

분산 컴파일 지원 시스템 설계 및 구현

조현주 박재규⁰ 정재형 조영우 김정애 유기영
경북대학교 컴퓨터공학과 정보보호연구실
i_hjcho@infosec.knu.ac.kr yook@knu.ac.kr

Design and Implementation of Distributed Compiling Support System

Hyeon-Ju Cho J.G. Park⁰, J.H. Jeong, Y. W. Cho, J.A. Kim Kee-Young Yoo
Dept. of Computer Engineering, Kyungpook National University.

요 약

모바일 분야의 비약적인 발전과 더불어 모바일 분야의 소스 크기 또한 커져 컴파일 시간이 심각한 문제로 대두 되고 있다. 본 논문에서는 컴파일 시간을 줄이기 위한 방법으로 분산 컴파일을 지원하는 시스템을 설계 및 구현하였다. 제안하는 분산 컴파일 지원 시스템은 소스를 분산하기 위하여 컴파일 옵션이 아니라 make 파일 내용 중에서 생성되어야 할 오브젝트와 오브젝트 생성규칙을 가지고 있는 부분만을 종속성 분석 결과에 따라 분할하여 재생성함으로써 분산 컴파일이 가능하도록 설계하여 분산컴파일을 위한 전처리부분에 드는 시간을 최대한 줄여 분산 컴파일 성능을 높일 수 있도록 하였다. 성능 테스트 결과에 따르면, 단일 시스템에서의 컴파일 할 때보다 약 4배 정도 컴파일 시간이 단축된 결과를 보이고 있다.

1. 서 론

모바일 분야의 비약적인 발전은 그와 관련된 프로그램의 개발 주기 또한 짧아질 것을 요구한다. 하지만 소프트웨어를 개발할 때 필수 과정 중의 하나인 컴파일 시간이 소스 크기의 증가로 인해 계속 늘어나고 있다. 따라서 컴파일 시간을 줄이기 위한 새로운 컴파일 환경이 요구된다.

본 논문에서는 컴파일 시간을 단축시키기 위해 분산 컴파일을 수행할 수 있도록 지원하는 분산 컴파일 지원 시스템(DCSS:Distributed Compiling Support System)을 설계 및 구현하고 그 성능 테스트 결과를 논한다. 제안하는 분산 컴파일 지원 시스템(DCSS)의 특징은 소스를 컴파일하기 위해 컴파일 옵션을 추출하는 것이 아니라, make파일과 make 파일을 구성하는 여러 min 파일(*.min)을 분석하여 명시된 소스 파일들간의 종속성에 따라 소스를 분산 할 수 있도록 make파일을 구성하는 min 파일의 일부를 변경하여 재생성함으로써 분산 컴파일이 가능하도록 설계한 것이다. 이렇게 설계함으로써 분산 컴파일을 하기 위한 전처리 작업에 드는 시간을 최소화 시키고자 하였다. 소스 파일마다 컴파일 옵션을 추출하여 분산 컴파일이 가능하도록 설계할 경우, 소스 파일이 커질수록 분산 컴파일의 전처리

작업에 드는 오버헤드가 더 커져 분산 컴파일의 성능을 저하시키는 문제로 대두될 것이기 때문이다. 또한, 계층적 구조의 make 파일을 가지고 있는 프로젝트라면 분산 컴파일 지원 시스템(DCSS)을 사용하여 컴파일 할 수 있도록 설계하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 분산 컴파일을 지원해주는 시스템에 대해서 고찰하고, 3장에서는 본 논문에서 제안하는 분산 컴파일 지원 시스템(DCSS)을 논하고, 4장에서는 분산 컴파일 지원 시스템(DCSS)의 분산 컴파일 성능 테스트 결과를 보여주며, 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

최근 시장에 나와있는 분산 컴파일 시스템으로는 IncrediBuild[1]와 distcc[2] 등이 있다.

