2003년도 대한전기학회 하계학술대회 논문집 2003.7.21-23

CEMTool환경에서의 분산시뮬레이션의 구현 및 냉간압연 제어시스템에서의 응용

이태일, 이영삼, 이관호, 권욱현
서울대 전기컴퓨터공학부

Distributed Simulation for Cold rolling Control System in CEMTool

Tairi Lee, Young Sam Lee, Kwan Ho Lee and Wook Hyun Kwon
School of Electrical Engineering & Computer Science, Seoul National University

Abstract - 본 논문에서는 CEMTool환경에서 수행된는 일반적 형태 분산 시뮬레이터의 구조를 제안한다. 먼저, 분산시뮬레이션을 할 수 있도록 SIMTool에서 전체 시스템을 패럴랠블록의 형식으로 여러개의 서브시스템으로 분할한다. 그런후에 CEMTool환경에서 분할된 시스템에 대하여 초기화, one step ahead 시뮬레이션, distribute와 ordering의 과정을 진행한 후 각각의 서브시스템에 대하여 독립적인 C code과 실행파일을 생성한다. 여러 대의 PC에서는 분할된 각각의 서브시스템을 독자적으로 실행시키는 동시에 서로간에 reflective memory를 통해 데이터를 주고받는다. 본 논문의 실험대상인 냉간압연 시스템은 각각의 서브시스템 내부의 계산량이 통신량보다 훨씬 많기 때문에 분산처리를 하기에 아주 적합하며, 본 논문에서는 냉간압연 시스템에 대한 분산시뮬레이션의 결과를 분석하고 제시된 방법으로 확실한 속도향상의 결과를 보여준다는 것을 설명하고자 한다.

1. 서 론

분산처리 또는 병렬처리는 대규모의 문제를 다루고 계산속도를 향상시키기 위한 방법으로서 폭넓게 연구되고 있는 분야이다. 본연구에서 다루는 분산시뮬레이션은 되먹임신호를 가지는 동적 시스템을 시뮬레이션 하기 위한 분산처리방법이다. 분산처리를 통해 해결할 수 있는 여러 가지 문제들중에서 동적 시스템에 대한 분산시뮬레이션은 매 순간마다 데이터를 주고받으며 동기를 맞추어야 하는 제약성이 따르는 관계로 상대적으로 연구가 많이 되지 않았다.

국내외 연구사례를 살펴보면, 분산처리 또는 병렬처리에 대한 개념을 여러 가지로 광범위하게 다루고있다. 예를 들면, 현재 많이 나와있는 분산처리 또는 병렬처리에 관한 서적과 논문[1,2]을 살펴보면 개념에 대한 해석 혹은 알고리즘에 대한 연구등이 대부분이다. 그리고 실제적인 시스템을 구축하여 분산시뮬레이션을 실행한 예제[3]는 있지만 분산시뮬레이션을 수행하기 위하여 시스템 분할 및 동기등에 대한 문제를 다루고 속도 향상을 목표로 하고 있지는 않다.

본 논문에서는, 국내에 많이 보급되어있는 CEMTool 환경에서 냉간압연 시스템을 구축한 후 그 시스템에 대하여 합리적으로 여러개의 서브시스템으로 분할한 후 여러대의 PC에서 reflective memory를 통하여 통신을 진행하면서 정확한 시뮬레이션 결과를 얻었으며 또한 최종목표인 속도향상을 실현한 과정과 기능을 설명하고 개발된 분산처리의 구조에 대해 논의를 할것이다.

본 논문은 다음과 같이 구성된다. 2장1절에서는 연속냉간 압연공정 제어시스템에 대하여 간단한 소개를 한 후 SIMTool에서의 시스템 분할 및 구조에 대하여 논의를 하고, 2절에서는 CEMTool에서의 여러개 서브시스템에 대한 멀티C코드 생성에 대하여 설명하며, 3절에서는 reflective memory를 사용한 실험결과 및 성능분석을 진행한다. 마지막으로 3장에서는 결론을 내린다.

