리플리케이션 관리 시스템을 위한 적응적 푸쉬 결함 모니터링 스타일 설계

김분희[0] 김영찬
중앙대학교 컴퓨터공학과
{bhkim[0], yckim}@sslab.cse.cau.ac.kr

Design of Adaptive Push Fault Monitoring Style for Replication Management System

Boon-Hee Kim[0] Young-Chan Kim
Dept. of Computer Science and Engineering, Chung-Ang University

요 약

인터넷을 구성하고 있는 수많은 종류의 하드웨어와 소프트웨어를 통합하여 효율적인 분산 환경을 구성하고자 많은 연구가 진행되어 왔다. 이에 대한 결실의 중요 핵심부분을 이루는 분산 객체 시스템은 인터넷에 접속한 수많은 이용자에게 분산된 환경의 네트워크 자원들을 자유로이 이용하면서, 그 이용의 불편함이 없도록 상당한 투명성을 부가해주고 있다. 본 논문에서는 기존 분산 객체 시스템에서 제공하는 리플리케이션 관리 시스템 내의 결함 모니터링 스타일에 복잡도를 줄인 푸쉬 모니터링 스타일 기반 절충형 장애 검출 시스템을 제안한다. 본 논문에서 제안한 절충형 장애 검출 시스템의 실험결과 나타난 장애 검출 과정에서 기존 방법에 비해 신속성과 정확성 측면에서 향상된 성능을 검증할 수 있었다.

1. 서 론

인터넷 관련 응용 분야들에서 신뢰성, 가용성, 안정성이 요구되는 분야가 늘어나고 있다. 이러한 시간 조건에 민감한(time-sensitive) 분산 응용 프로그램에 미들웨어 환경이 접목된 대상 응용프로그램으로 검색 엔진이나 전파 탐지기(radar tracking) 어플리케이션 등이 있을 수 있다. 이와 같은 분산된 환경은 신뢰성, 안정성, 응답시간 향상성을 제공하기위해 서버 리플리케이션 (server replication) 구조를 가지며, 각 리플리카(replica: duplicated server object)의 결함 발생을 진단하기 위한 구조가 요구된다.

분산 환경에 존재하는 다양성을 통합하는 역할의 미들웨어로써, 국제 표준화 활동의 결과인 CORBA 는 다양한 결함 관리(fault management) 방법을 제공해 주고 있다. CORBA의 결함 관리 방법은 다양한 구성 요소들의 복합적인 조합으로써 사용자의 요구에 따라 결함을 찾는 방법이 결정된다. 리플리케이션 관리자를 구성하는 요소 가운데, 특성 관리자(property manager)에 의해 관리되는 주요 요소로써 존재하는 결함 모니터링 스타일(fault monitoring style)은 리플리카의 결함을 감시하는 스타일을 결정한다. CORBA에서 제공하는 대표적인 결함 모니터링 스타일에 풀 모니터링 스타일(pull monitoring style)이 제시되어 있다. 이는 결함이 발생되지 않거나 서비스를 마칠 때 까지도 문제가 발생되지 않은 수많은 리플리카를 감시하기 위해 주기적으로 대상 리플리카를 감시해야 하며, 이러한 사항은 모니터 대상 객체의 결함 파악이 느린 단점을 지닌다.

결함 진단과 그에 따른 대응체제로써 고장 모니터링 메커니즘은 메시지 수신시간의 타임아웃을 기준으로 한다. 그러나 실제 비동기 분산 시스템에서는 리플리카 자체의 결함과 네트워크 과부하를 구별하기 어렵다. 평균 전송시간보다 낮은 타임아웃 값은 잘못된 장애 검출을 야기하고, 평균 전송시간보다 더 큰 타임아웃 값은 충돌 발생시 장애 검출의 지연 가능성이 높다[2]. 이에 메시지 평균 전송시간과 타임아웃의 상관관계에 따른 신뢰성 부여 모델의 장애 검출 시기 지연 문제를 해결한 절충형 모델이 요구된다.

