병렬처리 기반의 경제급전 계산

이상현
고려대학교 컴퓨터정보통신대학원
laraya7@naver.com

An Economic Dispatch Solution based on Parallel Processing

Lee, Sang Hyun
Graduate School, Korea University

요 약

전력계통의 경제급전은 전력거래시스템 및 에너지관리시스템(EMS)에서 사용되는 중요 기능 중의 하나이다. 경제급전의 계산은 대규모 행렬로 구성된 선형방정식에 기반하고 있다. 전력계통에 IT 기술을 이용한 여러 서비스가 개발 가능해짐에 따라 처리할 데이터량도 점차적으로 증가하고 있다. 본 논문에서는 이러한 경제급전을 병렬처리를 이용해 수행 속도를 향상시키는 방안을 제시하고자 한다.

1. 서 론

경제급전은 전력계통 운영의 여러 응용프로그램 중의 하나로 전 계통의 발전비용을 최소화 하기 위해 사용되고 있다. 최근 전력계통은 복잡한 대규모 시스템으로 성장함에 따라 여러 응용프로그램들이 분산환경으로 구성되고 운영되고 있으며, 계통해석 및 제어에 필요한 계산시간이 증가하고 있다. 하지만 이와 같이 시스템이 대규모로 복잡해지는 환경에서도 여러 전력계통 해석 응용프로그램은 주어진 시간 안에 계산을 끝내야 한다. 현재 순차적으로 계산 되고 있는 경제급전 식은 이러한 조건을 만족시키기가 점점 더 어려워지고 있다. 본 논문에서는 순차적으로 계산되는 경제급전 선형방정식의 행렬을 블록화하여 여러 프로세서가 각각 담당 블록을 병렬로 계산함으로써 속도 향상을 이룰 수 있는지 살펴 보았다. 행렬 블록화는 각 프로세스의 처리량이 균등하고, 통신시간이 최소로 될 수 있도록 하였다. 아울러 행렬의 블록을 담당하는 병렬 프로세스 수를 점진적으로 늘려감에 따라 프로세스 간 통신시간이 전체 성능 향상을 저하시킬 수 있는 직전까지의 최적 프로세스 수를 도출하고자 한다. 본 연구에서는 세가지 방법으로 경제급전 계산 행렬을 블록화 하고, 각 방법에 대해 프로세스 수를 증가 시켜 순차적 알고리즘과의 계산시간을 비교하여, 경제급전 병렬 처리 계산에서 가장 효율적인 데이터 분할 방법과 병렬 프로세스의 최적 한계 수를 알아보았다. 현 계통의 발전기 10기를 선택하여 수요와 출력 한계를 제약조건으로 세가지 방법에 대한 최적의 프로세스 수를 결정하는 실험을 진행 하였다. 병렬처리 계산을 위해 분산환경에서 병렬처리가 가능한 MPI (Message Passing Interface) 프로그래밍을 이용해 경제급전 병렬 프로그램을 제시하였다.

2. 경제급전 목적함수

경제급전 목적함수는 각 발전기의 발전비용함수의 합으로 이루어지고 그 합이 최소화되어야 한다. 발전기 최대/최소 출력 한계 및 총 발전출력량과 총수요가 같아야 하는 제약조건으로 이루어진 경제급전 식은

(1) 목적함수 : $Minimize \;\; C_{total} = \sum_{i=1}^{n} C_i(P_i)$

(2) 제약조건

$$\sum_{i=1}^{n} P_i = P_{load}$$

$$P_{i,\min} \le P_i \le P_{i,\max}$$

C_{total} : 총 발전비용

$C_i(P_i)$: 발전기 i의 발전 비용 함수

P_i : 발전기 i의 발전출력

P_{load} : 총수요

$P_{i,\min}$: 발전기 i의 최소출력 제약

$P_{i,\max}$: 발전기 i의 최대출력 제약

3. 행렬의 블록화

최적화 경제급전 순차적 계산은 목적함수와 제약조건

들에서 변환된 라그랑지안 함수(Lagrangian function)의 최소화 해를 찾는 것과 같다. 본 논문의 실험에서 사용된 라그랑지안 함수는 다음 그림 1과 같다.