2. CEMTool에서의 분산시뮬레이션의 구현

2.1 연속냉간 압연공정 제어시스템에 대한 소개 및 SIMTool에서의 서브시스템 분할

연속냉간 압연공정은 열간압연을 거쳐 생산된 코일을 소재로 상온에서 소비자가 필요로 하는 두께로 압연하는 최종 마무리 공정이다. 냉간압연은 압연기가 연이어 배치되어 압연모재의 두께를 점차적으로 줄여나간다. 압연공정에서의 제어목적은 압연공정의 안정성을 해치지 않고 압연기의 출측 판두께를 요구되는 사양내에 있도록 유지하는 것이다. 본 연구에서는 포항종합제철의 5개의 스탠드를 가지는 연속냉간 압연공정을 대상으로 하고있다. SIMTool에서 표현된 냉간압연 시뮬레이터는 크게 냉간압연 플랜트, 선형제어기 그리고 셋업 등 3개 서브시스템으로 구성되여 있다. <그림1>은 냉간압연 시스템의 구조이며 2000여개 블록으로 구성되였다.

먼저 블록의 종류를 보면, 일반블록과 슈퍼블록으로 나뉜다. 슈퍼블록은 특정된 기능을 가지고있지 않고 여러개의 블록을 한개의 그룹으로 묶어주는 작용을 한다. 사용하게 되는 패럴랠블록은 슈퍼블록의 특성을 가지고있으며 이런 슈퍼블록의 공능의 기초하에 나뉘여진 서브시스템을 분산시뮬레이션을 할수 있도록 되여있다. 즉, 전체 모델을 n개의 서브시스템으로 분산시뮬레이션을 하려면 모델을 n개의 패럴랠블록으로 나눈다. 예를 들면, 본 논문의 실험대상인 냉간압연모델은 플랜트, 컨트롤러와 셋업 등 세 개의 서브시스템으로 나눌수 있는데 이런 작업을 SIMTool에서는 플랜트, 컨트롤러, 셋업 등 세 개의 패럴랠블록으로 나누는것이다. <그림2>가 바로 <그림1>의 냉간압연시스템을 분산시뮬레이션을 위한 패럴랠블록형태로 나눈것이다.

<그림2>를 보면 세 개의 패럴랠블록사이에 연결이 없는것처럼 보인다. 하지만 실제로는 패럴랠블록사이에는 가상연결을 통하여 연결되여있다. SIMTool에서의 연결은 실제연결 외에도 goto와 from이란 블록으로 가상연결을 실현하는데 tag를 사용하여 자신과 연결된 가상블록을 찾는다. 이런 가상연결과 슈퍼블록은 SIMTool에서의 시스템을 설계할 때 간결하고 알아보기쉽게 표현할수 있다.

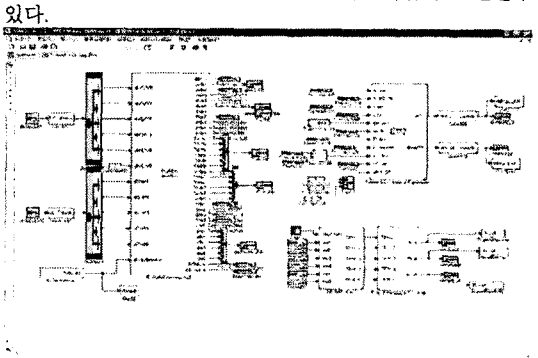


그림1 냉간압연 시스템

SIMTool에서 시스템을 설계하고 구성하는것은 matlab의 simulink와 비슷하며 여러개의 패럴랠블록으로 분할하여 CEMTool에서 분산시뮬레이션을 진행하는 것은 matlab에서는 없는 기능이다.

SIMTool에서 설계된 시스템은 CEMTool에서 계산하거나 시뮬레이션을 진행한다. SIMTool은 filename.out이라는 임시파일을 생성하고 시스템의 정보를 out파일에 써준다. 시스템의 정보를 규범되게 표현하기 위하여 먼저 시스템에 대하여 getflatcanvas과정을 거친다. 이 과정은 모든 블록을 펼친다는 의미인데 슈퍼블록을 해체하고 슈퍼블록내부와 외부를 연결해주고 가상연결이 있을 경우 만일 같은 패럴랠블록에 속해있다면 그대로 연결을 풀어주고 다른 패럴랠블록에 속해있다면 통신포트블록으로 생성된다. 통신포트블록은 분산시뮬레이션을 진행하기 위하여 특별히 생성되는 가상블록인데 패럴랠블록지간의 연결은 통신포트블록을 통하여 실현된다. 물론 통신포트블록은 실제연결과 가상연결 두가지를 모두 포함한다. out파일은 전반적인 시스템정보와 개개의 실제블록과 통신포트블록을 포함한 모든 가상블록에 대한 정보를 포함하고있다.

