韓國水資源學會論文集
第47卷 第7號·2014年 7月
pp. 599~614

J. KOREA WATER RESOURCES ASSOCIATION
Vol. 47, No. 7:599-614, July 2014
http://dx.doi.org/10.3741/JKWRA.2014.47.7.599
pISSN 1226-6280 • eISSN 2287-6138

MPI를 이용한 2차원 유한체적모형의 계산 성능 개선

Performance Improvement of Computing Time of 2 Dimensional Finite Volume Model using MPI

김 태 형* / 한 건 연** / 김 병 현***

Kim, Tae Hyung / Han, Kun Yeun / Kim, Byung Hyun

Abstract

In this study, two dimensional finite volume model was parallelized to improve computing time, which has been developed to be able to apply for the mixed meshes of triangle and quadrilateral. MPI scheme which is free from limitation of the number of cores was applied, and non-blocking point-to-point communication was used for fluxes and time steps calculation domain. The developed model is applied to analyze dam break in a L-shaped experimental channel with 90° bend and Malpasset dam breach event to calibrate the consistency between parallelized model and existing model and examine the speed-up and efficiency of computing time. Computational speed-up about the size of the input data was considered by simulating 4 cases classified by the number of meshes, Consequently, the simulation results reached a satisfactory accuracy compared to measured data and the results from existing model, and achieved more than 3 times benefit of computational speed-up against computing time of existing model. Simulation results of 3 cases classified by the size of input data lead us to the conclusion that it is important to use proper size of input data and the number of process in order to minimize the communication overhead.

keywords : 2D finite volume model, parallel programming, MPI, computing time, speed-up

요 지

본 연구에서는 삼각형 및 사각형 혼합격자의 적용이 가능하도록 기 개발된 2차원 유한체적모형의 계산속도를 개선하기 위해 모형의 병렬화를 수행하였다. 모형의 병렬화를 위해 코어 수의 제약에 자유로운 MPI 기법을 이용하였고, 프로그램 내의 흐름률 및 계산시간간격의 계산영역에 대해 논블록킹 점대점통신을 이용하였다. 병렬화 된 개발모형의 기존모형에 대한 계산결과의 일치성을 검증하고, 계산시간에 대한 성능향상도와 효율성을 검토하기 위해, 90°의 만곡이 존재하는 L자형 실험하도에 대한 댐 붕괴해석과 자연하천인 Malpasset 댐의 붕괴사상에 대해 모형을 적용하였다. 또한 격자수에 따라 4개의 Case로 구분하여 각각 모의함으로써, 입력규모의 크기에 따른 계산시간의 성능향상도를 함께 검토하였다. 분석결과 병렬화 모형에 의한 모의 결과는 기존모형 및 실측치와 비교하여 만족할 만한 정확도를 확보하였고, 기존모형에 대비해 약 3배 정도의 계산시간에 대한 성능이득을 얻을 수 있었다. 또한 입력자료 규모에 대한 Case별 모의 결과를 통해 적절한 입력자료의 규모와 프로세스 개수를 사용하는 것이 통신부하를 최소화할 수 있는 방안임을 확인할 수 있었다.

핵심용어 : 2차원 유한체적모형, 병렬프로그래밍, MPI, 계산시간, 성능향상도

* 경북대학교 건설·환경·에너지공학부 박사후연구원 (e-mail: sunz3515@hotmail.com)
Post-Doc., School of Architectural, Civil, Environmental and Energy Engineering, Kyungpook National Univ., Daegu 702-701, Korea
** 경북대학교 건설·환경·에너지공학부 교수 (e-mail: kshanj@knu.ac.kr)
Professor, School of Architectural, Civil, Environmental and Energy Engineering, Kyungpook National Univ., Daegu 702-701, Korea
*** 교신저자, 캘리포니아 주립대학교 얼바인, 토목, 환경공학과 및 UC 수문모델링 센터, 박사후연구원 (e-mail: bh.km@uci.edu, Tel: 82-16-642-4639)
Corresponding Author, Post-Doc., Dept. of Civil & Environmental Engineering, University of California, Irvine and UC Center for Hydrologic Modeling, CA, 92697, United States

1. 서 론

수공구조물 붕괴 등에 따른 2차원 침수해석을 위한 전산학적 접근은 프로그래밍 언어의 발전과 함께 20세기 후반들어 빠른 속도로 성장하기 시작하였다. 그 중 흐름의 전파양상을 정확하게 반영할 수 있는 상류이송기법인 Godunov 형태의 유한체적기법은 충격파와 같은 불연속적인 해를 가지는 문제의 정확한 해석능력과 비구조적 격자 사용의 용이성 등의 장점 때문에 다양한 수치기법들에 의해 연구 및 개발되어 왔다(Alcrudo and Garcia-Navarro, 1993; Chippada et al., 1998; Caleffi et al., 2003; Begnudelli et al., 2008). 하지만 이러한 기법은 양해법을 근간으로 하는 해석 기법으로써, 격자의 크기, 격자망의 구성, 그리고 계산 시간의 간격 등 모형의 구동에 있어 엄격한 제한이 필요하다. 특히 방대한 계산시간을 요구하는 기법의 약점은 홍수 예·경보 등을 위한 실시간 해석에 있어서 큰 제약이 되어 왔다.

이러한 계산시간에 대한 문제는 CPU의 성능이 지속적으로 발전하면서 점차 자연적으로 극복되는 듯하였으나 막대한 전력소모와 발열에 대한 문제로 인해 CPU 자체의 성능개발에 제약이 따르면서 더 이상의 CPU의 성능 개선에 의한 계산속도의 향상은 한계에 다다르게 되었다. 이에 대한 대안으로 2000년대 초부터 출시된 멀티코어 프로세서는 코어 수를 늘리는 방법으로 발열과 전력을 분산시킴으로써 컴퓨터 사용자의 멀티태스킹이 원활하도록 하였다(Wikipedia, 2014). 현재까지 2차원 홍수파 전파 해석을 위해 연구되고 개발되어 온 다양한 모형들은 특별한 처리 없이는 단일 코어만을 사용하여 계산하기 때문에 멀티코어의 장점을 전혀 이용할 수 없다. 이러한 단점을 극복하기 위해 프로그램을 병렬화함으로써 멀티코어를 활용할 수만 있다면 계산시간 단축에 큰 효과를 거둘 수 있을 것이다.

IT 분야에서 프로그램을 병렬화하기 위해 다양하게 개발된 도구들 중 OpenMP와 MPI는 정보통신 분야를 제외한 다른 공학분야에서 가장 많이 적용되고 있는 병렬화 기법들 중 하나이며, C, C++, Java, Fortran 등의 다양한 프로그래밍 언어에 대해 지원하고 있다. 홍수흐름해석, 강우-유출 해석, 침수해석 등 수자원공학의 여러 분야에 대해서도 OpenMP와 MPI 기법을 활용한 프로그램의 병렬화가 다양하게 시도되어 왔다. 국외의 연구 중 OpenMP 기법을 적용한 경우는 Neal et al. (2009)과 Paz et al. (2011) 등이 있다. Neal et al. (2009)은 1차원 운동파방정식(kinematic wave equation)과 래스터 기반의 저류방정식을 하도와 홍수터에 각각 적용하여 연계한 2차원 홍수해석 모형을 OpenMP를 이용하여 병렬화 한 후 프로그램 내의 각 서브루틴들의 성능개선정도를 시험하였고, Paz et al. (2011)은 하도와 홍수터에 대해 각각 1차원 동수역학(hydrodynamic) 모형과 2차원 래스터기반의 저류방정식에 의한 모형을 연계하여 병렬화한 후 실제하도에 적용함으로써 코어 개수에 따른 성능향상도를 시험하였다. MPI 기법을 이용한 병렬화는 OpenMP기법보다 최근 들어 더욱 다양하게 시도되었다(Pau and Sanders, 2006; Sanders et al., 2010; Yu, 2010; Wang et al., 2011). Pau and Sanders (2006)는 수심평균된 동수역학 방정식의 해를 찾기 위한 양해적 유한체적모형을 MPI기법으로 병렬화 한 후 동기화 과정을 생략한 경우와 논블록킹 송수신을 이용한 경우로 나누어 각각에 대한 성능향상도를 시험하였고, Wang et al. (2011)은 특정 지점에서의 홍수예경보를 위해 유전자 알고리듬을 적용한 데이터 기반 모형을 MPI 기법을 이용하여 병렬화 하였다. 한편, Neal et al. (2010)은 실제하도에 적용 가능한 2차원 홍수범람 해석 모형에 대해 OpenMP, MPI 기법 등으로 병렬화하여 기법별 성능을 비교하기도 하였다.

