한국농림기상학회지, 제19권 제3호(2017) (pISSN 1229-5671, eISSN 2288-1859)
Korean Journal of Agricultural and Forest Meteorology, Vol. 19, No. 3, (2017), pp. 153~163
DOI: 10.5532/KJAFM.2017.19.3.153
ⓒ Author(s) 2017. CC Attribution 3.0 License.

농업기상재해 조기경보시스템에서의 고해상도 격자형 자료의 처리 속도 향상 기법

박주현[1*] · 신용순[1] · 김성기[1] · 강위수[1] · 한용규[1] · 김진희[2] · 김대준[2] · 김수옥[2] · 심교문[3] · 박은우[4]
[1]에피넷, [2]국가농림기상센터, [3]국립농업과학원, [4]서울대학교
(2017년 8월 7일 접수; 2017년 9월 15일 수정; 2017년 9월 18일 수락)

Speed-up Techniques for High-Resolution Grid Data Processing in the Early Warning System for Agrometeorological Disaster

J. H. Park[1*], Y. S. Shin[1], S. K. Kim[1], W. S. Kang[1], Y. K. Han[1], J. H. Kim[2], D. J. Kim[2], S. O. Kim[2], K. M. Shim[3] and E. W. Park[4]
[1]R&D Center, EPINET Co., Ltd., Anyang 14056, Korea
[2]National Center for Agro-Meteorology, Seoul National University, Seoul 08826, Korea
[3]National Institute of Agricultural Sciences, RDA, Wanju 55365, Korea,
[4]Department of Agricultural Biotechnology, Seoul National University, Seoul 08826, Korea
(Received August 7, 2017; Revised September 15, 2017; Accepted September 18, 2017)

ABSTRACT

The objective of this study is to enhance the model's speed of estimating weather variables (e.g., minimum/maximum temperature, sunshine hour, PRISM (Parameter-elevation Regression on Independent Slopes Model) based precipitation), which are applied to the Agrometeorological Early Warning System (http://www.agmet.kr). The current process of weather estimation is operated on high-performance multi-core CPUs that have 8 physical cores and 16 logical threads. Nonetheless, the server is not even dedicated to the handling of a single county, indicating that very high overhead is involved in calculating the 10 counties of the Seomjin River Basin. In order to reduce such overhead, several cache and parallelization techniques were used to measure the performance and to check the applicability. Results are as follows: (1) for simple calculations such as Growing Degree Days accumulation, the time required for Input and Output (I/O) is significantly greater than that for calculation, suggesting the need of a technique which reduces disk I/O bottlenecks; (2) when there are many I/O, it is advantageous to distribute them on several servers. However, each server must have a cache for input data so that it does not compete for the same resource; and (3) GPU-based parallel processing method is most suitable for models such as PRISM with large computation loads.

***Key words*: Agrometeorological early warning service, GPU PRISM, Speed-up raster calculation**

* Corresponding Author : J. H. Park
(parkjh@epinet.kr)

I. 서 언

공간적 고해상도의 자료 처리 및 생산은 기상, 농업, 수문, 산림 등의 다양한 분야에서 그 중요성이 지속적으로 증가해 오고 있다(Ahn *et al*., 2013). 또한 관측 기술 및 통신 기술의 발달로 수집되는 자료의 시간적 해상도 역시 증가함에 따라 계산에 이용될 수 있는 데이터의 양이 점차 증가하고 있다. 이러한 상황의 변화에 따라 데이터 처리, 저장 및 배포에 요구되는 소프트웨어와 하드웨어의 성능 개선이 과거부터 지속적으로 진행되어 왔다. 하지만 물리적 한계로 인해 단일 프로세서에 대한 하드웨어적 개선은 2000년대 초반부터 이미 둔화되기 시작되었으며 현재까지 계속 이어져 오고 있으며 이에 대해 프로세서 제조사들은 단일 코어에서는 저전력을 위주로 하는 대신 다수의 코어를 하나의 물리적 프로세서에 집적함으로써 연산의 총합을 향상시키는 방향으로 나가가고 있다. 하지만 이러한 개선은 상당수의 전통적 응용프로그램인 von Neumann 방식이라고 명명된 순차 프로그램으로는 얻을 수 있는 이득이 거의 없다. 따라서 변화된 하드웨어의 개발 방향에 맞추어 소프트웨어의 개선 작업을 지속하여 증가하는 입력 데이터를 실시간 처리에 충분히 활용할 수 있도록 해야 한다.

Zhang는 최신의 GPU 하드웨어를 이용하여 각 폴리곤의 다양한 통계치를 래스터 자료로 부터 계산하는 구역 통계(zonal statistics)를 고속으로 처리하는 방법을 제시하였다(Zhang *et al*., 2015). Mielikainen 은 WRF (Weather Research and Forecasting) 과 관련된 모델을 GPU 4개를 상호 연계하여 최대 715배의 속도 향상을 이끌어 내었다(Mielikainen *et al*., 2013). 최신 ArcGIS 버전인 10.5에서는 Spatial Analyst™에 포함된 다양한 연산 중 비교적 입력 자료 대비 계산량이 큰 Aspect, Slope, Viewshed에 대해 GPU 기반 연산을 지원하기 시작하였으며 향후 다양한 Toolbox™로 확대해 나갈 것으로 보인다(https://www.arcgis.com). Ozimek은 ArcGIS 경우와 유사하게 GRASS GIS 모듈로서 동작할 수 있는 전파 확산 예측 수행을 약 14배 향상 시키는데 병렬처리 기술을 이용하였다(Ozimek *et al*., 2013). 구글은 자사의 오픈소스 기계학습 및 딥러닝 라이브러리인 텐서플로우에서 NVIDIA GPU 및 Multi-GPU를 지원하면서 데이터 처리 속도를 크게 향상 시켰다(https://www.tensorflow.org/). 데이터 처리 속도는 앞서 기술된 연산 장치의 성능뿐만 아니라 내부 버스, 네트워크, 입출력장치 등 다양한 소요들에 의해서 결정된다. Kim *et al*.(2014)은 클라이언트 SSD 캐시를 이용한 분산 파일 시스템으로 가상 이미지 배포 및 부팅 수행시간을 2~3배 정도 향상시키는 기법을 제시하였다.

