원격 컴파일 서비스를 위한 분산 시스템에 관한 연구

구민오[O] 조나연[OO] 민덕기[O1]
[O]건국대학교 컴퓨터공학부
[O]happykus@konkuk.ac.kr, [OO]nycho@konkuk.ac.kr, [O1]dkmin@konkuk.ac.kr

A Study on Distributed System for Remote Compile Service

Mino Ku[O], Na-Yun Cho[OO], Dugki Min[O1]
[O]School of Computer Science and Engineering, KonKuk University

요 약

컴파일러 설치 및 환경정보 설정과 같은 일련의 컴파일러 설정과정은 개발환경 구성과정 중 중요한 과정이며, 집단 개발환경에서의 개발환경 동일화 과정은 많은 비용을 초래할 수 있게 된다. 더욱이 개발 초급자 또는 공학교육 학습자에게는 앞서 언급한 개발환경 구성과정이 개발언어에 대한 학습과정의 진입장벽으로 작용할 수 있다. 뿐만 아니라, 다양한 개발언어에 대한 학습을 이루고자할 경우, 이에 수반되는 다양한 컴파일러 확보·설정 등의 과정은 개발 시스템과 컴파일 시스템이 일치함에 따라 환경구성의 복잡도가 증가하게 된다. 따라서 본 논문에서는 컴파일 환경을 구성한 분산 시스템을 기반으로 원격 컴파일 서비스를 제공하는 분산 시스템 아키텍처를 제시하며 이에 대한 구현 사항을 제공한다. 특히 브로커 아키텍처를 기반으로 한 분산 시스템 상에서 처리 시스템들에 빈번히 발생할 수 있는 고장 상황에 대해 고가용성(High Availability)을 달성하기 위해 처리 시스템간의 가용 여부를 관리하는 Coordinator 처리 시스템 선출을 위해 선출 알고리즘으로 Bully 알고리즘을 적용하였다.

1. 서 론

컴파일러 설치 및 환경정보 설정과 같은 일련의 컴파일러 설정과정은 개발환경 구성과정 중 중요한 과정이다. 이에 따라 컴파일러 최적화(Compiler Optimization)은 개발 생산성에 중요한 영향을 미침으로써 개발 도구 중 중요한 역할을 하는 컴파일러에 대한 시스템 및 개발 내용에 대한 최적화는 다양하게 시도되고 있다.[1,2,3,4,5] 또한 많은 수의 개발자들이 개발에 참여하는 집단 개발환경에서는 개발 코드에 대한 컴파일 환경을 일치시키는 동일화 과정에서 시간적·비용적 문제를 초래할 수 있게 된다. 더욱이 개발 초급자 또는 공학교육 학습자에게는 앞서 언급한 개발환경 구성과정이 개발언어에 대한 학습과정의 진입장벽으로 작용할 수 있게 됨으로써 초급적인 개발시도에 난항을 겪고 있으며, 이는 향후 개발 학습에 있어서 악영향을 미칠 수 있게 되므로 위와 같은 진입장벽의 문제는 공학교육자에게는 고려대상이 될 수 있다. 그리고 공학교육 환경에 있어서 제한된 학습공간을 다양한 수업에 활용함에 따라 각기 요구되는 개발환경 역시 다양화 되어야 하고, 이에 따라 동일한 개발 시스템의 변이 가능성이 높아지며, 특히, 개발환경의 고장 등의 문제로 인한 불능 상태에 빠지게 될 경우, 학습 환경의 저해를 초래할 수 있게 된다.