IncrediBuild[1]는 마이크로소프트사의 Visual C++ 과 통합하여 네트워크로 연결된 워크스테이션을 통해 분산된 컴파일 환경을 제공하여 컴파일 시간을 줄이는 시스템이다. IncrediBuild는 컴퓨터 한대에 한 개의 소스 파일만을 컴파일하는 프로그램으로, 네트워크에 있는 각 컴퓨터에 소스를 분산시켜서 컴파일하고 그 결과를 반환

받는 방식으로 동작한다. 그러나 이 시스템에서는 요청된 컴파일 작업의 우선순위를 가장 낮은 값으로 설정해서 다른 작업의 처리 효율을 낮추지 않도록 설계되어 있기 때문에 이로 인해 컴파일 완료시간이 지연되는 단점을 가지고 있다.

Distcc[2]는 클라이언트와 서버로 구성되며 c와 c++ 파일을 분산 컴파일할 수 있도록 해주는 시스템이다. 클라이언트는 전처리를 돌리고 서버에 요구를 전달한 후 반환되는 결과를 사용자에게 알려주며, 서버는 컴파일 명령과 소스 코드를 포함하고 있는 요구를 받아 목적 코드를 생성하거나 혹은 에러가 발생했을 경우 에러 메시지를 클라이언트로 반환하는 역할을 수행한다.

IncrediBuild[1]와 distcc[2]는 c 혹은 c++ 파일들을 분산시켜 목적 코드를 생성함으로써 컴파일 시간을 줄였지만 소스 파일 형태가 c와 c++ 소스로 한정되어 있다는 단점을 가지고 있다. 본 논문에서 제안하는 분산 컴파일 지원 시스템은 다양한 형태의 소스를 지원하며, 네트워크 상의 각 호스트에 컴파일 툴이 설치되어 있고, Make유틸리티[4]로 빌드되는 프로젝트라면 분산 컴파일이 가능하도록 설계되어 있다.

3. 분산 컴파일 지원 시스템 (DCSS)

3.1 Make 파일 구조

분산 컴파일을 하고자 하는 프로젝트의 Make 파일 구성이 그림 1과 같이 계층적 구조를 가지고 있어야 분산 컴파일 지원 시스템(DCSS)을 사용하여 컴파일을 할 수 있다.

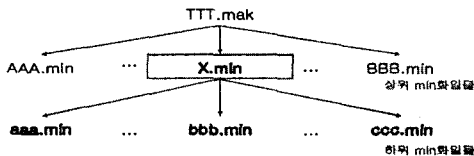


그림 1 Make 파일 구성도

3.2 분산 컴파일 지원 시스템 (DCSS)의 구성 및 특징

분산 컴파일 지원 시스템의 구성은 그림2와 같다. 컴파일에 필요한 여러 가지 정보를 입력 받는 사용자 인터페이스(UI)와 하위 min파일 간의 종속성과 소스 파일

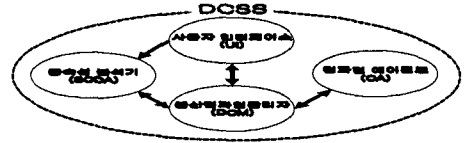


그림 2 분산 컴파일 지원 시스템(DCSS) 구성도

간의 종속성을 분석하는 종속성 분석기(SCDA), 그리고 종속성에 따라 네트워크상의 호스트들에게 소스를 배포하고 목적코드를 돌려 받으며, 이들 호스트들을 관리하는 분산 컴파일 관리자(DCM)와 요청된 컴파일을 수행하고 결과를 분산 컴파일 관리자(DCM)에게 알려주는 컴파일 에이전트(CA)로 구성 된다. 컴파일 에이전트로 참여하는 호스트를 결정할 때 단위시간 내에 평균 CPU 이용률이 특정 임계 값보다 큰 호스트는 시스템이 과부하 상태인 것으로 판단하고, 컴파일 에이전트에서 배제하여 컴파일 완료 시간이 지연되는 것을 방지하였다. 또한, 리모트의 컴파일 에이전트가 컴파일 작업 중에 에러나 예외 상황이 발생하여 컴파일을 완료하지 못할 경우 완료하지 못한 호스트의 컴파일 작업은 로컬 컴퓨터에서 컴파일을 수행하도록 설계하여 결함에도 견고하도록 하였다.