농업기상재해 조기경보시스템은 기후 상세화 모형, 생물계절 예측모형, 농업기상재해 예측모형, 전국기반 위험지수 산출모형 등 총 4가지 서브 시스템으로 구분될 수 있다. 본 연구에서는 각 서브 시스템을 구성하는 모형을 고속으로 처리하는데 사용될 수 있는 4 가지 기법(래스터 캐시, NFS 캐시, 분산 처리, GPU 병렬 처리)에 대해 소개한다. 그리고 각 기법에 대한 수행 성능을 평가하고 해당 기술 도입 시 기존 대비 얻을 수 있는 데이터 처리 속도 향상 비율을 확인하여 보았다. 현재 조기 경보 시스템의 하드웨어 적인 제약으로 인해 일부 기법 및 모델에 대해서는 조기경보 서버가 아닌 외부의 다른 서버에서 성능 측정을 하였으며, 해당 기술의 도입 가능성 여부를 살펴보았다.

II. 재료 및 방법

2.1. 래스터 캐시 적용

NCAM (National Center for Agro-Meteorology, 국가농림기상센터)은 기상청으로부터 다양한 형식, 영역, 공간 해상도, 시간 해상도 등의 자료를 수신한다. 이러한 데이터의 다형성은 각 데이터의 특별한 용처에 맞는 적절한 형식으로서, 농업기상재해조기경보시스템 역시, 각 모듈의 입력 자료와 출력 자료를 특정 형식으로 통일하여 모듈 간의 종속성을 완화하고 데이터 변환에 대한 효율을 높일 필요가 있다. 하지만 입출력 자료를 몇 가지 정도로만 한정하게 되면, 특정 모듈을 지원할 수 없게 되는 문제가 발생할 수 있다.

다양한 형식의 데이터를 지원하기 위해 적용할 수 있는 적절한 한 가지 방법은 입출력 자료의 변화에 능동적으로 대처 할 수 있도록 실시간으로 데이터 변환 방식을 사용하는 것이다. 예를 들어 과거 하동 일부 지역에서만 수행하던 계산을 섬진강 10개 시군으로 확장하면서 처리 시스템 소프트웨어가 대폭 수정될 필요가 있었다. 이를 최소화하기 위해 계산이 필요한 공간적 영역을 특정하지 않고, 매 반복 주기 마다 처리해야 할 영역과 해상도를 실시간 판단하는 방식을 취하는 것이다. 또한 기상청에서 발표하는 원본 데이터의 변화에 보다 강인하게 대응할 수 있도록 현재 발표

하고 있는 데이터의 공간적, 시간적 속성은 영속적인 것이 아닌, 일시적인 것으로 간주하고 이에 대응할 수 있도록 하여 다양한 형식의 데이터를 지원함은 물론 데이터 처리 시스템의 안정성을 확보해야 한다.

이 방식이 갖는 단점은 과거 자료의 계산 또는 원본 자료가 갱신되지 않는 한 재처리가 불필요한 데이터를 2회 이상 반복하면서 발생하는 시스템 자원의 소모가 있을 수 있다는 점이다. 또한 멀티 프로세스 시스템에서는 동일한 작업을 여러 프로세스가 반복하는 문제가 있고, 분산 시스템에서는 여러 서버의 여러 프로세스가 동일한 작업을 비슷한 시간대에 중복 처리한다는 문제가 있다. 따라서 본 연구에서는 아래와 같은 방식의 래스터 캐시 정책을 두어, 분산 메모리 병렬 처리 시스템에서도 효율적으로 래스터 데이터를 사용할 수 있는 LRU (least recently used, 최근 최소 사용) 캐시 시스템을 구축하여 보았다

최초 N개의 프로세스가 동시에 하나의 래스터 데이터에 대한 접근을 수행한다고 가정한다. N개의 프로세스는 각각 원본 래스터 파일에 대한 캐시 파일이 존재하는지 검사하며 캐시 파일이 있는 경우 그 파일이 잠겨있는지 검사한다. 잠겨 있지 않으면 사용해도 무방하다고 판단하고 캐시 생성에 필요한 높은 비용의 연산을 건너뛰고 해당 파일을 메모리로 읽어 들일 준비를 한다(연산을 위한 오버헤드는 없지만, I/O 에서 발생하는 오버헤드가 존재한다. 이는 2.2에서 언급한다). 하지만 캐시 파일이 없다면 프로세스는 다른 프로세스가 캐시 파일을 생성하고 있는지 확인하는 절차를 거친다. 만약 다른 프로세스가 캐시 생성 작업을 하고 있다면, 해당 프로세스는 캐시를 처리 하거나, 처리해야 할 캐시가 없다는 단순이 아무것도 하지 않고 대기하는 것이 자원의 낭비를 막는 방법이다.

타 프로세스의 캐시 생성 작업 수행 여부를 확인하기 위해 파일 잠금 기법을 사용한다. 예를 들어 N개의 프로세스 중 최초로 캐시 생성 작업에 도착한 하나의 프로세스(1번 프로세스라고 명명)는 다른 프로세스들이 파일을 사용할 수 없도록 파일을 잠그고 캐시 생성 작업에 들어간다. 이 프로세스 보다 나중에 도착한 프로세스(N-1개의 프로세서들이라고 명명)들은 캐시 파일은 없지만 잠겨 있으므로 다른 누군가가 캐시 생성을 하고 있는 중이라고 판단하고 처리해야 할 다른 캐시가 있는지 찾아보거나 파일의 잠금이 해제될 때까지 허용하는 시간 동안 대기 한다. 1번 프로세스가 캐시 생성을 완료하고 파일의 잠금을 해제하면, 대기하고 있던 N-1개의 프로세스들은 일제히 생성된 캐시에 접근하여 메모리로 가져오게 된다.