앞서 언급한 문제들을 해결하기 위해서는 개발 시 개발언어를 활용하는 저작 환경과 저작 결과물을 양산하는 컴파일 환경을 분리함으로써 달성할 수 있다. 그러나 집단 사용자에 의한 컴파일 서비스 요청이 이루어질 수 있는 환경에 대응하기 위해서는 컴파일 환경이 구성된 다중의 시스템이 요구되며, 이러한 다중 시스템을 기반으로 하기 위해서는 분산 컴퓨팅이 적용되어야 보다 원활한 원격 컴파일 서비스 제공이 가능하게 된다.[6] 따라서 본 논문에서는 컴파일 환경을 구성한 분산 시스템을 기반으로 원격 컴파일 서비스를 제공하는 분산 시스템 아키텍처를 제시하고 내고장성(Fault Tolerant) 메커니즘을 적용하여 보다 신뢰성 높은 시스템 구현을 통해 앞서 언급한 문제를 해결하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제시하는 분산 시스템에서 채용한 분산 아키텍처에 대해 설명하고 있으며, 3장에서는 본 논문에서 제시하는 분산 원격 컴파일 시스템 아키텍처의 각 요소들에 대해 설명하고 있다. 4장에서는 각 시스템들 간의 시스템 상호과정에 대해 설명을 하고 있으며, 5장에서는 처리 시스템(Compile System)에 대한 내고장성 기능의 구현 사항에 관해 일반 처리 시스템 고장과 Coordinator 처리 시스템의 고장 상황으로 나누어 설명하고 있다. 6장에서는 처리 시스템에서의 컴파일 서비스 방식을 언급하고 있으며, 7장에서는 본 논문에서 제시한 시스템의 구현사

[1] 교신저자: 민덕기 (dkmin@konkuk.ac.kr)

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2010-C1090-1031-0003)

항 및 구동환경에 대해서 언급하고 있다. 8장에서는 결론 및 향후 계획을 제시하고 있다.

2. 관련 연구

본 논문에서 제시하는 원격 컴파일 서비스를 위한 분산 시스템 아키텍처는 브로커 패턴(Broker Pattern)[7, 8,9]을 응용하여 처리작업을 위한 데이터를 Worker 노드에 전송하는 Dispatcher의 위치정보를 Client에 전송한다. 브로커 패턴에서는 중개자에 해당하는 브로커(Broker)를 통해 원격 컴포넌트에 대한 투명성을 제공하는 것이다. 따라서 원격 시스템에 위치하는 컴포넌트는 서비스 호출을 통해 다른 컴포넌트에서 제공하는 서비스에 접근할 수 있으며, 런타임시 컴포넌트의 추가 및 제거가 가능한 특징을 가지고 있다. 브로커패턴의 대표적인 사례로는 CORBA (Common Cbject Request Broker Archi-tecture)[10,11], DCOM(OLE 2.x)[12], EJB2[13] 등이 있다. 본 논문에서는 브로커 패턴의 기능적인 역할 측면에서 원격 컴파일 서비스를 제공하는 Dispatcher와 Client A-gent간의 중개하는 역할로써 Broker를 활용하고 있다.

본 논문에서 제시하는 분산 원격 컴파일 시스템에서는 컴파일 작업을 수행하는 처리 시스템 (Compile System)에 발생할 수 있는 고장 상황에 대해 내고장성(Fault Tolerant)을 달성할 수 있도록 Heartbeat를 통해 시스템의 가용 여부 검사 하며, 처리 시스템 중 여타 처리 시스템들의 상태를 관리하여 Dispatcher에 가용 처리 시스템 목록을 제공하는 등의 역할을 하는 Coordinator 처리 시스템의 고장상황 발생 시 Coordinator 처리 시스템을 선출할 수 있도록 선출 알고리즘 (Election Algorithm) 중 Bully Algorithm[14,15,16,17]이 적용 되었다. Bully Algorithm에서 채용한 선출 알고리즘은 처리 시스템 중 고장 상황 발생시 가장 먼저 발견한 처리 시스템이 선출 메세지(Election Message)를 자신보다 높은 우선순위로 할당된 처리 시스템에게 보내게 되고, 응답메세지가 올 경우 Coordinator 처리 시스템이 어떤 시스템이 될 것인지를 대기하게 된다. 이를 반복함으로써 가장 높은 우선순위가 할당된 처리 시스템은 더이상 보낼 곳이 없게 되므로 스스로 Coordinator 처리 시스템이 됨을 모든 처리 시스템에 보내게 되면서 Coordinator 처리 시스템을 선출한다.

3. 시스템 아키텍처

본 논문에서 제시하는 원격 컴파일 서비스를 위한 분산 시스템은 (그림 1)에서 도시한 바와 같이 구성 및 커뮤니케이션이 일어나며, 핵심적인 시스템들은 아래와 같다.

