분산 실시간 시스템에서 실시간성 분석을 위한 통신 요소 분석

구현우° 홍영식

동국대학교 컴퓨터공학과

{ hwgoo°, hongys}@dongguk.edu

Analysis of Effecting Factor of Communication for Timing Analysis in Distributed Real-Time Systems

Hyun-Woo Koo° ,Young-Sik Hong

Dept. of Computer Engineering, Dongguk University

요 약

실시간 시스템은 논리적 정확성뿐만 아니라 시간적 정확성을 요구한다. 시간적 정확성을 만족시키기 위해서 실시간 시스템의 설계자는 작업들의 스케줄 가능성에 대한 연구를 선행하여야 한다. 스케줄 가능성 분석을 위해 작업들에 대한 실행 시간 예측이 필요하다. 작업들의 실행 시간 예측을 위한 방법으로 측정과 정적 분석이 연구되었다. 측정 및 정적 분석은 비용 및 확장성에 문제점을 지니고 있고 실시간 시스템의 발전을 따라가지 못하여 분석 결과의 정확성이 만족스럽지 못하다. 본 논문에서는 정적 분석을 단일 시스템이 아닌 분산 실시간 시스템에 적용할 수 있는 확장된 정적 분석 도구의 개발을 위해 분산 실시간 시스템으로 전환에 의해 발생되는 통신 영향 요소의 분석 및 통신 영향 요소 분석기를 설계한다. 실행 시간에 영향을 미치는 요소들의 분석을 통해 원시 프로그램에서 자동적으로 예측된 실행 시간의 정확도와 신뢰도를 높인다.

1. 서 론

실시간 시스템의 가장 큰 특징은 시스템에서 요구되는 논리적 정확성, 뿐만 아니라 시간적 정확성을 필요로 한다. 이는 시스템의 올바른 결과뿐만 아니라 어느 정해진 시간 내에 결과를 도출하여야 함을 뜻 한다. 시간적 정확성을 만족시키기 스케줄 가능성(Schedulability)에 대한 연구가 필요하다.

스케줄 가능성 분석은 실시간 시스템 설계에 가장 중요하고 어려운 작업이다. 스케줄 가능성 분석을 위해 시스템에서 작동할 원시 프로그램의 실행 시간 분석 및 예측이 선행되어야 한다.

사용자들의 컴퓨팅 파워에 대한 요구가 증대되고 하드웨어의 발달 및 네트워크의 발달로 인해 실시간 시스템에서 작동될 많은 프로그램들이 복잡해지고 프로그램의 크기가 증가되었다. 단일 실시간 시스템을 이용하던 사용자들이 분산 실시간 시스템을 이용하여 보다 효율적인 시스템 구축을 원하게 됨으로써 기존의 정적인 실행 시간 분석만으로는 정확하고 신뢰할 수 있는 최악실행 시간의 예측이 불가능하게 되었다.[1]

분산 실시간 시스템에서의 정확하고 신뢰할 수 있는 실행 시간의 예측을 위해서는 운영체제 및 통신에 의한 부하를 예측하여 기존의 정적 분석 기법 추가하여 확장된 정적 분석 기법이 필요하다. 이에 본 논문에서는 분산 실시간 시스템에서 통신에 의해 발생되는 부하의 정적이고 자동적인 분석을 위한 실시간성 통신 요소의 분석 및 설계를 제시한다.

본 논문의 2장에서는 정적 최악 실행 시간 분석을 위한 관련 연구를 살펴보고 3장에서는 분산 실시간 시스템 설계의 편이성을 제공하기 위해 개발된 TMO에 관해 살펴보고 실시간성 분석도구의 전체 구조를 살펴본다. 4장에서는 3장에서 살펴본 분산 실시간성 분석에 영향을 미치는 요소 중 통신 영향 요소를 분산 시스템을 운용하기 위한 가장 간단한 RPC(Remote Procedure Call)와 관련된 리눅스 커널 소스의 분석을 통해 살펴본다. 5장에서는 향후 연구과제로 분석된 요소들과 실시간성 분석 도구의 통합을 다룬다.

2. 관련 연구

실시간 시스템의 중요한 요소인 시간적 정확성을 만족하기 위해 작업들의 스케줄링 기법에 관한 연구가 진행되어 왔고 그와 병행하여 작업들의 스케줄 가능성의 분석을 위한 실시간 시스템에서 실행될 프로그램의 최악 실행 시간 분석에 관한 연구 또한 활발히 진행되어 왔다.

