제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

컴포넌트 분석 큐를 적용한 자바 가상머신 스케줄러 설계

기영택*, 이승룡**
*경희대학교 전자계산학과
e-mail:{li2th, sylee}@oslab.kyunghee.ac.kr

Design of Java virtual machine scheduler using component analysis queue

Youngtek Ki*, Sungyoung Lee**
*Dept of Computer Engineering, Kyung Hee University

요 약

내장형 시스템 환경에서는 급변하는 시장의 적시성 요구(time-to-market)와 저렴한 비용으로 다양한 사용자의 요구사항을 효율적으로 반영해야 한다. 그러한 특징에 의해 내장형 시스템 환경에서는 재구성 가능한 컴포넌트 기반 소프트웨어 개발방법이 주목받고 있다. 또한 내장형 시스템 소프트웨어 개발에는 이식성, 신뢰성, 재사용성에서 강점을 가진 자바 가상머신이 주목받고 있다. 따라서 본 논문에서는 컴포넌트 기반 내장형 자바 가상머신에 적합한 스케줄러를 개발하기 위해, 추상 컴포넌트 조립기법과 의존성 검사 방법을 적용한 컴포넌트 스케줄링 큐를 적용하여, 컴포넌트 간의 문맥교환을 줄여 수행성능을 향상시킨 스케줄러의 설계 기법에 대하여 소개한다.

1. 서론

차세대 컴퓨터 시대를 맞이하여 내장형 시스템에 대한 관심이 증대되고 있어 내장형 시스템 개발에 대한 연구가 많이 진행되고 있다. 내장형 시스템 환경에서는 시장의 적시성 요구(*time-to-market*)와 저렴한 개발비용으로 다양한 사용자 요구사항을 반영하기 위하여 재사용과 재구성이 가능한 컴포넌트기반 소프트웨어 개발 방법이 주목받고 있다.

재활용 성을 높이기 위해, 작은 컴포넌트로 구성되어진 컴포넌트 기반 시스템은 동작할 때, 잦은 컴포넌트간의 문맥교환(Context-Switching)으로 인한 오버헤드가 발행한다. 이러한 부하에 따른 속도 저하를 막기 위한 방법으로 컴포넌트 조립기법이 있다. 컴포넌트 조립기법은 일련의 순서성을 가진 컴포넌트들을 조립하여 하나의 컴포넌트로 만드는 기법으로 컴포넌트의 개수를 줄여서 문맥교환에 대한 오버헤드를 줄일 수 있다.

한편, 내장형 시스템 소프트웨어 개발에는 이식성, 신뢰성, 재사용성에서 장점을 가지고 있는 자바가 개발언어로 급부상하고 있다.

컴포넌트 기반 자바가상머신은 기존 자바가상머신의 모듈을 더욱 세분화해서 컴포넌트로 구성한다. 기존 자바 가상머신의 모듈 기반 함수 호출 구조에 최적화된 스케줄러로는 컴포넌트 간의 문맥교환 구조를 갖는 컴포넌트 기반 자바 가상머신에서 수행 성능 저하가 예측된다.

※이 논문은 2002년도 한국 과학재단 목적기초연구(과제번호: R01-2001-000-00357-0)의 연구비에 의하여 연구되었음

따라서, 본 논문에서는 컴포넌트 기반 자바 가상머신의 수행성능 향상을 위해 추상 컴포넌트 레이어 기법을 적용한 자바 가상머신 스케줄러 설계 기법에 대하여 소개한다.

본 논문에서 제안한 스케줄러는 동적인 환경에서 컴포넌트를 추상 컴포넌트 레이어로 결합하여 컴포넌트간의 문맥교환에 드는 부하를 줄이는 추상 컴포넌트 레이어 기법과, 각각의 컴포넌트가 결합이 가능한지 분석하는 의존성 분석 기법을 수용한 컴포넌트 분석 큐를 적용하여 수행성능을 향상시킨다.