SIMTool에서 out파일을 생성한 후, dde통신을 사용하여 CEMTool에서는 시스템의 정보를 받아들이고 상응한 out파일을 읽어들여서 필요한 작업을 진행한다.

2.2 CEMTool에서의 멀티C코드 생성

과거에 CEMTool의 한 모듈로 대규모시스템의 시뮬레이션에 적합한 고속 시뮬레이터를 개발한 적 있다. 그리고 고속의 시뮬레이션 성능을 얻기 위해 자동C코드 생성기를 개발하였다. 이런 C코드 생성을 통해 얻어진 C코드를 컴파일하여 얻은 실행파일은 계산속도가 크게 향상되여서 고속 시뮬레이션의 목적을 충족시키는데 큰 역할을 하였다. 하지만 이미 개발한 시뮬레이터는 단일 PC에서만 시뮬레이션을 할수 있게끔 되여있고 C코드생성도 단일C코드생성에만 국한되여있다.

본 논문에서 논의하는 분산시뮬레이션은 이미 얻은 C코드 생성을 통해 얻은 속도 향상의 효과를 분산 시뮬레이션의 구조와 결합하여 추가로 속도 상승효과를 얻도록 하는것이다. 이렇게 하기 위하여 우리는 패럴랠블록의 수만큼 멀티C코드를 생성하도록 하였다.

멀티C코드를 생성하는 과정은 크게 초기화, one step ahead 시뮬레이션, 멀티C코드 생성 등 단계로 나뉘여져 있다. one step ahead 시뮬레이션은 먼저 한 스탭만큼 미리 시뮬레이션을 진행하는 과정인데 이것은 분산처리에서 꼭 필요한 단계이다. 전체 시스템을 여러개의 서브시스템으로 분할한 후 서브시스템사이를 연결해주는 가상의 ComInport와 ComOutport통신블록이 생기게 되는데 이들은 각각의 서브시스템에 대해서는 입력과 출력으로 볼 수 있다.

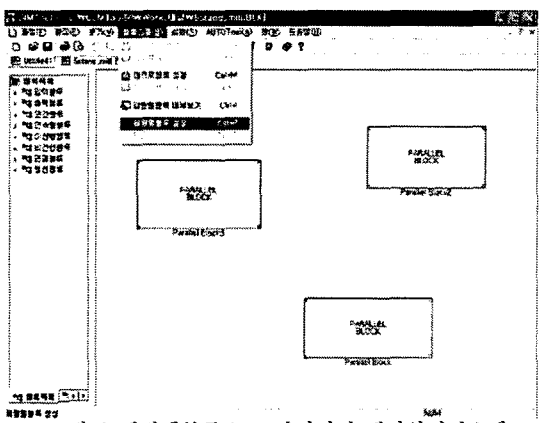


그림 2 패럴랠블록으로 나뉘여진 냉간압연시스템

하지만 이러한 통신포트블록은 가상의 블록으로서 먼저 초기화가 되여있지 않은 상태이므로 먼저 한개 스탭을 미리 시뮬레이션하여 이러한 가상블록의 값들을 얻은 후에 비로소 통신포트블록에 대해 초기화를 진행할수 있다. 그리고 매번 스탭이 끝난후에도 통신포트블록의 값들을 받아들이는 것이 필요하다. 이러한 과정은 또한 정확한 시뮬레이션 결과를 얻는데 필요하기도 하다.

이러한 초기화 과정을 끝낸후에는 멀티C코드생성을 하기 위한 distribute block과 패럴랠블록의 ordering이 필요하다. 즉, 매개 블록을 어느 패럴랠블록에 속하는가에 근거하여 패럴랠블록의 배열에 저장하는 과정과 매개 배열에서 우선순위에 근거하여 블록들의 계산 순서를 정하는 과정이다. 블록분리과정은 out파일에서 매개블록의 정보에 근거하여 블록을 분리하는 동시에 그 블록과 연결되여있는 입출력 관계정보를 생성한다. <그림3>은 블록을 나누는 과정을 설명한것이다.