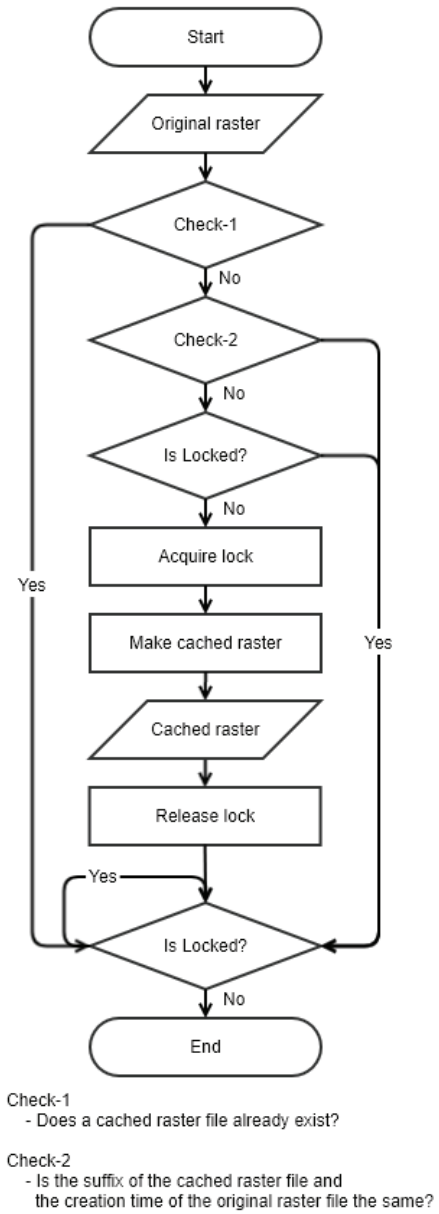

Fig. 1. Flowchart of raster cache policy.

제시된 캐시 생성 작업에 투입되지 못한 프로세스들이 대기하면서 시스템 유휴가 발생하는 것처럼 보일 수도 있고 실제로 하나의 작업만을 수행하면 그렇게 된다. 하지만 후술될 Celery를 이용한 분산 처리와 같이 하나의 작업만을 수행하는 것이 아니라 병렬로 처

리될 수 있는 여러 작업(예를 들어 한낮기온과 아침기온, 강수량은 종속관계 없이 실행될 수 있음)이 동시에 실행되므로 이러한 문제점을 줄일 수 있도록 하였다. 또한 생성된 래스터 캐시는 원본 자료의 '파일 생성 시각'이 변경되지 않는 한 향후 지속적으로 사용할 수 있으므로, 파일 잠금으로 인한 시스템 유휴 시간은 크지 않을 것으로 보인다.

2.2. NFS 캐시 적용

농업기상재해조기경보 현업 및 개발 시스템은 2대의 고성능 서버(GIS 서버, 계산 서버), 2대의 준고성능 서버(Web 서버, 개발 서버)가 1GBE (Gigabit Ethernet) Switch를 중심으로 연결되어있다. 또한 1대의 고용량 스토리지가 GIS 서버에 DAS (Direct-attached storage)로 연결되어 다른 서버들이 NFS를 이용하여 접근하도록 구성되어 있다. 4대의 각 서버는 명백히 구분된 역할이 있고 각자의 역할만 수행했었으나 처리해야 할 데이터의 양이 증가함에 따라 기후 상세화, 생물 계절 예측, 기상 재해 추정과 같은 연산을 분업하여 처리해야 할 필요가 발생하였다.

농업기상재해조기경보시스템의 연산 작업은 특정 시각에 주기적으로 수행된다. 이때 4대의 각 서버들이 한 곳에 집중된 공유 자원에 동시 접근하고, GIS서버를 제외한 3대의 서버들이 1GBE의 대역폭을 짧은 시간 동안 모두 채우게 된다. 이때 데이터 처리의 속도는 각 서버의 연산 능력에 관계없이 1GBE의 대역폭의 이하로 제약되는 문제가 발생하였다. 따라서 한 곳에 집중된 데이터 입력의 부하를 연산을 담당하는 각 서버에 분산함으로써 데이터 입력에 소모되는 지연을 감소시킬 수 있는 기법을 적용하기로 하였다.

대용량 스토리지 구성을 위해 널리 사용되고 있는 분산 파일 시스템(Distributed File System, DFS)인 Ceps FS, Lustre FS, Gluster FS등을 고려해 보았으나, Fig. 2와 같이 하나의 큰 스토리즈를 공유하는 현재 구축되어 있는 시스템에는 도입 대비 얻을 수 있는 효과가 크지 않을 것으로 판단되었다. 따라서 NFS를 통해 이동해야 할 데이터를 각 서버의 캐시 디스크에 직업 파일 형식으로 저장하는 방식으로 공유 자원 접

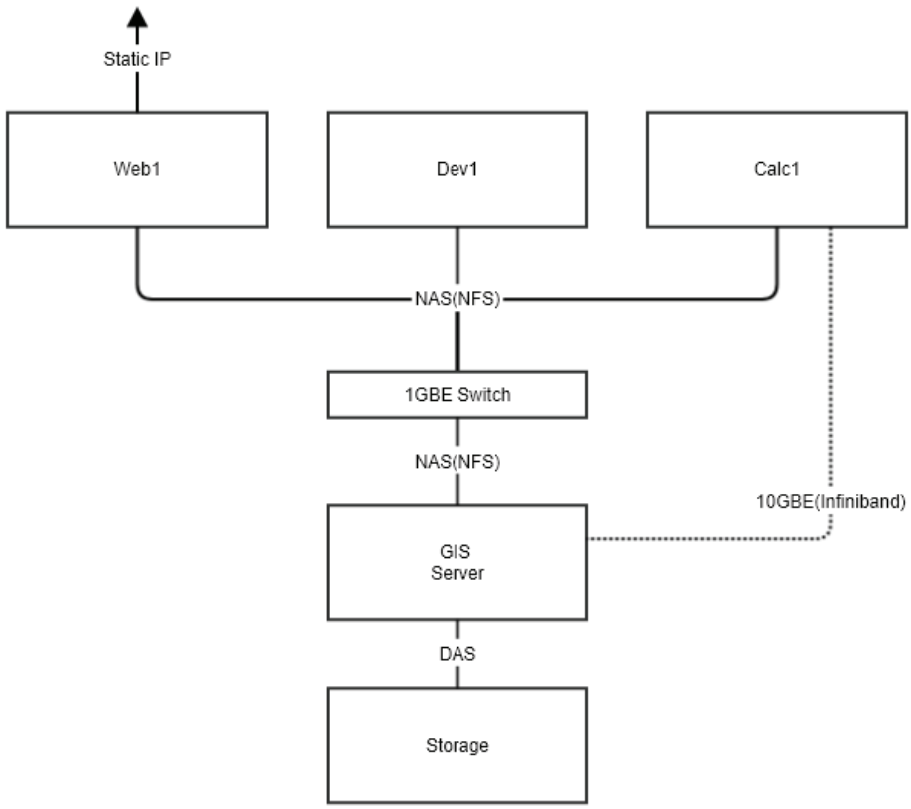

Fig. 2. Basic system architecture with 4 servers.

