2001년도 대한전기학회 전문대학교육위원회 학술대회 논문집 2001.7.10-11

전력 조류 계산의 분산 병렬처리기법에 관한 연구

°이춘모, 이해기
충청대학 전기자동화과

A Development of Distributed Parallel Processing algorithm for Power Flow analysis

°Chun-Mo Lee, Hae-Ki Lee
Chung Cheong College Dep. of Electric automatic Eng.

Abstract - Parallel processing has the potential to be cost effectively used on computationally intense power system problems. But this technology is not still available is not only parallel computer but also parallel processing scheme. Testing these algorithms to ensure accuracy, and evaluation of their performance is also an issue. Although a significant amount of parallel algorithms of power system problem have been developed in last decade, actual testing on processor architectures lies in the beginning stages.

This paper presents the parallel processing algorithm to supply the base being able to treat power flow by newton's method by the distributed memory type parallel computer. This method is to assign and to compute teared blocks of sparse matrix at each parallel processors. The testing to insure accuracy of developed method have been done on serial computer by trying to simulate a parallel environment.

Key words : Power flow, Parallel Processing, Block method, MPRLD ordering scheme, Distributed memory type parallel computer, Jacobian matrix

1. 서 론

병렬처리(parallel processing)는 데이터 교환장치를 가지고 있는 두개 이상의 프로세서를 이용 어떤 문제의 해를 분담, 처리하는 것을 의미한다.[1] 최근에는 웹 기반으로 개발할 수 있는 자바 프로그램이 대중화되면서 인터넷을 통한 병렬처리의 실현이 가능하게 되고 있어 비록 효율에서는 떨어지지만 값비싼 슈퍼 컴퓨터를 이용하지 않는 기법이 연구되고 있다.[2]-[4] 병렬 컴퓨터의 구성형태는 명령(instruction)과 데이타의 순차적인 수행 형태에 따라 SIMD(single instruction stream-multiple data stream) 와 MIMD(multiple instruction stream-multiple data stream) 두가지로 분류할수 있다.[5] SIMD형 컴퓨터는 다수의 프로세서들이 배열형태로 연결되어 하나의 제어장치에 의하여 동기적으로 병렬처리를 수행하는 컴퓨터를 말하는데 이 기종은 주어진 어느 한순간에 각 프로세서들이 동일한 명령을 수행하기 때문에 계통방정식의 계수행렬이 스파ㅅ하면서 복잡한 연산과정을 수반하는 전력계통과 같은 문제에 대하여는 대부분의 프로세서들이 연산을 수행하지 않고 대기(idleing)하고 있어 효율적인 적용은 어렵게 된다.[6]

MIMD형 컴퓨터는 스스로 연산 능력을 갖고 있는 여러개의 프로세서와 데이타 기억장치, 그리고 프로세서들간의 통신을 처리하기 위한 상호 연결망(interconnection network)등으로 구성되어 있는데, 이 컴퓨터는 다시 크게 대용량의 공유 메모리를 이용한 공유 메모리 시스템(shared memory system)과 각각의 프로세서들이 자체의 국부 기억장치(local memory)를 갖고 있는 분산 메모리 시스템(distributed memory system)으로 나눌 수 있다.[7]

전력 계통 해석을 위하여는 MIMD형 컴퓨터를 이용하여 해석하게 되는데 전력계통의 문제를 해결하기 위한 병렬처리기법은 크게 요소기법(elemental scheme)과 블록기법(block scheme)으로 구분할 수 있다. [8]

요소기법은 각 프로그램에 대하여 작은 단위의 프로그램 코드들로 구성된 타스크 그래프(task graph)를 구하여 연산과정의 병렬성을 최대로 이용하는 방법이다. 이러한 요소기법은 각 연산단계마다 많은 데이타 정보들의 교환이 이루어져야 하므로 프로세서간에 데이타 교환을 위한 통신부담(communication overhead)이 매우 커지게 되는 단점이 있다. 블록기법은 선형방정식의 행렬을 작은 블록(block)으로 분할하여 이 블록들을 각 프로세서에 할당하여 처리하는 방법으로서 병렬성은 타스크가 아닌 블록단위로 존재하여 각 프로세서가 연산을 수행할 때 다른 프로세서와의 데이터 이동에 의한 통신부담을 최소가 되도록 하는 것이다. [9-11] 따라서 컴퓨터 구조상 요소기법은 데이터 교환이 용이한 공유 메모리 시스템에 적합하고, 블록기법은 데이터 교환을 최소화하므로 분산 메모리 시스템에 적합하다. 그러나 병렬컴퓨터의 가격은 분산 메모리 시스템에 비하여 공유메모리 시스템이 구조상 매우 고가이기 때문에 비록 요소기법의 알고리즘이 정립되어 있기는 하지만 실용화되고 있지 못한 실정에 있다.

