The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 2, pp.271-277, Apr. 30, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.2.271
JIIBC 2017-2-39

CUDA를 이용한 실시간 대기질 예보 자료동화

Data Assimilation of Real-time Air Quality Forecast using CUDA

배효식*, 유숙현**, 권희용***

Hyo-Sik Bae*, Suk-Hyun Yu, Hee-Yong Kwon*****

요 약 현대에 들어서면서 대기오염 물질이 심각하게 국민의 건강을 위협하는 단계에 이르렀기 때문에 이에 대한 예보의 중요성은 점점 높아지고 있다. 대기질을 예보하는데 있어서 예보 모델에 입력되는 초기장은 예보의 정확성에 영향을 미치는 요소이기 때문에 신뢰도 높은 초기장을 생성하는 것이 매우 중요하며, 이때 필요한 기법 중 하나가 자료동화이다. 자료동화는 대상 지역이 넓어지고, 관측소의 수가 증가될수록 더 많은 연산이 필요하기 때문에 그 수행시간이 길어진다. 때문에 예보 규모가 커질수록 기존의 순차처리 방식으로는 빠른 처리속도를 요구하는 현업에 적용하기 어렵다. 이에 본 논문에서는 자료동화 기법 중의 하나인 크레스만 방법을 CUDA를 이용하여 실시간으로 처리할 수 있는 방법을 제안하였다. 그 결과, 제안한 CUDA를 이용한 병렬처리 방법이 최소 35배 이상 속도가 향상되었다.

Abstract As a result of rapid industrialization, air pollutants are seriously threatening the health of the people, the forecast is becoming more and more important. In forecasting air quality, it is very important to create a reliable initial field because the initial field input to the air quality forecasting model affects the accuracy of the forecast. There are several methods for enhancing the initial field input. One of the necessary techniques is data assimilation. The number of operations and the time required for such data assimilation is exponentially increased as the forecasting area is widened and the number of observation sites increases. Therefore, as the forecast size increases, it is difficult to apply the existing sequential processing method to a field requiring fast processing speed. In this paper, we propose a method that can process Cresman's method, which is one of the data assimilation techniques, in real time using CUDA. As a result, the proposed parallel processing method using CUDA improved at least 35 times faster than the conventional sequential method and other parallel processing methods.

Key Words : Data assimilation, Cresman's method, CUDA

Ⅰ. 서 론

급속한 산업화로 인해 발생되는 다양한 대기오염 물질과 미세먼지 및 중국의 주요 공업도시지역에서 발생되어 유입되는 황사 등은 심각하게 국민 건강을 위협하는 요소로서, 이러한 대기오염 물질에 대한 예보의 중요성이 점점 높아지고 있다. 대기오염 예보 정보는 일상생활에서도 활용할 수 있는데, 그것은 외출 시 마스크의 착용 여부나 외부 활동의 결정 유무처럼 국민들의 일상과 밀접한 관련을 갖고 있다.[2]

*정회원, 안양대학교 컴퓨터공학과
**정회원, 안양대학교 정보통신공학과
***정회원, 안양대학교 컴퓨터공학과(교신저자)
접수일자 : 2017년 3월 27일, 수정완료 : 2017년 4월 7일
게재확정일자: 2017년 4월 7일

Received: 27 March, 2017 / Revised: 7 April, 2017
Accepted: 7 April, 2017
***Corresponding Author: hykwon@anyang.ac.kr
Dept. of Computer Science & Engineering, Anyang University, Korea

대기질 예보 시 가장 중요한 요소는 예보의 정확성이고, 이를 향상시키 위해서 사용하는 기법중의 하나가 자료동화이다. 자료동화는 대기질 예보 모델링의 결과로 나온 예보 값을 관측소에서 최근에 측정한 데이터로 보정하여 새로운 배경 값을 생성하는 기법이다.