본 논문에서는 분산 객체 시스템에서 제공하는 리플리케이션 관리 시스템의 풀 모니터링 스타일에서 나타날 수 있는 단점을 극복한 절충형 장애 검출 기반 푸쉬 모니터링 스타일을 제안한다. 이 시스템을 FlexPush (FLEXible PUSH model)이라 하고, 제안된 푸쉬 결함 모니터링 스타일의 실험결과 시간 복잡도 측면에서 향상된 성능을 나타내며, 장애 검출의 신속성과 정확성을 증명할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서 관련연구, 3장에서는 본 논문의 핵심 부분인 FlexPush 설계 및 평가를, 마지막으로 4장에서 결론을 맺는다.

2. 관련 연구

2.1 CORBA의 결함 검출 기법

본 절에서는 CORBA에서 제공하는 replica의 결함 처리 방법을 나타낸다. 그림. 은 코바에서 제공하는 리플리케이션 관리자의 구성요소를 계층화 한 것이다. 본 논문에서 다루고자 하는 부분은 리플리케이션 관리자를 구성하는 요소 가운데, 특성 관리자(property manager)에 의해 관리되는 결함 모니터링 스타일이다. 리플리카의 결함을 감시하는 스타일을 결정해 주는 결함 모니터링 스타일에는 그림. 에서와 같이 PULL, PUSH, NOT_MONITORED로 구분된다. PULL 스타일은 고장 모니터 호출시 정기적으로 생존 여부를 모니터 대상 객체에게 물어보는 방법으로써 "is_alive" 메서드가 그 역할을 한

다. 이 스타일은 모니터 대상 객체가 해당 시간 간격 내에서 응답이 없으면, 고장 모니터는 이를 고장으로 판단한다. PUSH 스타일은 모니터 대상 객체가 주기적으로 결함 모니터에게 생존 여부 메시지를 "i_am_alive" 메서드를 통해 호출한다. 이 스타일은 모니터 대상 객체가 정해진 시간 간격내에 "i_am_alive" 메서드를 호출하지 않았다면, 고장 모니터는 그것이 고장이 아닐까 의심한다. 주요한 이점은 결함 모니터에 의해 모니터 대상 객체의 결함을 빨리 찾을 수 있다는 점이다[5].

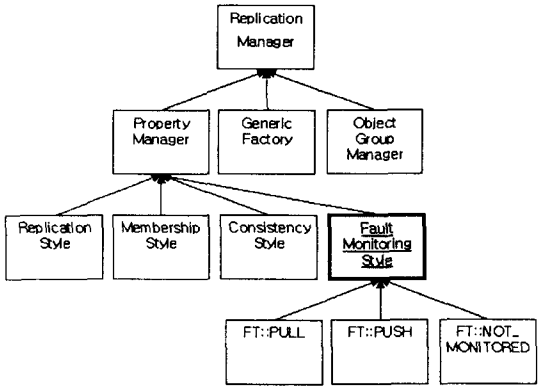

그림 1. 리플리케이션 관리자 구성 요소

2.2 CORBA 결함 모니터링 스타일

그림 2는 코바의 결함 감지 기법을 보여주는데, 상단의 RM(replication manager)은 PM(property manager), OGM(object group manager), GF(generic factory) 인터페이스를 구현하였다. 즉, RM은 PM, OGM, GF를 모듈로 하는 시스템이다. RM는 결함허용 시스템의 하부구조에서 중요한 역할을 하며 결함 허용 관리 도메인 내의 모든 호스트 상에 존재할 필요는 없다. 그러나 하나의 도메인 내에는 하나의 리플리케이션 관리자가 있어야 한다. RM을 구성하는 모듈 가운데, PM은 객체 그룹의 특징적인 면을 관리하는 역할을 한다.