따라서 본 연구에서는 분산메모리 시스템을 이용하기 위한 블록 기법에 대한 연구를 수행하였다. 전력계통을 여러개의 블록으로 분할하여 병렬컴퓨터로 처리하기 위하여는 계통의 모선을 최적서열 및 계통의 분할 알고리즘 뿐만 아니라 각각의 프로세서에서 전력계통 해석의 연산처리를 병렬로 수행할 수 있도록 하는 병렬처리기법이 필요하다. 병렬처리를 위한 삼각화 기법 및 전진 및 역진대입에 대한 병렬처리기법은 이미 연구를 수행한바 있다.[12][13] 그러나 실제 전력계통 조류계산을 병렬처리하기 위하여는 초기의 서열단계에서부터 마지막 반복해까지의 전 과정을 병렬처리 할 수 있는 기법이 필요하게 된다. 따라서 전력계통의 조류계산시 꼭 수반되는 Jacobian 행렬 연산의 병렬처리를 위한 모델링은 필수적으로 이루어져야 하나 현재까지 이 분야에 대한 연구가 미진한 상태로서 남아있는 실정이다. 따라서 본 논문에서는 실제 전력계통 해석 문제에서 자주 접하게 되는 전력 조류계산을 병렬처리 할 수 있도록 Jacobian 행렬의 연산과정을 블록기법에 의한 병렬 처리가 가능하도록 모델링 기법을 제시하여 전력조류계산의 전 단계를 이미 개발한 서열 및 반복해를 적용. 병렬

처리가 가능하도록 하였으며, 개발된 기법이 정확히 시행될 수 있는지를 평가하기 위하여 모델전력계통에 대한 각 프로세서의 병렬처리기능을 프로그램으로 구성, 시뮬레이션을 실시하였다.

2. 병렬 컴퓨터의 구조

2.1 공유 메모리형 병렬 컴퓨터

그림 2.1은 공유메모리 시스템의 구조를 나타낸 것으로 CRAY-2를 비롯한 ALLIANT, ULTRACOMPUTER, RP3 등의 시스템이 개발되어 있는데,[13] 이들은 각 프로세서가 별도로 자체의 프로그램을 수행하는 것이 아니라 전체 프로그램중 일부분의 프로그램 코드(code)들을 할당 받아 수행하는 것이다.

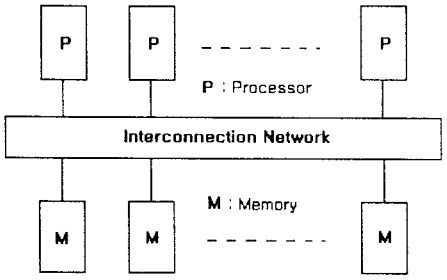


그림 2.1 공유 메모리형 MIMD 병렬 컴퓨터의 구조
FIg 2.1 The structure of shared memory type MIMD
parallel processing computer

이 시스템은 각각의 프로세서가 하나의 커다란 기억장치를 공유하므로 프로그래밍이 편리한 잇점은 있으나 둘 이상의 프로세서가 동일한 메모리를 참조(access)하려 할때 발생하는 회선쟁탈(bus contention)로 인한 시간지연과 여러개의 프로세서가 공유메모리를 참조(access)하기 위한 메카니즘의 문제로 프로세서와 메모리 사이의 상호 연결망 구성을 어렵게 할 뿐만 아니라 포함되는 프로세서의 수가 제한 받게 되며, 대용량의 문제로 시스템 구성의 비용이 매우 커지게 된다.[6]

2.2 분산 메모리형 병렬 컴퓨터

그림 2.2는 분산 메모리 방식의 구조를 나타낸 것인데, 이 시스템은 자체의 메모리를 이용하여 다른 언어나 다른 타입으로 작성된 프로그램을 처리할 수 있는 많은 프로세서를 포함할 수 있으며, 프로세서의 수를 간단하고 쉽게 확장시킬 수 있다. 다른 프로세서의 메모리에 대한 참조시간(access time)은 공유 메모리 방식에 비해 늦어지고 프로그래밍이 복잡한 문제는 있으나 전력계통을 분할하여 각 프로세서에 적절한 연산처리 분담이 가능할 경우에는 각 프로세서 사이의 정보교환의 수를 비교적 적게 유지할 수 있으며,특히 컴퓨터 가격이 저렴한 장점이 있다.[6][7]

분산 메모리형 컴퓨터에서 표준화된 소프트웨어의 개발은 효율적인 컴퓨터의 상호 연결망 구성과 함께 해결되어야 할 매우 중요한 과제로 대두되고 있는데, 이 문제의 해결을 위한 필수적 과제는 첫째,명확한 타스크의 분할 및

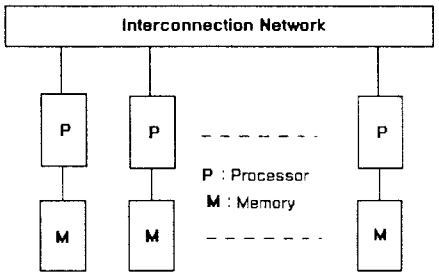


그림 2.2 분산 메모리형 MIMD 병렬 컴퓨터의 구조
FIg 2.2 The structure of distributed memory type MIMD parallel processing computer

할당을 위하여 해석하고자 하는 문제들을 규칙적인 방법에 의하여 분할이 가능케 하는 방법을 제공하는 것과 둘째 각 프로세서의 연산과 데이타 교환을 일반화 할 수 있는 병렬처리 알고리즘의 개발이다.[14]

3. 병렬처리 알고리즘

3.1 요소기법

요소기법은 프로세서의 수가 제약받지 않는다고 가정하면 프로그램의 작은 단위인 코드(code)들에 대한 연산을 각 프로세서에 할당하여 연산시키는 방법이다.

이러한 요소기법은 일명 슈퍼 컴퓨터(super computer)로 표현되는 CRAY-2와 같이 하나의 커다란 기억장치를 각 프로세서가 공유하는 공유 메모리형 MIMD 컴퓨터를 이용하여 각 프로세서에서 수행된 데이타를 하나의 공유 기억장치에 저장하여 각 프로세서들이 연산단계에서 필요한 데이타가 저장된 기억장치에 접근(access)하도록 하고 있다.