자료동화의 목적은 '대기(혹은 해양)의 흐름 상태를 가능한 한 정확하게 산정하기 위하여 모든 가용 정보들을 이용하는 것' 이다.[2] 이를 이용하면 전혀 새로운 데이터를 가지고 예보를 하는 것 보다는 조금이라도 더 정확도를 높일 수 있다.[3] 본 논문에서는 이러한 자료동화 기법 중 가장 범용적으로 사용하며, 구현이 간단하여 현업 적용성이 큰 크레스만 방법을 사용한다.

크레스만 방법을 기존의 순차처리 및 병렬처리 방법으로 구현할 때의 문제점은 오랜 수행시간이 걸려서 빠른 처리를 요구하는 현업에 적용하기 어렵다는 점이다. 이는 예보 지역이 확대되고, 측정소의 개수가 증가될수록 더 심화된다. 물론 슈퍼컴퓨터 같은 대형프레임을 이용하여 병렬처리 방법을 사용하면 빠른 수행속도를 얻을 수 있지만 이를 위해서는 상당히 많은 비용이 필요하다.

본 논문에서는 CUDA를 이용한 병렬처리 기법으로 크레스만 방법을 구현하여 개인용 컴퓨터와 같이 비용이 적게 드는 환경에서도 현업에서 요구하는 실시간 처리가 가능한 정도로 수행속도를 향상시켰다.

제안한 방법을 설명하기 위해서 이어지는 2장에서는 관련연구에 대하여 기술하고, 3장에서는 크레스만의 구현에 대해 설명하며, 4장에서는 실험 및 결과를 제시하고, 5장에서 결론을 내리고 끝맺는다.

II. 관련 연구

자료동화란 현재의 대기질 상태를 설명하기 위하여 자구상의 유용한 모든 관측 자료를 사용하여 최적의 수치모델 초기자료를 생성하는 과정을 말한다. 아무리 완벽한 수치예보모델이라고 하더라도 예보시간이 지날수록 예보 오차는 증가할 수 밖에 없는데, 자료동화 과정은 수치모델의 예보오차가 커지는 것을 막는 안전장치라고 할 수 있다. 이때, 관측자료를 보정하는 과정에는 단순히 자료만을 섞는 것이 아니라, 수치예보모델이 요구하는 물리법칙, 역학적 조건 등을 잘 활용하여 잡음을 최소화하고 조화로운 모델 초기자료를 만드는 과정을 포함한다. 이러한 자료동화의 개념을 그림 1.에 나타내었다.

자료동화란 그림 1.에서 나타낸 바와 같이 예보 값(X_b)을 영향반경(R) 내에 속하는 불규칙하게 분포하는 측정값(y_i)들을 이용하여 보정하는 방법이다.[9]

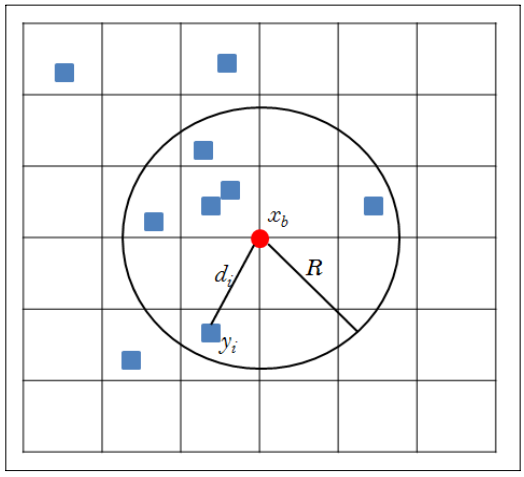

그림 1. 자료동화의 개념
Fig. 1. Concept of Data Assimilation

크레스만 방법은 George P. Cressman이 1959년 Monthly Weather Review에 소개한 보간법으로 예보측 모델에서 얻은 배경 값과 관측소에서 측정한 관측 값을 사용하여 선형조합에 의해 배경의 그리드 지점의 값을 보정하는 보간법이다. 그 보간법은 다음과 같다.[5]

$$x_a(u) = x_b(u) + \frac{\sum_{i=1}^{n} w_i [y(i) - x_b(i)]}{\sum_{i=1}^{n} w_i}$$

x_a : 보간된 값, x_b : 배경값,
y : 관측값, w : 거리에 따른 가중치

단, 거리에 따른 가중치는

$$w_i = \frac{R^2 - d_i^2}{R^2 + d_i^2} \qquad (d_i^2 \le R^2)$$
$$w_i = 0 \qquad (d_i^2 > R^2)$$

