

기술 문서

Java 성능 튜닝의 필수 요소

개발 프로세스 전 과정에서 애플리케이션 스피드, 확장성, 신뢰도 관리하기

알 F. 마나리노, Embarcadero 테크놀로지

2008년 8월

Corporate Headquarters

100 California Street, 12th Floor
San Francisco, California 94111

Asia-Pacific Headquarters

L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

DEVGEAR

서울특별시 서초구
반포동 743-14
데브기어 4층

왜 Java의 성능 튜닝이 중요할까요

요즘과 같은 압축된 개발 사이클, 멀티티어의 애플리케이션 아키텍처, 복합 테크놀로지 환경에서, 많은 조직들은 시의적절하게 확장성 있는 그러나 신뢰할만한 기업용 Java 애플리케이션에 대한 필요를 절감하고 있습니다. 성능 장애물들을 확인 및 수정하기 위해 개발 프로세스 전반에 걸쳐 작은 에너지를 투입하는 것만으로도 형편없이 실행되는 애플리케이션에 대한 리스크와 처리 비용을 극적으로 경감할 수 있습니다. Java 성능 튜닝은 간단히 말하면 직접 만든 코드를 속도, 신뢰성, 확장성, 유지보수라는 측면에서 최적화하는 것입니다. 정말로 확장성 있고, 번개처럼 빠른 Java SE 와 Java EE 애플리케이션을 만들기 위해서는 분명한 목적과 이해할 수 있는 프로그래밍 우선순위가 요구됩니다. 일반 성능 튜닝 사이클을 도입한 경우 가장 큰 이익은 애플리케이션의 어느 부분이 위태로운 병목구간이고, 어느 부분이 효과적으로 동작중인지 즉시 알 수 있게 됩니다.

성능 튜닝: 개발의 좋은 습관

Java 의 큰 장점은 플랫폼 독립적인 바이트코드 방식과 가비지 컬렉션의 자동 핸들링입니다. C/C++과 다르게, 개발자들은 애플리케이션의 비즈니스 요구사항에만 집중할 수 있고, 플랫폼과 관련된 것에서 상당히 자유롭습니다. 그러나 경험이 많은 개발자들은 애플리케이션의 기능에만 포커스를 맞추지 않습니다. 이런 추상화 레벨의 하부에는 가비지 컬렉션, 스레드 스케줄링, Java 가상 머신(JVM)에 의해 관리되는 상당한 작업과 제일 하부에 있는 OS의 패턴과 제약처럼 메모리와 프로세싱 파워의 견고한 한계가 존재하고 있는 것이 현실이기 때문입니다. 성공하는 개발자 —그리고 생산성있는 개발 프로세스 —는 개발 프로세스 전반에 걸쳐 초기 코드 생성 단계부터 QA 테스트와 그 이후까지 일상적으로 성능 분석을 합니다. 주요 개발자가 작은 코드 모듈을 앞단에서 검사하고 테스트한다는 것은 만들어진 Java 애플리케이션이 빠르고, 신뢰할만하고, 확장성이 있다는 것을 보증해주는 가장 좋은 방법입니다.

큰 그림을 안다면 깊게 팔 수 있습니다

최근 Java 애플리케이션 서버 세대는 증가된 메모리 용량과 프로세싱 파워를 제공합니다. 그러나, 이 상황에서도 애초에 결함이 있는 코드를 극복하기 위해서 더 많은 하드웨어를 투입하는 경우도 있습니다. 일단 실제 제품에 포함되면 한 줄의 버그 라인이 파문을 일으켜 애플리케이션 전체의 병목현상을 발생시키거나 미스터리하게 다운되는 방아쇠 역할을 할 수 있습니다. 개발자의 가장 큰 도전은 성능 병목현상 또는 메모리 이슈를 유발하는 부분이 Java SE 인지 Java EE 쪽인지 결정하는 것입니다. Java 와 다양한 플랫폼이라는 장점은 높은 추상화 레벨, 객체의 재활용, 처리 영역과 시스템 의존성을 격리한 것입니다. 그러나 광대한 저수준의 복잡도로부터 보호해주는 캡슐화로 인해 성능 이슈에 대한 여지를 남겨놓게 됩니다. 직관을 확장할 때와 큰 수고를 들이지 않고도 Java 애플리케이션이 어떻게 실행되는지 이해할 수 있도록 도와주는 도구가 필요합니다. Java 처럼 고도로 추상화된 객체지향 언어의 도래에 대해서, 스탠포드 컴퓨터 과학 명예 교수인 도널드 크누스 교수는 프로그래머들은 자신들이 짠 코드가 잘 실행되고 확장할 수 있는지를 결정하는 요소들에 대한 접촉점을 잃어버릴 위기에 처해 있다고 우려를 표명했습니다. “ 우선, 저수준 영역에서 일어나고 있는 자세한 것들을 무시하려고 할 것입니다. 그러나 디버깅을 하게되면, 분석해 볼 수 있는 능력이 부족할 것입니다. 이미 고도로 추상화된 것들만 알 수 있게 되었기 때문입니다.”