- Client
- Broker
- Dispatcher
- Compile System

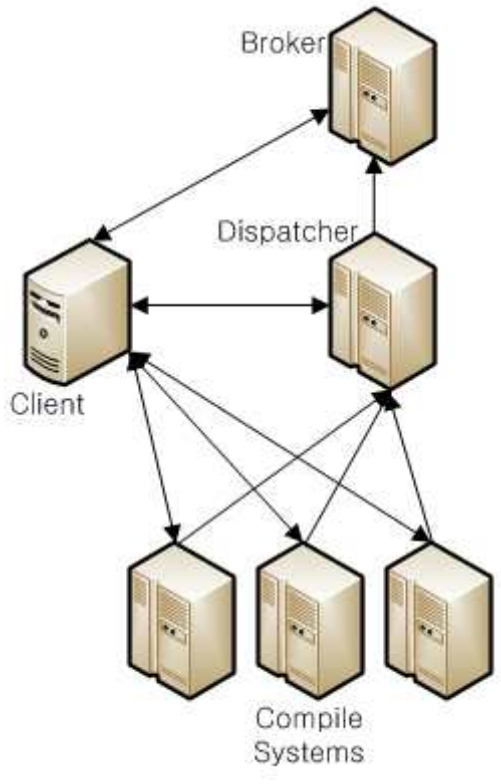
3.1. Client

클라이언트는 컴파일 서비스를 이용하는 시스템으로써, Broker와 통신을 통해 컴파일 서비스를 제공하는 Dispatcher의 위치 정보 (IP와 Port 정보)를 제공받으며, Dispatcher로부터 현재 가용할 수 있는 Compile System의 위치 정보 (IP와 Port 정보)를 제공받는다.

Dispatcher로부터 제공받은 Compile System의 위치 정보 (IP와 Port 정보)를 기반으로 Compile System에 컴파일러 선택정보, 소스코드를 제공을 통해 컴파일 서비스 요청하고 서비스 요청 결과를 통해 컴파일 완료시 컴파일 결과물 또는 컴파일 중 발생하는 에러메세지 등을 수신한다.