그림 2. 크레스만 방법
Fig. 2. Cressman's method

그림 2.에는 그림 1.의 자료동화의 기본 개념에서 거리에 따른 가중치(w)를 반영하여 예보 값을 보정하는 원리를 나타내었다. 이것은 불규칙하게 분포하는 측정값(y_i)

들이 예보 값(X_b)과 떨어져 있는 정도에 따라 그 가중치를 두어 가까이 있는 측정값은 예보 값에 영향을 더 미치고 멀리 떨어져 있는 측정값은 그 영향이 덜 미친다는 사실을 반영한 방법이다.

본 연구에서 제안하는 시스템을 구현하기 위해 필요한 CUDA(Compute Unified Device Architecture)는 NVIDIA사에서 그래픽 카드의 GPU를 범용적으로 활용할 수 있도록 제공하고 있는 기술로 '프로그램 모델', '프로그램 언어', '컴파일러', '라이브러리', '디버거', '프로파일러'를 제공하는 통합 개발 환경을 말한다.[12]

CUDA를 이용한 프로그램은 높은 연산 처리 능력, 병렬 프로그램의 확장성, 저렴한 가격, 편리한 설치 등의 장점을 가지게 된다. CPU보다 월등한 연산 처리능력을 가지고 있고 또한 다수의 ALU로 인해 대규모 데이터를 멀티스레드로 실행하여 처리하는 것이 가능하다. 같은 처리능력의 시스템을 구축하는데 CPU를 증가시키는 것보다 훨씬 저렴하게 구축할 수 있다. 그림 3.을 보면 CPU와 비교하여 GPU가 월등한 연산처리 능력을 가지고 있음을 확인할 수 있다.[11, 13]

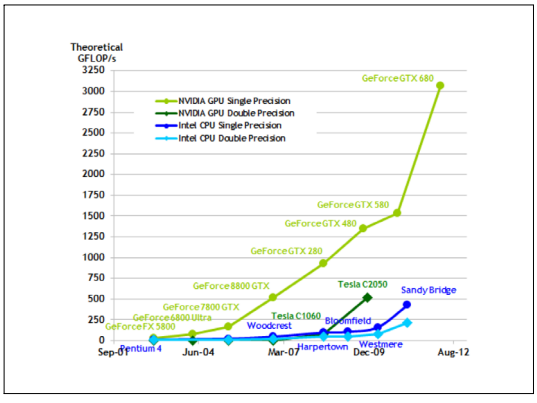

그림 3. CPU와 GPU의 초당 부동소수점 연산 속도
Fig. 3. Floating-Point Operations per Second for the CPU and GPU

CUDA는 그림 4.와 같은 방법으로 많은 수의 스레드를 실행하고 관리한다. 그리드 안에 2차원 혹은 3차원의 스레드 블록이 있고 스레드 블록 안에 여러 개의 스레드가 존재한다. 이러한 구조를 이용하여 CUDA의 커널함수를 작성하게 된다.