동적 컴포넌트 기반 자바 가상머신에 있어서 추상 컴포넌트 레이어 기법 적용으로 문맥교환 빈도를 낮춤으로 수행성능이 향상되는 특징이 있고 의존성 분석 기법의 적용으로 환경에 맞는 수준으로 추상 컴포넌트 레이어기법을 최적화 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 자바 가상머신 스케줄러와 컴포넌트 프레임워크인 PBO, VEST에 대해 소개하고, 3장에서는 본 논문에서 제안하는 컴포넌트 분석 큐를 적용한 자바 가상머신 스케줄러의 설계에 대해 설명하며, 4장에서는 결론을 맺으며 향후 과제에 대해 언급한다.

2. 관련연구

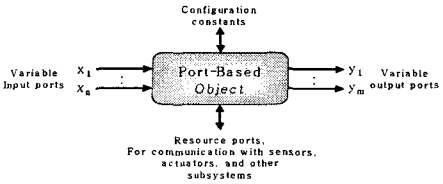
본 장에서는 컴포넌트 모델인 PBO(Port-Based Object)와 컴포넌트 툴셋인 VEST(Virginia Embedded Systems Toolset)에 대해 살펴본다.

PBO는 제안할 스케줄러를 적용할 자바 가상머신 시스

템을 컴포넌트화 하는 기법이며, VEST는 내장형 실시간 시스템을 위한 분석 및 의존성 기법 부분에 있어서 뛰어난 시스템이다.

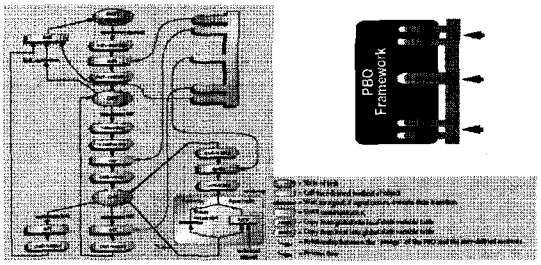
● PBO(Port-Based Object) 모델

PBO[1,2,3]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 최소화이며 [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 자바 가상머신의 각 컴포넌트를 생성할 경우, 융통성은 보장이 되지만 실행성능이 낮다는 단점이 있다.

●VEST(Virginia Embedded Systems Toolset)

VEST[4, 5]는 Univ of Virginia에서 만들었으며 내장형 시스템을 위한 컴포넌트 툴셋으로 컴포넌트기반 실시간 시스템의 개발, 구현, 평가를 목적으로 한다.

VEST는 내장형 시스템에 대한 원활한 지원을 위해서 infrastructure와 다양한 의존성 검사기능을 지원한다.

infrastructure는 계층적 컴포넌트 조립기술로써, 기반이 되는 소형 컴포넌트를 조립하여 하나의 컴포넌트를 만든다. 다수의 컴포넌트간의 문맥교환에 필요한 오버헤드를 줄이고 명시적인 시스템 구성에 도움을 준다.

의존성 검사기술은 컴포넌트들이 조립되어 구성되는 컴포넌트 기반 시스템이 요구사항에 적합하게 구성되어있는지 검사하는 시스템으로 각각의 역할은 [표 1]과 같다.

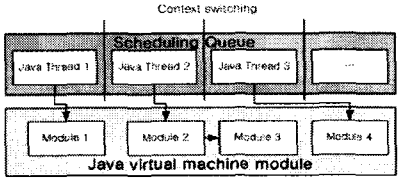
기능	검사 수준 및 주요 검사항목
factual checking	• 컴포넌트별 검사 • WCET(worst case execution time) • timing constraints
inter-component	• 컴포넌트 간의 관계 검사 • 선행관계(precedence requirements) • 동기화(exclusion requirements)
aspect	• 명세화 될 수 없는 시스템의 특정 검사 • 종단간 데드라인 검사(End-to-end deadline)
general	• 시스템 전체적인 요구사항을 검사 • 데드락(Dead-lock)의 적용여부

[표 3] dependency check의 세부 기능

의존성 검사수준이 높으면 안정성이 증가하지만 수행성능이 낮아지는 단점이 있다.

●자바가상머신 스케줄러

자바가상머신 스케줄러는 자바 소프트웨어가 요구하는 쓰레드와 자바 가상머신 내부에서 관리 목적으로 사용하는 쓰레기 수집기와 객체 최종처리기 등을 스케줄링 한다.