그림 3.1은 LU 삼각화 직접해법의 전진대입 과정을 데이타 흐름(data flow) 즉, 연산순서로 배치한 타스크 그래프(task graph)를 이용하여 요소기법의 병렬처리 과정을 나타내 보인것이다. 즉 전진대입 과정중 요소기법은 비대각 요소들의 연산을 수행하기 위하여는 그 열번호의 대각요소가 계산될때 까지 각 프로세서들은 대기(idle)하고 있어야 하며 각 프로세서에서 수행된 비대각요소들의 값은 다음 대각요소의 계산을 위하여 대각요소를 연산하는 프로세서에 전송되어져야 한다. 이와 같이 요소기법은 각 연산단계마다 포함되는 대기시간(idle time)과 연산단계마다 각 프로세서들이 다른 프로세서에 의해서 구해진 연산 결과를 전송받아야 하기 때문에 많은 양의 데이타 교환으로 인한 통신부담(communication overhead)이 커다란 문제로 지적되고 있다.[15]

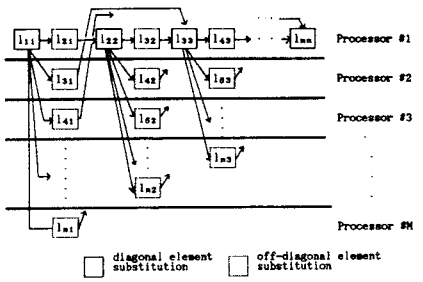


그림 3.1 전진대입 과정에 대한 요소기법 타스크그래프
Fig 3.1 Task graph depicting the forward substitution for element scheme

3.2 블록기법

블록기법(block scheme)은 행렬을 작은 블록으로 분할하여 각 블록들을 병렬처리시스템의 각 프로세서에 할당시켜 데이타 정보 교환의 수를 최소화하는 방법이다. 일반적인 전진대입 연산과정을 나타내면 그림 3.2와 같다. 이 그림에서 보는 바와 같이 각 타스크에 대한 연산순서는 요소기법과 같으나 연산과정을 블록으로 분할하여 각 프로세서에서 처리토록 하므로써 데이타 교환에 의한 통신부담을 감소시킬 수 있게 한 것이다. 이러한 블록기법은 전력계통과 같이 계통을 작은 cluster들로 분할이 가능한 경우 각 분할된 cluster들의 독립성이 커져 병렬처리 과정중 발생하는 데이타 교환에 의한 통신부담을 최소화 할수 있게 된다. 그러나 계통을 완전히 분리하는 것은 사실상 불가능하여 계통을 분할할때에는 각 분할된 블록들을 연결하는 상호 연결부분 모선(interconnection bus)이 존재하게 된다. 그러므로 분할된 블록과 공통부분모선의 요소들에 대한 연산을 병렬처리하기 위하여는 각 프로세서에 분할된 블록을 할당하는 것 뿐만 아니라 상호 연결 부분 모선들의 연산처리를 위한 병렬처리기법이 요구된다.

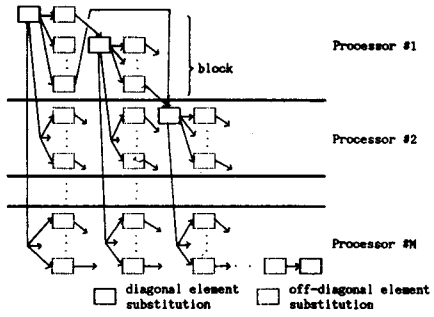


그림 3.2 블록기법의 타스크 배분
Fig 3.2 Task distribution among processing elements for block schemes

4. 전력조류계산의 병렬처리

4.1 뉴톤법에 의한 병렬 처리

모선 p에서의 전력 W_p는

$$W_p = P_p + jQ_p = V_p I_p^* \qquad (4\text{-}1)$$

로 나타내어진다. 한편

$$I_p = \sum_{q=1}^{n} Y_{pq} V_q \qquad (4\text{-}2)$$

이다. 따라서 식 (1-1)은

$$W_p = P_p + jQ_p = V_p \sum_{q=1}^{n} Y_{pq}^* V_q^* \qquad (4\text{-}3)$$

로 된다.

이 경우의 뉴톤법 (Newton's method)에 의한 전력조류계산은 식 (4-4)같이 표현되는 Jacobian 행렬을 포함한 계통 선형방정식의 일반식을 이용하여 해석하게 된다.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ M & L \end{bmatrix} \begin{bmatrix} \Delta\theta \\ |\Delta V|/|V| \end{bmatrix} \qquad (4\text{-}4)$$

식 (4-4)의 Jacobian행렬의 계수는 어드미턴스 행렬(Y_{BUS})과 전압을 유효분과 무효분으로 나누어 다음과 같이 구해진다.

i) 비대각 요소(J_{pq})

$$H_{pq} = L_{pq} = |V_p||V_q| (G_{pq}SIN\theta_{pq} - B_{pq}COS\theta_{pq})$$

$$N_{pq} = -M_{pq} = |V_p||V_q| (G_{pq}COS\theta_{pq} + B_{pq}SIN\theta_{pq}) \qquad p \neq q \qquad (4\text{-}5)$$

ii) 대각 요소(J_{pp})

$$H_{pp} = -Q_p - B_{pp}|V_p|^2$$
$$L_{pp} = Q_p - B_{pp}|V_p|^2$$

$$N_{pp} = P_p + G_{pp}|V_p|^2$$
$$M_{pp} = P_p - G_{pp}|V_p|^2 \qquad p = q \qquad (4\text{-}6)$$

여기서 $V_p = |V_p| \angle\theta_p$; $\theta_{pq} = \theta_p - \theta_q$

$Y_{pq} = G_{pq} + j\,B_{pq}$

이러한 뉴톤법에 의한 전력조류계산을 병렬처리하기 위하여는 분할기법 뿐만아니라 이 연산과정에 있는 Jacobian 행렬 연산처리를 포함한 전진 및 역진 대입의 전 과정에 대한 병렬처리를 위한 알고리즘의 개발은 필수적인 것이다.