분산 컴파일 지원 시스템의 특징은 Make 파일을 구성하는 상위 min 파일 중에서 컴파일에 의해 생성되어야 할 전체 오브젝트와 오브젝트 생성 규칙을 가지고 있는 하위 min 파일들의 정보를 담고 있는 상위 min파일 즉,그림 1의 X.min파일의 정보를 MIN 종속성 분석 결과에 따라 분할하여 컴파일 에이전트 개수만큼 재생성함으로써 분산 컴파일을 위한 전처리를 수행한다는 것이다.

3.3 종속성 분석

종속성 분석은 MIN 종속성분석과 소스종속성분석으로 나뉘어진다. X.min 파일에 기술되어 있는 하위 min 파일들 간의 의존 관계에 따라 하위 min 파일을 그룹핑하는 것을 MIN 종속성 분석이라 하며, 알고리즘(MinDepanal)은 다음과 같다.

- // $S = \{s_i \mid s_i \text{ is a source file for } i \geq 1\}$
- // $M = \{m_j \mid m_j \text{ is a min file for } j \geq 1\}$
- // $F = \{F_{f_k} \mid F_{f_k} \text{ is a folder in which}$
- // $f_k \text{ is located, for } f_k \in S \vee f_k \in M, k \geq 1\}$
- // $MG = \{m_{ijk} \dots G \mid m_{ijk} \dots G \text{ is } \{m_i \cup m_j \cup m_k, \dots\}$
- // for $m_i, m_j, m_k \in M, i, j, k \geq 1\}$

```
//When  $m_j$  includes  $s_i$  for  $m_j \in M, s_i \in S, i, j \geq 1$ 
// We denote  $s_i \alpha m_j$ 
//When  $m_j$  includes  $m_i$  and  $m_k$  for  $m_i, m_j, m_k \in M, i, j, k \geq 1$ 
// We denote  $m_i, m_k \alpha m_j$ 
// ModuleGrouping( $m_i, m_j$ ) forms a group  $m_{ij}G$ 
// for  $m_i, m_j \in \{M \cup MG\}, m_{ij}G \in MG$ 
Module MinDepanal ( $X, min$ )
{
  if ( $s_j \alpha m_i \wedge Fs_j \neq Fm_i \wedge Fs_j = Fm_j$ ) then
     $m_{ij}G = Grouping(m_i, m_j)$ 
  if ( $s_k \alpha m_{ij}G \wedge Fs_j \neq Fm_{ij}G \wedge Fs_k = Fm_k$ ) then
     $m_{ijk}G = Grouping(m_{ij}G, m_k)$ 
}
```

소스중속성분석은 프로젝트 최상위 폴더 아래에 있는 소스 파일들의 전처리 부분에 있는 헤더 및 기타 파일 그리고 매크로 정의에 의한 의존 관계 분석에 의해 이루어지며, 그림 3과 같은 구조로 메모리에 적재되어진다. 중속성 정보를 나타내는 자료 구조가 메모리에 적재되는 양을 최소화시키기 위해 각 배열 요소에 연결되는 노드에 있는 소스와 헤더 정보는 소스와 헤더 배열의 인덱스를 사용하도록 설계하였다. 그리고 검색 연산의 성능을 높이기 위해 배열은 힙소팅(heap sorting)한 다음, 이진검색(binary Search)을 사용하여 정보를 검색하도록 설계하였다.

