

Java 애플리케이션 성능 최적화를 개발 단계에서부터

J Optimizer 2009



데브기어 (02-595-4288, www.devgear.co.kr)

- 종류 : Java 프로파일링/튜닝 IDE
- 특징 : 개발 단계에서 Java 애플리케이션 튜닝을 위한 다양한 성능 최적화 기능 지원
- 평가 : ★★★★★
- 가격 : 별도 문의(238페이지 이벤트 참조)

웹 애플리케이션(Web Application)의 성능 문제는 개발자와 현업 담당자 모두에게 극복해야 할 과제이며 갈등을 빚게 되는 가장 흔한 요인이다. 특히, C/S 프로그램과는 다르게 웹 애플리케이션의 경우 웹 서버(Web Server), WAS, DB 그리고 각종 미들웨어와 리포팅 툴 등으로 복잡하게 구성되어 있다. 이런 복잡성에 따라 성능 문제는 단순한 개발자 역량 차원의 문제를 넘어 서게 되었으며, 소프트웨어 아키텍처와 맞물려 문제점 및 그에 대한 해결책을 찾기란 쉽지 않게 되었다.

성능 문제를 야기하는 구간을 찾기 위해 APM 혹은 Profiling 툴이라고 불리는 툴들이 이미 많이 존재하며(Jennifer, Wily, Open MCM, Jprobe, DEV FOR JAVA, Topaz 등), 이런 툴들은 대체로 WAS의 관점 혹은 End to End 관점에서 각기 다른 데이터를 축적하여 개발자와 현업 담당자에게 정보를 제공한다. 하지만, 이들 툴들의 경우 개발 초기 단계부터 사용하기에는 다소 어려움이 있으며, 개발자보다는 관리자나 현업 담당자의 관점에서 더 초점을 맞추는 경향이 있다. J Optimizer는 엠바카데로사가 볼랜드사의 Optimizeit 기술을 인수한 후 새롭게 개발한 제품으

로서, 개발 초기 단계부터 시스템 취약성을 파악하여 개발을 진행할 수 있도록 유용한 데이터들을 제공해주는 강력한 툴이다. 일반적인 APM 툴들이 개발 완료된 시스템의 운영 단계에서 모니터링에 치중하는데 반해, J Optimizer는 개발 단계에서부터 성능 최적화를 할 수 있게 지원해준다.

Memory and CPU Profiler

J Optimizer는 기본적으로 시스템이 사용하는 Resource에 대해 프로파일링을 할 수 있다. Java Heap Memory, Garbage Collection, CPU Usage, Thread Count, JDBC 연결 상태 등 기본적인 리소스의 사용 추이를 그래프로 표현해 시스템이 리소스를 적절하게 사용하는지 판단할 수 있는 정보를 제공해준다. 즉, 시스템 운용에 있어서 JVM의 Heap Memory Size와 WAS의 Container 구성 등과 같은 시스템의 전반적인 설정 상태의 유효성을 점검하는 데 매우 유용한 정보를 제공한다.

Thread Debugger

시스템의 운영적 측면에서 보았을 때 문제가 되는 프로그램의 원인은 매우 다양하다. 대표적으로 중요 자원의 미반납, SQL의 Bad Response Time 등을 들 수 있다. Connection이나 Statement 객체의 리소스 미해제는 오래 전부터 문제로 대두되어 누구나 중요한 문제임을 인식하고 있다. J Optimizer를 포함한 모든 APM 툴은 이를 지원하므로 이런 자원 반납 문제는 논의로 치더라도, 서버가 어느 날 갑자기 Hang이 걸리면서 정상적인 작동을 못하거나 갑자기 다운되어 버린다면 시스템 운영자는 난



〈화면 1〉
Memory Profiler

감하고 막막해지게 마련이다. 이런 상황에서는 처음부터 모든 로그를 찾아가면서 상황을 유추해 나가거나 서버의 Core Dump를 분석하는 것이 일반적이지만, 이런 방식으로 명쾌한 해답을 얻을 수 있는 경우는 드물다. 실제 상황에서 작동되는 스레드를 모니터링하면서 스레드의 상태 전이를 파악할 수 있다면 시스템 담당자는 어깨의 무거운 짐을 벗어버린 것처럼 홀가분한 기분일 것이다. J Optimizer가 바로 이러한 기능을 제공한다. JVM 상의 모든 스레드의 활동 상태를 감시할 뿐만 아니라 잠재적인 deadlock 상태일 수도 있음을 예견하는 기능을 제공한다. 서버의 hang 현상은 단 하나의 스레드에서 시작되고 이것이 누적됨으로써 다운됨을 시스템 운영자는 많이 경험했을 것이다.

