정보학연구 제9권 제4호 2006년 12월(pp.25-34)
The Journal of Information Technology, Vol.9, No.4


# 분산객체 기반 경량화 결함허용 기술의 성능 비교

김식* · 현무용**

목 차



## Ⅰ. 서론

응용 프로그램들이 보다 복잡해지고 분산 컴퓨팅을 지향함에 따라 이를 지원하기 위한 성공적인 패러다임으로써 분산객체 기술이 주목받고 있으며, 현재, 분산객체 프레임워크를 지원하는 다양한 미들웨어들이 제안되어 있다. 분산 시스템을 위한 객체지향 디자인은 프로그램의 이식성과 재사용성을 기본적으로 제공함으로써 복잡한 시스템 구축에 적합하다는 장점 때문에 현재 널리 사용되고 있는 분산 소프트웨어 개발 기법 중의 하나이다.[1] 현재, 객체 지향 분산 시스템을 지원하기 위한 방안으로써, OMG의 CORBA[2], Microsoft의 DCOM[3], IBM의 DSOM[4], Java RMI[5] 등 여러 가지 미들웨어 플랫폼들이 제안되어 있다.

이러한 분산 미들웨어 시스템들은 객체 기반 분산 응용 프로그램의 품질과 재사용성을 향상시켜 주지만, 결함허용 기능을 지원하지 않음으로써 가용성과 신뢰성이 보장된 객체 기반 분산 응용 프로그램의 설계 및 구현을 복잡하게 한다. 현재, 객체 기반 분산 응용 프로그램의 신뢰성 및 가용성을 향상시키기 위한 다양한 연구들이 진행 중에 있으며, 그 결과 Electra[6], Eternal[7], OGS[8], JavaGroup[9] 등의 시스템들이 제안되어 있다. 그러나, 위에서 언급한 대부분의 연구들은 결함허용 기능의 추가에 대한 비용 대 효과를 감안하여 신뢰성 있는 그룹 통신에 기반한 능동(*active*) 복제 메커니즘에 집중되어 있으며, 높은 등급의 가용성과 신뢰성이 요구되는 *mission critical* 응용프로그램에 적합한 방식이다.

분산 컴퓨팅이 일반화됨에 따라 저가의 컴퓨팅 환경 상에서 동작하는 *non mission critical*

* 세명대학교 정보통신학부 교수
** 한전KDN(주) 전력IT연구원 선임연구원

응용프로그램의 수가 증가 추세에 있다. 예를 들면, 슈퍼컴퓨터에서 동작하는 병렬 응용 프로그램들은 분산된 서버 클러스터 상에서 수행될 수 있다. 이러한 응용프로그램들에 있어서 결함허용 기능은 필수적이지만, 제한된 자원 및 수행 시간을 사용하여 결함허용 연산에 수반되는 오버헤드를 줄이는 방향으로 제공되는 것이 바람직하다.[10,12]

이 논문에서는 현재 다각도로 연구되고 있는 분산객체의 경량화 결함허용 기술들을 살펴보고 그 장·단점을 비교분석한다. 또한, 실험을 위한 환경 및 벤치마크 모델들을 소개한 뒤, 제시된 결함허용 기술의 성능을 평가하기 위한 다양한 실험 결과들을 제시한다.

# II. 결함허용 기법에 대한 비교 분석

## 2.1. 복제 방식

이 절에서는 결함허용 서비스를 제공하기 위한 대표적인 객체복사 방식인 능동 복제 방식과 수동 복제 방식의 장단점을 비교·분석하고, 경량화 방식의 결함허용 서비스 지원을 위한 적합성 여부를 평가한다. 평가는 관련 참고문헌들에 기반하여 체크포인트 관련 비용, 네트워크 대역폭 점유율, 결함 복구 시간, 호스트 가용 자원 독점률, PWD*(piecewise deterministic)* 가정[8] 요구의 5가지 비교 항목을 기준으로 실시된다. 표 1은 항목별 비교 결과를 보여주고 있다.

