재구성된 마이크로 커널의 실시간 특성 분석

박종현(朴鍾賢) 임강빈(任綱彬) 정기현(鄭己鉉) 최경희(崔景熙)
아주대학교 전자공학과
전화:(0331)219-2976 / 팩스: (0331) 212-9531

Real-time Characteristic Analysis of A Micro Kernel for Supporting Reconfigurability

Jong Hyun Park
Electronics Ajou University
E-mail: happyday@madang.ajou.ac.kr

Abstract

Goal of this paper is to design and develop core kernel components for single processor real-time system, which include real-time schedulers, synchronization mechanism, IPC, message passing, and clock & timer. The goal also contains the basic researches on dynamic load balancing and scheduling which provide mechanism for the distributed information processing and efficient resource sharing among various information appliances based on network.

제 1 장 서론

하드웨어의 발전 속도 향상과 정보 가전 기기의 생명 주기가 6 개월 이내로 상품 개발 기간의 단축으로 매번 달라지는 내장형 시스템 환경 하에서 기존 일체형 RTS 및 응용 프로그램을 수정보완하기 보다는 필요한 커널 구성 요소들만 골라 RTOS 를 재구성할 필요성이 대두되게 되었다.

그러나, 현 세대의 실시간 운영체제와 개발 도구들은 개발자들이 새로이 만들 응용들을 응용들의 환경에 맞게 지원하지 않는다. 게다가, 현재의 실시간 운영 체제들은 시스템 서비스가 고정된 세트를 가진 목적 시스템이다. 이러한 시스템들은 크기나 기능성에서 높거나 낮게 확장될 수 없다.

본 연구에서는 위에서 언급된 것에 비추어 확장 가능한 구조로, 커널의 최소화 및 컴포넌트의 모듈화를 통해서 재구성과 확장성이 뛰어난 내장형 실시간 운영체제의 커널을 제시하고, 이를 시험적으로 구현하여, 그 타당성을 확인하고자 한다. 본 연구의 대상은 재구성 기능을 지원하는 실시간 마이크로 커널의 설계이며, 실시간 시스템 RTS 는 견성 실시간 성능(hard real-time performance), 재구성 능력, 확장성 구조, 열린 인터페이스, 표준 인터페이스 제공, 이식성(portability) 등의 목표로 설계되었다.

제 2 장 RTS 실시간 커널

실시간 커널의 재 구성할 수 있는 능력(reconfigurability)을 논의 하고, 각각의 컴포넌트의 특징을 자세하게 알아본다.

1. 컴포넌트 개념의 구조

본 논문은 제반 실험의 환경으로서 실시간 마이크로 커널 RTS 를 이용한다. 실시간 마이크로 커널 RTS 는 본 연구에 의하여 개발되었으며 다른 서비스를 허용하는 매우 유연성 있는 구성 요소와 PK(primitive kernel)에 기반을 둔 컴포넌트 방식의 실시간 커널이다. 다음은 RTS 의 구조와 이를 이루고 있는 각 컴포넌트의 특징을 논한다.

타겟 시스템으로 운용하는 실시간 응용들을 지원하는 것이 요구되는 본질적인 서비스는 primitive 커널에 의해 제공된다. 그리고 요구에 따라 RTS 커널에 더해지게 될 수 있거나, RTS 커널로부터 제거 될 수 있는 개개의 런 타임 구성요소(runtime component)로서 운영체제의 각 옵션적 특징이 구현 된다. 이것은 실시간 시스템 RTS 가 주어진 응용 프로그램 또는 환경의 정확한 필요성들을 부합하기 위해 정확히 형성될 수 있고, 메

모리에 저장함으로 인하여, 성능을 향상시킬 수도 있다는 것을 의미한다.

Primitive 커널은 주어진 응용 프로그램을 지원하는 것은 요구된 실시간 시스템 RTS 특징을 추가하기 위해 RTS 커널 컴포넌트로 추가될 수 있다. 이들의 컴포넌트는 <그림 1>에 보여진다.

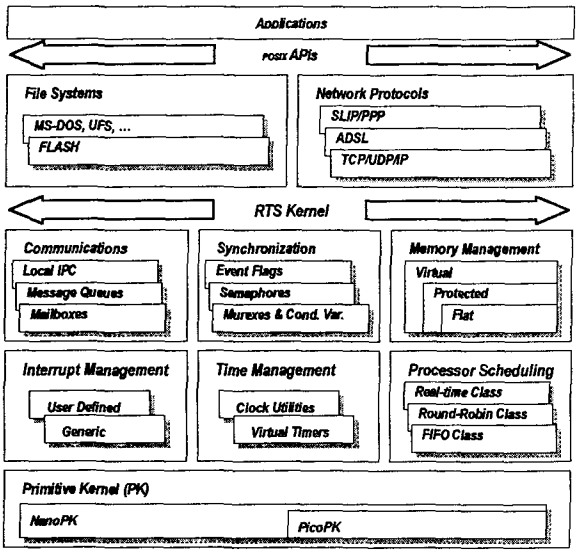

그림 1 실시간 커널 RTS 의 컴포넌트에 기반을 둔 구조

2. 재구성 능력과 확장성

컴포넌트에 기반을 둔 아키텍처를 이용하는 것의 의해, 응용 프로그램 설계는 간단한 스케줄링과 메모리 옵션들, 또는 완전히 특징 되어진 다중-API 소프트웨어 플랫폼사이에서 선택할 수 있다.