패럴랠블록의 ordering은 서브시스템을 구성하고 있는 패럴랠블록의 계산 순서를 정하는 알고리즘이다. 패럴랠블록의 계산순서가 잘못 정해질 경우 계산결과가 틀리거나 통신이 제대로 되지 않을 수가 있으므로 ordering과정은 매우 중요한 부분이다. 통신입력포트는 일반입력포트와 마찬가지로 계산 우선순위가 최상위에 속한다. 통신의 간결함과 편리를 위하여 통신포트에 대한 처리과정이 필요하다. <그림4>는 서브시스템 pblk3에서 pblk2으로 데이터를 전송하는 과정을 설명한것이다. 처리하는 데이터는 백터도 지원하기 때문에 백터길이에 대한 계산을 해주어야 한다. 그리고 pblk3에서 pblk2로 가는 모든 통신출력포트의 값들을 한개 배열에 저장하여 한번에 메모리에 써준다. pblk2에서도 이러한 데이터를 한번에 읽어들이고 앞에서 수행한 블록분리과정에서 얻은 입출력 관계에 근거하여 그 값들을 상응한 블록들로 분배한다.

이러한 과정이 완료된 후 멀티C코드를 생성하는데 system파일의 1번째 패럴랠블록에 대응되는 코드들은 system1.h, system1.prm, system1.c, rmfunctions1.c등 4가지이다. 그중 *.h는 헤더파일이고 *.prm은 파라미터파일이며 .c는 메인파일이고 rmfunctions1.c는 reflective memory를 사용한 통신파일이다. 이러한 파일들을 batch파일을 이용하여 간단하게 system0.exe, system1.exe, system2.exe 등 서브시스템의 수만큼 exe파일을 생성할

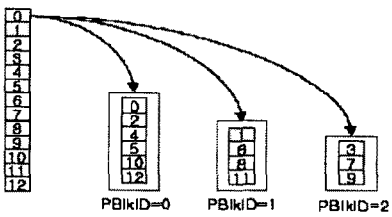


그림3 distribute block과정

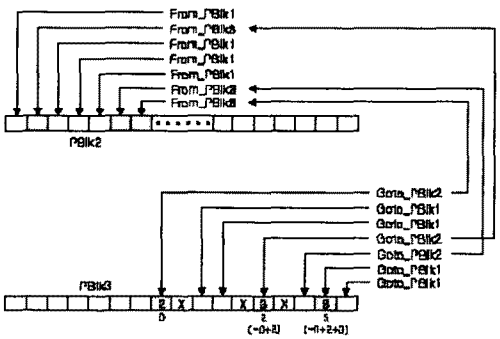


그림4 패럴랠블록 통신포트의 연결과정

수 있다. 만일 시스템을 3개의 서브시스템으로 분할하면 세 개의 exe파일이 생성되여서 세대의 pc에서 서로 통신하면서 실행할 수 있다.

2.3 실험결과 및 성능분석

이 논문의 실험의 대상은 2000여개의 블록을 가진 냉간압연 제어시스템이다. 우리는 이 시스템을 플랜트, 컨트롤러와 셋업 세 개의 서브시스템의 형식으로 세 개의 패럴랠블록으로 분할하고 세 개의 서브시스템 실행파일을 생성했다. 세대의 PC에서는 reflective memory와 광케이블을 이용하여 서로의 통신을 진행한다. reflective memory에서는 어느 한 노드의 메모리에 쓰기를 실행하면 나머지 다른 노드에서도 같은 주소에 동일한 값이 쓰게 됨으로써 데이터를 공유할 수 있게 된다. 본 실험에서 사용된 reflective memory는 VMIPCI-5579이다.

<그림5>는 이번 분산시뮬레이션 실험의 순서도이다. 처음에 시작하면 먼저 reflective memory와 시스템의 초기화 state의 초기화 등을 실행한다. 그리고 시작시간은 0초이고 결속시간은 40초이며 한개 스텝은 0.001초로 설정되여있다. 매개 스탭마다 evaloutput()과 시스템에 대한 업데이트를 진행하고 세대의 pc사이에 동기를 맞추면서 reflective memory에서 쓰기와 읽기를 진행한다.

이번 실험에서 사용한 세 대의 PC의 CPU속도는 각각 750MHz, 500MHz, 400MHz이다. 한대의 PC에서 단일 C코드를 생성하여 실행하면 각각 30초, 52초, 63초의 결과를 얻었다. 세 개의 서브시스템으로 나뉘여서 실행한 결과는 <표1>에서 볼 수 있다싶이 서브시스템의 분할결과에 따라서 약간씩 차이가 나는데 전반적으로 시간이 일정하게 줄었으며 제일 빠를 때 20초의 결과를 얻은 것을 볼 수 있다. 이러한 결과가 나온 것은 플랜트에서 약 60-70%의 계산로드를 갖기 때문에 대략 1.5배의 속도상승효율을 나타냈다. 만일 시스템분할이 더 균등하고 사용한 pc의 속도가 균등하다면 더욱 좋은 속도상승효과를 나타낼것이다.

