LaTTe 에서 사용하고 있는 컴파일러 최적화 기법들은 상당히 제한적으로 이루어지고 있는데 이는 실시간 컴파일러의 특성상 시간이 많이 걸리는 최적화 기법을 적용하기 힘들기 때문이며 실제 변환되는 코드들 중에는 실제로 수행되지 않는 부분들도 현저한 양을 차지하고 있기 때문이다. 현재 LaTTe 에서는 코드변환의 단위로 메소드를 선택하고 있는데 이러한 코드 변환 단위를 자주 수행되는 메소드로 제한하고 또한 메소드 안에서도 자주 수행되는 부분으로 제한한다면, 보다 시간이 많이 소모되는 최적화 방법을 사용하더라도 지금보다 더 적은 변환시간으로도 더 최적화된 코드를 얻을 수 있을 것으로 예상하고 있다. 현재 이러한 코드변환 단위를 줄여나가는 것과 보다 집중적인 최적화 방법에 대해 연구하고 있다.

LaTTe 의 소스 코드와 사용법, 그리고 벤치마크에 대한 실험 결과와 관련된 논문들은 http://latte.snu.ac.kr에서 download 받을 수 있다. 지난 1 년간 약 1000 카피가 전 세계적으로 학계와 기업에 배포되었으며 금년 9 월 IBM 으로부터 IBM Faculty Award 를 수상하였다.



<그림 1> LaTTe 자바 가상 기계