4.2 병렬처리를 위한 Jacobian 행렬 모델링

본 연구에서는 동시에 여러 프로세서들간의 데이터 교환이 가능한 하이퍼 연결망 구조를 갖는 분산 메모리형 병렬컴퓨터를 이용하여 전력조류계산을 병렬처리할 수 있도록 어드미턴스 행렬(Y_{BUS})의 분할된 블록들의 타스크 할당과 Jacobian 행렬의 모델링을 수행한다.

그림 3-1은 N 모선 계통을 m-1개의 크러스터로 분할한 후의 어드미턴스 행렬을 m개의 블록 형태로 나타낸 것이다. 여기서 m번째의 블록행렬은 계통을 분할하는 모선들에 해당하는 부분이다. Jacobian 행렬은 Y_{BUS}행렬과 같은 구조로 구해지게 되는데, 그림 3-1과 같이 블록으로 분할된 경우 Jacobian 행렬의 비대각 블록(non-diagonal block)은 비대각 요소 "J_{pq}"로 부터 구해진다. 식 (4-5)에서 J_{pq}는 동일한 모선번호 p,q의 전압 V_p,V_q와 어드미턴스 요소 Y_{pq}에 의하여 구해지므로 Jacobian 행렬의 비대각 요소의 연산은 해당 블록 모선의 전압과 어드미턴스만 필요하다.

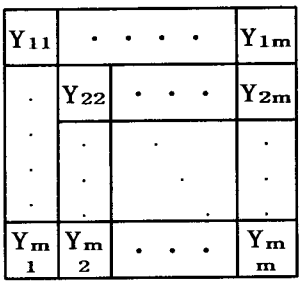
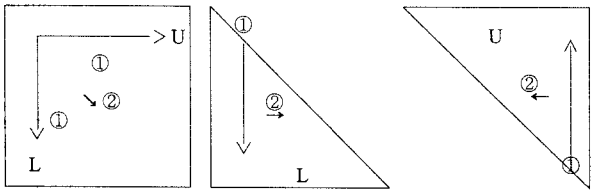


그림 4.1 m개의 대각 블록으로 구성된 Y_{BUS}
Fig 4.1 Y_{BUS} composed with m diagonal block

그러나 대각블록(diagonal block)의 요소 "J_{pp}"는 식(4.6)에서 처럼 모선 p의 행에 포함되어 있는 어드미턴스 행렬의 영이 아닌 요소와 이 요소들의 열 번호에 해당하는 전압을 필요로 하게 된다. 따라서 Jacobian 행렬 연산을 병렬처리하기 위하여는 그림 4.1과 같이 행으로 구분된 Y_{BUS}행렬의 대각블록, 비대각블록 및 해당 전압을 같은 프로세서의

기억장치에 할당하여야 한다. 이때 마지막 부분에 위치한 상호 연결 부분 모선 (interconnection bus)의 전압 벡터는 각 프로세서의 기억장치에 공통으로 기억시켜 연산을 수행하게 된다.

그림 4.2는 LU분할 및 전진/역진대입의 연산순서를 나타낸 것이다. 이 그림에서 보는 바와 같이 LU 분할의 연산 및 전진/역진 대입은 먼저 ①과 같이 행렬의 각 행과 열에 대하여 연산을 수행한 후 ②의 방향으로 순차적으로 수행된다.



(a)LU 분할 (b)전진대입 (c)역진대입

그림 4.2 LU분할 및 전진/역진대입의 연산순서
Fig 4.2 The calculation sequence of LU factorization and forward/backward substitution

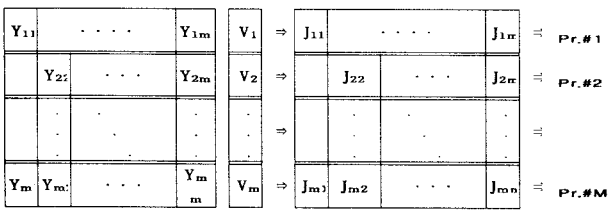


그림 4.3 Y_{BUS} 행렬 및 Jacobian행렬의 타스크 배분
Fig 4.3 Task distribution for Y_{BUS} and Jacobian matrix

따라서 Jacobian행렬이 그림 4.3과 같이 할당될 경우 LU분할에 의한 직접해법 적용시 각 연산단계마다 프로세서들은 "processor #M"의 국부 기억장치(local memory)에 저장된 각각의 비대각 블록행렬 $[J_{m1}]$,$[J_{m2}]$, ···· ,$[J_{m,m-1}]$의 데이터들을 전송받아 연산을 수행해야만 한다.