[그림 3] 스케줄링 큐

[그림 3]은 스케줄링 큐에서 스케줄링 된 자바 쓰레드가 자바 가상머신의 모듈을 호출하는 흐름을 나타낸다. 이런 구조는 모듈 기반의 함수호출 구조의 자바 가상머신에는 적합하나 컴포넌트 기반 자바 가상머신에는 적합하지 않다. 다수의 컴포넌트로 구성되어있는 컴포넌트 기반 자바 가상머신의 경우 자바 쓰레드에서 자바 가상머신을 구성하는 컴포넌트를 호출하였을 때, 그 컴포넌트가 연속적으로 다른 컴포넌트를 호출할 경우 컴포넌트들 간에 문맥교환에 따른 오버헤드가 발생한다.

3. 컴포넌트 기반 자바가상머신을 위한 동적인 계층형 조립 기법을 적용한 스케줄러

3.1 추상 컴포넌트 레이어의 적용

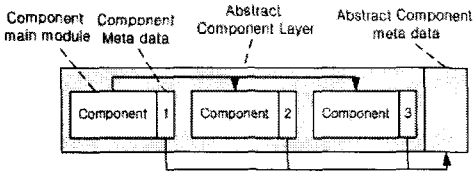
추상 컴포넌트 레이어 기법은 계층적 컴포넌트 조립 기법을 동적 환경에 최적화 시킨 것으로 계층적 컴포넌트 조립 기법과 유사하나, 결과물이 완전한 컴포넌트가 아니라는 점에서 차이점을 보인다. 추상 컴포넌트 레이어의 특징은 컴포넌트의 성격과 특성을 포함한 추상 레이어로, 하나의 완벽한 컴포넌트가 되기 위해서는 컴포넌트가 가져야할 복잡한 조건들을 만족해야 하지만, 추상 컴포넌트 레이어의 경우 프레임워크나 시스템이 요구하는 최적화된 조건으로 구성된다.

추상 컴포넌트 레이어는 포함 된 컴포넌트들의 메타데이터를 분석 및 통합하고, 필요한 데이터를 추출하여 추상 컴포넌트 레이어의 메타데이터를 갱신한다. 위와 같은 과정은 컴포넌트 프레임워크가 추상 컴포넌트 레이어의 요구사항을 분석할 수 있도록 한다.

각각의 컴포넌트의 수행 모듈을 직접 연결하여 추상 컴

포넌트 레이어에 포함된 컴포넌트 간에는 문맥교환이 아닌 직접적인 호출 구조로 치환된다. 추상 컴포넌트 레이어 안에 존재하는 컴포넌트 간에 문맥교환이 단순한 함수 호출 관계로 전환됨으로 수행의 성능의 상승이 있다.

추상 컴포넌트 레이어 구성은 [그림 4]와 같다.



[그림 4] 추상 컴포넌트 레이어
(Abstract Component Layer)

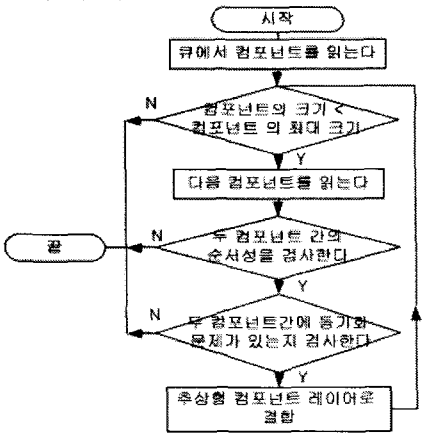
추상 컴포넌트 레이어는 포함하고 있는 컴포넌트들 간을 직접적인 호출관계로 연결하고 각각의 수행정보를 포함한 메타데이터를 추출해 추상 컴포넌트 레이어에 적합하게 수정하여 자신의 메타데이터를 갱신한다.

3.2 의존성 검사(Dependency check)

컴포넌트 기반 시스템을 구성할 때 각각의 컴포넌트가 정상적으로 동작하는지, 시스템이 컴포넌트들의 요구사항을 만족하는가에 대한 의존성 검사가 필요하다.

추상 컴포넌트 레이어 기법을 사용하기 위해서는 추가적으로 컴포넌트들이 추상 컴포넌트 레이어로 결합이 가능한지 추가적인 의존성 검사가 필요하다. 의존성 검사로는 선행관계 분석, 동기화 관련 검사와 같은 컴포넌트 간 의존성 검사와, 시스템의 메모리 자원을 고려한 추상 컴포넌트 레이어의 크기 제한 검사와 같은 전체적인 의존성 검사가 있다. 또한 컴포넌트 간의 의존성은 알고리즘의 순환(cycle) 고착을 피하기 위해서 단방향으로 구성된다.