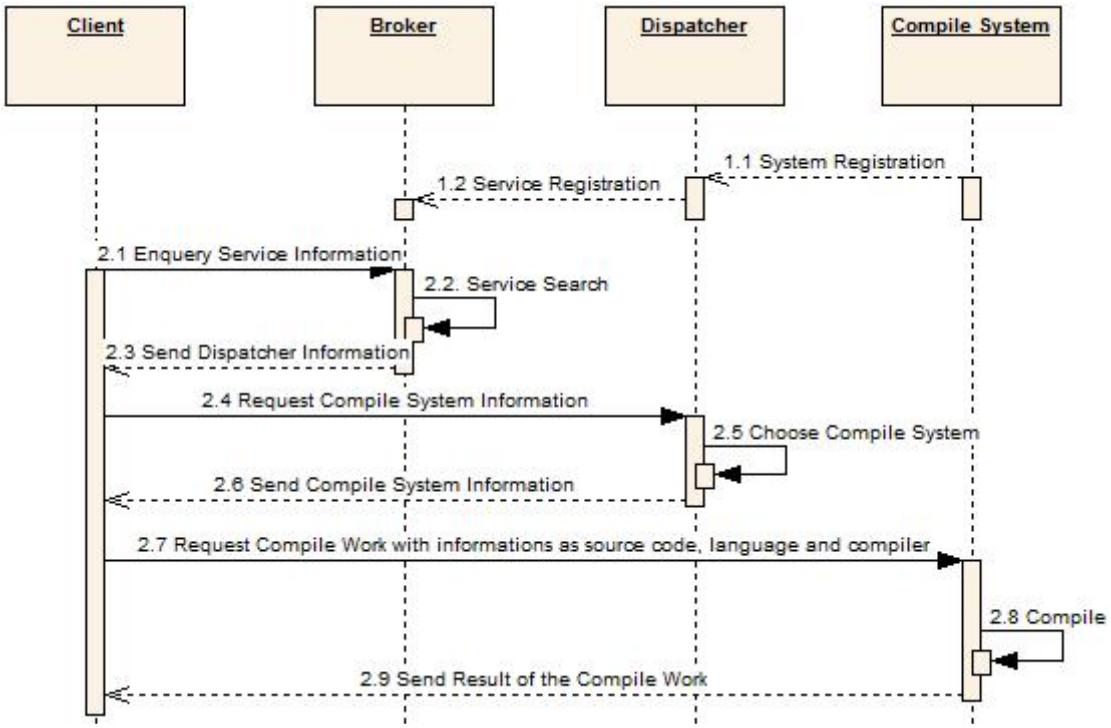
3.2. Broker

Broker는 Dispatcher에서 제공하는 서비스에 대한 목록의 검색, 등록, 수정, 삭제 등의 작업을 수행하는 역할을 하는 시스템이다. Dispatcher가 제공하는 서비스에 대한 등록, 수정, 삭제에 관한 요청을 Dispatcher로부터 받으며, 이를 기반으로 Broker는 “서비스 목록” 관리 작업을 수행한다. Client에게는 Broker내에 등록된 서비스들에 대한 검색 서비스를 제공해주며, 요청에 상응하는 서비스 존재 시 Broker는 Client에게 해당 Dispatcher의 위치정보(IP, Port 번호)를 제공한다.



(그림 1) 분산 원격 컴파일 시스템 구성

만 아니라, 이러한 내고장성(Fault Tolerance)을 보장하



(그림 2) 시스템 상호과정

3.3. Dispatcher

Dispatcher는 Client로부터 요청된 컴파일 작업을 수행할 Compile System의 위치 정보 제공을 수행하는 시스템이다. 또한 Dispatcher는 Broker에 자신이 제공하는 서비스항목에 대해 Broker에 등록, 수정, 삭제 등을 요청함으로써 Client로 부터의 서비스 요청 사항에 대해 Compile System들의 상태 변이에 대해 능동적으로 대처할 수 있다.

Dispatcher는 컴파일 작업 수행 시 수반되는 소스코드 및 컴파일러 선택 정보 등을 Compile System에 중개하지 않는 것을 특징으로 하고 있다. 이는 다수의 Client에 의해 빈번한 컴파일 작업 요청 시 컴파일 작업을 위한 데이터 전송에 대해 Dispatcher를 점유하게 되는 문제를 예방하기 위해서이다.

한편 Dispatcher는 Client로 부터의 Compile System 위치정보 요청 시 특정한 Compile System의 위치정보를 제공하기 위해 부하분산 (LoadBalancing) 메커니즘이 적용되어야 한다.

3.4. Compile System

Compile System은 C, C++, PASCAL 등의 다양한 컴파일러들이 설치·설정된 시스템으로써, Client로부터 요청된 컴파일 작업에 대해 상응하는 작업을 수행하고 작업 수행 결과를 Client에 전달하는 역할을 주된 역할로써 수행한다.

다수의 Compile System을 기반으로 하는 본 논문에서의 시스템은 Compile System의 고장 및 오류 상황에 대해 대응할 수 있는 메커니즘이 마련이 되어야 한다. 뿐만 아니라, 이러한 내고장성(Fault Tolerance)을 보장하기 위해 Dispatcher에 Health Check 메커니즘을 적용할 경우, Dispatcher에 이 또한 부하로 작용할 수 있게 되므로, 본 논문에서 제시하는 원격 분산 컴파일 시스템에서는 선출 알고리즘(Election Algorithm)을 기반으로 한 Compile System간의 특정한 처리 시스템 (Coordinator 처리 시스템)이 다수의 Compile System에 대한 관리 작업을 수행하며, 주기적으로 서비스 가능한 Compile System 목록을 Dispatcher에 제공하는 방식이 적용되었다.

4. 시스템 상호과정

(그림 2)는 본 논문에서 제시하는 원격 컴파일 서비스를 위한 분산 시스템 아키텍처의 상호과정을 표현한 그림으로써, 대표적인 시스템 상호과정은 다음과 같이 크게 2가지로 나뉠 수 있다.

- 과정 1. 서비스 등록과정
- 과정 2. 서비스 요청과정

4.1. 서비스 등록과정

다수의 Compile System을 기반으로 하는 본 시스템에서는 서비스의 전제 조건으로 컴파일 서비스를 제공할 수 있는 Compile System이 Dispatcher에 등록됨으로써 서비스 발생과정이 이루어져야 하며, 다음으로는 이를 기반으로 서비스 제공의 기반을 갖춘 Dispatcher의 Broker 등록으로써 서비스 제공을 완성할 수 있다. 따라서 (그림 2)에서 확인할 수 있는 바와 같이 Compile System은 Dispatcher에 등록 과정을 가지고 (과정 1.1), 이

를 통해 서비스 발생이 완료된 Dispatcher는 Broker에 서비스 등록 과정을 가진다 (과정 1.2).