$$L = \sum_{i=1}^{10}\int C_i(P_i) + \lambda(TotalDemand - \sum_{i=1}^{10} P_i)$$
$$+ \sum_{i=1}^{10}\beta_i^{\min}(P_i^{\min} - P_i + y_i^2) + \sum_{i=1}^{10}\beta_i^{\max}(P_i - P_i^{\max} + x_i^2)$$

그림 1. 경제급전 목적함수/제약조건의 라그랑지안 함수

$C_i(P_i)$: 발전기 i의 발전 비용 함수

λ : 총수요($TotalDemand$)에 대한 라그랑지 승수

$\beta_i^{\min}$: 발전기 i의 최소출력 제약 라그랑지 승수

$\beta_i^{\max}$: 발전기 i의 최대출력 제약 라그랑지 승수

$P_i^{\min}$: 발전기 i의 최소출력 제약

$P_i^{\max}$: 발전기 i의 최대출력 제약

y_i^2 : 발전기 i의 최소출력 제약 여유(slack) 변수

x_i^2 : 발전기 i의 최대출력 제약 여유(slack) 변수

위에서 구한 라그랑지안 함수로부터 구한 자코비안 행렬은 다음과 같이 구할 수 있다.

$$\mathbf{J} = \begin{bmatrix} [\nabla^2 L] & [\mathbf{G}] \\ [\mathbf{G}]^T & [\mathbf{0}] \end{bmatrix}$$

그림 2. 자코비안 행렬

$\nabla^2 L$: 라그랑지안 함수의 Gradient

$\mathbf{G}$ 행렬 :

$$\begin{bmatrix} \frac{\partial h_1}{\partial p_1} & \frac{\partial h_2}{\partial p_1} & \cdots & \frac{\partial h_{22}}{\partial p_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_1}{\partial p_{10}} & \frac{\partial h_2}{\partial p_{10}} & \cdots & \frac{\partial h_2}{\partial p_{10}} \\ \frac{\partial h_1}{\partial y_1} & \frac{\partial h_2}{\partial y_1} & \cdots & \frac{\partial h_{22}}{\partial y_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_1}{\partial y_{10}} & \frac{\partial h_2}{\partial y_{10}} & \cdots & \frac{\partial h_{22}}{\partial y_{10}} \\ \frac{\partial h_1}{\partial x_1} & \frac{\partial h_2}{\partial x_1} & \cdots & \frac{\partial h_{22}}{\partial x_1} \end{bmatrix}$$

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_1}{\partial x_{10}} & \frac{\partial h_2}{\partial x_{10}} & \cdots & \frac{\partial h_{22}}{\partial x_{10}} \end{bmatrix}$$

h_i : 총수요 및 최소/최대 출력 제약에 대해 발전기 출력 p_i와 여유변수 x_i, y_i 로 표현된 식

경제급전 해는 라그랑지안 함수로부터 구한 자코비안(Jacobean)행렬의 역행렬(행렬 $\mathbf{J}^{-1}$)과 라그랑지안 함수의 기울기 벡터 즉, ∇L (벡터 $\mathbf{m}$)의 곱의 결과(결과벡터 $\mathbf{c}$)가 일정 허용치(본 논문에서는 1.0e - 14)를 초과하지 않을 때까지 반복하여 구함으로써 해를 계산하였다.
위 순차적 계산을 병렬로 처리할 때는 자코비안 역행렬 블록을 각 프로세스가 맡아 별도로 계산하고, 각 병렬 프로세스에서 계산된 결과를 통합하여 최종 해를 만들어낸다. 이때, 각 프로세스에 할당되는 계산과정이 서로 독립적이어야 하고, 할당되는 블록이 가능한 균등해야 하며, 통합과정의 통신시간이 전체 계산시간에 비해 너무 많이 차지하지 않아야 한다.

3.1 행 기반 블록화

프로세스 i는 $\mathbf{J}^{-1}$ 행렬의 행 i블록과 $\mathbf{m}$ 벡터 사본을 갖고, 결과 $\mathbf{c}_i$를 계산하기 위해 필요한 요소들은 모두 갖추고 있다. 계산 결과는 각 프로세스 i에서 계산한 부분 결과 벡터 블록 $\mathbf{c}_i$를 최종 결과 벡터로 통합하기 위해 프로세스 간 통신이 필요하다.