성능 도구로 최적화의 달인이 될 수 있습니다

크누스 교수는 개발자들이 코드의 확장성, 신뢰성, 스피드를 위해서 수면 아래 일어나고 있는 일에 대한 통찰력이 필요하다고 조언합니다. J Optimizer 는 Java 가상 머신에 대해서 이러한 통찰력을 줄 수 있는 이해하기 쉽고 파워풀한 정보를 개발자에게 효과적으로 전달해 줍니다. 근원적인 질문은 다음과 같습니다: “ 이 모듈 또는 애플리케이션에 있어서 가장 중요한 성능 이슈는 무엇인가?” Java 성능 튜닝을 위해 특화되어 설계된 Embarcadero J Optimizer 같은 도구는 이 질문에 이상적인 방안을 제시합니다—게다가 코드 향상에 대한 방법과 효율성까지 확신할 수 있게 됩니다. 핵심 Java 문제 영역을 구별하도록 도와주는 도구가 없다면, 애플리케이션의 전반적인 성능에 실제 영향을 주는 중요한 최적화 이슈를 찾느라고 중요하지 않은 영역까지 세부 최적화를 하느라 많은 시간을 허비하기 쉽습니다.

목표는 개발팀 각각의 멤버들이 각 단계의 성능 튜닝에 대하여 스마트하게 대응할 수 있도록 도구에 익숙해지는 것입니다. 현명한 성능 튜닝은 애플리케이션의 전반적인 비즈니스 요구사항의 맥락에서 이루어집니다. 어떤 미세한 성능 이슈는 성능 개선 노력을 보장하지 않을 수도 있습니다. 다른 중요한 최적화 트레이드 오프는, 전체 아키텍처가 수정을 유발할지 이해하는 지점에서, 컴포넌트가 함께 다뤄질 때 발생하게 됩니다.

코드 모듈을 개발하거나 연결할 때마다, 반복적으로 코드를 빠른 성능이 나오도록 튜닝한다면, 프로젝트 종료 시점에서 미칠 것 같은 문제가 발생할 여지를 최소화시키는 가장 좋은 방법입니다. 반대의 경우, 실제 제품에서는 사소한 문제라도 비싸고 복잡한 과정을 통해서 해결해야하는 골치덩이가 되어버립니다. QA 와 고객에게 전달되는 최적화된 애플리케이션은 가볍고, 안정적이고, 확장성이 있으면 억 소리 나게 빠를 것입니다.

결론: Java 성능 튜닝은 아주 중요합니다

성능 목표에 대한 계획, 설계, 테스트는 성능이 떨어지는 애플리케이션 또는 장애가 나타난 다음에 실행하는 것 이상입니다. 개발 프로세스 전반에 걸쳐 코드의 실행에 대해 적절하게 경고를 인지하게 되면, 마지막 단계에서 과도한 비용과 절망적인 상황은 피할 수 있습니다. 빠르고, 확장할 수 있는 최고 성능의 코드는 처음부터 요구되는 설계입니다. (격리된 성능 팀처럼 특별한 스킬이 없는) 개별 개발자들을 위해 개발 프로세스에서 정기적으로 테스트된다는 것, 역시 중요합니다.

팀의 개별 개발자들이 J Optimizer 같은 도구를 사용한다고 전제한다면 아주 간단해집니다: 코드를 작성한 사람만큼 코드 세세한 것을 이해하는 사람은 없을 것이고, 저자 외에 어느 누구도 그 코드의 로직과 구현을 개선할 수 있는 책임자가 되지 못할 것입니다.