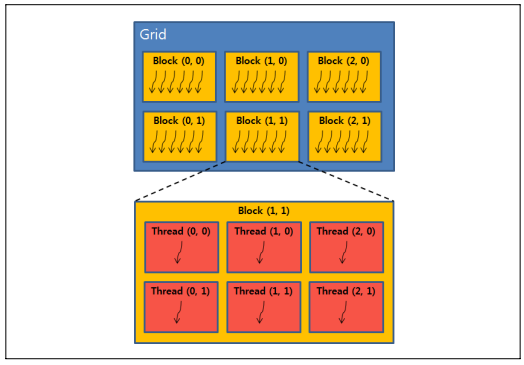

그림 4. 그리드를 구성하는 스레드 블록
Fig. 4. Thread Block that makes up the Grid

Ⅲ. 크레스만 방법의 구현

크레스만 방법을 순차처리 알고리즘으로 구현하면 그림 5.와 같다. 영역 안에 있는 모든 점이 값을 구해야 하는 격자점(i, j)이고, 각 점에 대하여 지정한 거리 안에 있는 주변 관측점(k)의 측정값을 가지고 새로운 값을 계산하고 그 값을 저장한다.

```
모든 격자점 (i, j)과 모든 관측점(k)에 대해
  만약 격자점 (i, j)과 관측점(k) 사이의 거리 d 가 L 이하인
  모든 관측점(k)에 대해
    각 관측점(k)의 거리에 따른 가중치 w 는 L^2 - d^2 / L^2 + d^2로 하고
    해당하는 모든 관측점(k)의 w의 합 sw와
    w * (관측점(k)의 값 y - 격자점(i, j)의 값 xb)의 합 sv를 구한다.
  격자점(i, j)의 새 배경값 xa 는  xb + sv / sw 이다
```

그림 5. 순차처리 알고리즘
Fig. 5. Sequential processing algorithm

그림 6.는 멀티스레드 기법을 이용한 알고리즘이다. 순차처리 알고리즘을 멀티스레드 기법에 맞게 수정 하였다. 그림 7.의 빨간 점은 스레드를 나타내고 색상영역은 각 스레드에 할당되어 있는 격자점들을 나타낸다. n개의 스레드 t_n는 동시에 각각 할당되어 있는 격자점(i_t, j)에 대하여 차례대로 새 배경 값을 계산한다. OpenMP는 멀티스레드 기법과 그 형태와 동작이 비슷한데 단지 OpenMP의 키워드를 이용하여 멀티스레드 기법을 컴파일러 차원에서 좀 더 간편하게 구현하도록 지원한다.

각 스레드 t_n 에 i를 기준으로 할당된 일부 격자점 (i_t, j)과
모든 관측점(k)에 대해
  만약 격자점 (i_t, j)과 관측점(k) 사이의 거리 d 가 L 이하인
  모든 관측점(k)에 대해
    각 관측점(k)의 거리에 따른 가중치 w 는 $L^2 - d^2 / L^2 + d^2$로 하고
    해당하는 모든 관측점(k)의 w의 합 sw와
    w * (관측점(k)의 값 y - 격자점(i_t, j)의 값 x_b)의 합 sv를 구한다.
  격자점(i_t, j)의 새 배경값 x_a 는 x_b + sv / sw 이다

그림 6. 멀티스레드 기법 알고리즘
Fig. 6. Multi-Thread technique algorithm

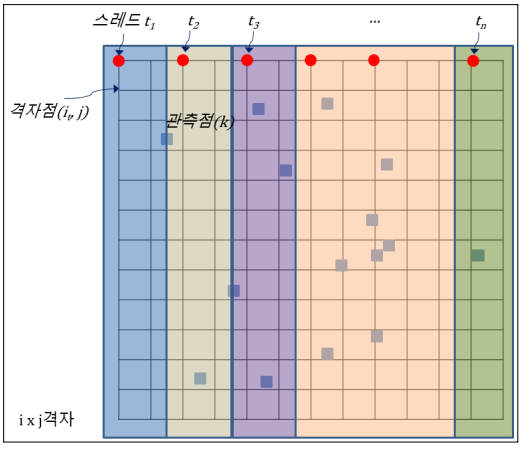

그림 7. 멀티스레드 동작
Fig. 7. Multi-Thread behavior

마지막으로 그림 8.은 CUDA의 알고리즘이다. 본 논문의 목적인 실시간 처리가 가능한 자료동화를 구현하기 위한 병렬처리 알고리즘이다. CUDA는 논리적으로 한 그리드 내에 여러 개의 블록을 생성하고 각 블록에 여러 개의 스레드를 생성해서 관리한다.