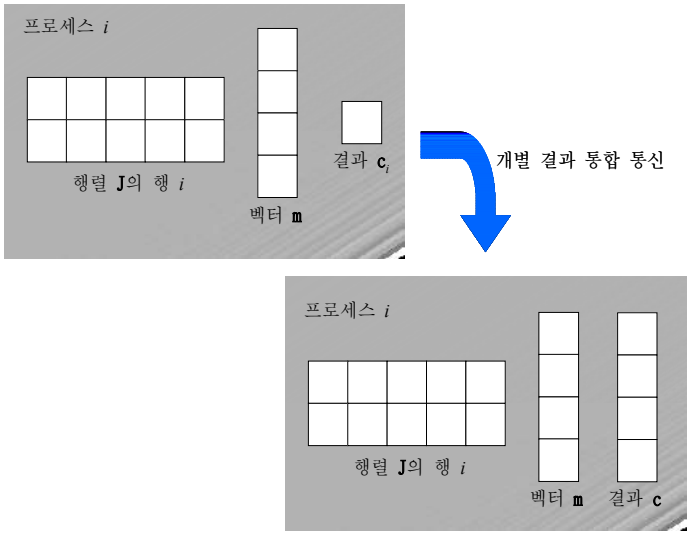


그림 3. 행 기반 병렬처리 계산

3.2 열 기반 블록화

프로세스 i 는 $\mathbf{J}^{-1}$ 행렬의 열 i 블록을 기반으로 데이터를 분할한다. $\mathbf{m}$ 벡터는 한 요소, 즉 m 벡터 배열의 한 요소만 가지고 있으면 프로세스 i 가 담당하는 계산 요소를 갖게 된다. 각 프로세스 i 에서 계산한 결과 $\mathbf{c}$ 는 결과 벡터의 부분 덧셈 항목이므로 최종 결과 벡터를 구성하기 위해 각 프로세스 i 간에 자신이 갖고 있는 결

과 c를 서로 주고 받을 수 있는 통신을 해야 한다.

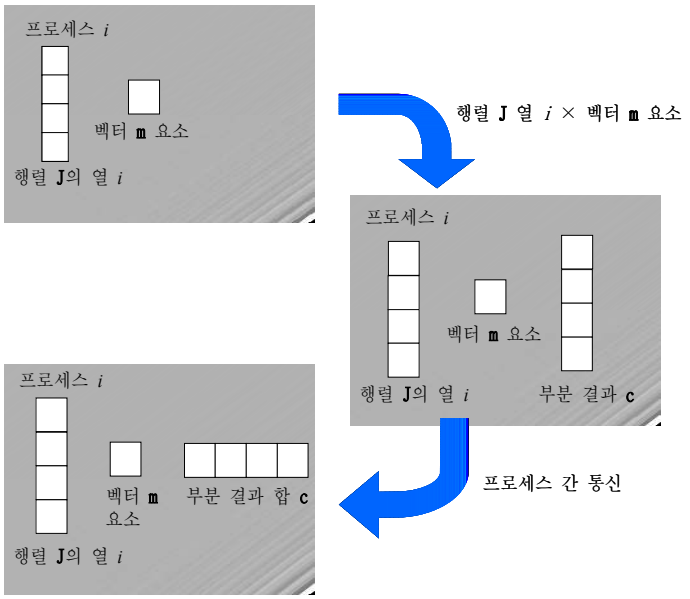
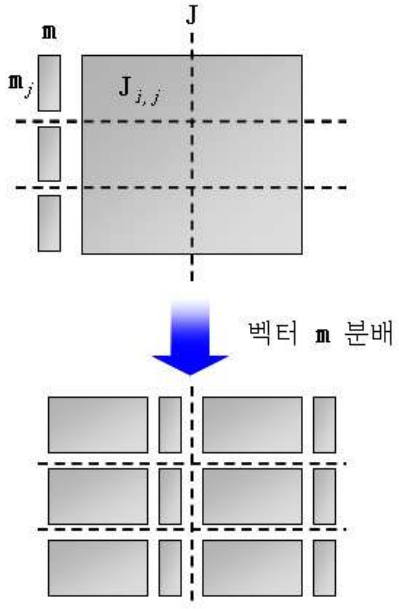