2차원 범람모의를 위한 홍수파 전파해석에 대해 MPI 기법을 활용한 예로써, Yu (2010)는 확산파에 기초한 2차원 홍수범람 모형에 대해 도메인 분할을 이용하여 MPI 기법을 이용하여 병렬화하였고, Sanders et al. (2010)은 2차원 천수방정식을 해석하기 위해 비구조적 격자를 적용한 Godunov형 유한체적모형을 MPI 기반으로 병렬화하여 실제하도에 적용함으로써 다양한 코어 개수에 대한 성능향상도를 시험하였으나, 도메인을 직접 분할함으로써 서브 도메인간의 데이터 교환에 신경을 써야 하는 문제를 드러냈다.

수자원 분야에 있어서 병렬화에 대한 국내의 연구사례는 양적으로 부족한 편이다. Park et al. (2003)은 유전자 알고리듬과 MPI기법을 이용한 병렬처리를 통해 해안지역에서 최적 지하수 양수량 또는 최적 관정위치를 평가할 수 있는 모형을 개발하여 적용한 바 있고, Jung et al. (2010)에 의해 운동역학적인 이론에 근거한 분포형 모형을 MPI 기반으로 병렬화하여 계산속도에 대한 성능향상도를 시험하는 등의 연구가 수행된 바 있으나, 2차원 홍수파 전파해석에 대한 병렬화를 위한 국내의 연구는 이루어진 바가 없다. 국외의 연구 또한 2차원 침수해석을 위한 기법의 지배방정식이 단순하여 정확도 및 적용성에 문제를 나타낼 것으로 사료되고, 도메인 분할 방법에 따라 계산 속도의 성능향상이 달라지는 등 여러 가지 제약이 따르는 문제를 안고 있다.

본 연구에서는 정확도와 적용성이 입증된 바 있지만 격자수와 계산시간 간격에 따라 막대한 계산시간이 소요되는 단점을 안고 있는 기 개발된 2차원 홍수파 해석 모형(Kim

et al., 2009; Kim et al., 2011)의 계산속도를 개선하고자 하였다. 코어 수의 제약에 자유로운 메시지 패싱 병렬프로그래밍 모델의 표준인 MPI (Massage Passing Interface) 기법을 이용하여 모형을 병렬화하였고, 병렬화 작업 이전에 불필요한 서브루틴을 제거하고 프로그램의 병목지점 및 데이터 의존성(Dependency)이 나타나는 부분을 제거하는 최적화 작업을 실시하였다. 병렬 처리된 모형의 기존 모형에 대한 계산결과의 일치성을 확인하기 위해 작은 규모의 입력자료(L자형 실험하도)로 코어별 계산결과에 대한 검증을 우선적으로 실시하였고, 코어 별 성능향상도도 함께 확인하였다. 또한 실제하도에 대한 적용성 및 큰 규모의 입력자료에 대한 성능향상도를 확인하기 위해 1959년 12월에 발생한 프랑스 Reyran 강 하류의 Malpasset 댐 붕괴사례에 적용함으로써 기존모형 대비 계산시간 단축의 효과를 입증하고자 하였다. 이에 더하여 격자수의 조정을 통해 다양한 크기의 입력자료로 Case를 구분하여 모의함으로써, 입력자료의 규모가 계산시간에 대한 성능향상에 미치는 영향과 가장 효율적인 성능향상을 얻기 위한 조건에 대해 검토하고자 하였다.

2. 병렬 프로그래밍

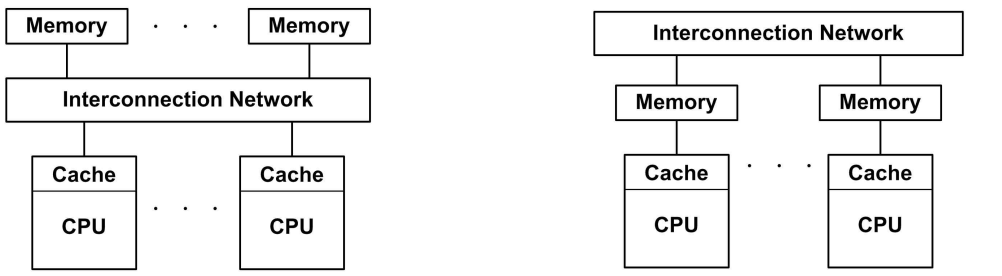
2.1 병렬프로그래밍 개요

병렬 처리란 “프로그램 내의 계산 영역을 여러 개로 나누어 각각에 대한 계산을 여러 프로세서에서 동시에 수행하는 것”을 말한다. 프로그램을 병렬화하는 주된 목적은 프로그램의 방대한 수행시간을 줄이는데 있다. 네트워크로 연결된 다수의 컴퓨터를 이용해 병렬 처리가 가능하지만, 2000년대 들어 멀티코어의 시대가 열리면서 한 대의 컴퓨터만으로도 병렬처리가 가능해 짐에 따라 계산속도의 향상 및 높은 성능향상도를 구현할 수 있게 되었다. 특히 코어의 수가 25,000여개에 이르는 슈퍼컴퓨터(Tachyon2, KISTI)를 활용하기 위해서는 병렬 처리를 위한 기술개발이 필수적이고, 현재까지 병렬 처리를 위한 다양한 프로그래밍 기법들이 개발되고 있다.

현재까지도 개발되고 있는 다양한 병렬 프로그래밍 기법들은 병렬화하여 실행하고자 하는 하드웨어 또는 시스템의 아키텍처 특성과 밀접한 관련이 있다. 병렬 시스템의 아키텍처는 메모리가 공유되어 있는지 아니면 분산되어 있는 구조인지에 따라 공유 메모리 구조(Shared-Memory Multiprocessors)와 분산 메모리 구조(Distributed-Memory Multiprocessors)로 나뉜다(Andrews, 2000). 공유 메모리 구조는 Fig. 1(a)와 같이 프로세서와 메모리가 상호연결망(Interconnection Network)에 의해 물리적으로 연결되어 있는 구조를 말하며, 상호연결망에 접속되어 있는 구조에 따라 UMA모델과 NUMA모델로 다시 구분된다. 분산 메모리 구조는 Fig. 1(b)와 같이 각각의 프로세서가 개별 메모리를 가지는 구조를 말하며, 각각의 메모리는 상호연결망에 의해 데이터를 주고받는 메시지 패싱(Massage Passing)을 지원받는다. 최근의 고성능 시스템들은 분산 메모리 시스템과 공유 메모리 시스템의 장점을 모두 가지는 분산-공유 메모리 아키텍처로 개발되고 있다(Pacheco, 1997).