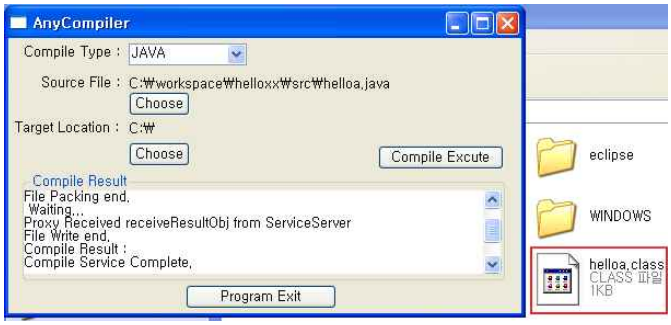
4.2. 서비스 요청과정

(과정 1)로써 서비스 요청의 기반이 완성된 다음에는 Client가 컴파일 서비스를 요청할 수 있게 된다.

Client는 Broker에 컴파일 서비스를 제공하는 Dispatcher 유무를 의뢰하는 조회과정을 가진다 (과정 2.1). Client로부터 조회요청을 받은 Broker는 내부에 등록된 서비스 목록 중 Client가 요청한 컴파일 서비스 유무를 검색하는 과정을 가진다 (과정 2.2).

만약 Broker에 내부에 등록된 서비스 중 Client가 요청한 컴파일 서비스 존재 시 (과정 2.3)과 같이 해당 Dispatcher의 위치정보를 제공한다. (과정 2.3)의 과정으로 확보된 Dispatcher 위치정보를 기반으로 Client는 Dispatcher에 컴파일 서비스를 수행할 수 있는 Compile System의 위치정보를 요청하고 (과정 2.4), Dispatcher는 Round Robin 부하 분산 알고리즘 방식을 통해 등록된 Compile System 중 하나를 선택한다 (과정 2.5). 다음으로 Dispatcher는 (과정 2.5)를 통해 선정된 Compile System 중 하나에 대한 위치정보를 Client에 전송하고 (과정 2.6), Client는 소스코드, 소스코드 제작언어, 컴파일러 등과 같은 정보들과 함께 컴파일 서비스를 (2.6 과정)에서 확보한 Compile System의 위치정보를 기반으로 컴파일 작업을 요청한다 (과정 2.7). Compile System은 (과정 2.7)을 통해 Client로부터 전송된 컴파일 작업 정보를 기반으로 컴파일 작업을 수행하고 (과정 2.8), 이에 대한 작업 결과를 (과정 2.9)와 같이 전송한다. (과정 2.8) 중 컴파일 시 에러가 나지 않을 경우, 컴파일 된 결과물을 전송하며, 컴파일 시 에러가 날 경우, 에러메시지를 (과정 2.9)의 과정에서 Client에 보내게 된다.

(그림 3)은 클라이언트 프로그램의 실행 및 구동 화면으로써, 컴파일 언어에 대한 정보 및 소스코드를 제공한 뒤, 컴파일 성공 시 컴파일 결과물을 클라이언트 시스템에 생성 된 화면을 나타내고 있다.



(그림 3) 컴파일 서비스 요청 및 생성물

5. 고장 극복

5.1. 일반 처리 시스템 고장

분산 처리 환경에서는 처리 작업을 수행하는 시스템들의 고장 상황이 임의적·비예측적 발생할 수 있다. 따라서 본 논문에서 제시하는 분산 원격 컴파일 시스템에서는 특정한 처리 시스템의 고장 상황에도 불구하고 전체적인 시스템의 서비스 제공 메커니즘에 영향을 극소화 시킬 수 있도록 내고장성(Fault Tolerant)을 지원하고 있다.

본 논문에서 제시하는 분산 원격 컴파일 시스템에서는 처리 시스템 (Compile System) 중 하나는 Coordinator가 되어서 다른 처리 시스템들로부터 일정한 간격으로 Heartbeat 메시지를 수신하고 일정 시간내 Heartbeat 메시지가 수신되지 않았을 경우, Coordinator로 지정된 처리 시스템은 해당 처리 시스템을 제외 시킨 처리 시스템 목록을 Dispatcher로 전송함으로써, 고장 시스템에 처리 작업이 할당 되지 않도록 하고 있다.