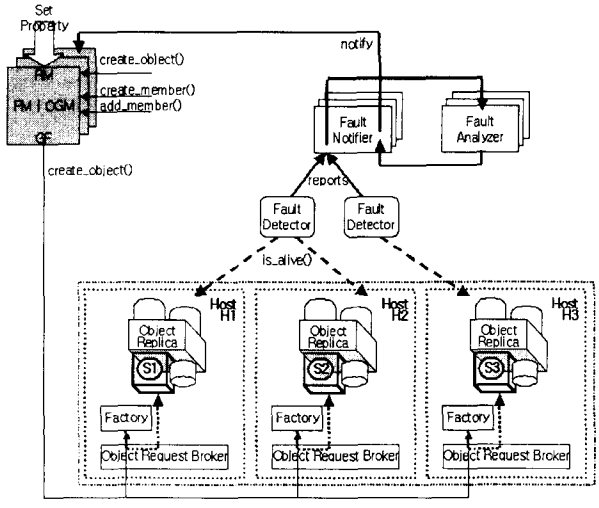

그림 2. CORBA 리플리케이션 관리자 동작 방식

리플리케이션 관리자는 위의 동작 방식에 따라 각 객체의 결함을 검출하게 된다. 여기서 결함 검출기(fault detector)는 PullMonitorable 타입(one of the monitoring style for CORBA)의 어플리케이션 객체에 주기적으로 "is_alive" 명령을 수행하게 되고, 이에 대한 해답의 유무에 따라 결함을 감지하게 된다. 결함이 감지된 경우 결함 검출기는 결함 통보기(fault notifier)에 보고하게 되고, 결함 통보기는 최종적으로 리플리케이션 관리자에게 특정 리플리카의 결함을 기정사실화 하는 통지를 하게 되는 구조이다. 이러한 풀 결함 모니터링 스타일(Pull Fault Monitoring Style)에서 결함 검출기는 결함이 발생되지 않은 또는 서비스를 마칠 때 까지도 문제가 발생되지 않은 수많은 리플리카를 감시하기 위해 주기적으로 감시 대상 리플리카에 "is_alive" 명령을 수행해야 한다. 이러한 사항은 결함 검출기를 관리하는 시스템 측면에서 봤을 때, 시스템 자체적 결함에 대한 대처를 할 수 없고 미미하지만 시스템 성능저하 요인이 될 수 있다. 그리고 이러한 중앙 집중적인 관리보다는 리플리카가 존재하는 각 시스템에서 자신의 시스템의 상태를 자체적으로 관리하는 것이 효율성 측면에서 이점이 있다. 이때 나타날 수 있는 자체적 관리의 측면에서 각 시스템을 이루고 있는 미들웨어에 추가되어야 할 모듈이 차지하는 성능 및 용량 등의 요소에서 나타날 수 있는 부가적 요인은 거의 없음을 본 논문의 평가 결과 증명된다.

3. FlexPush 설계

3.1 제안된 시스템

FlexPush 모델은 분산된 비동기적 결함 관리 모델로써 통신 장애가 없는 충돌 방지(crash-stop)형이다. 메시지 평균 전송시간과 타임아웃의 상관관계에 따른 신뢰성 부여 모델[6]의 장애 검출 시기 지연 문제를 해결한 절충형 방식[5]에서 검출 시간이 향상된 성능 모델을 제안하고 따른다. FlexPush 결함 모니터링 알고리즘은 타임아웃 이벤트, 응답호출 이벤트를 기반으로 동작되는데, 타임아웃 이벤트는 특정한 타임아웃 주기가 끝날 때 모니터 대상 객체와 시스템 과부하에 의해서 발생되고, 응답호출 이벤트는 푸쉬 모니터가 몇몇 모니터 대상 객체에 의해 호출된 "i_am_alive" 메서드를 활성화 시킬 때에 발생한다.