2.2 병렬프로그래밍 모델

프로그램을 병렬화 하는 주된 목적은 방대한 양의 데이터의 처리속도를 단축함으로써 프로그램의 효율성을 높이는 것에 있다. 따라서 최적화된 병렬 프로그래밍 모형을 개발하기 위해서는 병렬 시스템 아키텍처에 따라 적절한 병렬처리 기법을 선택할 필요가 있다. 현재까지 개발된 병렬 프로그래밍 모델은 다양한 종류로 구분될 수 있지만 크게 두 가지 종류로 구분된다. 다중 스레드들이 하나의 주소공간을 공유하여 데이터를 병렬 처리하는 공유 메모리 병렬 프로그래밍 모델과 병렬 시스템을 구성하는 노드들이 같은 주소공간을 공유하고 있지 않아 각 프로세스 간에 서로 연결된 네트워크를 통하여 데이터를 주고받을 수 있도록 구성된 메시지 패싱 병렬프로그래밍 모델이 바로 그것이다. 또한 두



(a) Shared-Memory Multiprocessor (b) Distributed-Memory Multiprocessor

Fig. 1. Structure of Multiprocessor

가지 모델을 혼합하여 사용하는 방법인 하이브리드 병렬 프로그래밍 모델에 대한 연구도 지속적으로 진행되고 있다.

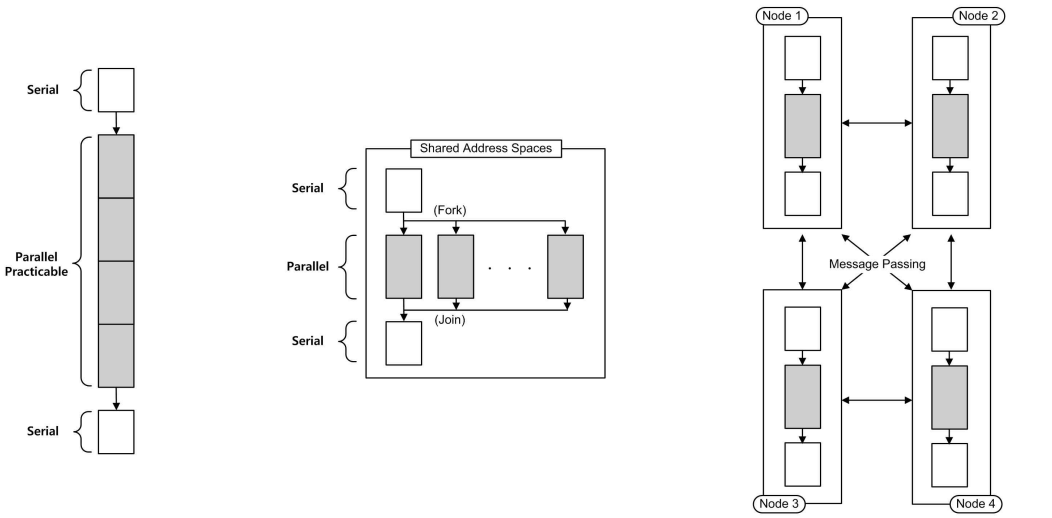
공유 메모리 병렬 프로그래밍 모델은 하나의 스레드가 아닌 다중스레드로 데이터를 분산하여 처리하는 모델로써 같은 프로세스에서 생성되는 모든 스레드는 동일한 주소 공간을 가지며 대표적인 기법으로 OpenMP가 있다. Fig. 2(a)는 단일 스레드 프로그램이 순차적으로 데이터를 처리하는 것을 보여주는 것에 반해 Fig. 2(b)는 다중 스레드 프로그램이 병렬 처리 가능 영역에서 다중 스레드를 생성(Fork)하여 병렬로 데이터를 처리한 다음 결합(Join)하여 처리하는 OpenMP의 병렬 처리 방법을 보여주고 있다. 같은 프로세스에서 생성되는 모든 스레드는 동일한 주소 공간을 사용하고 있기 때문에 다른 스레드가 갱신하는 데이터를 참조하기가 쉬운 장점이 있지만 스레드의 생성과 결합에 따른 부하가 발생하는 단점이 있다(Pacheco, 1997).

이에 반해 메시지 패싱 병렬프로그래밍 모델은 병렬 시스템을 구성하는 노드들이 동일한 주소 공간을 사용하고 있지 않아, 각 프로세스 간에 갱신되는 데이터에 접근하기 위해 네트워크를 통한 연결을 지원하도록 구성되어 있는 모델이다. Fig. 2(c)는 메시지 패싱 병렬프로그래밍의 개념도를 나타내며 각 노드별로 각각의 메모리 공간을 사용하기 때문에 다른 프로세스의 데이터에 접근하기 위해 데이터 통신(메시지 패싱)을 필요로 한다. 이러한 메시지 패싱 방식의 병렬 프로그래밍 기법은 PVM (Parallel Virtual Machine), HPF (High Performance Fortran), MPI (Message Passing Interface)등이 있고, 본 연구에서는 MPI기법을 이용하여 병렬화를 수행하였다.

2.3 MPI 프로그래밍

본 연구에 적용된 기법인 MPI(Message Passing Interface)는 메시지 패싱 프로그래밍 모델 중 가장 많이 적용되고 있는 기법으로써, 이름 그대로 프로세스들 사이의 데이터 통신을 위한 인터페이스이며, 서브루틴(Subroutine) 또는 함수 등의 형태로 제공되는 라이브러리이다. 컴퓨터 기술이 급격히 발전하기 시작하던 1980년대부터 세계 각국에서 다양하게 개발되어 오던 MPI 라이브러리는 1992년부터 미국과 유럽의 메시지 패싱 시스템 전문가들에 의해 표준화 작업이 이루어지기 시작했다. 그 결과 1994년 5월에 발표된 MPI-1은 메시지 패싱에 대한 표준규약으로 완성되었고, 그 이후 여러 차례에 걸친 수정과 확장을 포함하는 업데이트를 통해 MPI-2.2까지 발표되었다(Lee et al., 2010). 현재까지도 MPI 포럼 등을 통해 지속적인 수정 및 확장 작업이 이루어지고 있으며, 향후 MPI-3의 출시를 위한 연구가 지속적으로 이루어지고 있다.

Fig. 3에서 보는 바와 같이 분산 메모리 구조는 각 프로세스가 메모리를 따로 가지는 분산된 시스템 환경이기 때문에 각 프로세스 간에 데이터들이 다른 프로세스에 위치한 메모리에 직접적으로 접근할 수 없다. 따라서 데이터들은 통신네트워크를 통해 메시지를 송신 및 수신할 수 있도록 MPI 라이브러리의 지원을 받는다. 두 프로세스 사이의 통신은 점대점(Point to Point) 통신, 집합통신 등 다양한 방식이 존재하지만 주로 점대점 통신으로 이루어지게 되며, 이때 통신이 이루어지는 두 프로세스는 각각 송신 및 수신 프로세스가 된다. 메시지를 주고받을 때 유의해야 할



(a) Single Thread Process

(b) Shared-Memory Parallel Programmin Model

(c) Massage Passing Parallel Programming Model

Fig. 2. Process of Single Thread and Multi Thread

점은 송신 변수는 통신이 완료되어야 다른 곳에서 다시 사용될 수 있고, 수신변수도 통신이 완료되어야 비로소 사용이 가능하게 된다는 점이다(Lee et al., 2010). 따라서 통신 완료여부를 검사하기 위한 루틴이 필요하며 통신간의 교착여부, 데이터간의 의존성, 동기화 필요여부 등을 확인하는 것 또한 MPI를 사용하는데 있어 중요한 사항이 된다.

이렇듯 공유 메모리 병렬 프로그램들에 비해 다소 복잡하고 적용이 쉽지 않은 MPI를 가장 많이 사용하는 주된 이유 중 하나는 MPI가 다양한 플랫폼에서 구현가능하다는 점 때문이다. 즉, MPI로 작성된 프로그램은 분산 메모리와 공유 메모리 구조에 대해 모두 적용가능하며, 개인 PC에서부터 다중워크스테이션, 슈퍼컴퓨터에 이르기까지 모든 환경에 대해서 구현이 가능하다.