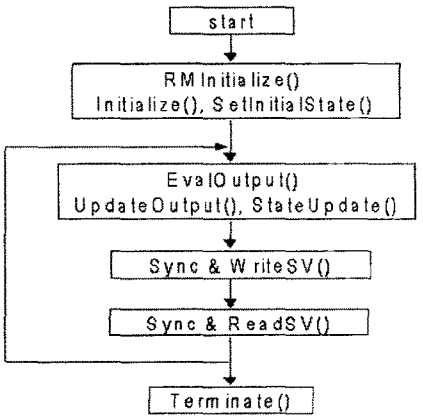


그림 5 분산시뮬레이션 순서도

Time	Cpu1(750Mhz)	Cpu2(500Mhz)	Cpu3(400Mhz)
20s	Cold_rolling plant	Controller	Setup
22s	Cold_rolling plant	Setup	Controller
38s	Controller	Cold_rolling plant	Setup
40s	Controller	Setup	Cold_rolling plant
45s	Setup	Cold_rolling plant	Controller
47s	Setup	Controller	Cold_rolling plant

표 1 냉간압연 분산시뮬레이션 결과

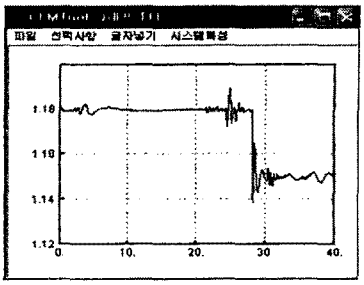
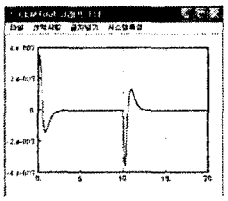


그림 6 냉간압연시스템의 판두께의 변화그래프



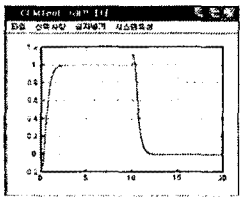


그림7 inverted pendulum시스템의 제어그래프

만약 속도상승만 있고 정확한 결과를 얻을수 없다면 분산시뮬레이션의 의미가 없어지는것이다. 앞에서 one 스탭시뮬레이션을 먼저 하고 멀티C코드를 생성하는것도 바로 정확한 초기값을 얻고 최종적으로 정확한 결과를 내오기 위한것이다. 냉간압연 제어시스템을 분산시뮬레이션<그림6>을 실험해본 결과 정확도가 99.99%이상이였다.

또한 분산시뮬레이션의 범용성을 확보하기 위하여 아주 민감한 inverted pendulum시스템<그림7>에 대하여 분산시뮬레이션을 한 결과도 아주 안정적이고 한개의 pc에서의 시뮬레이션과 거의 완전히 같은 결과를 보여주었다.

3. 결 론

본 논문에서는 CEMTool환경에서 냉간압연 제어시스템을 분산시뮬레이션을 구현하는 과정과 결론을 자세히 서술하였다. SIMTool에서 한개의 동적시스템을 여러개의 서브시스템으로 분할한 후 reflective memory를 사용하여 통신하면서 분산시뮬레이션을 진행하였으며 또한 이런 분산시뮬레이션이 임의의 시스템에 적용할 수 있도록 범용성 있는 코드를 구현하도록 하였다.

실험결과 대규모 시스템인 냉간압연모델에 대해서도 정확한 결과를 가져왔으며 좋은 속도상승효과도 나타냈다. 그리고 아주 민감한 시스템에 적용한 경우에도 정확한 결과를 가져왔으며 이것은 범용적으로 사용되여도 안정한 결과를 얻을 수 있다는 것을 보여주었다.

본 연구의 이러한 분산시뮬레이션은 아직까지 실현해본적 없는 분산처리방식이고 확실한 속도상승효과를 가져왔다는데서 큰 의의를 갖는다.

[참 고 문 헌]

[1] Barry Wilkinson and Michael Allen, "Parallel Progamming", An Alan R.Apt Book, 1999

[2] C.D.Pham, "Comparison of Message Aggregation Strategies for Parallel Simulations on a High Performance Cluster", Modeling, Analysis and Simulation of Computer and Telecommunation Systems, 2000

[3] L.Pollini and M.Innocenti, "A synthetic environment for dynamic systems control and distributed simulation", IEEE Control Systems Magazine, vol.20, pp.49-61, April 2000