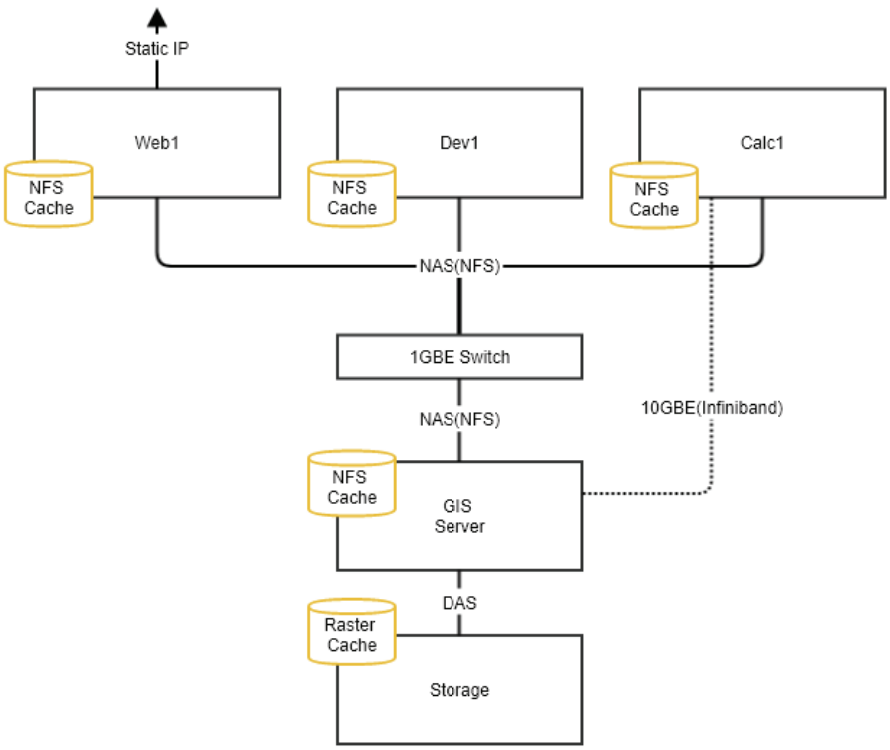

Fig. 3. Enhanced server system architecture by applying local/remote cache.

근의 경쟁 문제와 1GBE의 대역폭 한계에 대응하기로 하였다.

GIS 서버에 연결된 스토리지는 2.1에 기술된 바와 같은 '래스터 캐시'를 갖는다. 이곳은 모든 캐시를 담고 있는 하나의 대용량 캐시 저장소이며 최소의 횟수로 접근하는 것이 성능상 유리한 가장 느린 캐시이다. 또한 각 서버는 로컬에 각자의 NFS 캐시를 갖는다. 이 캐시는 NFS 서버의 스토리지 서버가 가지고 있는 전체 캐시의 일부 복제본이다. 해당 서버 스스로가 생성한 캐시일 수도 있고(이 경우는 생성한 캐시를 로컬 캐시에 저장한 후 NFS 캐시 서버로 전송한 경우) 다른 서버가 생성한 캐시를 복제해온 경우일 수도 있다.

모형을 위한 입력자료를 준비할 때 각 프로세스는 우선 로컬 캐시의 존재 여부를 검사한다. 로컬 캐시가 존재하고 파일이 잠겨있지 않을 경우 해당 파일을 메모리로 읽어 들여 모형의 입력자료로 사용한다. 1차적으로 검사한 로컬 캐시가 없으면 NFS를 통해 원격의 캐시의 존재 여부를 검사한다. NFS 서버에 캐시가 존재하면 단순히 해당 캐시를 로컬로 복제해 오는 것으로 끝나지만, NFS 서버에도 캐시가 없으면 앞서 기술한

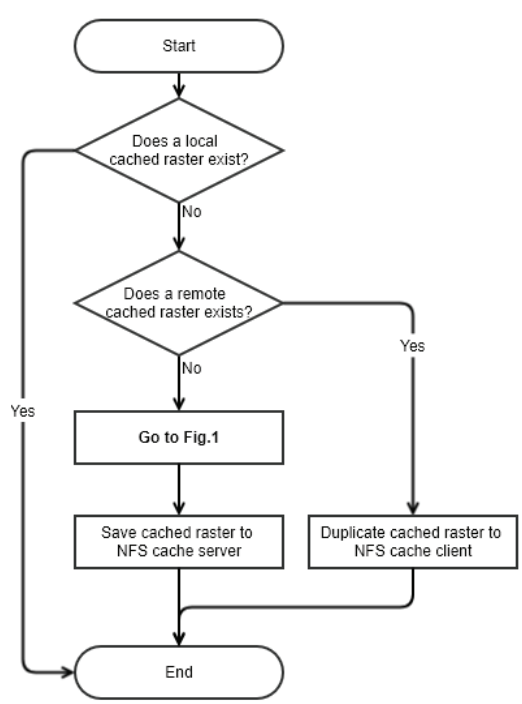

Fig. 4. Flowchart of local/remote cache policy.

Fig. 1과 같은 작업을 거쳐 캐시를 생성하고 로컬에 캐시를 저장한 후 서버에 이를 전송하여 다른 서버가 이를 이용할 수 있도록 한다. 캐시와 원본의 일치 문제는 고비용의 일치 검사 알고리즘 대신 캐시 파일 이름의 접미사와 원본 파일의 생성 시각 일치 여부를 이용하여 빠르게 결정할 수 있도록 하였다.

2.3. 분산 처리 적용

분산 처리(Distributed Computing)는 네트워크 상에 분산된 컴퓨팅 처리 자원을 이용하여 하나의 처리 장치에서 수행하기 어려운 거대 문제를 해결하고자 하는 컴퓨팅 모델이다. 앞서 기술된 바와 같이 4대의 준고성능 서버가 하나의 네트워크상에 연결되어 있으며 각 서버는 최소 8개의 작업을 동시에 전개할 수 있는 능력을 가지고 있다.