3. 2차원 유한체적 모형의 병렬화

기 개발된 고정확도 2차원 유한체적 모형(Kim et al., 2009; Kim et al., 2011)은 마른하도 및 자연하천에서의 홍수파 전파해석을 위해 2차원 천수방정식을 Unsplit 유한체적기법과 HLLC Riemann 해법을 이용하여 해석한 모형으로(Kim et al., 2009), 복잡한 지형의 효율적 처리를 위해 삼각형 및 사각형 혼합격자에 적용 가능하도록 확장, 발전시켰다(Kim et al., 2011). 흐름률과 생성항의 균형을 위해 수면경사법을 사용하였으며, 시간과 공간에 대해 2차 정확도를 가지는 MUSCL 기법으로 확장하였다. 개발모형을 실험하도 및 자연하천의 홍수파 전파해석에 적용함으로써 적용성 및 정확성을 입증한 바 있지만 양해적 수치기법의 특성상 계산거리와 계산시간의 간격에 대한 제한 때문에 전체 계산시간이 길어지는 문제점은 계속 안고 있었다. 본 연구에서는 이러한 단점을 극복하고자 메시지 패싱 병렬 프로그래밍 모델인 MPI를 이용하여 모형을 병렬화 함으로써 프로그램의 계산 수행시간을 단축하고자 하였다.

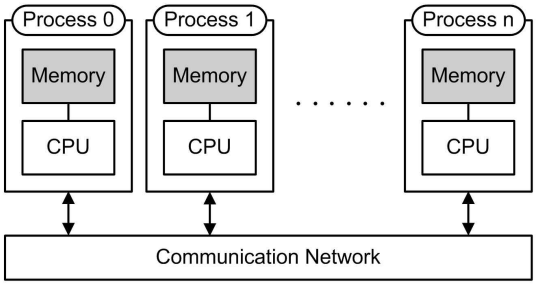


Fig. 3. Structure of Distributed Memory System

모형을 병렬 처리하기 위해 프로그램 내에서 병렬화 가능 영역을 확인하였고, 데이터 의존성 때문에 메시지 패싱이 필요한 영역도 함께 확인하였다. 또한 데이터 의존성이 나타나는 부분을 최소화 하도록 프로그램을 수정하고, 불필요한 서브루틴 또는 반복계산을 제거함으로써 프로그램에 대한 최적화 작업을 우선적으로 수행하였다. Fig. 4는 고정확도 2차원 유한체적 모형의 개략적 구조를 보여주고 있다. 프로그램이 시작되면 지형자료 및 제어자료의 입력 후 가장 먼저 TOPOG에서 프로그램이 인식 가능 하도록 지형자료를 재구축하고, 초기조건 및 경계조건을 설정한다. 흐름률 및 생성항 등 보존변수의 갱신에 필자에 의해 입력된 CFL 조건을 만족하는 적절한 Δt를 찾는 요한 변수들을 계산하기에 앞서 CFLCON 단계에서 사용

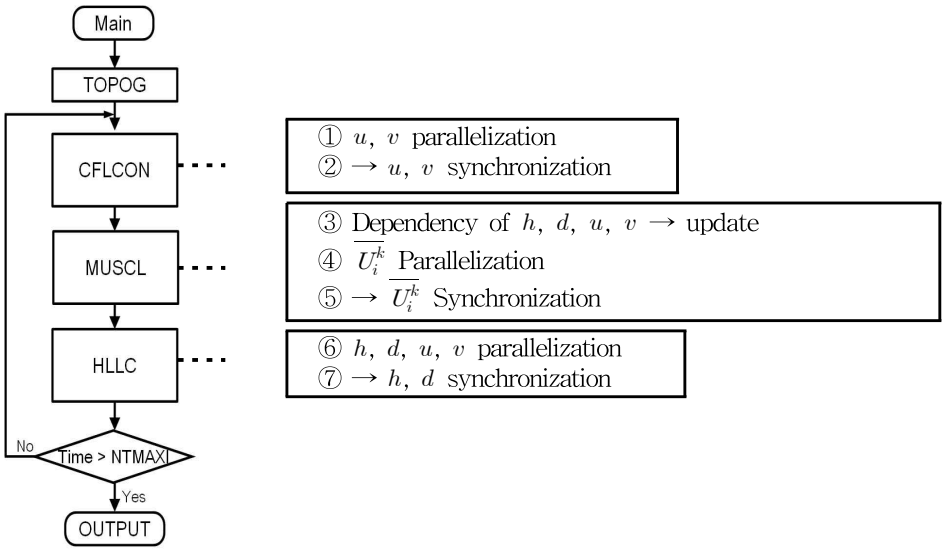


Fig. 4. Basic Structure of 2D Finite Volume Model and Details of Parallelization

과정을 거친다. 이후 예측단계(Predictor Step)인 MUSCL에서 수면경사법을 이용해 보존변수를 재구성하여 갱신하고, 계속해서 보정단계(Corrector Step)인 HLLC에서 격자경계로 유·출입하는 흐름률과 하상경사, 마찰경사 등을 포함하는 생성항을 계산하여 현재의 시간단계에 대해 보존변수를 갱신한다. 마지막으로 현재의 시간단계와 제어자료로 입력된 계산종료시간(NTMAXI)과의 비교를 통해 프로그램을 계속 수행할지, 최종 결과를 출력하고 종료할 것인지를 결정하게 된다. 프로그램이 계산종료시간에 이르지 못한 경우에는 다시 CFLCON으로 가서 새로운 계산시간간격을 계산함으로써, 매 시간단계마다 갱신되는 보존변수를 Δt 계산에 반영하도록 구성되어 있다.

모형의 병렬화는 프로그램 시작 후 반복 실행되는 CFLCON, MUSCL, HLLC를 포함하는 서브루틴에 대해 수행하였다. TOPOG에서도 병렬 처리가 가능한 영역이 존재하였으나, 프로그램이 수행된 후 한 번만 실행되는 서브루틴으로써 전체 프로그램의 속도 개선에 큰 영향을 주지 못한다고 판단하여 생략하였다. 각 프로세스 마다 통신할 데이터양이 다르고 주고받는 데이터 버퍼가 동일한 구조를 가지고 있는 프로그램의 특성상 동기화(Synchronization) 과정에서 각 프로세스 사이의 통신은 모두 논블록킹 점대점(Non-Blocking Point-to-Point) 통신을 사용하였다.

세부적인 병렬화 및 데이터 통신에 의한 동기화 과정은 다음과 같다. 먼저 CFL 조건에 의해 x방향의 유속(u)과 y방향의 유속(v)을 이용하여 Δt를 계산하므로 u와 v를 병렬 처리하여 최적화된 Δt를 찾는 시간을 단축하였고, 병렬 처리된 u, v는 다음 단계인 MUSCL단계의 여러 내부 계산을 위해 동기화 되었다. MUSCL 내부의 서브루틴 실행 시 보존변수들의 데이터 의존성이 나타난 것을 확인하였으므로, h(수위), d(수심), u, v를 갱신하였고, 예측단계(Precitor Step)에서 갱신해야 할 보존변수($\overline{U_i^k}$)를 병렬화하여 계산함으로써 계산시간을 단축하였다. 또한 다음 단계인 HLLC에서의 데이터 사용을 위해 병렬화된 $\overline{U_i^k}$를 동기화 하였다. HLLC에서는 흐름률 계산을 위해 활용되는 여러 변수들을 병렬 계산한 후 다음 계산단계의 MUSCL에서 사용될 h와 d를 동기화 하였고, 프로그램 가동 중에 특정시간에서의 결과를 출력할 사항이 발생하는 경우 만족되는 조건 하에서 변수들을 동기화 하여 출력하였다.