그림 4.4는 LU 분할 및 전진/역진대입의 연산순서를 감안한 데이터 할당 방법을 나타낸 것으로 Y_{BUS} 행렬의 대각 블록행렬과 해당 비대각 블록행렬을 동일한 프로세서의 기억장치에 할당시키도록 하였다. 이와 같은 데이터 할당은 Jacobian 행렬의 연산시에는 데이터 교환량이 증가하나 조류계산의 연산량의 대부분을 점유하는 LU 삼각화 직접해법의 병렬처리시에는 각 프로세서가 연산 수행중 필요한 대부분의 데이터를 자신의 국부 기억장치에 저장하고 있어 데이타 교환을 최소화 할 수 있고 동일한 데이터의 공유를 억제하므로 전체.

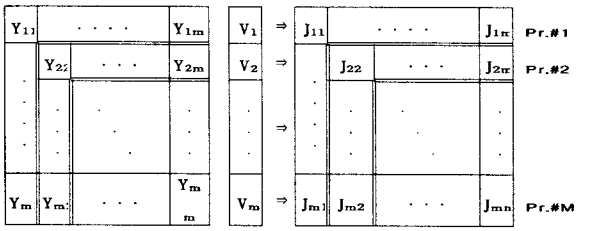


그림4.4 제안된 Jacobian 행렬 연산의 타스크 배분
Fig4.4 Proposed task distribution for Jacobian matrix calculation

의 기억용량을 감소시킬 수 있게 된다. 이때 각 프로세서는 각각의 저장장치에 기억된 비대각 블록행렬 $[Y_{m1}][Y_{m2}]\cdots[Y_{m-1,m}]$의 데이터와 식 (4-6)에 의하여 공통부분 모선에 대한 H_{pp}, N_{pp}, M_{pp}, L_{pp}의 요소들의 부분 해가 구해진다.

여기서 공통부분의 Jacobian 행렬인 $[J_{mm}]$은 각 프로세서에서 구해진 공통부분의 $[J^1{}_{mm}]$,$[J^2{}_{mm}]$ ···· $[J^M{}_{mm}]$의 데이터를 전송 받아 식 (4-7)의 방법에 의하여 연산이 완성된다.

$$[J_{mm}]=[J^1{}_{mm}]+[J^2{}_{mm}]+\cdots+[J^M{}_{mm}] \qquad (4\text{-}7)$$

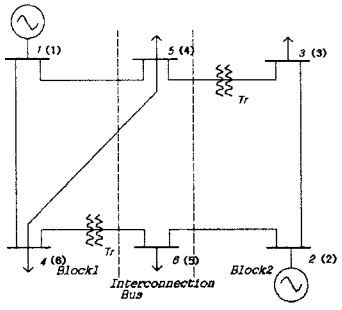
이때 병렬처리 연산 단계에서 주로 상호 연결 부분에 대한 연산을 위한 데이터 전송 및 각 프로세서의 연산 결과에 대한 덧셈이 필요하게 되는데 이 과정이 블록 병렬처리기법으로 처리할 때 추가로 소요되는 시간이 된다.

5. 적용예 및 결과 고찰

5.1 전력계통의 구성과 타스크 할당

본 논문에서 제시한 Jacobian 행렬의 모델링을 적용하기 위하여 그림 5.1의 모델 계통에 적용하였는데 이 6모선 모델 계통에 대한 그림은 병렬처리를 위한 분할기법 11)에 의하여 분할한 결과를 나타낸 것이며 모선 "5"와 "6"에 의하여 두개의 블록으로 분할되었다. 모선 번호는 분할의 효율을 향상시키기 이한 MPRLD 서열산법12)을 적용하여 재부여 된 것이며 ( ) 안의 번호가 원래의 모선 번호이다.

이 모델 계통에 대한 데이터는 표 5.1과 표5.2에 50MVA를 기준으로 한 단위법으로 표시하였으며 기준모선은(slack bus)은 1번 모선으로 가정하였다.



No. Type
1 Slack
2 P-V
3 P-Q
4 P-Q
5 P-Q
6 P-Q

그림 5.1 분할된 6모선 모델 전력 계통
Fig. 5.1 The tearing of 6 bus model power system

표 5.1 모델 전력계통의 선로 데이터
Table 5.1 Line Data of Model Electric Power System

Between Buses	Line impedance	Half line charging admittance	Off-nominal turns ratio(a)
1 - 5	0.08 +j0.37	0.007	-
1 - 4	0.123+j0.518	0.010	-
2 - 3	0.723+j1.05	0.	-
2 - 6	0.282+j0.64	0.	-
5 - 3	0. +j0.133	0.	0.909
5 - 4	0.097+j0.407	0.0076	-
4 - 6	0. +j0.3	0.	0.976

A static shunt capacitor at bus 5 is present with admittance=j0.005

표 5.2 모델 전력계통의 모선 데이터
Table 5.2 Bus Data of Model Electric Power System

Bus No.	Type	Bus Loading P(MW)	Bus Loading Q(MVAR)	Specified Voltage[V]	Reactive power Limit Q_{min}	Reactive power Limit Q_{max}
1	Slack	-	-	1.05	-	-
2	P-Q	0.5	-	1.10	0.13	0.2
3	P-Q	0.55	0.13	-	-	-
4	P-Q	0.5	0.05	-	-	-
5	P-Q		-	-	-	-
6	P-Q	0.3	0.18	-	-	-

본 실험은 각 프로세서의 연산을 프로그램으로 구성하여 한대의 컴퓨터로 시뮬레이션 시키는 방법에 의하여 수행하였다. 프로세서간의 연산 데이터의 교환은 각 프로세서의 연산 결과를 필요로 하는 프로세서, 즉 시뮬레이션된 프로그램에 데이터를 연산전에 미리 입력시키는 방법에 의하여 처리하였으며, 병렬처리기법의 연산 수행 결과를 검증하기 위하여 분할하지 않은 계통을 순차적인 방법에 구한 Jacobian 행렬을 얻은 결과와 2개의 크러스터로 분할된 계통에 대해서 Jacobian 행렬의 모델링 기법을 시뮬레이션하여 얻은 결과를 비교, 검토하였다.