분산 처리를 도와주는 여러가지 라이브러리 중 본 연구에서는 기존부터 현업 서비스에 사용되어 온 Celery라는 Python 기반의 라이브러리를 사용하였다. Celery는 기본적으로 RabbitMQ라는 메시지 큐를 사용하며, 이 메시지 큐에 쌓여있는 작업들이 Celery에 의해 생성된 독립적인 프로세스(worker)들에 의해 동시에 처리되면서 높은 성능과 가용성을 제공하게 된다. 또한 한 개 또는 그 이상의 작업이 완료되었을 때 다음 작업을 수행하는 동기 작업을 수행할 수 있어 다양한 응용의 지원이 가능하다.

작업 큐(또는 메시지 큐)에 들어 있는 모든 작업들이 동시에 처리될 수 있다면 가장 높을 효율을 내겠지만 특정 자료는 다른 자료에 종속적일 수도 있고(Fig. 5에서 한낮 기온은 하늘 상태에 종속적임), 자기 자신에 대해 종속적일 수도 있다(Fig. 5에서 아침 기온은 어제의 아침기온에 대해 종속적임). 이러한 관계는 동시에 처리될 수 없는 선행과 후행이 명확히 구분된 순차 작업이 되어야 하므로 처리 속도에 큰 영향을 미친다.

조기경보시스템에서 서로 독립적으로 동시에 처리될 수 있는 작업(Task)을 그룹(Group)으로 묶어 보면 도합 7개의 그룹이 순차적으로 처리되어야 함을 볼 수 있다(Table 1). 각 작업에 대한 괄호 안의 숫자는 동시에 처리될 수 있는 예보 기간(일 수)을 뜻한다. 또한 Number of total tasks 열은 동시에 처리될 수 있는 작업의 수를 뜻한다. 예를 들어 Group Index 4에서 Number of total tasks의 값이 69인 것을 볼 수 있는데, 이는 3일 치의 Evapotranspiration, 3일 치의 Sunscald, 9일 치의 Sugiura × 2(아침, 한낮 기온에 대해), 9일 치의 GDD × 5(1월 1일 기준온도 0, 5, 10, 12 도 및 10월 1일 기준온도 5도에 대해), 즉 3 + 3 + 9 × 2 + 9 × 5 도합 69개의 작업이 있고 동시에 처리될 수 있음을 나타낸다. 각 그룹의 인덱스는 순차적으로 실행되어야 하는 순서를 나타낸다. 즉 Index 1의 작업이 완료되지 못하면 Index 2 뿐만아니라 Index 7 까지의 모든 작업이 완료될 수 없다. 따라서

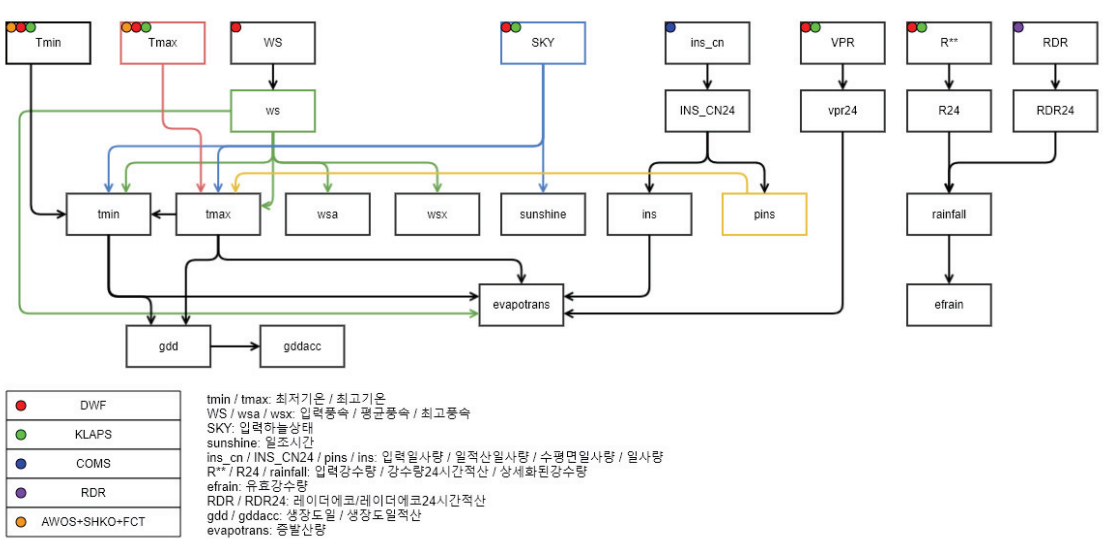

Fig. 5. Weather data generation dependency diagram.

Table 1. List of 7 sequential Groups (each group has multiple tasks that can be processed simultaneously)

Group Index	Category	Task and forecasting days	Number of total tasks
1	Weather	Wind(+3), Rain(+3), Insolation(+1), Insolation-forecasting(+2)	9
2	Weather	Base-temperature(+9), Sunshine hour(+3)	12
3	Weather	Maximum temperature(+9), Minimum temperature(+9), Effective rain(+3)	21
4	Weather	Evapotranspiration(+3), GDD(+9), Sunscald(+3), Sugiura(+9)	69
5	Weather	GDD accumulation(+9), Effective rain accumulation(+3), Evapotranspiration accumulation(+3), Sunshine hour accumulation(+3)	51
6	Plant	Growth stage(+3～+9)	100 or more
7	Risk	Risk & Risk index(+3～+9)	100 or more

각 그룹의 처리 속도 향상이 최종 적인 시스템 속도 향상으로 직접 이어진다. 반면 그룹 안의 특정 작업의 속도 향상이 전체 시스템의 속도 향상에 전혀 영향을 주지 못하는 경우도 있다. 예를 들어 그룹 Index 1 포함되어 있는 Wind, Rain, Insolation 등의 작업에서 Wind(+3)을 수행하는데 가장 오랜 시간인 30분이 소요된다고 했을 때, Rain(+3)과 Insolation(+1)을 아무리 개선시킨다 해도, Wind(+3)의 소요시간에는 변화가 없으므로 전체 시스템의 성능 향상은 없다. 따라서 처리 속도가 문제되어 개선 작업을 수행하고자 한다면 동시에 수행되는 다른 작업들도 함께 고려해 보아야 한다. 또한 동시에 처리될 수 있으며 수행시간이 비슷한 작업들끼리 그룹화하는 것이 일반적으로 성능 향상에 도움이 된다.