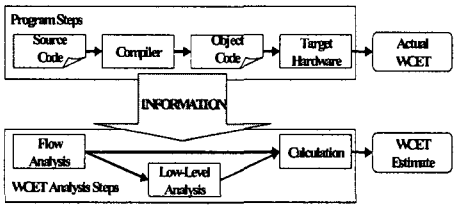
실행 시간 분석을 위한 기법으로는 크게 측정과 정적 분석으로 나눌 수 있다. 측정 기법은 가능한 모든 테스트 케이스를 시뮬레이션을 통해 작업의 스케줄 가능성을 판단한다. 이 기법의 가장 큰 문제점은 프로그램에 대한 가능한 모든 테스트 케이스의 설정이 어렵고 이에 따른 많은 시간과 비용을 초래한다. 하지만 비교적 정확한 실시간성 분석이 가능하기 때문에 현재 많은 실시간 시스템의 설계시 측정을 통해 실시간성 분석을 진행한다.

측정의 문제점인 많은 시간과 비용을 감소시키기 위해 정적 분석기법들이 개발되었고 현재 상용화에 이르렀다.

본 연구는 정보통신부 대학 IT연구센터 육성•지원사업의 연구결과로 수행되었습니다.

[그림 1]은 전통적인 정적 최악 실행 시간 분석 기법을 보여준다. [그림 1]에서 실시간성 분석을 위해 원시 프로그램 단위 요소의 흐름을 분석하는 Flow 분석과 하드웨어 영향 요소의 분석, 그리고 분석된 요소들의 실행 시간을 계산하는 단계로 나뉜다. 하드웨어의 요소에 대한 연구는 캐쉬와 파이프라인에 대해 많은 연구가 진행되었다. [3, 6]

그리고 분석된 요소들의 실행 시간을 계산하는 단계도 많은 연구가 진행되어 왔다. 이 단계에 적용되는 기법으로는 Tree-Based Calculation, Path-Based Calculation 그리고 IPET-Based Calculation이 있다.[2]

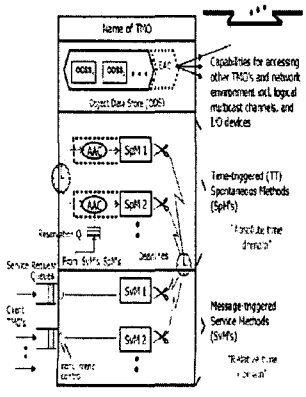


[그림 1] 전통적인 정적 WCET 분석 기법

측정 기법에 의해 실행 시간 예측의 부정확성과 더불어 점차 발전하는 컴퓨팅 파워와 분산 기술에 의해 단일 실시간 시스템 환경에서 분산 실시간 시스템 환경으로 프로그램의 크기가 거대해지고 점점 더 복잡해짐에 따라 앞서 살펴본 정적 실시간성 분석 도구의 정확성 문제점이 언급 되고 있다. 동적 입력에 대한 문제점 해결 및 보다 정확한 실행 시간 분석의 필요성이 대두되고 있다. [1, 4]

3. 실시간성 분석 도구

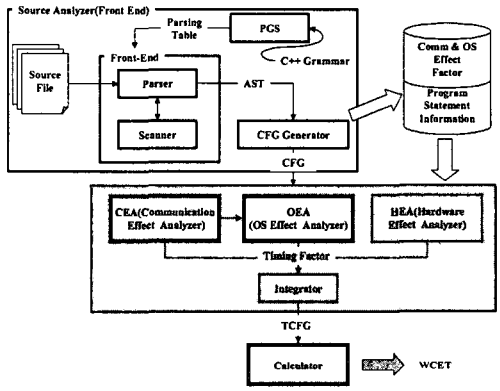
신뢰성 있는 분산 실시간 환경 구축을 위해 TMO(Timg-triggered Message-triggered Object)를 이용하여 기존 실시간 시스템 환경에 객체 지향 개념을 적용하고 원격 객체(Remote Object) 호출에 대한 시간적 처리 과정, 시스템 및 데이터의 동기화를 부여하여 통신에 의해 발생할 수 있는 응답 시간의 미보장성을 줄여 개발자가 고려해야 하는 통신에 대한 문제를 최소화한다.