그림 4. 열 기반 병렬처리 계산

3.3 부분 행렬 기반 블록화

행렬 $\mathbf{J}^{-1}$를 부분 행렬 $\mathbf{J}^{-1}_{i,j}$으로 분할하여 각 프로세스가 분할된 $\mathbf{J}^{-1}_{i,j}$의 부분 행렬 계산을 담당한다. 이 때, m 벡터도 부분 벡터 $\mathbf{m}_j$으로 분할하여 각 담당 프로세스는 $\mathbf{J}^{-1}_{i,j}$ × $\mathbf{m}_j$를 계산하여 부분 결과 벡터 $\mathbf{d}_{i,j}$를 구한다. 분할 행렬 기반 블록화 병렬 처리는 다음의 방법으로 구현 된다.

- 벡터 m을 각 프로세스가 균등하게 담당할 수 있도록 분배한다.
- 프로세스는 담당한 부분 행렬 $\mathbf{J}^{-1}_{i,j}$와 부분 벡터 mj 계산한다.
- 각 프로세스가 계산한 부분 결과 $\mathbf{d}_{i,j}$를 통신하여 합해 최종 결과 벡터 c를 생성한다.



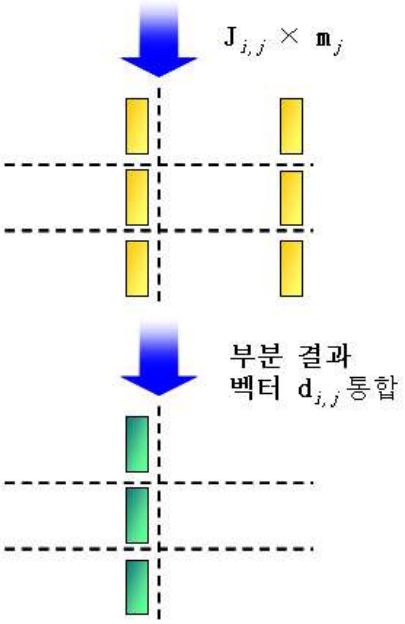


그림 5. 부분행렬 기반 병렬처리 계산

4. 실험결과

실험 환경은 Intel Dual Core CPU E6790 의 리눅스 OS 하에서 메시지 전달 방식의 병렬 라이브러리인 MPI(Message Passing Interface)를 사용하였다. <표 1>과 같이 현 계통에서 사용되는 발전기 10 기의 발전 비용 함수를 기반으로, 총 수요 제약 4850MW 와 <표 2>의 발전기 출력 한계를 제약조건을 만족시키도록 시뮬레이션을 수행했다. 제시한 세가지 행렬 블록화와 프로세스 수를 늘려감에 따라 순차적 계산 보다 성능 향상이 어느 정도 이루어 지는지 위의 환경 하에서 실험을 수행 하였다.

표 1. 발전기 비용함수

발전기	발전 비용 함수
1	$C_1(P_1) = 78.65612 + 1.299039\,P_1 + 0.000562\,P_1^2$
2	$C_2(P_2) = 63.828389 + 2.011941P_2 + 0.000003P_2^2$
3	$C_3(P_3) = 79.812305 + 1.797541\,P_3 + 0.000175\,P_3^2$
4	$C_4(P_4) = 76.529163 + 1.700427\,P_4 + 0.00029\,P_4^2$
5	$C_5(P_5) = 76.606258 + 1.764123\,P_5 + 0.000288\,P_5^2$
6	$C_6(P_6) = 59.260233 + 1.607545\,P_6 + 0.000013\,P_6^2$
7	$C_7(P_7) = 98.398851 + 1.52508\,P_7 + 0.000793\,P_7^2$
8	$C_8(P_8) = 26.614692 + 1.628199\,P_8 + 0.000001\,P_8^2$
9	$C_9(P_9) = 56.989269 + 1.583316\,P_9 + 0.000164\,P_9^2$
10	$C_{10}(P_{10}) = 32.903604 + 1.597836\,P_{10} + 0.000022\,P_{10}^2$