2.4. GPU 병렬 처리

어떤 작업을 병렬화하여 얻을 수 있는 처리 속도의 향상은 그 모델에서 병렬화가 가능한 부분이 차지하는 비율에 따라 크게 좌우된다(Amdahl, 1967). 전체 수행시간 대비 병렬화가 가능한 부분이 20%를 차지한다고 가정하면, 이 부분을 무한대로 빠르게 해도, 최대 1.25배 이상의 속도 향상은 이룰 수 없다. 반면에 병렬화 가능 부분이 99%라면, 이 부분을 2배 빠르게 했을 때 얻을 수 있는 속도 향상은 최대 약 1.98배가 된다.

조기경보시스템의 다양한 모듈들 중 특정 알고리즘이 적용된 모듈과 데이터의 입출력 부분을 제외하면, 대부분의 래스터 연산들이 격자마다 독립적으로 처리될 수 있는 경우가 많다. 이는 GPU를 이용한 데이터 처리에 적합한 특징이다. 하지만 각 격자의 처리 과정

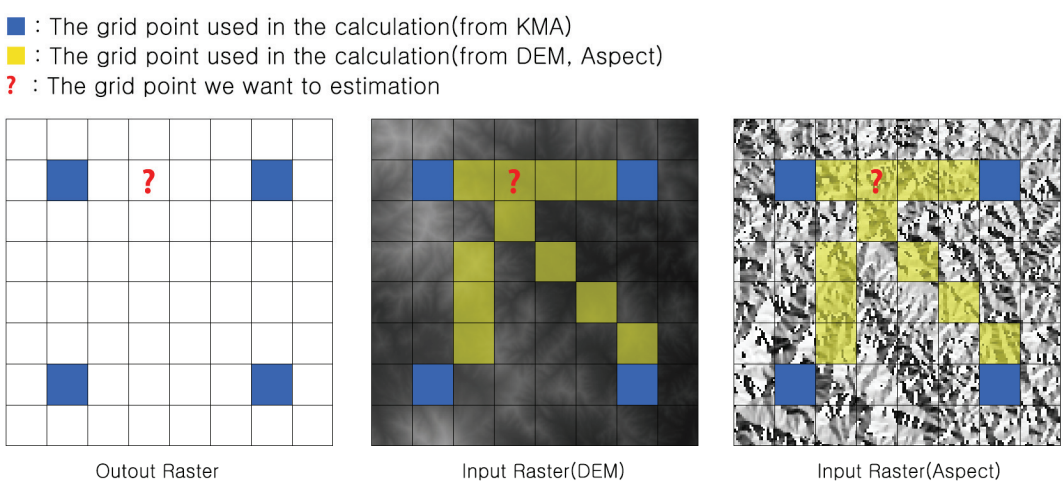

Fig. 6. Grids used to estimate precipitation value for one grid.

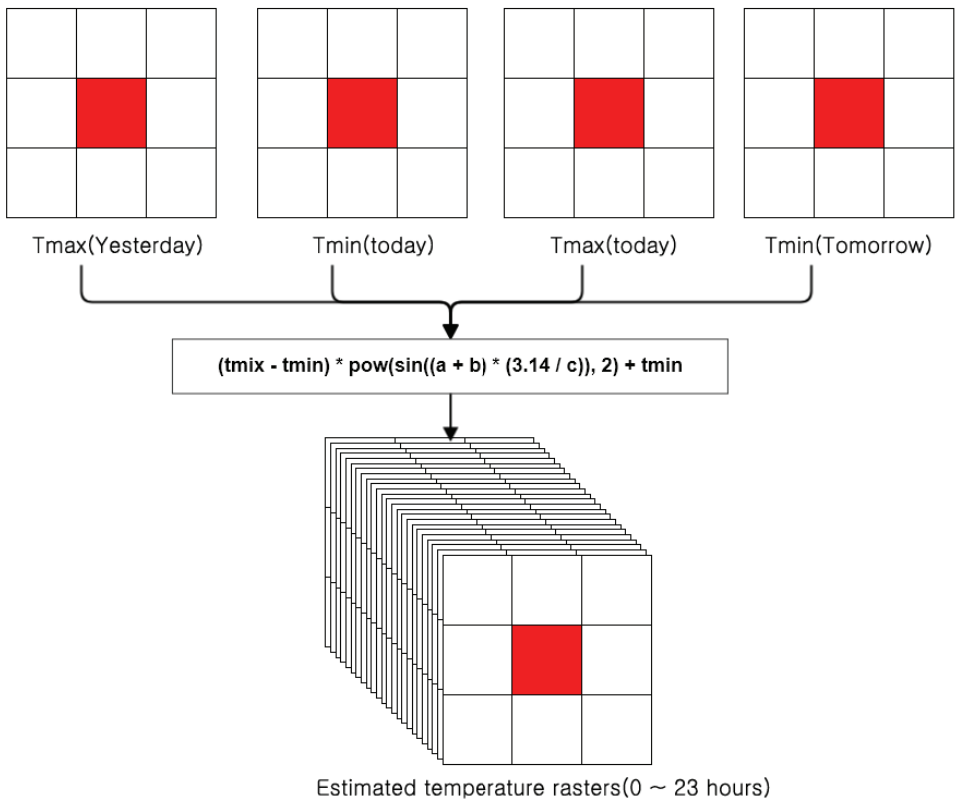

Fig. 7. Grids used to estimate 1-hourly temperature value for 24 grids at same coordinate.

이 비교적 단순하면서도 적은 개수의 연산으로 구성되어 있다면 전체 계산 시간 중 연산에 투입되는 시간 대비 병렬화 준비 작업에 소모하는 시간 비율이 증가하게 되어 큰 성능 향상이 없을 가능성이 높아진다.