수동 복제 방식 중 cold 대기 모드에서는, 무결함 동작 중 주기적으로 발생하는 주 객체의 체크포인트 비용을 고려해야 한다. 만약 주 객체의 상태가 다량의 정보를 포함한다면, 주 객체에 대한 주기적인 체크포인트에는 상당한 비용이 소모된다. Warm 대기 모드인 경우, 주 객체의 상태가 주기적으로 백업 복사본들에게 전송되므로 상당한 오버헤드를 초래한다. 그러나 참고문헌에 따르면 체크포인트의 회수와 결함 복구 효율은 서로 비례하지 않으며, 최적의 체크포인트 회수는 응용 프로그램과 문제 크기에 의존한다. 따라서 주 서버 객체의 상태 및 응용 프로그램의 특성을 고려한 체크포인트 회수의 신중한 선택을 통해서 그 오버헤드를 줄일 수 있다.[12,13]

<표 1> 복제 방식간의 비교

| 비교 항목 \ 복제 방식 | 능동 복제 | 수동 복제 |
|---|---|---|
| 체크포인트 비용 | 해당사항 없음 | 높음 |
| 네트워크 대역폭 점유율 | 높음 | **낮음** |
| 결함 복구 시간 | **빠름** | 느림 |
| 호스트 가용자원 독점률 | 높음 | **낮음** |
| PWD 가정 | 필요 | 필요 |

능동 복제 방식에서, 모든 복사본들에게 원격 호출 메시지를 전달하기 위해서는 신뢰성 있는 다중전송 메커니즘의 지원이 필수적이다. 이는 각각의 원격 호출 메시지가 더 많은 다중전송 메시지를 양산할 수 있으므로, 네트워크 대역폭의 사용 증가를 초래한다. 또한 신뢰성 있는 다중전송 메커니즘에서 요구하는 메시지 정렬, 가상 동기화는 상당한 실행상의 오버헤드를 초래한다. 수동 복제 방식의 경우, 주 객체에게 전달되는 원격 호출 메시지만 존재하기 때문에 네트워크 대역폭의 점유율이 비교적 낮게 유지된다.

서버 객체에 결함이 발생한 경우, 능동 복제 방식은 상당히 빠른 결함 복구 능력을 발휘한다. 실제로 능동 복제된 모든 복사본들은 클라이언트의 모든 요청들을 동일하게 수행하기 때문에 그 중 한 복사본에 결함이 발생하더라도 나머지 복

사본들이 클라이언트의 요청을 중단 없이 수행한다. 따라서 능동 복제는 높은 수준의 신뢰성과 가용성을 요구하는 시스템에 적합한 방식임을 알 수 있다[10]. 그러나 수동 복제 방식에서 주 객체에 결함이 발생하면, 그 복구에 상대적으로 많은 시간이 필요하다. Cold 대기 모드의 경우에는, 새로운 주 객체 선정, 마지막 체크포인트의 전달 및 마지막 체크포인트 이후의 메시지 재실행 등의 과정을 거치게 된다. Warm 대기 모드인 경우, 마지막 체크포인트 이후의 메시지 재실행만 요구된다.

경량화 결함허용 서비스는 제한된 자원 및 실행 시간을 사용하여 결함허용 연산에 수반되는 오버헤드를 줄이는 방향으로 설계되어야 한다. 따라서 위에서 논의된 복사 방식간의 비교·분석 결과는 수동 복제가 경량화 방식의 결함허용 서비스를 위한 보다 적합한 객체 복제 방식임을 입증한다.

## 2.2. 롤백복구 프로토콜

표 2는 일관성있는 객체 복제를 위한 메커니즘 중의 하나인 롤백복구(rollback recovery) 프로토콜들[12]을 PWD 가정, 실행 오버헤드, 출력 커밋 지연(output commit latency)시간, 쓰레기 수집의 용이성, 결함 복구 동작의 단순성, 도미노 효과, 고아 프로세스의 발생, 롤백의 범위 등을 포함한 다양한 기준 항목으로 평가한 결과이다. 평가는 관련 참고문헌들의 분석 및 실험 결과에 근거하여 실시되었다.