3.2 FlexPush 모니터링 구조

본 알고리즘을 구성하는 각종 값은 msec(milliseconds) 단위이며, 초기 설정된 타임아웃 값과 모니터링 간격은 시스템 과부하와 실제 결함의 상호 관련성을 고려하여, 실제 실행하면서 최적의 값을 찾기 위해 변경되게 된다. 초기 모니터링 간격은 최적의 값으로 판명된 3 msec으로 설정되고, 타임아웃 값은 2 msec으로 초기 설정한다. 그리고, TRS[3] 정책을 기반으로 초기화 된다. 그림 3은 전체 리플리케이션 관리자의 동작 방식 중 본 논문에서 설계한 절충형 푸쉬 모델 부분만을 확대한 것이다. 리플리케이션 관리자는 결함 분석기와 결함 통보기의 상호 협조하에 결정된 결함 정보를 받게 되는데, 이러한 정보를 받고 처리하는 기법은 CORBA 표준 규격에 정의된 바를 수정하지 않는 것이 많은 부분에서 이점을 지닌다. 그래서 본 논문에서는 기본 동작 기법을 수정 하지 않는 방향으로 푸쉬 모델을 설계하였다.

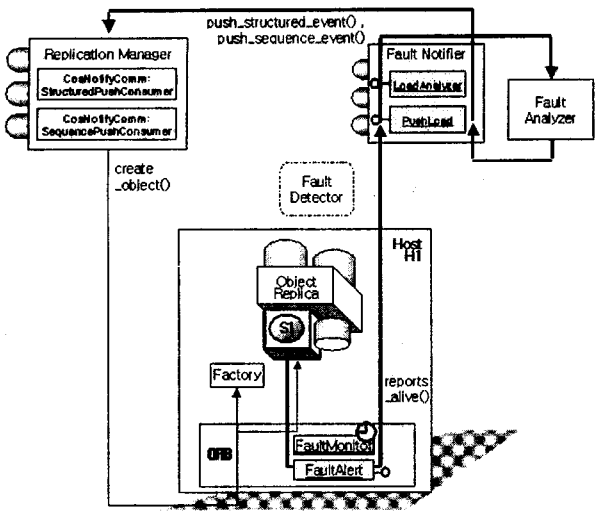

그림 3. 제안된 시스템의 동작 방식

결함 허용 하부구조에서 어플리케이션 또는 하부구조에서 제어되는 그룹 스타일(membership style)로써 객체 그룹의 수를 초기화 하게 되는데, 본 구조에서는 이용자의 선택에 따라 결함 검출기를 구성하는 모듈들을 그대로 이용할 수도 있고, 결함 검출기의 기능을 일시 정지시킬 수도 있겠다. 전자의 경우 결함 검출기의 핵심 기능인 "is_alive" 명령을 일시정지 시킨 후 서버 리플리카의 ORB(Object Request Broker)가 보고하는 내용을 받고, 이를 결함 식별기(fault identifier)에 보고해야 한다. 이는 해당 서버 측 ORB가 자체적으로 푸쉬하는 메시지를 결함 통보기에게 단지 넘겨주는 역할만 할 뿐임으로 그 유용성에서 문제가 된다. 이러한 기법보다는 풀 스타일을 선택 시 결함 검출기자체를 일시 정지시키는 것이 알고리즘의 복잡도 측면에서 효율적이다. 결함 통보기는 결함 검출기에 의해 발견된 결함을 리플리케이션 관리자에게 보고하는 역할을 하는데, 결함 결함 통보기와 클라이언트에 의해 선택되어진 통보의 형태에 따라서 결함 전달이 달라진다.

그림 4는 제안된 시스템의 결함 통보기를 통해 결함을 보고하기 위한 명령의 파생 과정을 보여준다. 리플리케이션 관리기 구현에 있어서 선택된 타입이 제안한 기법일 경우에 나타나는 결함 전파의 진행을 보여주고 있는데, 선택된 리플리카는 구현 시 PushMonitorable 인터페이스로부터 상속받게 된다. PushMonitorable 인터페이스를 구성하는 "i_am_alive" 오퍼레이션이 백그라운드 스래드로 작동하여 사용자가 정한 시간 간격 (Fault-MonitoringIntervalAndTimeout's TimeBase::TimeT)에 따라 주기적으로 메시지를 통보하고, 이 메시지는 초기 클라이언트가 선정한 통보 타입에 따라 해당 메서드에 파라메터로써 전해지게 된다. 이러한 형태는 consumer 내의 각종 하부구조를 이루는 모듈들을 개별적으로 다른 단위로 인식될 때 가능한 방법이다.