4. MPI를 이용한 병렬화 모형의 적용

MPI를 이용하여 병렬 처리된 본 연구모형의 적용성 및 계산시간에 대한 개선정도를 확인하기 위해 실측치가 존재하는 L자형 실험하도(Soares-Frazão and Zech, 1999)와 Malpasset 댐 붕괴 사상에 대하여 모형을 적용하였다. L자형 실험하도 모의를 통해 병렬화되기 이전 모형의 계산결과에 대한 일치성을 확인함과 동시에 계산시간에 대한 성능향상도를 확인하고자 하였다. 또한 보다 큰 규모의 입력자료에 대한 성능향상도 및 효율성을 측정하고자 1959년 12월에 프랑스 Reyran강 유역에서 발생하여 큰 피해를 입힌 Malpasset 댐 붕괴 사상에 모형을 적용하였다.

4.1 L자형 실험하도에 대한 병렬화 모형의 적용

Kim et al. (2011)은 본 모형의 이전 버전인 병렬처리 전 모형을 이용하여 90° 만곡을 가진 하도에 대한 댐 붕괴모의를 실시하였고, 혼합격자가 적용된 계산 결과를 사각형 격자 및 Petaccia (2003)의 계산 결과와 비교함으로써 혼합격자를 적용한 계산결과가 다른 모의결과에 비해 우수함을 입증한 바 있다. 검보정을 위한 실측자료는 Soares-Frazão and Zech (1999)에 의해 수행된 댐 붕괴 실험결과를 활용하였다(Fig. 5). Soares-Frazão and Zech (1999)는 0.01 m의 수심을 가지는 젖은하도조건과 수심이 0.0 m인 마른하도조건에 대해 실험을 수행하였으며, 저수지 및 하

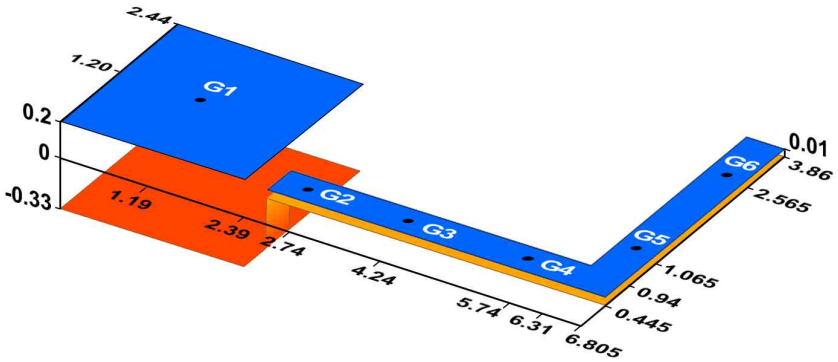


Fig. 5. 3-dimensional Layout of the Dam-break Experiment with 90° Bend Channel

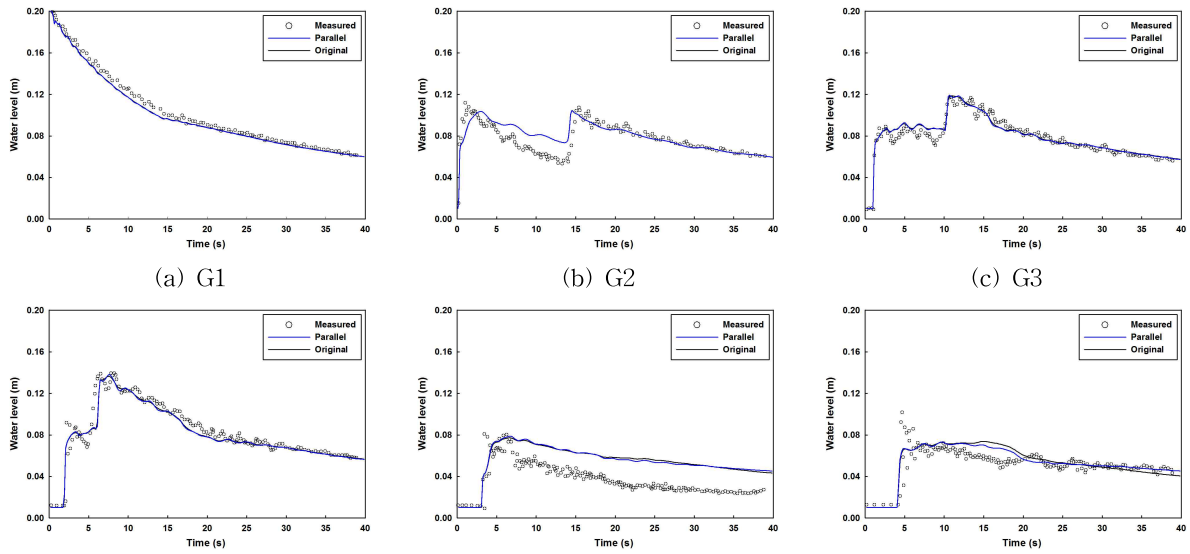
도의 주요 6개 지점에서 수위계측을 실시하였다.

4.1.1 모형의 적용성 및 기존모형과의 일치성 검토

Kim et al. (2011)은 실험조건과 동일하게 젖은하도조건과, 마른하도조건으로 구분하여 모의를 실시하였다. 병렬처리된 본 연구모형을 이용하여 Kim et al. (2011)의 모의와 동일한 조건으로 모의하기 위해, 격자의 사용(3,420개의 혼합격자), 조도계수(0.05), CFL 값(0.5), 계산시간(40초) 등 모든 입력변수를 같은 조건으로 부여하였다.

모형의 적용성을 입증하기 위해 본 연구모형에 의한 계산결과를 기존모형에 의한 계산 결과 및 실측수위와 비교하였다. 본 연구모형은 기존모형과 동일한 수치기법을 사용하였기 때문에 병렬처리 전 모형과의 일치성이 확보되어야 한다. 병렬화 모형과 기존모형에 대한 일치성을 확인하기 위해 G1~G6의 주요지점에 대해 두 모형의 계산결과를 정량적으로 비교 검토하였다. 또한 본 연구모형의 정확성 검토를 위해 실측치와의 비교 검토를 정량적으로 수행하였다. 정량적 평가를 위해 평균제곱근 오차(RMSE, Root Mean Square Error), 상관계수(CC, Correlation Coefficient), Nash-Sutcliffe 효율계수(NSE, Nash-Sutcliffe Efficiency coefficient), 평균절대편차(MAD, Mean Absolute Deviation)를 통계지표로 활용하였다.