5.2 Jacobian 행렬의 모델링에 의한 전력조류 계산의 병렬처리

모델계통의 선로정수로 부터 MPRLD 서열산법으로 재서열된 어드미턴스 행렬(Y_{bus})을 구하여 각 모선들을 분할된 계통의 그룹으로 표현하면 표 5.3과 같이 구해진다. 이 어드미턴스 행렬을 제시한 데이터 할당방법에 의하여 3개의 프로세서 "#a", "#b", "#c"의 각 기억장치에 표 5.4와 같이 할당시켰다. 이들 표의 제일 좌측과 상측에 부여된 번호는 모선번호를 나타낸 것이다.

표 5.3 모델계통의 어드미턴스 행렬 (Y_{bus})
Table 5.3 Admittance Matrix of Model System (Y_{bus})

	1	2	3	4	5	6
1	0.9922 -j4.4025			-0.4339 j1.8275	-0.5583 j2.5820	
2		1.0214 -j1.9545	-0.4449 j0.6401			-0.5765 j1.3085
3		-0.4449 j0.6401	0.4449 -j8.1649		0.0 j8.2715	0.0 j0.0
4	-0.4339 j1.8275			0.9880 -j7.6517	-0.5541 j2.3249	0.0 j3.4153
5	-0.5583 j2.5820		0.0 j8.2715	-0.5541 j2.3249	1.1124 -j13.9995	0.0 j0.0
6		-0.5765 j1.3085	0.0 j0.0	0.0 j3.4153	0.0 j0.0	0.5765 -j4.6418
	Block 1	Block 2		Block 1	Interconnection Bus	

표 5.4 각 프로세서의 기억장치에 할당된 어드미턴스 행렬
Table 5.4 Assigned admittance matrix at the memory of each processor

Processor #a

	1	4	5	6
1	0.9922 - j4.4025	-0.4339 + j1.8275	-0.5583 + j2.5820	
4	-0.4339 -j1.8275	0.9880 - j7.6517	-0.5541 + j2.3249	0.0 + j3.4153
5	-0.5583 + j2.5820	-0.5541 + j2.3249		
6		0.0 + j3.4153		

Processor #b

	2	3	5	6
2	1.0214 - j1.9545	-0.4449 + j0.6401		-0.5765 + j1.3085
3	-0.4449 + j0.6401	0.4449 - j8.1649	0.0 + j8.2715	0.0 + j0.0
5		0.0 + j8.2715		
6	-0.5765 + j1.3085	0.0 + j0.0		

Processor #c

	5	6
5	1.1124 - j13.9995	0.0 + j0.0
6	0.0 + j0.0	0.5765 - j4.6418

분할된 모델계통에 대하여 모선 분할에 의한 분할방정식(diakoptic equation)에 의하여 Jacobian 행렬을 이용한 계통방정식의 일반식은 다음과 같이 표현할 수 있다.

$$\begin{bmatrix} \Delta P_i \\ \Delta Q_i \\ \Delta P_c \\ \Delta Q_c \end{bmatrix} = \begin{bmatrix} J_{11} & & J_{13} \\ & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \begin{bmatrix} \Delta\theta_i \\ |\Delta V_i| / |V_i| \\ \Delta\theta_c \\ |\Delta V_c| / |V_c| \end{bmatrix} \quad (5\text{-}1)$$

각 프로세서의 기억장치에 할당된 어드미턴스 행렬과 모선 데이터로 부터 제시된 방법에 의하여 각 프로세서에 의하여 Jacobian 행렬의 모델링을 수행하면 표 5.5에서 표 5.7과 같이 구해진다.

표 5.5 "processor #a" 에서 구해진 Jacobian 행렬
Table 5.5 Jacobian matrix obtained by "processor #a"

[J_{11}]		[J_{13}]			
7.65907	.96631	-2.32490	-.55410	-3.41530	.00000
.96631	7.64433	.55410	-2.32490	.00000	-3.41530
-2.32490	.55410	5.03600	-1.14032		
-.55410	-2.32490	-1.14032	5.03600		
-3.41530	.00000			3.41530	.00000
.00000	-3.41530			.00000	3.41530
[J_{31}]		[J_{33}^a]			

표 5.6 "processor #b" 에서 구해진 Jacobian 행렬
Table 5.6 Jacobian matrix obtained by "processor #b"

[J22]			[J23]			
2.15006	-.71071	-.48939			-1.43935	-.63415
-.71071	8.98221	.40041	-8.27150	.00000	.00000	.00000
-.48939	.40041	7.34759	.00000	-8.27150	.00000	.00000
	-8.27150	.00000	8.27150	.00000		
	.00000	-8.27150	.00000	-8.27150		
-1.43935	.00000	.00000			1.43935	-.63415
-.63415	.00000	.00000			-.63415	-1.43935

[J32] [J33b]

표 5.7 "processor #c" 에서 구해진 Jacobian 행렬
Table 5.7 Jacobian matrix obtained by processor "#c"

[J33c]

.00000	2.22480	.00000	.00000
2.22480	27.99900	.00000	.00000
.00000	.00000	.00000	1.15300
.00000	.00000	1.15300	9.28360

다시 각 프로세서에서 구한 상호 연결 부분의 Jacobian 행렬은 식 (5-1)에 의해서 제시된 방법에 의하여 표 5.8과 이 구해진다.