EMBARCADERO® J OPTIMIZER™ 로 성능 튜닝이 쉬워집니다

20:80 원칙은 성능 튜닝을 위한 말입니다. 애플리케이션 80%의 성능은 일반적으로 20%도 안되는 코드에 의해서 좌우됩니다. 개발팀이 빠지기 쉬운 함정은 리스크가 많고 비용이 많이 드는 마지막 단계인 애플리케이션 재설계에 이르게 할 정도로 핵심 문제 영역의 확인까지 너무 많은 시간이 지체되는 것입니다. 미리 손을 쓰게 되면 이런 몰아치는 엔드 게임 재작업을 예방할 수 있습니다. J Optimizer 같은 도구를 도입하면 개발팀 모두가 애플리케이션의 스피드, 확장성, 신뢰성에 대해서 개발 프로세스 전반에 걸쳐 인식—오히려 더 적극적으로—하게 됩니다. 개별 루틴 또는 서브 시스템의 기본 기능이 정해져있지만, 모듈의 성능은 애플리케이션의 전반적인 비즈니스 요구사항의 맥락에서 평가되어야 합니다. 코드 조각들이 합쳐질 때 발생하는 어떤 핵심 이슈에 대해서 모든 코드를 동일한 강도로 살펴봐야 될 필요가 없다는 것은 명확합니다. 상황이 어떠하든, 개발 프로세스에서 이슈들의 경중을 정하는 것은 전체 프로젝트를 결길로 빠지게 하거나 실제 제품으로 출시되어 고객센터에서 비즈니스 장애를 일으키게 되는 지연, 급증하는 문제, 깊게 뿌리박힌 성능 결점 같은 것들을 피할 수 있습니다.

필자에 대하여

AI Mannarino는 현재 수석 시스템 엔지니어이자 Embarcadero 테크놀로지, Inc.의 에반젤리스트입니다. Embarcadero에 합류하기 전, 최근 3년간 볼랜드에서 나온 코드기어에서 그리고 그 전 5년간은 ALM/SDO 개발 도구 솔루션 매출을 지원하는 수석 시스템 엔지니어로 볼랜드에서 근무했습니다.

AI은 객체지향 분석설계(OOAD)와 상용 애플리케이션 개발과 배치에 대한 책임을 포함해서 소프트웨어 개발 25년 이상의 경력을 갖고 있습니다. 볼랜드에서 근무하기 전, AI은 Objectivity, Versant, Red Brick Systems, Information Builders 에서 SE 경력을 갖고 있고, 복잡한 전기-기계 시스템에서 돌아가는 실제 애플리케이션 구현을 담당한 Grumman Aerospace에서 전기 엔지니어였습니다. AI은 맨하탄 컬리지에서 전기 공학 학사 학위를 받았습니다.



Embarcadero Technologies Inc.는 애플리케이션 개발자 및 데이터베이스 전문가가 자신이 선택한 환경에서 소프트웨어 애플리케이션을 설계, 빌드 및 실행하는 도구를 사용할 수 있도록 합니다. 전 세계 3백만 이상의 커뮤니티와 Fortune지 선정 100대 기업 중 90개 기업이 Embarcadero의 CodeGear™ 및 DatabaseGear™ 제품군을 기반으로 하여 생산성을 향상시키고 개방적인 협업 및 자유로운 혁신을 추구하고 있습니다. Embarcadero는 1993년에 설립되어 캘리포니아 샌프란시스코에 본사가 있으며 전 세계에 사무소를 두고 있습니다. Embarcadero의 온라인 주소는 www.embarcadero.com입니다. Embarcadero의 주요 제품인 DatabaseGear의 도구에는 ER/Studio®, DBArtisan®, Rapid SQL® 및 Embarcadero® Change Manager™가 있습니다.



데브기어는 미국 Embarcadero Technologies Inc.와 기존의 코드기어 한국 지사의 협력으로 전략적으로 설립된 엠바카데로 솔루션 전문 공급 기업입니다. 데브기어는 Delphi, C++Builder, JBuilder, Delphi Prism 등 개발툴 제품들과 ER/Studio, PowerSQL, DB Artisan, EA/Studio 등의 데이터베이스 툴 제품들에 대한 한국 시장에 공급은 물론 기술지원 및 교육을 제공합니다. 데브기어 웹 사이트는 <http://www.devgear.co.kr/> 이며 제품에 대한 문의는 ask@embarcadero.kr 로 하면 됩니다.