Fig. 6은 G1~G6 지점에서 젖은하도조건에 대해 계산된 결과를 기존모형의 계산결과 및 실측수위와 비교하여 나타낸 그림이다. 병렬화 모형과 기존 모형과의 정량적 분석 결과를 Table 1에 제시하였고, 각 모형들에 의한 계산결과를 실측치와 비교하여 분석한 결과를 Table 2에 제시하였다. 병렬처리 전과 후의 모의결과는 G5와 G6를 제외하고 육안으로는 차이점을 거의 확인할 수 없으며, Table 1에 나타난 것처럼 G1~G4 지점에서의 결과는 모든 통계지표에서 일치된 범위를 나타낸다고 볼 수 있다. 하지만 G5와 G6지점에서는 육안으로도 식별가능한 정도의 차이가 나타나고 있고, 통계지표에서도 상대적으로 G1~G4지점보다 더 큰 오차가 나타나고 있다. 동일한 이론에 의한 수치처리 기법 및 같은 프로그램 구조로 구성된 두 모형의 계산결과는 이론적으로는 완벽히 일치해야 할 것 같으나, 각각 사용된 컴파일러가 다른 경우 계산결과에서 나타난 정도의 오차는 충분히 발생할 수 있다. 또한 병렬화



(a) G1 (b) G2 (c) G3

(d) G4 (e) G5 (f) G6

Fig. 6. Comparison of Numerical and Measured Data for Wet Bed Case

Table 1. Comparison of Parallel and Original Simulation Result for Wet Bed Case

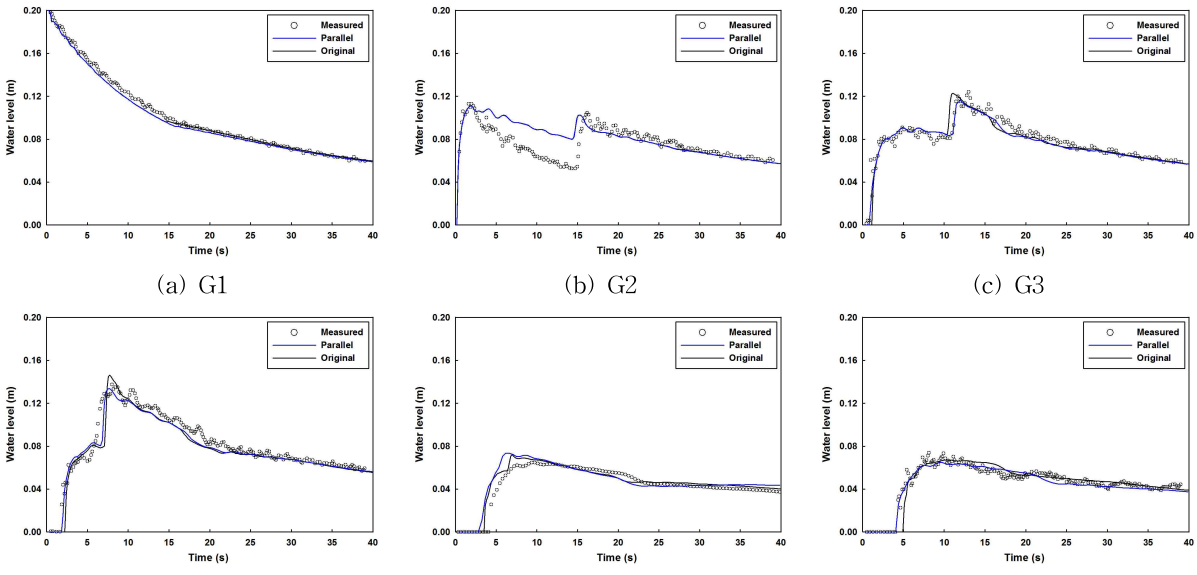
	G1	G2	G3	G4	G5	G6
RMSE	0.0004	0.0000	0.0006	0.0007	0.0012	0.0025
CC	1.0000	1.0000	0.9996	0.9997	0.9977	0.9919
NSE	1.0000	1.0000	0.9999	0.9998	0.9989	0.9949
MAD	0.0003	0.0000	0.0004	0.0005	0.0009	0.0019

작업을 수행하면서 기존 모형에서 단정도 실수형으로 사용된 일부 변수에 대해 배정도 실수형으로 수정하는 작업을 실시하였고, 병렬처리에 있어서 데이터 의존성을 유발하는 부분을 찾아서 최적화한 사항도 오차를 나타내는 원인이 될 수 있을 것으로 판단된다.

마른하도조건에 대해서도 젖은하도 조건과 동일한 분석을 실시하였다. G1~G6 지점에서 마른하도조건에 대해 계산된 결과를 기존모형의 계산결과 및 실측수위와 비교하여 Fig. 7에 도시하였고, 병렬화 모형으로 계산된 결과에 대해 기존모형으로 계산된 결과 및 실측치와의 비교분석결과를 Tables 3 and 4에 각각 나타내었다. G1과 G2 지점에서는 젖은하도에서의 계산결과와 비슷한 정도의 높은 일치성을 보여주었으나, 나머지 지점에 대해서는 육안으로도 확인할 수 있는 정도의 오차를 보여주고 있다. Table 3에 제시된 바와 같이 오차와 관련된 통계지표는 평균제곱근오차의 경우 약 0.2~0.5 cm 정도로 나타나고 있고, 상관성과 관련한 지표는 96%에서 98% 범위를 나타내는 것으로 보아, 젖은하도의 계산결과와 비교하여 상대적으로 높은 오차와 낮은 상관성을 나타냄을 확인할 수 있다. 특히 저수지에 가까운 G2 지점보다 붕괴파가 전파되면서 마른하도의 영향을 더 크게 받는 G3~G6 지점에서의 오차가 더 큰 것으로 보아 마른하도 상태가 계산결

Table 2. Comparison of Simulation Result and Measured Data for Wet Bed Case

	G1		G2		G3	
	Original	Parallel	Original	Parallel	Original	Parallel
RMSE	0.0042	0.0044	0.0112	0.0112	0.0054	0.0056
CC	0.9984	0.9982	0.7415	0.7407	0.9627	0.9589
NSE	0.9992	0.9991	0.9847	0.9847	0.9952	0.9947
MAD	0.0035	0.0037	0.0082	0.0082	0.0039	0.0041
	G4		G5		G6	
	Original	Parallel	Original	Parallel	Original	Parallel
RMSE	0.0090	0.0093	0.0212	0.0209	0.0102	0.0091
CC	0.9477	0.9445	0.7697	0.7934	0.7335	0.7678
NSEC	0.9883	0.9875	0.8273	0.8324	0.9648	0.9714
MAD	0.0053	0.0055	0.0199	0.0197	0.0074	0.0063



(a) G1 (b) G2 (c) G3

(d) G4 (e) G5 (f) G6

Fig. 7. Comparison of Numerical and Measured Data for Dry Bed Case

Table 3. Comparison of Parallel and Original Simulation Result for Dry Bed Case

	G1	G2	G3	G4	G5	G6
RMSE	0.0013	0.0002	0.0051	0.0055	0.0036	0.0115
CC	0.9998	1.0000	0.9712	0.9842	0.9808	0.9650
NSE	0.9997	1.0000	0.9982	0.9936	0.9976	0.9667
MAD	0.0009	0.0001	0.0017	0.0021	0.0022	0.0034

Table 4. Comparison of Simulation Result and Measured Data for Dry Bed Case

	G1		G2		G3	
	Original	Parallel	Original	Parallel	Original	Parallel
RMSE	0.0035	0.0038	0.0144	0.0144	0.0088	0.0061
CC	0.9982	0.9986	0.5897	0.5895	0.9232	0.9605
NSE	0.9991	0.9989	0.9661	0.9660	0.9882	0.9941
MAD	0.0027	0.0031	0.0102	0.0102	0.0049	0.0040
	G4		G5		G6	
	Original	Parallel	Original	Parallel	Original	Parallel
RMSE	0.0104	0.0078	0.0152	0.0144	0.0064	0.0043
CC	0.9411	0.9668	0.7842	0.8682	0.9248	0.9634
NSEC	0.9741	0.9853	0.8175	0.8413	0.9737	0.9872
MAD	0.0063	0.0053	0.0145	0.0134	0.0037	0.0034

과의 오차에 더 큰 영향을 미침을 확인할 수 있다. 이는 goto 문 등과 같은 데이터 의존성을 유발하는 프로그래밍 언어가 젖은하도보다 마른하도의 경우에 더 많이 포함되어 있는 프로그램 특성과, 이러한 데이터 의존성을 제거하기 위해 젖은하도보다 마른하도의 경우에 더 많은 수정작업을 가한 원인 때문인 것으로 판단된다.