Fig. 6은 특정 격자 지점의 강수량을 계산하기 위해 사용되는 격자점들을 시각화한 것이다. 적색 물음표로 표시된 격자와 같이 관측 값이 없는 격자의 값을 결정하고자 할 때, 주변의 관측 값이 있는 격자들 사이에 있는 모든 격자들이(Fig. 6에서의 청색 및 황색 격자들) 이용되며 서로의 계산 결과는 다른 격자에 영향을 주지 않는 높은 병렬성을 가지고 있다.

Fig. 7은 어제의 최고기온, 오늘의 최저기온 및 최고기온 그리고 내일 아침기온에 해당하는 4개의 입력을 이용하여 각 시각에 대한 24개의 기온값을 출력으로 생성하는 Sugiura 기법이 특정 격자에 대해 어떻게 처리되는지 시각화한 것이다. PRISM 강수와 같이 각 격자별 독립적 연산이 가능한 높은 병렬성을 가지고 있는 알고리즘이지만 연산량 대비(6개의 사칙연산 1개의 삼각함수 연산, 1개의 지수 연산) 출력 자료의 개수(24개)가 많기 때문에 성능 향상 폭이 상대적으로 낮다.

IV. 결 론

3.1. 래스터 캐시

래스터 캐시는 다양한 종류, 영역, 해상도의 입력 자료를 데이처 처리 시점에 동적으로 지원하고 독립된 프로세스의 중복 연산을 줄임으로써 결과적으로는 데이터 처리 속도를 일정하게 유지하면서도 다양한 형식의 데이터를 지원하는 것을 목표로 하였다. 그 결과 래스터 캐시를 사용하는 것은 사용하지 않는 것 대비 데이터의 처리 속도가 최대 3배 향상됨을 볼 수 있었다. 이는 이미 생성된 캐시를 단순히 읽기만 함으로서 캐시 생성에 소요되는 시간을 제거하는 것일 뿐만 아니라 여러 개의 각 프로세스는 각자의 캐시를 생성할 뿐이지만 시스템 전체로서는 모든 캐시를 함께 생성하

면서 성능이 향상된 것이다.

Table 2. The best raster cache performance (NFS cache enabled, Distribution processing enabled)

Category	Time (s)	Speedup
Weather	1408	3.2x
Growth stage	1594	2.8x
Risk & Risk index	1793	3.3x

3.2. NFS 캐시

NFS 캐시는 중앙 집중화된 NFS 서버의 Disk I/O에서 발생하는 지연과, 1GBE에 제한된 현재의 조기경보 시스템의 네트워크 한계를 극복하기 위해 도입하였다. 그 결과 NFS 캐시를 사용함으로써 처리 속도가 최대 3배 향상됨을 확인할 수 있었다. 다만 각 서버는 평균적으로 1년치 데이터를 캐시하기 위한 400GB 정도의 스토리지를 추가적으로 소모하므로, 안정적인 운영을 위해서는 약 512GB 이상의 캐시를 할당하는 유리할 것으로 보인다. 현재의 구현은 캐시가 디스크의 95%를 차지하면 가장 오래된 캐시를 삭제하는 작업을 수행하게 되는데, 만약 캐시에 할당된 디스크의 용량이 크지 않을 경우 오래된 캐시를 제거하는 작업이 빈번하게 발생할 수 있으므로 적절한 용량을 경험적으로 선정할 것이 요구된다.

Table 3. The best NFS cache performance (raster cache enabled, Distribution processing enabled)

Category	Time (s)	Speedup
Weather	1432	2.2x
Growth stage	1525	1.8x
Risk & Risk index	1821	3.3x

3.3. 분산 처리

분산 처리는 다수의 코어를 가진 하나의 서버뿐만 네트워크로 연결된 여러 서버를 이용하여 처리 해야 할 복수 개의 작업을 분산시켜 최종적으로 수행 시간을 단축 시키는데 목표를 두었다. 조기경보 시스템을 위해 운영되고 있는 4개의 모든 서버를 이용한 것과 1개의 GIS 서버를 이용한 것을 비교하는 것으로 성능을 측정하였다. 그 결과 래스터 캐시(2.1)와 NFS 캐시(2.2)를 활성화한 상태에서는 최대 2배의 성능 향상을 확인할 수 있었다. 성능 향상 폭이 투입된 서버의 개수 대비 높지 않은 이유는 기준이 되는 GIS 서버 대비 나머지 서버들의 각 성능이 낮기 때문인 것으로 추측된다.

Table 4. The best distributed processing performance (raster cache enabled, NFS cache enabled)

Category	Time (s)	Speedup
Weather	1699	1.7x
Growth stage	1676	1.7x
Risk & Risk index	1204	2.4x

3.4. GPU 병렬 처리

GPU를 이용한 병렬 처리는 하나의 래스터를 고속으로 처리하기 위해 그리고 GPU가 래스터를 처리하는 동안 CPU를 다른 작업에 이용할 수 있도록 하여 시스템의 성능을 향상 시키는 데 그 목적을 두었다. 조기경보 시스템은 별도의 GPU 하드웨어를 탑재되어 있지 않으며, CPU 또한 OpenCL™ (Open computing language)을 지원하지 않는 모델이기 때문에 추가 적인 자원 투입 없이 적용할 수 있는 기법이 아니다. 또한 본 연구에서는 모든 조기경보 모형에 대해 적용해 보지 못하였고 가장 높은 성능 향상이 있을 것이라고 추측되는 PRISM 강수(Kim and Yun, 2013)와 성능 향상이 평균적일 것이라고 추측되는 기온의 시간 분해 모델인 Sugiura 알고리즘(Sugiura, 1999)에 대해서만 적용해 보는 것으로 가능성을 확인해 보았다. 그 결과, Kim and Yun(2013) 기법을 이용한 PRISM 강수의 경우 평균 1,000배 이상의 속도향상이 있었고, Sugiura의 경우 평균 6배의 성능 향상이 있었다. CPU 구현의 경우 OpenMP와 같은 병렬화 모델을 사용하지 않았는데, 그 이유는 2.2에 제시된 바와 같이 Celery를 이용한 다중 프로세스 기법으로 다수의 CPU 코어가 이미 이용되고 있을 거라는 가정을 기반 했기 때문이다(30m

Table 5. The average GPU performance about two models (except disk I/O)

Model name	Time (s)	Speedup
PRISM rainfall (270m, Seomjin river watershed)	0.53	1000x
PRISM rainfall(30m, South korea)	332	?
Sugiura temperature	40	6.7x

해상도 전국 강수의 경우는 CPU로의 성능 측정에 실패하여 GPU로 측정된 수행 시간만을 제시한다).