클라이언트·서버 환경 하에서 동작하며 원격 함수 호출이 단위(atomic) 연산으로 간주되는 객체 기반 결함 허용 분산 시스템에서 PWD 가정은 일반적인 범주에 속하며, 빠른 출력 커밋 동작과 고아 프로세스가 발생하지 않는 복구동작을 지원한다.[12,13] 또한 가비지 컬렉션과 결함 복구는 결함 라인 계산 과정을 요구하므로, 무결함 동작시에 미리 결정된 결함 라인을 사용하는 비관적 로깅 프로토콜 하에서 보다 효율적으로 동작 가능하다.[1,14]

비조정 체크포인팅 프로토콜은 가장 적은 무결함 실행 오버헤드를 보여 주지만, 도미노 효과가 발생될 경우 심각한 결함복구 오버헤드를 초래한다. 도미노 효과는 PWD의 가정 하에 동작하는 로그기반 롤백복구 프로토콜에 의해 방지 가능하다.[14]

<표 2> 롤백복구 프로토콜간의 비교

| 비교 항목 \ 프로토콜 | 비조정 체크포인팅 | 조정 체크포인팅 | **비관적 로깅** | **낙관적 로깅** | **인과적 로깅** |
|---|---|---|---|---|---|
| PWD 가정 | 불필요 | 불필요 | **필요** | 필요 | 필요 |
| 실행시간 오버헤드 | 낮음 | 높음 | **높음↑** | 높음 | 높음 |
| 출력 커밋 | 불가능 | 느림 | **빠름** | 느림 | 빠름 |
| 체크포인트/프로세스 | 2개 이상 | 1개 | **1개** | 2개 이상 | 1개 |
| 쓰레기 수집 | 복잡 | 단순 | **단순** | 복잡 | 복잡 |
| 결함복구 | 복잡 | 단순 | **단순** | 복잡 | 복잡 |
| 도미노 효과 발생 | 가능 | 가능성 없음 | **가능성 없음** | 가능성 없음 | 가능성 없음 |
| 고아프로세스 발생 | 가능 | 가능 | **가능성 없음** | 가능 | 가능성 없음 |
| 롤백의 범위 | 무제한 (Unbounded) | 마지막 체크포인트 | 마지막 체크포인트 | 이전 몇 개의 체크포인트 | 마지막 체크포인트 |

※ PWD : *piecewise deterministic*
※ ↑ : 향후 성능 개선 가능성 내재

롤백의 범위는 각각의 프로세스가 유지하는 체크포인트의 최대 개수를 결정하며, 낙관적 로깅 프로토콜의 경우 로깅 과정에 따라 여러 개의 체크포인트 유지가 요구된다. 비관적 로깅 프로토콜의 경우, 높은 실행상의 오버헤드를 보여주고 있는데, 이는 전송자 기반 메시지 로깅 같은 구현상의 추가적인 처리 메커니즘을 통해 해결 가능하다.[14]

위에서 논의된 평가 결과는 비관적 로깅 프로토콜이 경량화 방식의 결함허용 연산을 위한 적절한 해결방안이 될 수 있음을 보여주고 있다.

## 2.3. 접근 방식

이 절에서는 분산 객체의 신뢰성을 높이기 위한 세 가지 접근방식들을 비교하고, 그 장·단점을 평가한다. 평가는 각각의 접근방식을 채택한 시스템들(Electra, Eternal, OGS, JavaGroup)에 기반하여 실시되었으며 투명성, 사용의 용이성, 이식성, 상호연동성, 모듈화 가능성, 표준규약과의 호환성, 성능, 단순성의 8가지 기준 항목 중심으로 실시된다.

표 3은 항목별 평가 결과를 보여주고 있으며, 양호(good), 만족(satisfactory), 제한적(limited)의 3가지 등급으로 평가한다. 다음에 기술된 내용들은 표에 제시된 평가 결과를 기반으로 9가지 항목별로 각 접근 방식의 장단점을 비교 분석한 결과이다.