선행관계 분석은 연속된 컴포넌트들이 순서성을 가지고 있는지 검사한다. 선행관계 분석은 가장 기본적인 의존성 검사 기법이다. 동기화 관련 검사는 컴포넌트들의 협력 관계를 검사한다. 동기화 관계에 있는 컴포넌트는 추상 컴포넌트 레이어에 포함될 수 없다. 크기 제약 검사는 내장형 시스템에서의 메모리 제약을 고려하여 결과물의 크기를 제한한다. 의존성 검사의 세부 과정은 [그림 5]와 같다.



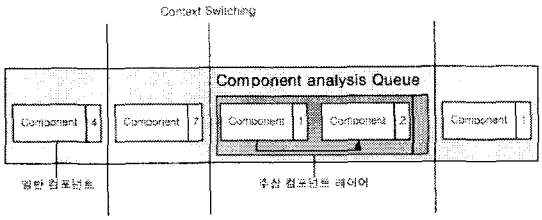
[그림 5] 의존성 검사 순서도

의존성 검사의 순서는 수행시간 단축을 위해 검사 알고리즘의 복잡도가 낮은 순서로 배치하였다.

3.3 컴포넌트 분석 큐

자바 언어로 작성된 소프트웨어의 쓰레드가 자바 가상머신 내부 컴포넌트를 호출하면, 컴포넌트 분석 큐는 호출되는 컴포넌트들을 순서대로 정렬하여 배치한다. 또한 의존성 검사 기법에 의해 순서성, 동기화 조건, 크기 제약 조건을 만족하는 컴포넌트 들을 실제적으로 추상 컴포넌트 레이어로 합치는 기능을 한다.

컴포넌트 분석 큐는 자바 소프트웨어 쓰레드가 자바 가상머신의 내부 컴포넌트를 호출한 시점에 컴포넌트를 재구성(reconfigure) 한다. 재구성 시점을 명시하는 것은 재구성 정책을 명확하게 하고 시스템의 분석을 용이하게 한다. [그림 6]은 컴포넌트 분석 큐의 동작과정을 나타낸 것으로 컴포넌트의 의존성을 검사하여 문맥교환을 줄이는 것을 보인다.

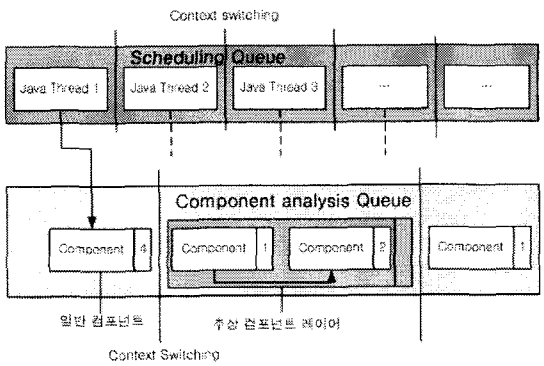


[그림 6] 컴포넌트 분석 큐

컴포넌트 분석 큐는 정렬된 컴포넌트 간에 의존성 검사 후, 컴포넌트들을 추상 컴포넌트 레이어로 결합한다.

3.4 컴포넌트 분석 큐를 적용한 자바 가상머신 스케줄러

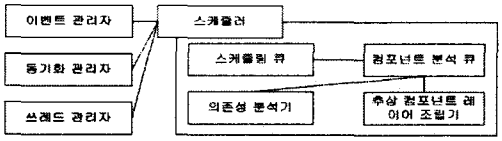
컴포넌트 분석 큐를 적용한 자바 가상머신 스케줄러는 기존의 스케줄러와 비교해서 호출관계의 변화에 의한 성능 향상에 이점을 갖는다. 스케줄링 큐에서 자바 소프트웨어 쓰레드가 자바 가상머신 내부의 컴포넌트를 호출하면, 컴포넌트 분석 큐는 컴포넌트들을 정렬하고 분석해서 추상 컴포넌트 레이어 기법을 이용하여 추상 컴포넌트 레이어로 조립한다. 다수의 컴포넌트를 포함한 추상 컴포넌트 레이어는 추상 컴포넌트 레이어가 포함한 컴포넌트들 간의 문맥 교환이 직접적인 호출 관계로 대체됨으로 수행성능이 향상된다.