j개의 블록과 각 블록당 i개 만큼 생성된 CUDA 스레드 C에 각각에 할당된 한 격자점 (i_c, j_c)과 모든 관측점(k)에 대해
  만약 격자점 (i_c, j_c)과 관측점(k) 사이의 거리 d 가 L 이하인
  모든 관측점(k)에 대해
    각 관측점(k)의 거리에 따른 가중치 w 는 $L^2 - d^2 / L^2 + d^2$로 하고
    해당하는 모든 관측점(k)의 w의 합 sw와
    w * (관측점(k)의 값 y - 격자점(i_c, j_c)의 값 x_b)의 합 sv를 구한다.
  격자점(i_c, j_c)의 새 배경값 x_a 는 x_b + sv / sw 이다

그림 8. CUDA 알고리즘
Fig. 8. CUDA algorithm

그림 9.를 보면 빨간 점은 스레드를 나타내고 색상영역은 블록을 나타내는데 격자의 j 만큼 블록을 생성하고 각 블록에 격자의 i 만큼 스레드를 생성한다. 즉 i x j 개의 많은 스레드가 동시에 생성되고 각 스레드는 자신이 위치한 격자점에 대하여 동시에 새 배경 값을 계산한다.

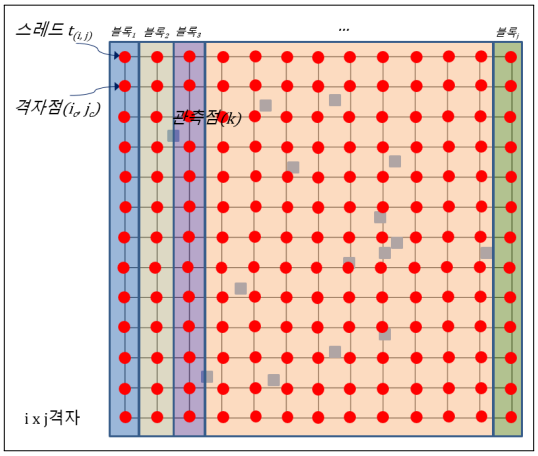

그림 9. CUDA 동작
Fig. 9. CUDA behavior

Ⅳ. 실험 및 결과

성능을 평가하기 위해서 크레스만 방법을 제안한 CUDA를 이용한 알고리즘과 기존의 기법인 순차처리, 멀티스레드, OpenMP로 구현하여, 각 알고리즘의 수행시간을 서로 비교하였다. 실험 환경은 표 1.에 나타내었고, 자료동화를 위한 환경 변수는 표 2.에 제시하였다.

실험 데이터는 2015년 2월 1일 2시의 동아시아의 27km 격자 간격으로 이루어진 미세먼지(PM10)의 사전 모델 데이터와 1,720개소의 관측소에서 실제 측정된 미세먼지 데이터를 사용하였다.

표 1. 실험 환경
Table 1. Experimental environment

구분	사양
CPU	Intel 쿼드코어 i5-2500K (3.30GHz)
RAM	DDR3 16GB(1333MHz)
Graphic Card	NVIDA GeForce GTX960

표 2. 변수 설정
Table 2. Variable setting

구분	값
가로 격자 수	146
세로 격자 수	122
격자간 실거리	27km
거리	2 격자 (54km)
관측소 수	1,720 개소
스레드 수	8

순차처리, 멀티스레드 기법, OpenMP, CUDA 알고리즘의 속도 측정을 위해 구현한 프로그램은 그림 10.과 같다. 그림 10.의 프로그램에서는 사전모델(Xb) 데이터와 관측(Yr) 데이터를 선택할 수 있다. 그리고 선택된 사전모델 데이터와 관측 데이터를 이용하여 수행한 각 알고리즘의 실행 화면은 그림 11, 그림 12, 그림 13, 그림 14와 같다.