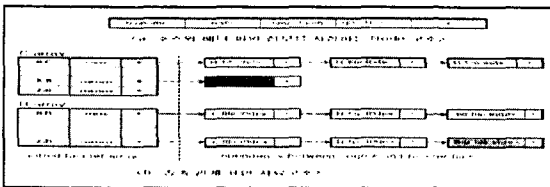


그림 3 소스간 중속성 분석용 자료 구조

4. 성능

분산 컴파일 지원 시스템(DCSS)의 성능 측정 결과가 그림 4와 그림 5에 나타나있다. 성능 측정은 Pentium® CPU HT 2.80GHz, 메모리 512MB인 데스크 탑 컴퓨터 30대가 100Mbps 속도의 네트워크에 접속되어 있는 환경에서 이루어졌다. 성능 측정에 사용된 분산 컴파일용 소스

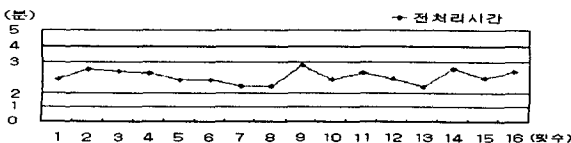


그림 4 분산 컴파일 전처리 성능 측정

의 크기는 651MB 크기의 5,800여개 파일이다.

분산 컴파일 전처리에 드는 시간은 그림 4에 나타난 것처럼 2-3분 정도밖에 소모되지 않는 것으로 나타났다. 로컬 컴퓨터 한 대로만 컴파일을 수행할 경우 59분 정도 걸리는데 반해, 분산 컴파일 지원 시스템을 사용하여 14대 컴퓨터로 분산 컴파일을 수행할 경우 15분 정도 걸려 컴파일 성능이 약 4배 향상되었다. 성능 테스트에 사용된 프로젝트의 경우, 컴파일을 수행하기 위해 항상 컴파일 에이전트로 전송되어야 파일들의 크기가 소스 파일 크기의 3배나 되었지만, 소스의 크기가 더 큰 프로젝트를 분산 컴파일한다면 성능이 더 나올 것으로 기대된다.

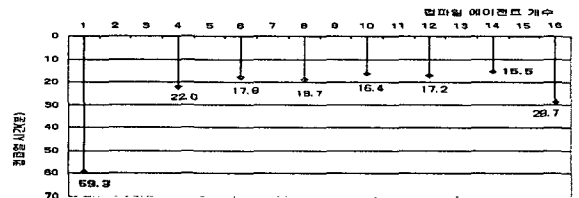


그림 5 컴파일 에이전트 개수에 따른 분산 컴파일 성능

5. 결론 및 향후 과제

본 논문에서는 컴파일시간을 단축시키고자 설계하여 구축된 분산 컴파일 지원 시스템에 관해 논하였으며, 그 성능 테스트 결과를 보여주었다. 대형 프로젝트를 개발할 때 분산 컴파일 지원 시스템을 사용하게 되면 컴파일 시간을 단축하여 프로그램 개발 속도를 향상 시킬 것으로 기대된다. 향후 연구과제로는 컴파일 성능을 향상시키기 위해 네트워크 성능을 고려한 분산 컴파일 에이전트 수 결정 알고리즘에 대한 연구와 과부하 상태에 있는 호스트를 컴파일 에이전트에서 배제하기 위한 CPU 이용률의 임계값 결정 알고리즘에 관한 연구가 남아있다.

참고문헌

- [1] IncrediBuild : The Fast C/C++ Build Solution, <http://www.xoreax.com/main.htm>
- [2] distcc : a fast, free distributed c/c++ compiler, <http://distcc.samba.org/>
- [3] Andy Oram, Steve Talbott, "Managing Projects with make, 2nd edition", O'Reilly, 167pages, Oct.,1991
- [4] GNUmake, http://www.gnu.org/software/make/manual/html_mono/make.html