5.2. Coordinator 처리 시스템 고장

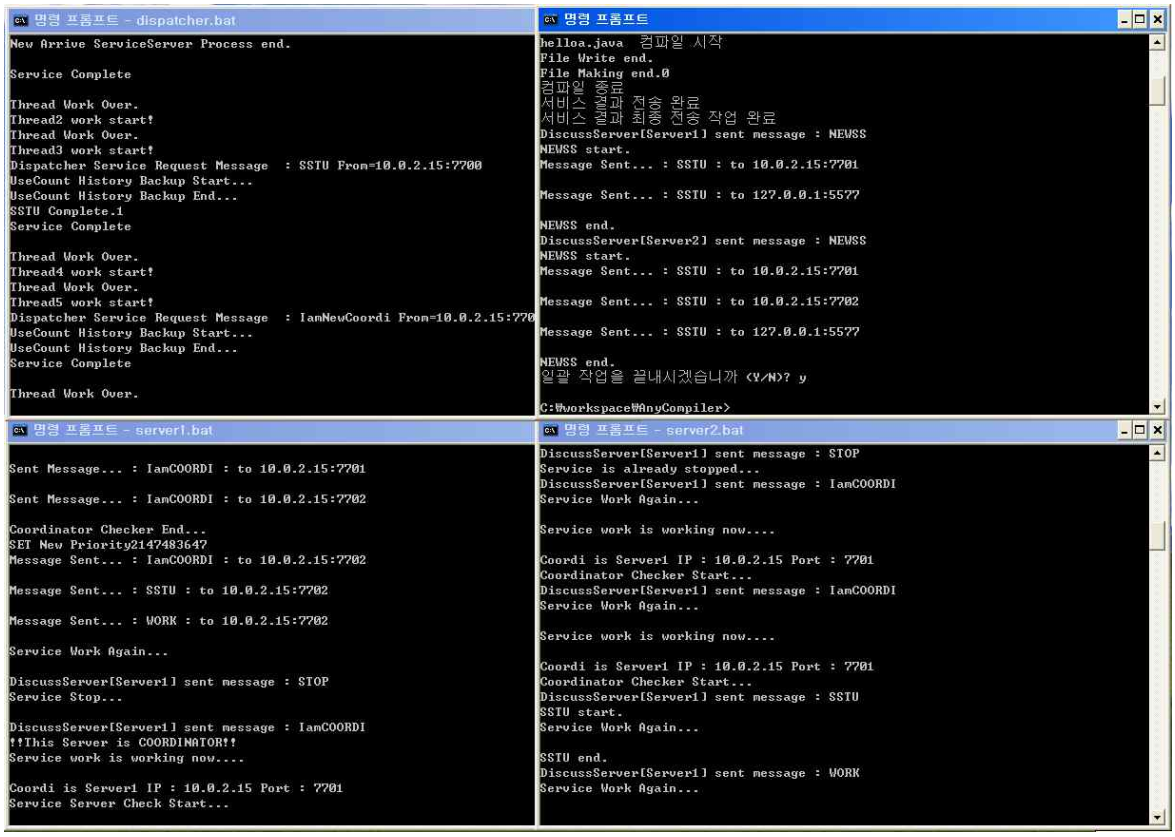
본 논문에서 제시하는 분산 원격 컴파일 시스템에서 Coordinator 처리 시스템은 처리 작업과 함께 처리 작업 (Compile 작업)을 수행하는 다른 처리 시스템들을 관리하는 역할을 병행하는 처리 시스템으로써, 해당 시스템은 주기적으로 Dispatcher 시스템에 가용 처리 시스템 목록을 제공한다. 그러나 고장 상황이 발생할 경우, 관리 작업 및 처리 작업을 동시에 불가능하게 되는 상황으로 연결이 되므로 Coordinator 처리 시스템의 고장은 본 논문에서 제시하는 시스템 구조상에서 발생할 수 있는 문제 중 비중있는 문제에 해당된다. 따라서 이러한 문제를 해결하고자 Coordinator 처리 시스템 고장 상황 발생 시 처리 시스템들은 선출 알고리즘 (Election Algorithm) 중 Bully 알고리즘 [14,15,16,17]을 적용하여 새로운 처리 시스템을 Coordinator 처리 시스템으로 선출하게 된다. (그림 4)는 앞서 설명한 Coordinator 처리 시스템에 고장 상황과 유사한 임의적인 프로그램 종료 상황 발생 시 새로운 Coordinator 처리 시스템을 선출하는 과정 중 출력하는 메시지들을 나타낸 그림이다.