투명성은 결함허용 메커니즘의 세부사항들을 프로그래머로부터 숨기는 것을 의미한다. 통합방식은 클라이언트의 투명성을 보장하기 때문에, 클라이언트는 자신이 원격 호출하는 서버가 그룹인지의 여부를 인지할 필요가 없다. 가로채기 방식을 채택한 Eternal 시스템의 경우는 투명성을 제공하고 있지만, 클라이언트가 다중 전송된 원격함수 호출에 대한 모든 응답들에 접근하는 것을 허용하지 않는다. 서비스 방식에서의 투명성은 환경설정에 의존적이다.[1,13]

사용의 용이성은 프로그래머가 오류를 범할 가능성을 줄임으로써 프로그램의 개발 시간을 단축시키며 응용프로그램들의 신뢰성을 높일 수 있다는 점에서 중요한 평가기준이다. 통합 방식이나 가로채기 방식에서 제공하는 투명한 그룹 통신 지원은 클라이언트 측에서의 명시된 구조를 요구하지 않기 때문에 사용이 용이하다. 한편 서비스 방식도 이미 그 인터페이스가 표준화되어 잘 알려진 *publish/subscribe* 프로그래밍 모델 등을 제공하므로 많은 프로그래머에게 익숙한 개발 방식이다.[8]

<표 3> 접근 방식간의 비교

| 비교 항목 \ 접근방식 | 통합 방식 | 가로채기 방식 | 서비스 방식 |
|---|---|---|---|
| 투명성 | 양호 | 양호 | 만족 |
| 사용의 용이성 | 양호 | 양호 | 양호 |
| 이식성 | 제한적 | 만족 | 양호 |
| 상호연동성 | 만족 | 양호 | 양호 |
| 모듈화 | 제한적 | 만족 | 양호 |
| 표준규약과의 호환성 | 제한적 | 양호 | 양호 |
| 성능 | 양호 | 양호 | 만족 |
| 단순성 | 제한적 | 만족 | 양호 |

통합방식을 채택한 시스템들은 자신의 고유한 ORB에 기초하여 설계되었고 ORB내에 그룹 통신을 지원하기 위한 코드들을 추가하는 형태로 구현되었기 때문에, 다른 하드웨어 구조를 가지는 시스템으로의 이식이 용이하지 않으며 개발된 응용프로그램의 이식성 또한 보장하지 않는다. 가로채기 방식은 Unix 환경하의 저수준 시스템 호출을 사용하기 때문에 결함허용 코드의 이식성은 보장되지 않으나, 응용프로그램은 시스템에 독립된 형태로 개발되어 그 이식성이 보장된다. 구현 종속(implementation specific)의 ORB 특징

들에 독립적이라는 가정 하에, 서비스 접근방식에서의 결함허용 코드 및 응용프로그램 코드의 이식성은 보장된다. 그리고 그룹통신 툴킷을 사용하는 다른 접근방식과 비교하여 그룹통신 툴킷에 의해 지원되는 구조들에 의존적이지도 않다.[1,8]

통합 방식을 채택한 시스템의 경우, 시스템 전용 그룹 통신 프로토콜들을 사용하는 툴킷들에 기반한 구현들은 서로 상호연동성이 보장되지 않는다. 서비스 방식은 ORB에서 제공하는 통신 프리미티브들만을 사용하므로 완전한 상호연동성을 제공한다

통합 방식은 단일구조를 지원하므로, 시스템의 일부를 변경하는 작업은 전체 시스템의 변경을 요구하게 된다. 가로채기 방식은 복제 메커니즘과 ORB를 서로 분리하여 유지하므로 시스템 자체의 모듈화가 보장된다. 서비스 방식은 IDL 인터페이스에 기반한 새로운 컴포넌트를 정의함으로써 재사용성과 모듈화를 증진시킨다. 그리고 서비스 방식을 채택한 시스템이 제공하는 객체 그룹 서비스는 서로 독립적이고 그룹 통신과 자연스럽게 연결되는 다수의 서비스 객체들로 구성된다.[1]