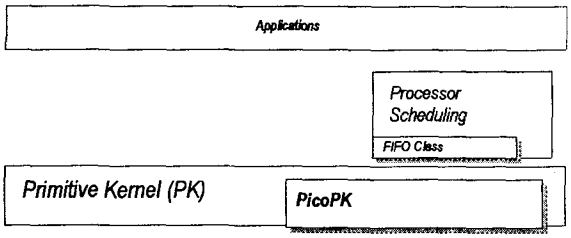

그림 2. 최소 실시간 커널 RTS 의 구성 예.

단일로 운용하는 라인카드(line card)와, 하드웨어와 물리적인 메모리를 관리하기 위한 정정 제어 응용 프로그램을 구현하기 위해서 PicoPK 은 기본적인 실시간 스케줄러 옵션으로 간단히 조합될 수 있다. 최소 구성의 결과로서, <그림 2>에 보여지고, 라인카드(line card) 그 자체에는 직접적으로 임베드(embed)될 만큼 충분히 작다.

여러 개의 그런 라인 카드을 합치고 분산된 클라이언트와-서버 환경에서 다중 응용 프로그램을 실행하는 point-of-sale 터미널을 실현하기 위해, NanoPK 는 향상된 스케줄링, 동기화, 메모리 관리, 태스크(task)의 원격 접근을 처리하기 위해 여러 추가 구성요소로 결합될 수 있다. 이런 방식으로, 동적으로 중앙의 저장소 서버로부터 다운로드 할 수 있고, 네트워크를 통해 원격적으로 데이터에 접근할 수 있을 수 있게, 터미널은 구성될 수 있다.

투명한 프로세스간 통신(IPC)을 제공하기 위해서 구성 요소를 포함하는 것에 의해, 터미널은 또한 분산 구성 관리(distributed configuration management)를 수행하기 위해 응용 프로그램을 실행할 수 있다

3. 실시간 스케줄러

스케줄러는 스레드들 또는 다른 어느 특정영역의 그룹 중에서 스레드 우선도, 스레드 deadline, 서비스 객체, 시간 분할 응답(time sharing response), 공평성 위에 스케줄링 결정을 기초로 한다. Primitive 커널은 전략의 유형과 또한,전략을 구현하기 위해 사용되는 데이터 구조들에도 관한 제약들을 두지 않는다. (예를 들면, 실행하는 큐 구조는 스케줄러에 맡기게 된다.)

만약 일부 사용자들이 정확하게 그들의 섬세한 응용 프로그램을 위해 새로운 전략을 디자인한다면, 스케줄러는 그 새로운 전략들을 채택하도록 허용한다. 그러나, 본 논문에서는 FIFO, 라운드 로빈, 실시간 스케줄링 전략만으로 스케줄러를 구현할 예정이다.

제 3 장. 성능 평가

본 논문의 성능 평가는 RTS 의 실 시간 특징들을 측정한다. 그러므로 Rate monotonic 스케줄링과 비슷한 방법이 사용될 수 있다. 성능 평가의 목적은 RTS 가 견성 실시간(hard real-time) 응용에 대해서 적당한지 아닌지를 밝히는 것이다. 응용의 행위가 미리 예측될 필요가 있다면, 그러면 기본적인 소프트웨어 즉 RTS 는 요구를 맞추기 위해서 필요로 하는 모든 특징을 가질 필요가 있다. 일반적으로, 운영체제의 모든 시스템 콜과

동작은 예측할 수 있는 행위가 보여져야 한다. 이것은 실행 시간이 한정되어야 하고, 예측 가능하고, 시스템의 작업부하의 독립성이 있어야 한다는 것을 의미한다.

본 논문에서 하고자 하는 테스트의 목적은 운영체제의 응답성과 처리량에 대처하는 것 뿐만 아니라, 운영체제의 실행시간이 어느 정도 결정되어 있느냐와 그 동작 상태와 성능을 분석하는 것이다.