3.3 평가

FlexPush의 실험의 있어서 시간 간격 값은 각각 0.1, 0.3, 0.5, 0.7의 네 개의 요소로써 타임아웃 비율을 각각 비교 분석 하였다. 비교 분석 값은 타임아웃 된 수와 응답 된 수의 비율로써 상대 측정 요소인 작업량과의 대응관계로써 성능 평가가 이루어진다. 작업량은 각각 1, 5, 10의 세 개의 요소로써 구성한다. 각 작업량의 평균 타임아웃 비율이 가장 높은 경우는 0.7 요소로써 약 18%에 이르며, 각 작업량 당 최상의 값은 작업량이 각각 1, 5, 10이고, 0.3, 0.1, 0.3 요소로써 각각의 타임아웃 비율이 6%, 3.1%, 2.5%였다. Irineu 논문에 비해 5-0.1-3.1% 부분에서 장애 발생율이 상당히 낮았으며, 다른 요소에 대해서는 오차범위($0 \leq x \leq 0.1$) 내에서 나타나고 있다.

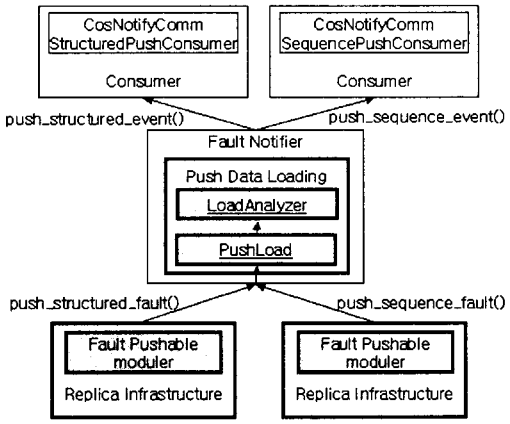

그림 4. 결함 통보기를 통한 결함 보고 전파

4. 결론 및 향후연구

메시지 수신시간의 타임아웃을 기준으로 하는 결함 진단과 그에 따른 대응체제로써 고장 모니터링 메커니즘은 실제 비동기 분산시스템에서는 리플리카 자체의 결함과 네트워크 과부하를 구별하기 어려운 체제이다. 평균 전송시간보다 낮은 타임아웃 값은 잘못된 장애 검출을야기하고, 평균 전송시간보다 더 큰 타임아웃 값은 충돌 발생시 장애검출의 지연 가능성이 높다. 이에 메시지 평균 전송시간과 타임아웃의 상관관계에 따른 신뢰성 부여 모델의 장애 검출 시기 지연 문제를 해결한 절충형 모델이 요구된다.

본 논문에서는 분산 객체 시스템에서 제공하는 리플리케이션 관리 시스템의 풀 모니터링 스타일에서 나타날 수 있는 단점을 극복하고, 시간 복잡도를 줄인 절충형 장애 검출 기반의 FlexPush를 제안하였다. 제안된 푸쉬 결함 모니터링 스타일의 실험결과 시간 복잡도 측면에서 향상된 성능을 나타내며, 장애 검출의 신속성과 정확성을 증명할 수 있었다.

참고문헌

[1] CORBA Specification v3.0, 2002.
[2] T. Chandra, "Unreliable Failure Detector for Reliable Distributed Systems", Journal of the ACM, 43(2), 225-267, 1996.
[3] B.-H. Kim, Y.-C. Kim, "Transparent Replica Strategy for Fault-Masking in Real-Time Distribution System", Proceedings of the 29th KISS fall conference, Oct, 2002.
[4] S. Bagchi, K. Whisnant, Z. Kalbarczyk, and R. K. Iyer, "Chameleon: A Software Infrastructure for Adaptive Fault Tolerance," Proc. of the 17th IEEE Symposium on Reliable Distributed Systems, Oct. 1998.
[5] K. P. Birman,"Reliable Distributed Computing with the Isis Toolkit, IEEE, 1994.