일반적으로 클라이언트와 서버 프로그램이 표준 규약에 의거하여 작성된 경우 표준규약과 호환된다고 간주한다. 통합 방식은 표준 규약을 변경하고 확장한 형태로 설계되었으므로 표준 규약과의 완전한 호환성은 보장되지 않는다. 구현 특유의 ORB 특징들에 의존하지 않는다는 가정 하에, 가로채기 및 서비스 방식은 표준 규약과의 호환성을 보장한다. 가로채기 방식을 채택한 Eternal 시스템의 경우에는 ORB로부터 완전히 분리되어 있고 IIOP 구조에만 의존적이다. 서비스 방식은 ORB 핵심부와 독립적이며 ORB 구현에 관련된 가정들을 포함하지 않는다.[8,15]

시스템은 문제를 해결하고 의도한 동작을 수행하는 범위 내에서 단순하고 경량화한 형태로 설계되는 것이 바람직하다. 현재 통합 방식과 가로채기 방식에 관련된 구현들은 신뢰성 있는 그룹 통신을 위한 외부 툴킷의 지원이 필수적이다. 그러나 이러한 툴킷들은 대부분 객체 그룹에 대한 적절한 지원이 부족한 상태이며 이를 보완하기 위한 소프트웨어 계층이 반드시 추가되어야 한다. 서비스 방식은 객체 간 통신에 필수적인 프리미티브들만 제공하며, 독립적이고 선택적인 컴포넌트 형태로 구축되어 있어서 문제 해결을 위한 적절하고 경량화 방식의 해결방안을 제공한다.[1,8]

각 접근 방식에 대한 장·단점 분석 결과는 서비스 접근방식이 표준 규약과의 호환성이 보장되며, 컴포넌트에 기반 한 모듈화가 용이할 뿐만 아니라, 문제 해결을 위한 적절하고 경량화 방식의 해결방안을 제공하고 있음을 보여준다.

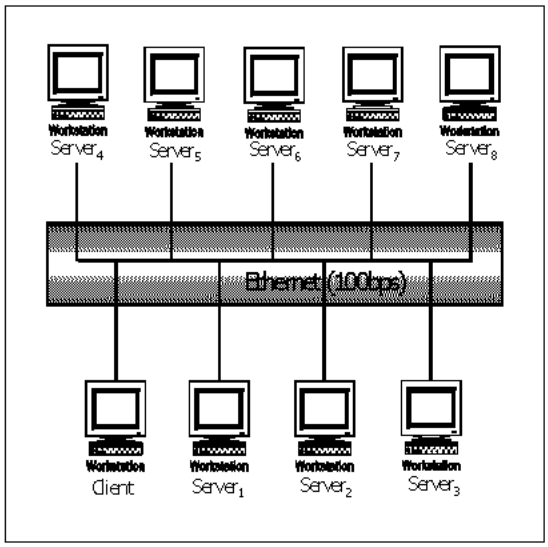
# III. 실험 및 성능 평가

이 장에서는 벤치마크 프로그램들을 이용한 일련의 실험들을 통해 2장에서 비교·분석된 다양한 결함허용 기술이 적용된 시스템들의 성능 및 비례확장성을 평가한다. 실험을 위해 2장에서 논의된 결함허용 기술들을 적용한 시스템 환경하에서 벤치마크 응용프로그램들을 개발하였고, 이 응용프로그램들 간의 성능 비교를 통해 논의된 경량화 결함허용 기술들의 성능을 평가하였다.

## 3.1. 실험 환경

제안한 시스템의 성능 평가를 위한 실험 환경

은 그림 1과 같다. 실험을 위해 9대의 Sun Ultra10 워크스테이션을 100Mbps 이더넷 네트워크로 연결하며, 그 중 한 워크스테이션에는 클라이언트 객체를 나머지 8 대의 워크스테이션에는 서버 객체들을 각각 배치한다. 실험에 참가한 워크스테이션에는 X Windows를 제외한 다른 응용 프로그램들이 실행되지 못하도록 제한한다. 메커니즘에 의해, 각각 클라이언트 및 서버와 동일한 사이트에 위치해야만 하는 복제관리자 및 HORB 데몬(daemon)을 제외한 모든 클라이언트와 서버 객체들은 서로 다른 호스트에 배치된다.