[그림 2] TMO의 기본구조

[그림 2]는 TMO의 기본 구조를 나타낸다. SpM과 SvM의 이용을 통해 작업의 동기화를 부여하여 시간적 정확성을 높이고 SvM을 통해 객체간의 통신을 손쉽게 설계 및 구현이 가능하다. TMO를 이용한 분산 실시간 시스템의 설계는 객체 지향 개념의 적용을 통해 체계적인 설계가 가능하다.

그러나 분산 실시간 시스템의 설계시 선행 되어야 하는 스케줄 가능성 검사를 위한 많은 실행 시간 분석 도구들은 구조적 프로그램 분석 방법을 택하고 있기 때문에 객체 지향 프로그램 분석 방법의 개발이 필요하다.[5, 7]



[그림 3] 실시간성 분석 도구

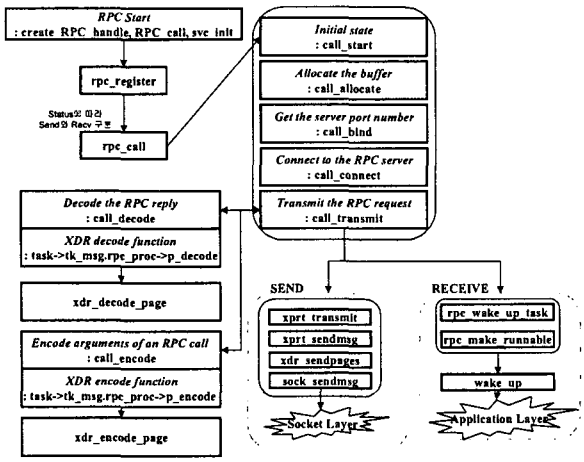
[그림 3]은 분산 실시간 시스템을 위한 실시간성 분석 도구의 전체적인 구조를 나타낸다.

소스 분석기는 컴파일러의 Front-End에 해당하는 작업을 진행하여 AST(Abstract Syntax Tree)를 먼저 생성한다. CFG Gen(Control Flow Graph Generator)의 역할은 입력으로 받은 AST를 순회하면서 원시 프로그램의 동작 의미를 가지는 문장별 흐름을 파악한 후 이후에 사용할 정보를 추출한다. 통신 영향 요소 분석기 및 운영체제 영향 요소 분석기에 사용될 통신, I/O 그리고 메모리 할당에 해당하는 문장 및 시스템 콜을 분석하여 저장한다. 통신, 운영체제 그리고 하드웨어 영향 요소 분석기는 원시 프로그램에서 추출된 영향 요소들의 시간 영향 요소를 분석 및 통합하여 TCFG(Timing Control Flow Graph)를 생성한다. 본 연구에서 계산(Calculation)의 단계는 관련 연구에서 언급한 TBC(Tree-Based Calculation)을 기반으로 하여 최악 실행 시간을 예측한다.

4. 통신 영향 요소 분석

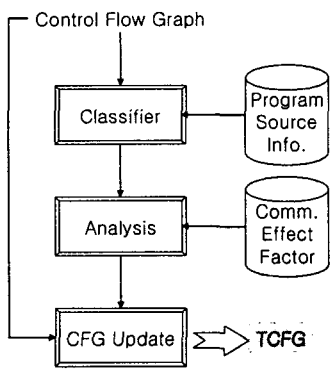
본 논문에서는 실시간성 분석 도구의 구성 요소 중 CEA(Communication Effect Analyzer)를 선행 과제로 한다. RPC(Remote Procedure Call)을 기초로 하여 분석을 진행한다. 클라이언트와 서버 모델에서 RPC가 실행 시간에 영향을 미치는 통신 요소는 크게 세 부분으로 나뉜다. 클라이언트에서 통신을 위한 준비 작업, 서버 측에서 해당 요청을 받고 응답을 준비하는 작업 그리고 통신 지연 시간으로 나뉜다. 통신 지연 시간은 통신 매체 및 사용량에 따

라 많은 변화를 가지므로 본 연구에서 통신 지연 시간을 일정함을 전제로 하여 클라이언트 측과 서버 측에서 통신을 위해 준비하는 작업을 먼저 분석한다. 동적인 통신 지연 시간의 예측에 관한 연구는 향후 연구과제로 남겨두기로 한다.