6. 컴파일 서비스

본 논문에서 제시하는 분산 원격 컴파일 시스템의 처리 시스템(Compile System)에서 채용하는 컴파일 서비스는 기존의 공개된 컴파일러를 프로세스 구동 방식 (Process Execution) 방식으로 컴파일을 진행함으로써 달성된다.

처리 시스템(Compile System)은 클라이언트로부터 전달된 소스코드를 컴파일 하고 컴파일 성공 시 클라이언트에게 컴파일 시 생성되는 컴파일 결과물을 클라이언트에게 반환한다. 만약 컴파일 시 에러가 발생될 경우, console 출력물로 제공되는 에러 사항들을 클라이언트 시스템에 반환함으로써 사용자는 코드 작성 과정에서 발생된 문제점을 확인할 수 있으며, 이를 수정 한 뒤 컴파일 과정을 반복할 수 있다.

목록이다.



(그림 4) Coordinator 처리 시스템 선출 화면

7. 구현 사항 및 구동 환경

본 논문에서 제시하는 분산 원격 컴파일 시스템은 JAVA 언어를 기반으로 SUN JDK 1.6와 Eclipse를 기반으로 개발되었으며, 구현 및 구동 시스템 환경은 아래의 [표 1]과 같다.

[표 1] 구현 및 구동 시스템 환경

항목	사양
CPU	AMD Phenom II X4 945 3.0 GHz
RAM	8 Gb
OS	Windows XP 32bit Windows 7 64bit
Java	Sun JDK 1.6
Eclipse	Eclipse 3.5 version

또한 처리 시스템(Compile System)에 적용된 컴파일 서비스는 JAVA, C, C++, Pascal이며, 처리 시스템의 운영체제를 Windows로 함에 따라 Windows 운영체제 하에서 구동 가능한 컴파일러를 중심으로 구성하였다. Java 언어에 대한 컴파일러는 Sun Microsystems사의 Sun JDK 1.6을 기반으로 하였으며, C와 C++ 언어의 경우 Gcw[18], Pascal 언어의 경우 Free Pascal[19] 을 이용하였다. [표 2]는 보다 상세한 컴파일 서비스에서 지원하는 언어 및 해당 언어를 지원하기 위한 컴파일러

[표 2] 컴파일 서비스 목록

지원 언어	컴파일러
Java	Sun JDK 1.6
C	Gcc for Windows (gcw)
C++	
Pascal	Free Pascal 2.4.0

8. 결론 및 향후계획

개발환경 구축 및 동일화 과정은 개발 결과물의 양산 과정을 포함하여 생산성, 효율성 측면에서 매우 중요한 이슈이며, 본 사안의 중요성으로 인해 컴파일러 최적화를 포함하는 개발 환경 최적화에 관한 연구는 지속적으로 이루어져 왔으며 앞으로 대규모화 분산화 되어가는 개발환경의 변화 속에서 더욱더 활발히 진행될 것이다. 뿐만 아니라, 공학 교육에 있어서 컴퓨터 프로그램 언어 교육 시 개발 환경에 대한 구축·실시·운영에 있어서 지속적으로 발생할 수 있는 문제점들 역시 개발 환경에 대한 동일화와 항상성 유지 문제에서 최종적인 개발 결과물 양산 환경과 개발 저작 환경의 분리는 대안으로 거론될 수 있다. 따라서 본 논문에서는 원격 컴파일 서비스를 위한 분산 시스템 아키텍처를 제시하고 이를 구현한 구현물을 제시하고 있다.

원격 컴파일 시스템은 크게 서비스를 제공받는 Client,

서비스를 제공하는 집단을 중개하는 Broker, 서비스 제공 집단의 세부적인 처리 시스템과의 중개를 담당하는 Dispatcher, 그리고 Client와의 상호작용을 통해 실질적인 컴파일 서비스를 제공하는 Compile System으로 나눠지며, 앞서 언급한 시스템 요소들 간의 "서비스 등록과정" 및 "서비스 요청과정"등의 상호작용을 통해 실질적인 서비스가 이루어질 수 있도록 설계 및 구현되었다.