(그림 1) 실험환경

## 3.2. 실험 모델

분산 응용프로그램의 전체 수행시간 중 통신이 차지하는 비율은 일부분에 불과하며 체크포인트와 결함 복구 동작은 아주 드물게 발생한다. 보다 실제적인 환경에서 메시지 저장의 오버헤드를 분석하기 위해, 이 논문에서는 *n* queens 문제(*n* queens problem), tsp 문제(traveling sales man problem), gauss 소거법(gaussian elimination)의 세 가지 벤치마크 프로그램을 앞 절에서 논의된 다양한 결함허용 기술을 적용한 시스템 환경 하에서 각각 작성한 뒤 그 실행시간을 측정하였다. 참고문헌[12,13]에 의하면, 채택된 세 가지 벤치마크 프로그램들은 통신량, 통신 패턴 및 메모리 요구 사항에 있어서 합리적인(reasonable) 분포를 나타내며, 네트워크로 연결된 분산 환경에서 동작하는 응용 프로그램들의 전형이다.

이 세 가지 프로그램들을 실험 모델로 채택한 이유는 서로 상이한 통신 량과 통신 패턴 때문이다. *n* queens 프로그램의 경우, 클라이언트 객체와 원격 서버 객체 사이의 통신은 프로그램 실행 시작 시점과 종료 시점에서만 발생하며, 원격 서버 객체 사이에는 아무런 통신도 일어나지 않는다. *n* queens 프로그램의 전체 통신 량은 문제 크기(problem size)에 관계없이 항상 일정하다.

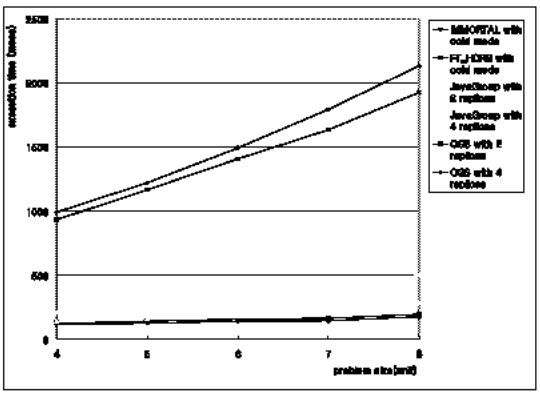
## 3.3. 성능 평가

그림 2, 그림 3, 그림 4는 *n* queens, tsp, gauss 프로그램을 Immortal[12], FT_HORB[13], JavaGroup (version 1.0), OGS(verson 0.8b) 하에서 각각 구현 한 뒤 그 실행 시간을 측정한 결과이다. X축은 문제크기를, Y축은 실행시간을 의미한다. Immortal, FT_HORB 하에서의 성능측정은 cold 대기 모드에서 체크포인트 횟수를 32로 고정하여 실시되었다. OGS의 경우에는 Visibroker for Java(version 4.5) 지원하에 클라이언트 객체가 GroupAccess 객체에 접속하도록 하였으며, Untyped 서버로 동작하도록 서버 객체를 수정하였다. JavaGroup와 OGS의 경우 그룹에 소속된 구성원의 개수를 2개, 4개로 구분하여 실험하였다.

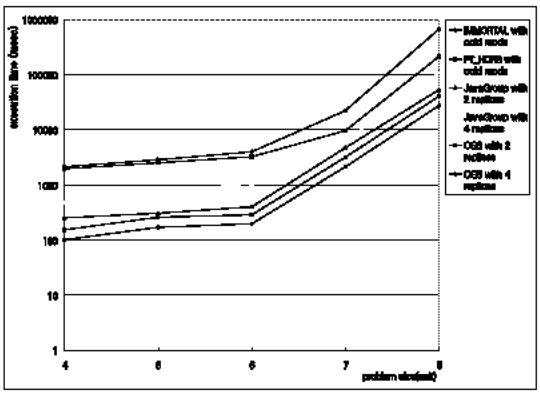
실험에 의하면, Immortal 환경에서 구현된 *n* queens, tsp, gauss 프로그램은 JavaGroup, OGS