Table 6. CPU and GPU system specification

	CPU	GPU
Manufacturer	Intel	NVIDIA
Model	Xeon E5-2650 v2	Tesla P100
RAM	64G	16G
Price	$1453	$4500

비교에 사용된 CPU는 현재 조기경보 시스템에서 가장 높은 성능인 GIS 서버에 탑재된 Intel xeon 프로세서가 이용되었고, GPU는 NVIDIA Tesla P100이 이용되었다. Table 6과 같이 성능 측정에 사용된 GPU가 CPU 대비 매우 높은 가격임을 볼 수 있는데 이는 배정밀도 연산 능력, 메모리 오류 정정 기능, NVLink™ 등의 기능에 기인하는 것으로 필수적인 요소는 아니다. 하지만 30미터 해상도의 전국 데이터가 메모리에 한번에 적재될 수 있는 12GB 이상의 메모리를 가진 모델을 사용하는 것이 현재 성능 측정을 위해 제작된 프로그램이 갖는 제약이다.

적 요

데이터 처리 속도는 예보 능력과 관련이 있다. 최신의 입력 자료를 이용한 예측 데이터의 고속 생산은 신속한 대처를 가능하게 한다. 또한 알고리즘 작성, 계산, 결과 평가, 알고리즘 개선으로 이어지는 순환 구조를 원활하게 할 뿐만 아니라 오류 발생시 빠른 시간 내에 복구할 수 있게 하는 등 매우 중요한 요소이다. 현재의 조기경보 시스템은 매 계산 주기 마다 섬진강 유역의 10개 시군에 대해 30미터 해상도의 격자형 자료를 400개 이상 생성하고 있으며(중간 데이터 포함) 최대 9일까지 예보되는 자료를 포함할 경우 600개 이상이다. 이는 전국을 30미터 해상도로 약 45개를 생성하는 계산양과 비슷하다. 또한 14,000여개의 필지에 대한 구역 통계와, 각 래스터의 평균, 최대, 최소 등의 통계 자료 생성도 함께 수행 해야 한다. 이와 같은 대량의 데이터를 한정된 시간 내로 처리하기 위한 몇 가지 기법을 적용하여 적용하였으며, 아직 적용은 못하였으나 가능성의 여부를 평가해 보는 것으로 본 연구를 진행하였다. 그 결과 앞서 제시된 래스터 캐시, NFS 캐시, 분산 처리를 모두 적용할 경우 데이터 처리 시간을 1/8로 단축 시킬 수 있음이 확인되었다. 또한 GPU를 이용한 연산을 적용할 경우 일부 모듈에 대해 매우 큰 폭으로 수행 시간을 단축 시킬 수 있음을 확인하였다. 다만 캐시를 위한 추가적인 디스크, GPU라는 별도의 하드웨어, 추가된 하드웨어 지원을 위한 고출력 전원 장치와 이에 따른 UPS (Uninterruptible power supply, 무정전 전원공급 장치)까지 상대적으로 높은 사양으로 준비해야 하는 비용적인 문제가 발생할 수 있다. 본 연구에서 제시한 네 가지 기법 중 세 가지는 계산 서버 추가를 통한 수평적 성능 확장에 관한 것이다. 하지만 서버의 추가가 처리 속도 향상으로 이어지지 않음은 물론 오히려 저하시키는 경우가 있다. 본 연구에서는 특정 시간 내로 작업을 완료 시키지 못하면 해당 작업을 반환하여 다른 서버가 처리하는 간단한 방식을 이용한다. 하지만 이런 문제를 지속적으로 발생시키는 계산 서버가 발견된다면 정해진 기준에 따라 계산 작업에서 완전히 퇴출 시켜야 성능 향상에 도움이 된다. 따라서 처리 속도에 대한 정확한 원인을 검사하고 이를 실시간으로 반영할 수 있는 기법이 필요하다.

감사의 글

본 논문은 농촌진흥청 국립농업과학원 농업과학기술 연구개발사업 (과제번호: PJ010007062017)의 지원에 의해 이루어진 것임.

REFERENCES

Ahn, J. B., J. N. Hur, and A. Y. Lim, 2014: Estimation of fine-scale daily temperature with 30 m-resolution using PRISM. *Atmosphere* **24**(1), 101-110.

Amdahl, G. M., 1967: Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings* **30**, 483-485.

Chung, U., K. Yun, K. S. Cho, J. H. Yi, and J. I. Yun, 2009: The PRISM-based rainfall mapping at an enhanced gridcell resolution in complex terrain. *Korean Journal of Agricultural and Forest Meteorology* **11**, 72-78.

Kim, C. Y., Y. C. Kim, Y. C. Kim, S. M. Lee, Y. K. Kim, and D. W. Seo, 2014: Performance enhancement of distributed file system as virtual desktop storage using client side SSD cache. *Computer and Communication*

System **3**(12), 433-442.

Kim, D. J., and J. I. Yun, 2013: Improving usage of the Korea meteorological administration's digital forecasts in agriculture: 2. Refining the distribution of precipitation amount. *Korean Journal of Agricultural and Forest Meteorology* **15**(3), 171-177.

Mielikainen, J., B. Huang, and H. L. A. Huang, 2013: Speeding up the computation of WRF double-moment 6-class microphysics scheme with GPU. *Journal of Atmospheric and Oceanic Technology* **30**(12), 2896-2906.

Ozimek, I., A. Hrovat, A. Vilhar, and T. Javornik, 2015: GPU computation acceleration of GRASS GIS modules for predicting radio-propagation. *International Journal of Communication* **9**, 76-85.

Rodgers, D. P., 1985: Improvements in multiprocessor system design. *ACM SIGARCH Computer Architecture News archive* **13**(3), 225−231.

Sugiura, T., 1999: Prediction of full bloom date of pear using air temperature. *Agriculture and Horticulture* **54**(10), 146-149.

Zhang, J., S. You, and L. Gruenwald, 2015: Efficient parallel zonal statistics on large-scale global biodiversity data on GPUs. *Proceedings of the 4th International ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, 35-44.