1. 측정 방법

절대적 시간 참조는 시간 간격을 측정하는데 요구된다. 대부분의 운영 체제들은 측정 도구로서 사용될 수 있는 소프트웨어 타이머를 포함한다. 그러나 다른 운영체제의 타이머가 반드시 같은 해상도(resolution)를 갖지는 않는다. 더욱이, 그런 운영체제는 반드시 정확한 측정을 수행하는데 충분한 정밀도를 항상 제공하지 않는다. 소프트웨어 타이머의 사용은 테스트를 실행하는 동안 운영체제에 의해 측정이 실행되기 때문에, 결과에 예측할 수 없는 부하(overhead)를 추가하게 된다. 더욱이, 소프트웨어 타이머 근거한 측정은 시스템 clock 을 가지는 동기적인 것이다. 소프트웨어 타이머 대신에, 우리는 다른 카운터 타이머를 사용한다.

카운터 타이머(타이머 번호 4)는 특정한 레지스터에 현재의 카운트 값을 유지하면서 50Mhz 오실레이터(oscillator)에 의해 생성된 tick 당 한 번씩 카운트 값을 감소시킨다.

하나의 테스트가 실행하는 동안, 현재 카운트 값은 카운터 타이머로부터 읽혀지고, 계산된 시스템 동작 전후에 유효한 timestamp buffer 에 쓰여진다. 이것을 trace 라고 불려진다. timestamp buffer 는 테스트되기 전 커널로부터 할당된 정상적인 시스템 메모리이다. 카운터 값을 읽는 것과 시스템 버퍼(unsigned long: 4 바이트)에 쓰는 것은 약 114 tick 이 소요된다. trace 는 코트에서 그 위치가 명백해지고, 분석 결과동안 테스트의 실행 통로를 따르고 식별하도록 가능하게 끔 만드는 것이다. <그림 2> 일반적인 성능 측정을 보여준다: 타임스탬프 버퍼에 써지는 trace T2 와 T1 의 시간 차는 시스템 동작에 대한 실행 속도를 공급한다. 모인 데이터가 시스템 동작의 처음과 끝을 표시함에 따라, 설계된 타임 간격을 계산하는데 사용될 수 있다. 각각의 테스트는 통계적인 도구를 사용하여 그것들을 분석하기 위해 충분한 샘플을 생성하는 최소 10000 번의 loop 을 수행한다.

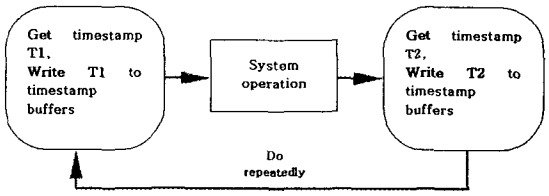

그림 2. 일반적인 성능 측정

2. 시스템 작업부하(Workload)

매개변수의 set 은 테스트의 실행동안 시스템의 작업부하에 의해 정의된다. 시스템 작업 부하는 시스템 상에서 스레드들의 개수에 의존한다. 예를 들면, 이용할 수 없는 자원을 요청하고 있는 스레드들은 시스템 큐로 커널에 의해 대기열로 넣어진다. 몇 몇의 테스트는 요청되었던 자원을 release 할 때, 재 스케줄하기 위한 시간이 요구자(requester)의 숫자 또는 실행할 준비가 되어있는 스레드의 숫자에 의존적인지 아닌지를 결정한다. 그러므로, 테스트를 위해 object 를 요구할 시간을 측정하고 있는 시리즈는 스레드들을 요구하는 변수로 반복 된다.

다른 한편으로는, 시스템 작업 부하는 context 와 스레드를 switch 하는 동안 조작되는 데이터의 양에 의해 영향을 주게 된다. 태스크 스위치 대기시간(latency) 줄이기 위해, 많은 RTS 의 구현된 것들은 스레드들을 호출하는 가벼운 정도의 태스크를 사용한다. 스레드는 프로세스보다는 적은 정보를 나르기 때문에 가벼운 정도의 가중치를 가지게 된다. 이것은 스레드 제어 블록이 프로세스 제어 블록보다 작은 것을 의미한다. 선점된 스레드의 제어 블록을 저장하고, 다음 실행 스레드의 제어 블록을 본래 상태로 되돌린 것에 기인된 부하(overhead)는 감소하게 된다. 스레드 제어 블럭에 포함되는 데이터의 양은 메모리 구성에 의존한다

3. 성능 평가 항목

성능 평가 항목에는 스레드 전환 latency, 스레드 생성 과 삭제, 동기화 객체, 배타적 객체에 관한 항목이 있다. 본 논문에서 측정하고자 하는 부분은 다음과 같이 정의 된 항목을 가지고 측정하였다.

먼저 스레드 전환 latency에서는 스레드를 선점하기 위해 운영체제에 대해 요구된 시간과, 다음에 실행할 준비가 되어 있고 후에 실행을 시작할 준비가 되어 있는 스레드를 검색해서 되찾는 시간.을 측정한다. 스레드는 같은 우선 순위 수준에서 실행하고 최대 이용 가능한 우선순위는 0으로 한다.