환경에서 구현된 프로그램들에 비해 확연한 성능 향상을 보여주었다. 실험 결과는 JavaGroup, OGS 시스템의 필수 서비스인 그룹 회원 서비스, 그룹 다중전송 서비스, 가상 동기화 실행 서비스가 유발하는 결함허용 연산 오버헤드가 제안한 시스템에 비해 매우 크다는 것을 의미하며, 그룹에 소속된 구성원의 개수가 늘어날수록 그 오버헤드도 비례적으로 증가함을 보여주고 있다. 실험 결과, 신뢰성 있는 그룹 통신에 기반한 결함허용 서비스는 비교적 낮은 등급의 신뢰성과 가용성 요구를 바탕으로 결함허용에 따른 오버헤드의 최소화를 지향하는 응용프로그램에는 적합하지 않음을 의미한다.

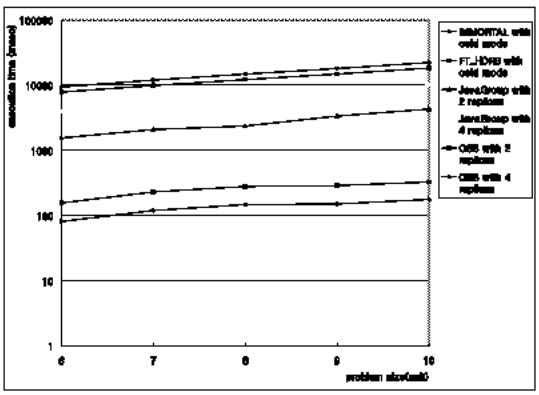
Immortal과 FT_HORB 환경에서 각각 구현된 프로그램들 간의 성능 비교에서는, Immortal이 보다 향상된 성능을 보여주고 있다. 이는 Immortal에서 채택한 송신자 기반 메시지 로깅 메커니즘이 중앙통제식 복제관리자의 병목현상을 개선한 결과이다. 실험 결과는 프로그램의 전체 통신량이 $O(c)$, $O(n)$, $O(n^2)$ 순으로 증가할수록 Immortal 시스템이 채택한 메시지 로깅 메커니즘이 보다 향상된 성능을 발휘함을 입증한다.



(그림 2) 신뢰성 있는 그룹 통신 시스템과의 성능비교(n-queens의 경우)



(그림 3) 신뢰성 있는 그룹 통신 시스템과의 성능비교(tsp의 경우)



(그림 4) 신뢰성 있는 그룹 통신 시스템과의 성능비교(gauss의 경우)

## Ⅳ. 결론

컴퓨터 및 통신 기술의 발전과 더불어 분산 컴퓨팅은 컴퓨터 분야의 주요 관심 기술들 중의 하나로서 주목받고 있다. 시스템들이 분산 컴퓨팅을 지향하고 그 구조가 점점 복잡해짐에 따라, 시스템의 부분 결함 혹은 통신 링크 결함 같은 문제점들을 해결하기 위한 연구들이 활발하게 진행되고 있다. 그러나 이러한 문제점들을 보완하

기 위해 제안되고 있는 대부분의 시스템들은 신뢰성 있는 그룹통신 기반의 능동복제 방식 프로토콜을 채택하고 있어서 결함허용 연산에 따른 심각한 오버헤드의 발생은 필연적이다.

인터넷의 보급과 더불어, 인터넷 기반의 응용 프로그램 형태로 개발되며 저가의 컴퓨팅 환경하에서 동작하는 *non mission critical* 분산 응용 프로그램의 수가 증가 추세에 있다. 이러한 응용 프로그램들에 있어서, 결함허용 기능은 필수적이지만, 제한된 자원 및 수행 시간을 사용하여 결함허용 연산에 수반되는 오버헤드를 줄이는 방향으로 제공되는 것이 바람직하다.

본 논문에서는 분산컴퓨팅 환경에서 경량화 결함허용 기능을 추가하기 위한 다양한 접근 방법에 대하여 비교하여 그 장·단점을 비교·분석하였다. 또한, 실험을 위한 환경 및 벤치마크 모델들을 소개한 뒤, 제시된 결함허용 기술들의 성능을 평가하기 위한 다양한 실험 결과들을 제시하였다.