표 2. 발전기 출력 제약

발전기	최대출력 한계	최소출력 한계
1	267	160
2	640	192
3	708	213
4	523	160
5	543	163
6	528	196
7	474	142
8	561	175
9	571	171
10	614	320

그림 5 는 행렬 블록화 별로 병렬 계산을 수행했을 시, 소요되는 시간을 측정한 것으로 통신시간도 포함 된 것이다. Y 축의 SpeedUp 은 다음 식에 의해 계산하였다.

$$SpeedUp = \frac{T}{T_p}$$

TP는 프로세스 수 P에서 병렬로 처리한 소요 시간이며, T는 프로세스 수가 한 개, 즉 순차적으로 프로그램을 실행했을 때의 소요시간이다.

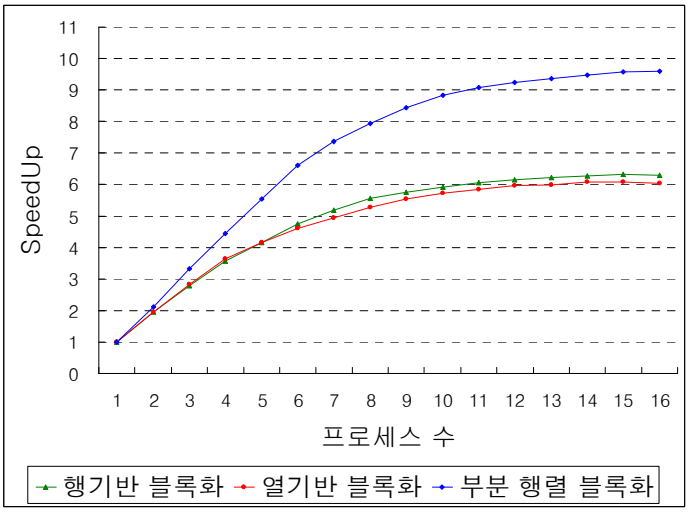


그림 6. 행렬 블록화 별 성능 비교 그래프

표 3. 행렬 블록화 별 성능 비교표

프로세스	행기반		열기반		부분 행렬	
	시간 (msec)	SpeedUp	시간 (msec)	SpeedUp	시간 (msec)	SpeedUp
1	66.46	1	63.81	1	63.41	1
2	32.74	1.938	32.94	1.937	29.83	2.126
3	22.75	2.789	22.66	2.816	19.13	3.315
4	17.82	3.561	17.57	3.632	14.25	4.45
5	15.22	4.17	15.32	4.165	11.47	5.528
6	13.37	4.746	13.83	4.614	9.61	6.598
7	12.26	5.176	12.94	4.931	8.62	7.356
8	11.4	5.567	12.11	5.269	7.98	7.946
9	11.03	5.753	11.51	5.554	7.51	8.443
10	10.74	5.909	11.13	5.733	7.18	8.831
11	10.48	6.055	10.92	5.843	6.99	9.072
12	10.31	6.155	10.71	5.958	6.86	9.243
13	10.19	6.228	10.66	5.986	6.78	9.353
14	10.12	6.271	10.51	6.071	6.69	9.478
15	10.06	6.308	10.5	6.077	6.62	9.579
16	10.07	6.302	10.59	6.025	6.6	9.608

그림 6 에 의하면 행렬을 부분 행렬로 블록화 하여 병렬처리를 했을 때, 가장 실행시간이 적게 소요되는 것을 알 수 있다. 행기반 블록화와 열기반 블록화의 병렬처리 수행시간은 프로세스 수가 증가할수록 행기반 블록화의 수행시간이 더 빠름을 볼 수 있다. 이는 열기반 블록화는 행단위로 읽어들인 행렬 데이터를 각 프로세스로 분산 시키는 통신 시간이 더 소요되기 때문이다.