Request Analyzer

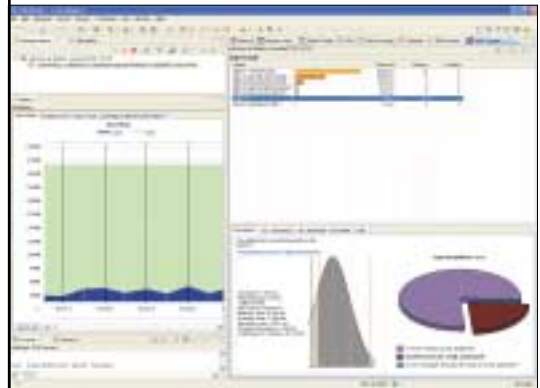
J Optimizer는 Request별 단위 프로세스의 분석이 가능하다. 이는 단순히 시스템 상의 SQL Response Time, Application Response Time의 통계학적 축적이 아니라, Request의 처리에 대한 종합적인 분석을 제공하기 위해 시점을 기준으로 Heap Memory, Thread, Class, JDBC 등의 사용 현황을 Relation이 맺어진 데이터의 형태로 프로파일링할 수 있음을 의미한다. 즉, Request를 처리함에 있어서 시스템에서 문제가 되는 SQL을 찾는 것부터 스레드의 활성 상태까지 파악해 성능 문제를 야기하는 구간을 찾아낼 수 있는 것이다.

Code Audits and Metrics

개발자의 관점에서 지나치게 쉬우면서 민감한 사항들 중의 하나는 클래스, 메소드, 필드 등의 명명 규칙 등을 포함한 코딩 습관의 다양함이다. 프로젝트의 초기에 항상 개발 표준서를 배포하더라도 성능상 큰 문제가 발생하지 않는 이상은 그냥 지나치게 쉬운 문제이다. 하지만 하나의 프로그램을 한 사람이 계속 유지 보수하지는 않으며, 규칙상의 사소한 문제라고 하더라도 이것이 쌓이면 추후에 관리상의 어려움이 발생할 여지가 크다. 더구나 그것이 단순한 코딩 습관의 차원이 아니라, 당장은 정상적인 결과를 내고 있더라도 다양한 상황에서 예외의 위험성을 내포한 잘못된 소스 코드라면 문제는 더 심각해진다. J Optimizer는 기존 프로젝트상에서 곧바로 Audit를 실행함으로써 추가적인 공수 없이 코드 검사를 수행하고 이에 대한 친절한 수정 가이드를 제시해준다. 이는 개발자 각자가 구현 완료 후 간단히 돌려봄으로써 좀 더 향상된 품질의 구현 산출물을 얻을 수 있음을 의미하기도 하지만, 또한 개발 리더나 소프트웨어 아키텍트의 관리 하에 소스 관리 통제가 더 용이해짐을 의미하기도 한다. 수많은 code inspection 툴이 존재하지만 J Optimizer의 간단한 실행과 구체적인 결과물 출력은 단연 돋보이는 기능이다.

Java 튜닝의 마침표, J Optimizer

이 외에도 J Optimizer는 애플리케이션의 죽은 코드 지점을 쉽게 찾아서



〈화면 2〉 Request Analyzer



〈화면 3〉 AuditsMetrics : Kiviat Chart



〈화면 4〉 Code Coverage

제거함으로써 품질을 높이고 애플리케이션을 보다 가볍게 만들도록 지원하는 Code Coverage 기능도 지원하며, 원격 프로파일링을 위한 J Optimizer Agent, 코드 변경 영향 평가를 위한 Progress Tracker 기능도 지닌다. J Optimizer는 BEA WebLogic, IBM WebSphere, Oracle Application Server, JBoss, Tomcat, Geronimo, GlassFish, Jetty 등의 다양한 WAS를 지원한다. +

박지훈 imp@embarcadero.kr