[그림 4] 리눅스 커널과 관련된 RPC Flow

[그림 4]는 RPC의 라이브러리 함수 흐름도를 나타낸다. 가장 먼저 RPC 등록을 한 후 서버와 클라인언트를 연결하고 데이터 전송시 마샬링(Marshalling) 또는 언마샬링(Unmarshalling) 과정을 거친 후 리눅스 커널의 네트워크 모듈 중 소켓 레이어에 해당하는 시스템 콜을 호출한다. 리눅스 커널에서 네트워크 모듈은 소켓 계층, 전달 계층, 연결 계층 마지막으로 물리 계층을 거치게 되고 각각의 계층별 커널 소스 분석을 통해 각 모듈 및 라이브러리 함수들의 실행 시간을 예측하여 통신에 관련된 정보를 저장한다. 이 때 저장되는 통신에 대한 정보를 통신 영향 요소(Communication Effect Factor)라 부른다. 모듈 분석은 Bottom-up 방법을 이용하여 최하위 단일 모듈에서 점차 최상위 모듈로 분석된 각 요소를 결합하여 통신에 의해 생기는 영향 요소의 TF(Timing Factor)를 생성한다.



[그림 5] CEA 동작 흐름도

[그림 5]는 통신 영향 요소 분석기(CEA)의 전체 동작 흐름도를 보여준다. 프런트 앤드에서 작성된 CFG의 노드 중 통신과 관련된 노드를 추출하고 프로토콜, 송신 및 수신 구분, 라우터 동작 여부, 데이터 크기 그리고 보안성 등 통신 정보 분석을 분류자(classifer)에서 담당한다. 이러한 정보를 이용하여 미리 분석된 통신 영향 요소와 비교 분석하고 시간 요소를 계산하는 작업을 분석(analysis)단계에서 진행한다. 마지막으로 프런트 앤드에서 작성된 CFG에 계산되어진 시간 요소를 추가하여 시간 요소가 적용된 TCFG(Timing Control Flow Graph)를 생성함으로써 통신 영향 요소의 분석을 종료한다.

5. 결론 및 향후 연구 과제

본 연구에서는 분산 실시간 시스템을 위한 실시간성 분석 도구의 설계에 있어 통신에 의해 발생하는 오버헤드를 통신 영향 요소라 정하고 리눅스 커널 소스를 기초로 이를 분석하기 위한 모델 설계를 논하였다. 통신 영향 요소들의 정형화를 통해 정보의 사용 및 저장을 용이하게 하는 방안을 제안하고 구현을 향후 연구과제로 남긴다. 그리고 실시간 시스템에서 고려되어야 할 영향 요소 중 하나인 운영체제 영향 요소의 분석을 병행한다. 특히 스케줄링과 관련된 모듈 분석 및 오버헤드 예측을 통해 전통적인 정적 분석 기법을 확장하여 보다 정확하고 신뢰성 있는 분석 도구의 개발이 필요하다.

6. 참고문헌

[1] Scott A. Brandt, "The Case for Dynamic Real-Time Task Timing in Modern Real-Time Systems", IEEE IPDPS'04, June, 2004

[2] Ermedahl. Andreas, "A Modular Tool Architecture for Worst-Case Execution Time Analysis", Doctoral Theis, Uppsala University, 2003

[3] J. Engblom, A. Ermedahl, M. Sjoedin, J. Gubstafsson, and H. Hansson. "Worst-case execution-time analysis for embedded real-time systems", Journal of STTT, vol. 4, p. 437-455, no. 4. 2003

[4] F. Mueller, "Timing Analysis : In Search of Multiple Paradigms", IEEE IPDPS'04, June, 2004

[5] K.H.(Kane) Kim, Lynn Choi and Moon Hae Kim, "Issues in Realization of an Execution Time Analyzer for Distributed Real-Time Object", 3rd IEEE ASSET'00, 2000

[6] G. Ottosson and M. Sjodin, "Worst-Case execution *time analysis for modern hardware architectures*". In Proceeding of ACM SINGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems. 1997

[7] J. Gustafsson, "Worst Case Execution Time Analysis of Object-Oriented Programs." 7th IEEE WORDS 2002, January, 2002