인터넷 기반의 응용프로그램 형태로 개발되는 많은 소프트웨어들에 대한 가용성 및 신뢰성 보장 요구들이 급증하고 있다. 이러한 추세를 감안할 때, 본 연구에서 논의되고 분석된 연구 결과들은 향후 결함허용 연산에 관련된 오버헤드의 최소화 및 낮은 등급의 가용성과 신뢰성만을 요구하는 중·소규모 분산 응용프로그램의 개발에 효과적으로 이용될 수 있다.

## 참고문헌

[1] P. Narasimhan, "Transparent Fault Tolerance for CORBA", Ph.D thesis, University of California, Santa Barbara, CA, 1999.

[2] Object Management Group, *CORBA Specification,* OMG Technical Committee Document, May, 2002.

[3] T.L. Thai & A. Oram, "Learing DCOM", *O'reilly,* April, 1999.

[4] IBM, *SOMobjects: A practical Introduction to SOM and DSOM,* International Technical Support Organization, 1994.

[5] Sun Microsystems, JDK Documentation, <http://www.javasoft.com/>, 2002.

[6] S. Maffies & D.C. Schmidt, "Constructing Reliable Distributed Communication Systems with CORBA", *IEEE Communications Magazine, 14*(2), February, 1997.

[7] L.E. Moser, P.M. Melliar Smith & P. Narasimhan, "Consistent object replication in the Eternal system", *Theory and Practice of Object Systems, 4*(2), 1998, pp.81 92.

[8] P. Felber, R. Guerraoui and A. Schiper, "The implementation of a CORBA object group service", *Theory and Practice of Object Systems, 4*(2), 1998, pp. 93 105.

[9] Mark Hayden, "The Ensemble System", Technical Report 98 1662, Cornell University, January, 1998.

[10] S. Rao & H. Vin, "Egida: An Extensible Toolkit For Low overhead Fault Tolerance", In *Proceedings of the 29th Fault tolerant Computing Symposium, Madison,* Wisconsin, June, 1999, pp. 48 55.

[11] Heon Y. Yeom, Namyoon Woo, & Hyungsoo Jung, "Providing Fault Tole

rance for Parallel Applications on Grids", *GlobusWORLD 2004,* January 19 22, 2004, (San Francisco, CA, USA)

[12] 현무용, 김식, 김명준, 야마키다 지로, "IM MORTAL : 원격 메소드 호출에 기반한 결함허용 분산 미들웨어 시스템", '정보과학회 논문지', 29(5), 2002.

[13] Shik Kim, Muyong Hyun and Jiro Yamakita, FT_HORB: A Fault Tolerant Distributed Programming Environment based on RMI, *IEICE Transaction on Information and Systems, Vol.E85 D,* No.3, March, 2002

[14] E. N. Elnozahy, D.B. Johnson & Y.M. Wang, "A Survey of Rollback Recovery Protocols in Message Passing Systems", *Technical Report CMU CS 96 181,* Carnegie Mellon University, October, 1996.

[15] Hyuck Han, Jai Wug Kim, Jongpil Lee, & Heon Y. Yeom, "Practical Fault Toleranct Framework for eScience Infrastructure", *IEEE e Science 2006, Dec. 4 6,* 2006, Amsterdam, Netherlands, 167 178, April, 2005)

# The Performance Comparison of Low-Overhead Fault Tolerant Services based on Distributed Object

Shik, Kim*, Muyong, Hyun*

## Abstract

As most application programs are more sophisticated and are adopted the distributed object tech nology, the object based distributed design became widespread since it supports portability and reusability. The approaches for fault tolerant distributed computing are categorized into the *active replica* mechanism for *mission critical* application programs and the *passive replica mechanism* for *non mission critical* ones, when fault tolerant facilities are added on.

Our paper introduces the *pros* and *drawbacks* of several approaches for the add on low overhead fault tolerant services by the surveys and shows the results of experiments for bench mark models in order to demonstrate their performance.

Key Words : low overhead fault tolerance, distributed object, RMI

* Professor, School of Information and Communication System, Semyung University
** Senior Researcher, R&D Institute of Korea Electric Power Data Network Co., Ltd