표 5.8 상호 연결 부분의 Jacobian 행렬
Table 5.8 Jacobian matrix of Interconnection bus

[J33]

13.30750	1.08448	.00000	.00000
1.08448	14.69150	.00000	.00000
.00000	.00000	4.85465	.51885
.00000	.00000	.51885	4.42895

최종적으로 병렬처리를 위한 Jacobian행렬의 모델링에 의하여 구해진 값은 표5.9와 같다. 동일한 계통에 대하여 순차적인 방법에 Jacobian행렬의 참값을 구하면 표 5.10과 같이 얻어진다.

표 5.9 병렬처리를 위한 Jacobian행렬의 모델링에 의하여 구해진 값
Table 5.9 The value by Jacobian matrix modeling for parallel processing

2.15006	-.71071	-.48939			-1.43935	-.63415		
-.71071	8.98221	.40041	-8.27150	.00000	.00000	.00000		
-.48939	.40041	7.34759	.00000	-8.27150	.00000	.00000		
	-8.27150	.00000	13.30750	1.08448	.00000	.00000	-2.32490	.55410
	.00000	-8.27150	1.08448	14.69150	.00000	.00000	-.55410	-2.32490
-1.43935	.00000	.00000	.00000	.00000	4.85465	.51885	-3.41530	.00000
-.63415	.00000	.00000	.00000	.00000	.51885	4.42895	-2.32490	.55410
			-2.32490	-.55410	-3.41530	.00000	7.65907	.96631
			.55410	-2.32490	.00000	-3.41530	.96631	7.64433

표 5.10 순차적인 방법에 의하여 얻어진 Jacobian행렬의 참값
Table 5.10 The value of Jacobian matrix by serial method

2.14998	-.71067	-.48935			-1.43931	-.63420		
-.71067	8.98135	.40037	-8.27060	.000000	.00000	.00000		
-.48939	.40041	7.34837	.00000	-8.26068	.00000	.00000		
	-8.27060	.00000	13.30672	1.08446	.00000	.00000	-2.32494	.55410
	.00000	-8.26068	1.08446	14.64265	.00000	.00000	-.55410	-2.32494
-1.43931	.00000	.00000	.00000	.00000	4.85597	.51885	-3.41667	.00000
-.63420	.00000	.00000	.00000	.00000	.51885	4.42762	.00000	-3.41667
			-2.32494	-.55410	-3.41667	.00000	7.66045	.96634
			.55410	-2.32494	.00000	-3.41667	.96634	7.57613

이와 같이 본 논문에서 제안한 병렬처리를 위한 Jacobian 행렬의 모델링의 결과와 순차적인 방법에 의하여 얻은 결과에서 나타나는 작은 오차는 시뮬레이션 과정에서 데이터 교환시 소수점 자리의 간략화 과정에서 발생한 것으로 병렬처리를 위한 Jacobian 행렬의 모델링이 정확히 수행되었음을 알 수 있다.

표 5.11: 각 프로세서에서 병렬로 수행된 전진대입 결과
Table 5.11: Parallel forward solution obtained by each processors

"Processor #1,#2"

열번호	$y(i) = [L]^{-1} \cdot b(i)$	
	Processor #1	Processor #2
1		.18030
2		-.04314
3		.10870
4	-.06245	
5	.00236	
6	-.14719	-.38137
7	-.02929	.89908
8	-.21430	.27319
9	.00804	.12036

"Processor #3"

열번호	$y(i) = [L]^{-1} \cdot b(i)$
6	-.10703
7	.08113
8	-.19139
9	.15450

표 5.12 각 프로세서에서 수행된 역진대입 결과
Table 5.12 Backward solution obtained by each processors

"Processor #3"

열번호	$x(i) = [U]^{-1} \cdot y(i)$
6	- .22392
7	.12751
8	-.21971
9	.15450

"Processor #1,#2"

열번호	Processor #1	Processor #2
1		.04475
2		-.27008
3		.25497
4	-.23791	
5	.14854	

또한 계통을 분할하지 않고 반복연산의 첫번째를 수행하여 얻은 결과는 표 5.13과 같은데 이 결과는 병렬처리기법으로 구한 표 5.12 와 동일함을 알 수 있다. 이와 같이 전력계통의 조류계산을 위하여 LU 분할 및 전진/역진 (forward/ back ward substitution)의 과정에 포함한 전체 전력조류계산에 대한 병렬처리 과정을 완성하였다.

표 5.13 Serial 로 수행된 첫번째 반복계산 결과
Table 5.13 Serial computation result of 1st iteration

x(i) = [$\Delta\theta$,|ΔV|/|V|]	
Δe_2	.04755
Δe_3	-.27008
$\|\Delta V_3\|/\|V_3\|$	.25497
Δe_6	-.23791
$\|\Delta V_4\|/\|V_4\|$	.14854
Δe_4	-.22392
$\|\Delta V_5\|/\|V_5\|$	.12751
Δe_5	-.21971
$\|\Delta V_6\|/\|V_6\|$	.15450

제안된 병렬처리기법의 연산시간은 각 블록의 크기 중 가장 큰 블록에 포함된 모선수와 이들 모선과 관련된 삼각행렬 [L],[U]에 있는 영이 아닌 요소의 수에 의하여 좌우되는데, 이 두요소들은 어드미턴스 행렬의 재서열에 따라 변할 수 있다. 또한, 경계모선의 숫자는 블록 병렬처리기법의 연산 효율에 매우 중요한 영향을 미치게 된다. 그러나 블록기법에 의한 병렬처리는 요소기법에 비하여 데이터 교환에 소요되는 시간은 무시할 수 있을 정도로 작고 , 계통의 크기가 커질 수록 공통부분의 모선 수는 분할그룹의 모선 수에 비하여 매우 작게 되므로, 상호 연결망 (interconnection network)을 이용한 정보교환의 부담(communication overhead)이 매우 작아지고, 블록기법의 공통부분을 처리하기 위한 데이터 교환은 전체 연산에 비하여 아주 작게 된다.