스레드의 생성과 삭제항목에서는 스레드를 생성하고, 삭제할 시간.을 측정한다.

동기화 객체항목에서는 동기화 객체를 생성하고 삭제하기 위해 필요한 시간을 측정한다. 동기화 테스트는 많은 스레들들이 대기하고 있을 때, 세마포의 1개의 송신권(token)을 release하기 위한 시간을 측정한다. 각각 다른 우선순위를 갖는 스레드가 대기하고 있는 개수를 2개 사용하여 측정한다.

배타적 객체를 측정하는 항목에서는 배타 객체(exclusion object)를 생성하고 삭제하기 위한 시간을 측정한다. mutex 의 생성과 삭제의 시간을 재고, 그리고 나서 경쟁 없이 자원을 얻고 release하는 시간을 잰다.

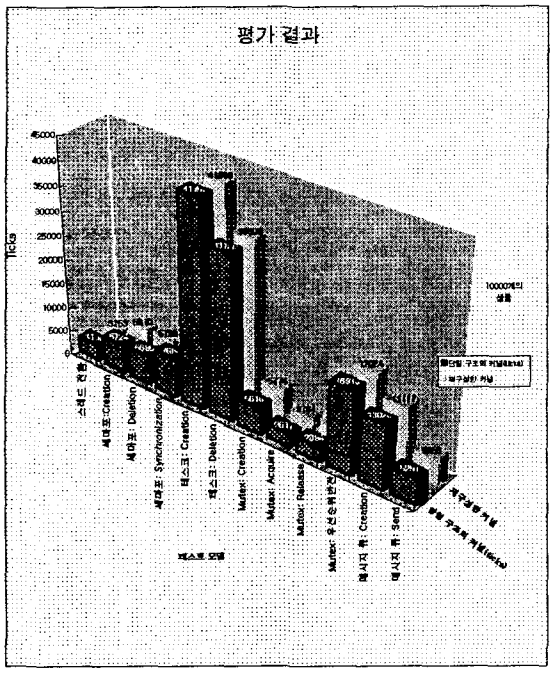

그림 3. 성능 평가 비교

메시지 전달 객체를 측정하는 항목에서는 메시지 전달 객체(message passing object)를 생성하고 삭제하기 위한 시간을 측정한다. 메시지 송수신하는데 있어, 메시지 크기의 사이의 차의 시간을 얻도록 메시지 크기를 변화 시키고, 각 테스크에서 하나의 엔트리를 갖는 메시지 큐를 사용한다.

이러한 평가 결과는 다음과 같이 기존의 단일 구조의 커널과 재구성기능을 지원하는 커널과 비교 분석되었다.<그림 3 참조>

제 3 장 결론 및 앞으로 할일

본 논문은 실시간 스케줄러, 동기화 메커니즘, IPC(Inter-Process Communication), 메시지 전달(message passing), clock과 타이머 등을 포함하는 단일 프로세서와 실시간 시스템(RTS라 불린)을 위한 여러 핵심 커널 요소들의 성능 분석하여 운영체제의 응답성 처리량 뿐만 아니라, 운영체제의 determinism과 성능을 테스트하는 것을 측정하도록 했다. 테스트 환경과 결과는 실험적인 요인들과 테스트 시리즈에 근거를 바탕으로 하여 성능을 기존의 단일 구조의 커널과 비교 분석하였다.

실제로 기존의 단일 구조의 커널이 성능이 뛰어나기는 하나 재구성 능력과 확장성을 고려한다면 재구성 기능을 지원하는 컴포넌트 기반의 커널을 사용하여 성능 향상을 도모하는 편이 현 추세의 임베디스 시스템 환경을 따라 잡는 데는 훨씬 더 용이하다고 본다.

앞으로 상업적 생산성에 있어서 완전히 이용되도록 모듈을 개발하기 위하여 포괄적인 성능 평가와 분석, 미세조정, 신뢰성 증진, 인기 있는 프로세서에 이식성 등등을 포함하는 좀더 많은 일들이 수행되어야 할 것이다.

참고 문헌

[1] Andrews G.R., "Concurrent Programming: Principles and Practice.", Benjamin/Cummings, 1991.

[2] Armstrong J., Virding R., Williams, M., "Concurrent Programming in Erlang.", Prentice-Hall International (UK) Ltd., 1993.

[3] Deitel H. M., "Operating System.", Addison-Wesley, 1990.

[4] Tanenbaum A. S., "Modern Operating Systems.", Prentice-Hall, 1992.

[5] J. J. Labrosse, "The μC/OS is a small real-time kernel", http://www.ucos-ii.com