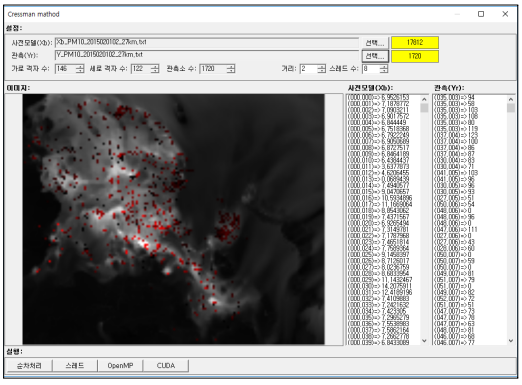

그림 10. 크레스만 방법 구현 프로그램
Fig. 10. Cressman Mathod Implementation Program

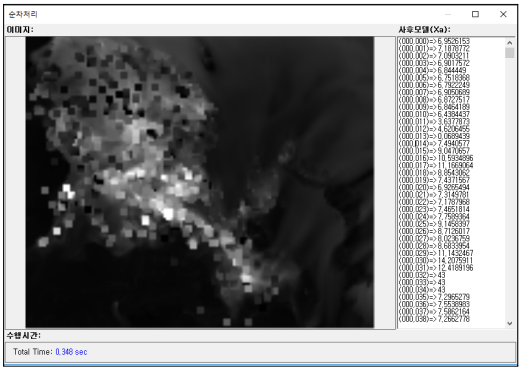

그림 11. 순차처리 알고리즘 실행화면
Fig. 11. Sequential processing algorithm execution

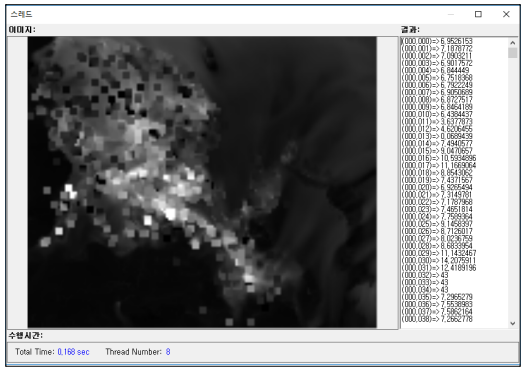

그림 12. Multi-Thread 알고리즘 실행화면
Fig. 12. Multi-Thread algorithm execution

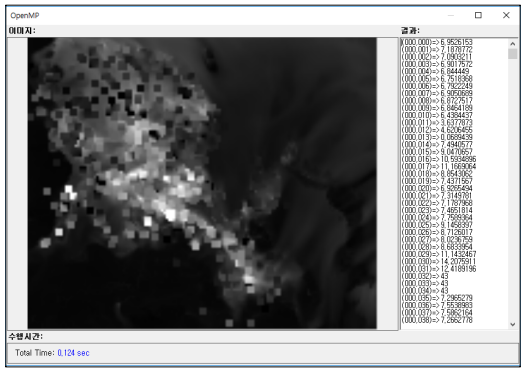

그림 13. OpenMP 알고리즘 실행화면
Fig. 13. OpenMP algorithm execution

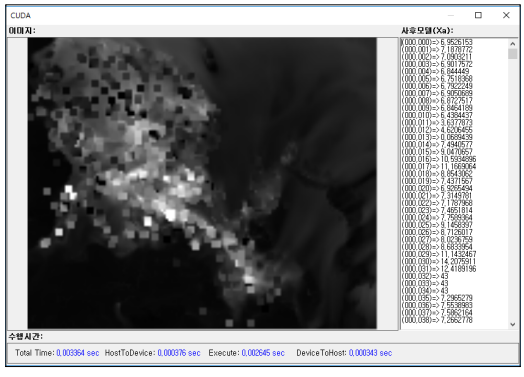

그림 14. CUDA 알고리즘 실행화면
Fig. 14. CUDA algorithm execution

성능 비교는 순차처리, 스레드, OpenMP, CUDA를 각각 20번씩 수행 한 후 각각의 수행시간 중 가장 높은 것과 낮은 것을 뺀 18개의 시간에 대한 평균을 가지고 비교하였다.