4.1.2 병렬화 모형의 계산시간 검토

병렬처리된 모형의 계산시간에 대한 개선정도를 확인하기 위해 프로세스 개수별로 모의를 실시하여 계산시간을 측정하였다. 모의를 위해 2.93 GHz의 클럭속도를 가지는 Intel Xeon X5570 프로세서를 탑재한 슈퍼컴퓨터를 사용하였고, Intel Compiler 11.0의 컴파일러와 MPICH-2를 MPI의 적용을 위한 라이브러리로 이용하였다.

병렬화된 모형의 계산시간에 대한 개선정도를 측정하기 위해 가장 많이 적용되고 있는 지표인 성능향상도(Speed-up)와 효율성(Efficiency)을 사용하였다. 성능향상도는 프로그램이 이전의 순차프로그램과 비교해 상대적으로 얼마나 더 성능이 좋아졌는가를 판단하기 위한 지표로써, Eq. (1)과 같은 식으로 계산할 수 있다(Pacheco, 1997).

$$S(n) = \frac{T_1}{T_p} \tag{1}$$

여기서, T_1은 최적화 전의 기존순차모형의 실행시간 또는 병렬처리된 프로그램을 1개의 프로세스를 이용하여 계산한 실행시간을 의미하고, T_p는 n개의 프로세스를 이용한 병렬프로그램의 실행시간을 의미한다. 효율성은 Eq. (2)와 같은 식으로 계산되며, 프로세스 개수(n)에 대해 어느 정도의 성능향상을 나타냈는지를 나타내 주는 지표이다(Pacheco, 1997).

$$E(n) = \frac{T_1}{T_p \times n} = \frac{S(n)}{n} \tag{2}$$

병렬처리된 고정확도 2차원 유한체적 모형을 이용하여 프로세스 개수별로 90°만곡을 가진 하도에 대한 댐 붕괴 모의를 실시하였다. 총 8개의 스레드에 대해 프로세스 개수별로 프로그램 실행시간, 성능향상도, 그리고 효율성을 계산하였고, Table 5에 젖은하도에 대한 분석결과를, Table 6에 마른하도에 대한 분석결과를 각각 제시하였다. 또한 Fig. 8에 동일한 분석결과를 그래프로 제시하였다.

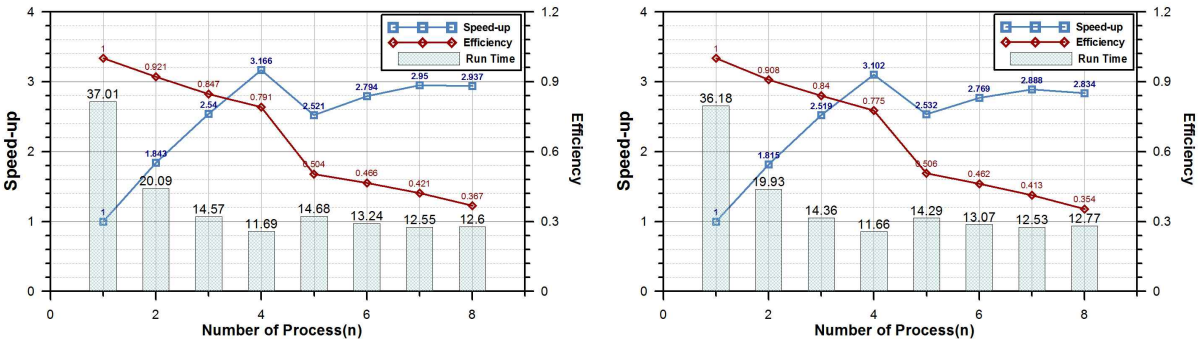
프로세스 개수가 4개까지 증가함에 따라 프로그램 실

Table 5. Simulation Result according to Number of Process for Wet Bed Case

Number of Process	Program Run Time (sec)	Speed-up	Efficiency
1	37.009	1.000	1.000
2	20.086	1.843	0.921
3	14.57	2.540	0.847
4	11.69	3.166	0.791
5	14.681	2.521	0.504
6	13.245	2.794	0.466
7	12.547	2.950	0.421
8	12.602	2.937	0.367

Table 6. Simulation Result according to Number of Process for Dry Bed Case

Number of Process	Program Run Time (sec)	Speed-up	Efficiency
1	36.184	1.000	1.000
2	19.934	1.815	0.908
3	14.364	2.519	0.840
4	11.665	3.102	0.775
5	14.291	2.532	0.506
6	13.067	2.769	0.462
7	12.527	2.888	0.413
8	12.766	2.834	0.354



(a) Wet Bed Case (b) Dry Bed Case

Fig. 8. Simulation Result by Number of Process

행시간이 감소되어 병렬처리에 대한 계산시간 단축효과를 확인할 수 있었으며, 그에 따른 성능향상도는 젖은하도의 경우 1.843, 2.54, 3.166으로, 마른하도의 경우 1.815, 2.519, 3.102로 선형적으로 증가함을 알 수 있다. 마른하도 조건에 비해 젖은하도 조건에서 미세하게 더 좋은 성능향상을 나타내었지만, 0.01 미만으로 큰 차이가 없음을 보여주고 있다. 프로세스 개수 증가에 비례하는 성능향상도를 얻지 못하는 이유는 프로그램이 순차코드영역과 병렬코드영역으로 나누어져 있고, 병렬코드영역만 병렬화를 통해 성능 이득을 얻을 수 있기 때문이다. 어떠한 프로그램이든 순차코드영역은 존재할 수밖에 없고, 본 연구모형의 경우 병렬화를 통해 얻을 수 있는 성능향상이 4개의 프로세스를 이용한 경우 젖은하도의 경우 3.166배, 마른하도의 경우 3.102배 정도 되는 것으로 나타났다. 하지만 프로세스 개수가 5개 이상 증가하는 경우에는 오히려 4개의 프로세스를 이용하는 경우보다 계산시간이 증가되고, 성능

향상도 또한 그에 미치지 못하였으며, 효율성의 감소폭도 크게 증가되었다. 이는 MPI 기법이 메시지 통신을 이용하여 프로세스간 데이터를 전달하는 방식이기 때문에, 프로세스 개수가 필요 이상 증가하는 경우 메시지 전달에 필요한 데이터의 양이 증가되고 그에 따른 통신부하(Communication Overhead)가 발생하여 오히려 계산시간이 증가하는 현상이 발생하였기 때문이라 판단된다. 또한 여러 개의 서브루틴을 태스크 병렬화를 통해 작업을 분배하여 동시에 실행하는 경우 작업량이 이상적으로 분배되기 힘들기 때문에 가장 먼저 작업이 끝난 서브루틴은 다른 서브루틴의 실행이 끝날 때 까지 기다려야 하는 문제가 발생한다. 이러한 현상을 로드언밸런싱(Load Unalancing)이라고 하며, 병렬화된 프로그램이 프로세스 개수의 증가에 따라 원하는 성능향상도를 얻지 못하는 중요한 이유 중 하나가 된다. 본 연구모형의 경우 작업분배에 의한 병렬화 부분이 많지 않기 때문에 프로세스 개수가 5개 이상 증가하면서 원하는 성능이득을 얻지 못하는 이유는 주로 통신부하 문제 때문이라고 판단된다.

프로세스 개수별로 모의하여 계산시간 및 성능향상도를 검토해 본 결과 필요 이상의 프로세스 개수 증가는 원하는 성능이득을 얻을 수 없으며, 본 연구모형의 경우 4개의 프로세스를 이용하는 경우가 성능향상도나 효율성 면에서 가장 적절하다고 판단된다. 병렬코드 영역을 증가시키고 로드언밸런싱 문제를 최소화하는 프로그램의 개선이 추가로 이루어진다면 더 많은 프로세스 개수 증가에 대한 성능이득을 얻을 수 있을 것으로 기대된다.