행 기반 블록화의 병렬 처리 속도는 순차적 계산 보다 최대 6.3 배 향상 되었지만, 프로세스 수가 16 을 기점으로 속도가 둔화되는 현상을 보였다. 열 기반 블록화도 마찬가지로 최대 6.1 배 가량 병렬 처리 속도가 개선 되었고, 행 기반 블록화와 비슷하게 프로세스 수 15 이상부터 프로세스 간 통신시간 증가로 속도가 둔화되었다. 부분 행렬 블록화는 행 기반과 열 기반 블록화보다 높은 속도 개선을 이루어 최대 9.6 배 속도 향상을 보여 주고 있다. 부분 행렬 블록화는 프로세스 수 16 에서부터 프로세스 수 20 까지 속도 개선이 거의 정체 상태에 머무르다가 프로세스 수 20 이상 부터는 속도가 점차 둔화되는 현상을 보여 주었다.

5. 결 론

본 연구에서는 향후 전력계통시스템이 대규모의 데이터와 및 다양한 서비스 개발로 지금보다 복잡하게 시스템이 성장할수록, 요구되는 시간 내에 경제급전 해를 구하기 위한 방법으로 병렬 처리 기법을 제시하였다. 경제급전의 선형방정식의 해를 병렬로 구하기 위해서는 각 프로세스들이 최대한 효율적으로 계산 되도록 자코비안 역행렬과 벡터를 블록화 하였다. 제시한 세가지 블록화 방법 중에서 부분 행렬 기반 블록화 방법이 병렬 처리에서 가장 우수한 수치를 보여 주었고, 순차적인 계산방법에 비해 병렬로 처리했을 때, 성능 향상 효과가 큰 걸 알 수 있었다. 하지만, 발전기 10 대의 경제급전 환경에

서 행렬 블록을 담당하는 병렬 처리를 위한 프로세스 수의 한계점은 16 이상이 되었을 때, 프로세스간 통신에 소요되는 시간이 실제 계산에 소요되는 시간을 초과하여 점차 속도 개선이 둔화되는 것을 알 수 있었다.

전 계통의 발전기와 송전선로 제약을 고려한 경제급전을 계산할 때, 병렬처리 속도 향상을 위해 적절한 행렬 분할과 프로세스 수를 정하는 것이 중요한 속도 개선 요소가 되겠다. 향후, 여러 대의 PC 가 클러스터로 구성된 환경에서 프로세스 수를 늘려 가면서 최적의 성능 향상을 가져올 수 있는 방안을 찾고, 실제 계통에서 운전되고 있는 모들 발전기들과 보다 많은 제약조건을 가지고 경제급전 병렬처리 실험을 하고자 한다.

6. 참고 문헌

[1] Chitra Yingvivatanapong, Wei-Jen Lee, and Edwin Liu, “Multi-Area Power Generation Dispatch in Competitive Markets,” IEEE Trans. Power Syst., vol. 23, no. 1, pp. 196-203, Feb 2008.

[2] Stephen G. Nash, and Ariela Sofer, “BTN : Software for Parallel Unconstrained Optimization,” ACM Trans. Mathematical Software., vol. 18, no. 4, pp. 414-448, Dec 1992.

[3] Michael J. Quinn “Parallel Programming in C with MPI and OpenMP” 1st Ed. McGraw Hill.

[4] Narayan S. Rau “Optimization principles : practical applications to the operation and market of the electric power industry” 1st Ed. Wiley-Interscience.

[5] Allen J. Wood, and Bruce F. Wollenberg “Power Generation, Operation, and Control” 2nd Ed. Wiley-Interscience.

[6] Jack Casazza, and Frank Delea “Understanding Electric Power Systems : An Overview of the Technology and the Marketplace” 1st Ed. Wiley-Interscience.

[7] Mohammad Shahidehpour, and Yaoyu Wang “Communication and Control in Electric Power Systems : Applications of Parallel And Distributed Processing” 1st Ed. Wiley-Interscience.

[8] 박정도 “경제급전을 고려한 발전기 기동정지계획 알고리즘에 관한 연구”. 연세대학교 2000.6.

[9] 김정선. “MPI 병렬프로그램을 위한 성능가시화 시스템”. 공학기술 논문집. vol. 6, no. 1, 1997.8.

[10] 오세영 “최적화이론(I)” 초판. 교우사.