표 3. 수행 시간 비교
Table 3. Execution performance comparison(단위 초)

회수	순차처리	스레드	OpenMP	CUDA
1	0.343	0.167	0.103	0.003058
2	0.335	0.155	0.110	0.002955
3	0.336	0.142	0.124	0.003523
4	0.337	0.192	0.127	0.002878
5	0.339	0.198	0.113	0.002885
6	0.339	0.173	0.094	0.003322
7	0.337	0.170	0.094	0.003105
8	0.337	0.173	0.122	0.002964
9	0.335	0.169	0.095	0.002868
10	0.338	0.203	0.094	0.002937
11	0.338	0.152	0.112	0.002901
12	0.337	0.145	0.093	0.003142
13	0.341	0.197	0.102	0.002909
14	0.340	0.175	0.094	0.002901
15	0.336	0.219	0.099	0.002948
16	0.338	0.158	0.107	0.002916
17	0.340	0.169	0.112	0.002940
18	0.338	0.178	0.109	0.003069
19	0.344	0.178	0.100	0.002878
20	0.342	0.197	0.125	0.004841

각 방식의 수행시간 중 가장 높은 것과 가장 낮은 것을 뺀 나머지 시간에 대한 평균값은 표 4.에 나타내었다.

표 4. 평균 수행 시간
Table 4. Average run time **(단위 초)**

순차처리	스레드	OpenMP	CUDA
0.3383889	0.174944	0.106056	0.003013

이 결과를 보면 CUDA를 이용한 병렬처리의 경우 순차처리에 비해 그 시간이 약 112배, 스레드에 비해 58배, OpenMP에 비해 35배 더 빨리 수행됨을 알 수 있다.

V. 결 론

최근 들어 대기오염이 심화되고 있어서 대기질 예보의 중요성이 점점 커지고 있다. 이러한 대기질 예보에 있어서 초기장은 예보의 정확성에 큰 영향을 미치기 때문에 자료동화를 통해 초기장의 신뢰성을 높이는 것이 필요하다. 하지만 자료동화를 위한 기존의 순차처리 및 병렬처리 방식은 오랜 수행시간으로 인해 실시간 처리를 요구하는 현업에 적용하기가 어렵다.

이에 본 논문에서는 NVIDIA사의 GPGPU 연산에 최적화된 프로그래밍 개발환경인 CUDA를 이용하여 수행시간을 비약적으로 향상시켜 실시간으로 자료동화를 할 수 있는 방법을 연구하였다.

그 결과, 제안한 방법은 기존의 순차처리방식, 멀티스레드, OpenMP 방식보다 최소 35배에서 최대 112배까지 극적으로 수행속도가 향상되었다. 또한, 적용 지역을 동아시아뿐 만이 아니라 더 넓은 지역으로 확장하고, 관측소의 수가 증가될수록 효과는 더욱 극대화 될 것으로 예상된다. 이러한 성능으로 볼 때, 제안한 CUDA를 이용한 자료동화 방법은 실시간 처리를 요구하는 현업에 적용하기 효과적인 방법이라 사료된다.

References

[1] National Institute of Environmental Research, *A Study of Accuracy Improvement of Numerical Air Quality Forecasting Model(I)*, NIER-SP2015-064, 11-1480523-002327-01, 2015.