[그림 7] 컴포넌트 분석 큐가 적용된 스케줄링 큐

[그림 7]은 컴포넌트 분석 큐가 적용된 스케줄링 큐를 나타낸 것으로 추상 컴포넌트 레이어의 적용으로 문맥 교

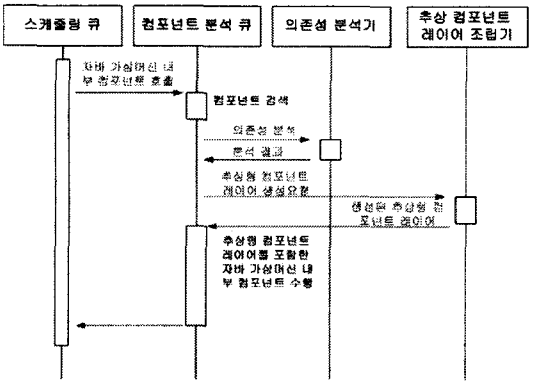
환이 줄어든 것을 볼 수 있다.



[그림 8] 스케줄러 및 관련 모듈 구성도

[그림 8]은 제안된 스케줄러와 관련된 모듈간의 관계를 보인다.

스케줄링 큐에서 자바 가상머신 내부로의 호출이 발생하면 스케줄링 큐는 컴포넌트 분석 큐를 호출한다. 컴포넌트 분석 큐는 요구된 컴포넌트를 정렬하고 의존성 분석 결과에 따라서 추상 컴포넌트 레이어 조립기를 호출하여 컴포넌트를 조립한다. 각각의 모듈간의 호출관계는 [그림 9]와 같다.



[그림 9] 시퀀스 다이어그램

4. 결론 및 향후연구

본 논문에서는 컴포넌트 기반 자바가상머신을 위한 컴포넌트 분석 큐를 적용한 스케줄러의 설계에 대해 소개하였다. 제안된 스케줄러는 컴포넌트를 조립하여 컴포넌트 문맥 교환 빈도를 낮추는 추상 컴포넌트 레이어 조립 기법과 각각의 컴포넌트가 조립가능한지 파악하는 의존성 분석 기법을 제공함으로서, 컴포넌트 모델인 PBO 모델에 의해 설계된 내장형 자바 가상머신을 위한 수행성능이 향상된 스케줄러를 제공할 수 있도록 하였다.

본 논문에서 제시하는 컴포넌트 분석 큐를 적용한 자바 가상머신 스케줄러의 특징은 다음과 같다. 추상 컴포넌트 레이어 기법을 적용한 동적 컴포넌트 결합 기법의 적용함으로 수행 성능을 높이고, 의존성 검사 기법의 적용으로 동적 컴포넌트 결합의 안정성을 보장한다.

본 논문에서 제시한 스케줄러는, 추상 컴포넌트 레이어 기법이 포함하는 컴포넌트의 메타데이터 수준에 대한 최적화에 대한 연구가 필요하다. 또한 컴포넌트 기반 자바가상머신에 적합한 의존성 검사수준의 최적화 방안에 대한 연구도 필요하다. 추가적으로 실시간 시스템으로 확장을 위한 컴포넌트들의 수행시간 분석기법에 대한 연구가 필요하다.

참고문헌

[1] 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.

[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] John A. Stankovic, VEST: A Toolset For Constructing and Analyzing Component Based Operating System For Embedded and Real-Time Systems, 2000, University of Virginia.

[5] J. Stankovic, et. al, VEST: Virginia Embedded Systems Toolkit, IEEE/IEE Real-Time Embedded Systems Workshop, Dec 2001.

[6] Premysl Brada, Metadata Support for Safe Component Upgrades, University of West Bohemia in Pilsen, 2002.

[7] J. Goslin, B. Joy, and G. Steele, Java Language Specification, Sun Microsystems, 1996.

[8] Tim Lindholm, Frank Yellin, The Java Virtual Machine Specification, Sun Microsystems, 1996.