본 논문에서 제시하는 원격 컴파일 서비스를 위한 분산 시스템은 다수의 Compile System을 기반으로 서비스를 제공하게 되므로, 분산 환경에서 흔히 발생할 수 있는 시스템 오류 및 고장 상황에 대해 높은 서비스 가용성 (Availability) 및 신뢰성 (Reliability)을 제공하기 위해 내고장성 (Fault Tolerant) 기능을 적용하였다. 뿐만 아니라, 다수의 Client로부터 요청되는 컴파일 작업을 분배함에 있어서 Dispatcher내에 등록된 다수의 Compile System에 원활한 분배작업을 위해 저비용 부하 분산 알고리즘 중 RoundRobin 기법을 적용함으로써 이를 해결했다. 따라서 향후 연구에서는 처리 시스템(Compile System)간의 보다 효율적인 선정 작업 (Election)을 가능하게 하는 기법에 대해 보완적 측면의 연구·구현을 진행할 계획이며, Dispatcher에 등록된 다수의 처리 시스템(Compile System)에 컴파일 작업 요청을 분산함에 있어서 보다 다양한 부하분산 알고리즘 및 메커니즘을 선택적으로 적용할 수 있도록 현재 내재되어 있는 부하분산 알고리즘 부분에 대해서 모듈화 및 추가적인 부하분산 모듈에 관한 연구 및 구현을 진행할 계획이다.

6. 참고문헌

[1] John Cavazos, Grigori Fursin, Felix Agakov, Edwin Bonilla, Michael F.P. O Boyle, Olivier Temam "Rapidly Selecting Good Compiler Optimizations using Performance Counters", 2007

[2] M. Haneda, P. M. W. Knijnenburg, H. A. G. Wijshoff. "Automatic selection of compiler options using non-parametric inferential statistics", Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, pp. 123-132, 2005.

[3] Z. Pan, R. Eigenmann, "Fast and effective orchestration of compiler optimizations for automatic performance tuning", Proceedings of the International Symposium on Code Generation and Optimization, pp. 319-332, 2006.

[4] Z. Pan, R. Eigenmann, "Fast automatic procedure-level performance tuning", Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 2006.

[5] S. Triantafyllis, M. Vachharajani, N. Vachharajani, and D. I. August, "Compiler optimization-space exploration", Proceedings of the International Symposium on Code Generation and Optimization", pp. 204-215, 2003.

[6] Andrew S. Tanenbaum, Maarten Van Steen, "Distributed Systems: Principles and Paradigms (2nd Edition)", Prentice Hall, 2007

[7] M. Kircher and P. Jain, "Pattern-oriented software architecture", vol. 3 , Patterns forResource Management, J. Wiley, 2004.

[8] Hayden, Ra C. and Carrick, Christina, Yang, Qiang, "Architectural design patterns for multiagent coordination", the 3rd Int. Conf. on Autonomous Agents, 1999.

[9] D. Schmidt, M. Stal, H. Rohnert, F. Buschmann, "Pattern-oriented softwarearchitecture", vol. 2 , Patterns for concurrent and networked objects, J. Wiley, 2000.

[10] OMG, http://www.corba.org/

[11] J. Siegel, " CORBA 3 Fundamentals and Programming, 2nd Edition", John Wiley & SonS, 2000.

[12] R. Sessions, "COM and DCOM : Microsoft's Vision for Distributed Objects", John Wiley & Sons, 1997

[13] D. M. John Crupi, Deepak Alur, "Core J2EE Patterns: Best Practices and Design Strategies, 1st edition", Prentice Hall PTR/Sun Microsystems Press, 2001.

[14] Garcia-Molina, H., "Elections in a Distributed Computing System", IEEE Transactions on Computers, Vol. C-13, No. 1, pp. 48-59, 1982.

[15] Tanenbaum, A. S., "Distributed Operating System, Pearson Education", 2002.

[16] Sinha, P. K., "Distributed Operating Systems Concepts and Design, Prentice Hall", 2002.

[17] Ehsanul Q, Mamun K, Masum SM, Abdur M, Mustafa R. "Modified Bully Algorithm for Electing Coordinator in Distributed Systems", CD Proceedings of the 3rd WSEAS International Conference on Software Engineering, 2004.

[18] GCW, http://gcw.sourceforge.net

[19] Free Pascal, http://www.freepascal.org