[2] S. Yu, Y. Koo, and H. Kwon, “Inverse Model Parameter Estimation Based on Sensitivity Analysis for Improvement of PM10 Forecasting,” Journal of Korea Multimedia Society, Vol. 18, No. 7, pp. 886-894, 2015.
DOI: https://doi.org/10.9717/kmms.2015.18.7.886

[3] K. Lee, S. Lee, and E. Kim, “Assessment of Global Air Quality Reanalysis and Its Impact as Chemical Boundary Conditions for a Local PM Modeling System,” Journal of Environmental Science International, Vol. 25, No.7, pp. 1029-1042, 2016.
DOI : https://doi.org/10.5322/JESI.2016.25.7.1029

[4] R. Park, K. Han, C. Song, M. Park, S. Lee, S. Hong, et al, “Current Status and Development of Modeling Techniques for Forecasting and Monitoring of Air Quality over East Asia,” Korea Society for Atmospheric Environment, Vol. 29, No. 4, pp. 407-438, 2013.
DOI: https://doi.org/10.5572/KOSAE.2013.29.4.407

[5] G.P. Cressman, “An Operational Objective Analysis System,” Monthly Weather Review, Vol. 87, No. 10, pp. 367-374, 1959.

DOI: https:/doi.org/10.1175/1520-0493(1959)087<0367:AOOAS>2.0.CO;2

[6] K. Ide, P. Courtier, M. Ghil, and A.C. Lorenc, "Unified Notation for Data Assimilation: Operational, Sequential and Variational," Journal of the Meteorological Society of Japan, Vol. 75, No. 1B, pp. 181-189, 1997.
DOI: https://doi.org/10.2151/jmsj1965.75.1B_181

[7] A.C. Lorenc, "Analysis Methods for Numerical Weather Prediction," Quarterly Journal of the Royal Meteorological Society, Vol. 112, Issue 474, pp. 1177-1194, 1986.
DOI: https://doi.org/10.1002/qj.49711247414

[8] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering, Vol. 82, Issue 1, pp.35-45, 1960.
DOI: https://doi.org/10.1115/1.3662552

[9] E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability*, Sigma Press, Seoul, 2012.

[10] S. Jason and K. Edword, *CUDA by Example : An An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, Boston, Massachusetts, 2011.

[11] NVIDIA, *CUDA C Programming Guide 7.5*, 2015.

[12] H. Kwon, C. Joo, and H.Y. Kwon, "A Study on High Speed Image Rotation Algorithm using CUDA," The Journal of the Institute of Internet, Broadcasting and Communication(JIIBC), VOL. 16 No. 5, pp.1-6, 2016.
DOI: https://doi.org/10.7236/JIIBC.2016.16.5.1

[13] K. Cho, B. Park, and T.Yoon, "A Study on Improved Image Matching Method using the CUDA Computing," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), Vol. 16, No. 4, pp. 2749-2756, 2015.
DOI: https://doi.org/10.5762/KAIS.2015.16.4.2749

저자 소개

배 효 식(정회원)

- 2003 : 안양대학교 컴퓨터공학과 (학사)
- 2013 : 안양대학교 컴퓨터공학과 (석사)
- 2016 : 안양대학교 컴퓨터공학과 (박사과정 수료)
- 2003 ~ 현재 : (주)넥스트소프트 SI사업부

<주관심분야 : 패턴인식, 영상처리, 병렬처리응용>

유 숙 현(정회원)

- 1999 : 안양대학교 컴퓨터공학과 (학사)
- 2002 : 안양대학교 컴퓨터공학과 (석사)
- 2011 : 안양대학교 컴퓨터공학과 (박사)
- 2012 ~ 현재 : 안양대학교 정보통신공학과 조교수

<주관심분야 : 패턴인식, 신경망, 영상처리, 병렬처리응용>

권 희 용(정회원)

- 1983 : 서울대학교 전자계산기공학과 (학사)
- 1986 : 서울대학교 전자계산기공학과 (공학석사)
- 1993 : 서울대학교 전자계산기공학과 (공학박사)
- 1986 ~ 1995 : 한국통신 연구개발단 선임연구원
- 1995 ~ 현재 : 안양대학교 컴퓨터공학과 교수

<주관심분야 : 패턴인식, 신경망, 영상처리, 병렬처리응용>