6. 결 론

전력계통의 조류계산시 꼭 수반되는 Jacobian 행렬 연산의 효율적인 데이터 할당과 병렬처리는 전력조류계산 전체의 효율을 좌우하게 된다. 본 논문에서는 실제 전력계통 해석 문제에서 자주 접하게 되는 Newton 법에 의한 전력 조류계산을 위하여 Jacobian 행렬 연산을 포함한 LU삼각화 직접해법의 전 과정에 대한 병렬처리를 효율적으로 처리할 수 있는 기법을 제시하였으며, 실제 전력조류 계산에 적용하기 위하여 모델 전력 계통 해석에 적용하여 개발된 기법이 정확히 시행될 수 있음을 보였다. 본 실험은 한대의 컴퓨터로 각 프로세서의 연산을 프로그램으로 구성하여 시뮬레이션 시키는 방법에 의하여 수행하였다. 프로세서간의 연산 데이터의 교환은 각 프로세서의 연산 결과를 필요로 하는 프로세서, 즉, 시뮬레이션된 프로그램에 데이터를 연산 전에 미리 입력시키는 방법에 의하여 처리하였다. 본 논문에서 수행된 중요한 내용은 다음과 같다.

1) 병렬처리를 위하여 개발된 MPRLD 서열산법과 전력계통의 분할기법을 적용하여 전력조류계산의 전체를 처리 할 수 있는 병렬처리 프로그래밍 기법을 완성하였다.
2) 병렬컴퓨터의 각 프로세서에 대한 어드미턴스 Y_{bus}행렬의 타스크 할당 방법을 제시하였다
3) 전력조류계산의 병렬처리를 위한 Jacobian 행렬의 연산과정에 대한 모델링 기법을 제시하였으며, 모델전력계통에 대한 각 프로세서의 병렬처리기능을 프로그램으로 구성, 시뮬레이션을 실시하여 신뢰성을 확보하였다.

이러한 결과는 향후 프로그래밍이 어렵지만 경제적인 분산메모리형 컴퓨터의 이용을 가능하게 되어 병렬처리에 의한 전력계통해석의 실용화를 앞당기게 될 것이다.

[참 고 문 헌]

[1] M. Feilmeier, G. R. Joubert, U. Schendel, "Parallel Computing" Volume 3.(1)M. Feilmeier, G. R. Joubert, U. Schendel, "Parallel Computing" Volume 3. North-Holland-Amsterdam, 1986.

[2] Qusay H.Manmoud, "Distributed Programing With Java", 인포북 출판사 , 2001

[3] Merin hughes, "Java Network Progrming", 인포.북 출판사, 2000

[4] M.Pedro Silva, Alexandra V.Sousa, "Web based DMS-distribution management system", IEEE Trans. on power system 2000.

[5] M. J. Flynn, " Very high-speed computing system," Proc. IEEE,Vol. 54, pp.1901-1909, 1966

[6] 맹 승열 , " 병렬형 컴퓨터 구조" 전자공학회지, 제18권 제7호, 7월 1991

[7] D. D. Gajski and J. K. Peir, "Essential Issues in multiprocessor systems," IEEE Computer, pp. 9-27, June 1985

[8] 이 춘모, 서 희석 "스파스 선형방정식의 병렬처리에 관한 연구", 충청전문대학 논문집 P225-246, 1997. 12.

[9] C. P. Arnold, M. I. Parr,and M. B. Dewe,"An efficient parallel algorithm for the solution of large sparse linear matrix equations", IEEE Trans. on Computers, Vol. C-32, No.3, pp.265-272, March 1983

[10] O. Wing, J. w. Huang,"A computation model of parallel solution of linear equations",IEEE Trans.Comput.,Vol.C-29, pp.632-638, July, 1980

[11] C.Hong, and R.Y.Liu, "A new parallel algorithm for solving large sparse patrix equations and parallel Newton's power flow" ICEE, 1999

[12] 이 춘모, 신 명철, "스파스백터법에 유용한 서열산법에 관한 연구" 대한전기학회지,제40권, 제10호 pp.991-998, 1991. 10.

[13] 이 춘모, 신 명철, "전력계통해석의 병렬처리를 위한 분할기법에 관한 연구" 대한전기학회지, 제45권 ,제12호 pp.1672-1678, 1996. 12.

[14] J. P. Hayes, T. Mudge, and Q. F. Stout, " A microcessor-based Hyper-cube super computer," IEEE Micro, Vol. 6. No. 5, pp. 6-17 , Oct. 1986.

[15] Danial J. Tylavsky, Anjan Bose " Parallel Processing In Power Systems Computation," An IEEE committe report by task force of the computer and analytical methods subcommittee of the power systems engineering comittee,IEEE Trans.PAS Vol. 7, No.2, pp 629-638, May 1992