4.2 Malpasset 댐 붕괴사상에 대한 병렬화 모형의 적용

본 연구모형의 계산시간에 대한 성능향상도를 보다 더 큰 규모의 입력자료를 이용하여 파악하고자 댐 붕괴 시 흔적수위와 수리실험에 의한 실측수위가 존재하는 Malpasset 댐 붕괴사상에 대해 모형을 적용하였다. Malpasset 댐은 프랑스 남부의 Fréjus 지역으로부터 약 7 km 북쪽에 위치한 Reyran 강에 건설된 아치형 댐으로써, 불충분한 지형조사와 하류부의 고속도로 건설현장의 발파, 그리고 붕괴직전 24시간동안의 130 mm의 집중호우를 포함한 10일간의 총 500 mm 이상의 강우로 인해 1959년 12월 2일에 붕괴되었다. 댐 붕괴에 의해 생성된 최대 40 m의 홍수파는 약 70 km/hr의 파속으로 Reyran 강을 따라 하류부에 위치한 Fréjus 지역으로 전파되면서 총 433명의 사상자와 약 6,800만 달러의 재산피해를 발생시킨 후 인근의 지중해 연안으로 빠져나갔다.

4.2.1 모형의 적용성 검토

병렬화 모형에 대한 정확성 및 적용성을 검토하고자 Malpasset 댐 붕괴시의 흔적수위와 수리모형실험에 의한 실측수위를 검토하였다. 댐 붕괴 홍수파가 지나간 후 경찰에 의해 약 100여개 지점에서 흔적수위가 조사되었으며, 그 중 Reyran강의 좌·우안에 대해 가장 중요한 지점들을 Fig. 9의 P1~P17로 나타내었다. 또한 1964년 프랑스전력(EDF, Electricite de France)의 국립수리실험실(LNH, Laboratoire National d'Hydraulique)에 의해 1/400의 축척으로 Malpasset 댐 붕괴에 대한 수리모형실험이 수행되어 앞서 조사된 흔적수위와 검증을 실시하였다(Goutal, 1999). 실측된 지점들을 Fig. 9에 S6~S14로 나타내었으며, 현장조사에 의한 흔적수위와 함께 개발모형의 검보정 자료로 활용하였다.

입력자료의 규모에 따른 병렬화 모형의 성능향상도 및 효율성을 검토하기 위해 지형자료에 대한 격자수를 약 5,000개, 10,000개, 15,000개, 그리고 20,000개의 4가지 경우로 구성하여 각각 Case 1~Case 4로 명명하였다. 격자

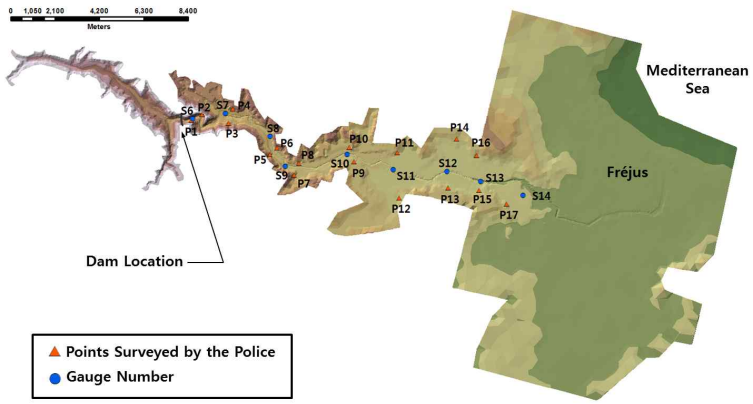


Fig. 9. Location of Gauges and Points Surveyed by the Police

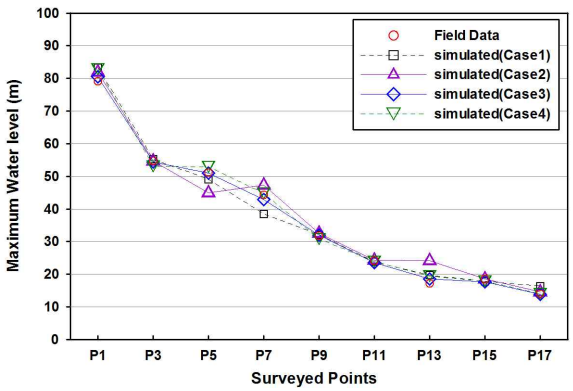
는 삼각형 및 사각형의 혼합격자로 구성하였고, 댐 인근으로부터 붕괴 홍수파가 전파되는 하류부로 이동할수록 격자의 크기가 점점 증가되도록 구성하였다. 각 Case에 대한 격자수와 노드수, 그리고 격자의 면적에 대한 상세정보를 Table 7에 제시하였다.

모의를 위해 저수지의 수위를 100 m, 댐 하류부의 수심을 0.0 m로 하는 마른하도 조건으로 초기조건을 구성하였고, 댐 붕괴유량에 비해 저수지 유입유량이 매우 적어 상류단 경계조건은 고려하지 않았다(Valiani et al., 2002; Kim et al., 2011). 또한 하류단 경계조건은 해안으로 유출되도록 하는 자유유출 경계조건으로 고려하였으며, 조도계수는 CADAM의 연구(Soares-Frazão and Zech, 1999)에서 제안된 0.033의 동일한 조건으로 전단면을 고려하였다(Valiani et al., 2002; Yoon and Kang, 2004; Brufau et al., 2004; Liang et al., 2007; Kim et al., 2011).

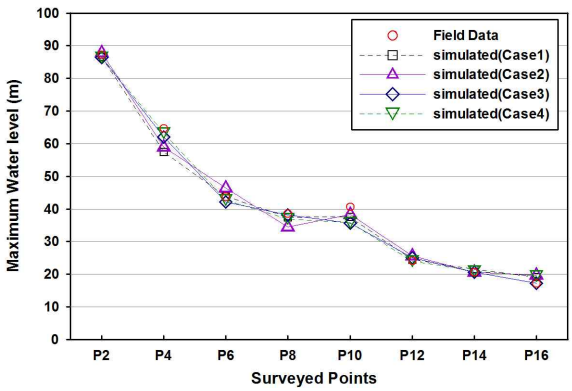
Fig. 10에 본 연구모형에 의해 모의된 최대수위를 현장조사에 의한 흔적수위와 수리모형실험에 의한 실측수위를 각 Case 별로 도시하였다. 계산된 최대수위와 하천의 우안 및 좌안에 대한 흔적수위와 비교를 Figs. 10(a) and 10(b)에 각각 나타내었고, 수리모형실험에 의한 실측수위와의 비교를 Fig. 10(c)에 나타내었다. Fig. 10에서 보는 바와 같이 본 연구모형에 의해 각 지점에서 계산된 최대수위는 각

Table 7. Information of Cells and Nodes in each Case

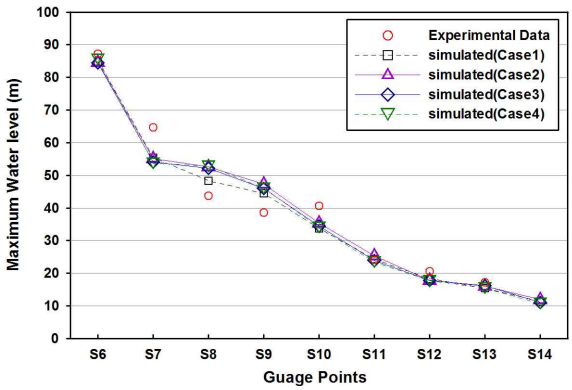
	Case 1	Case 2	Case 3	Case 4
Number of Cells	5,091	9,962	14,925	20,012
Number of Nodes	4,341	8,419	12,610	16,780
Average Cell Area($10^5 m^2$)	12.525	8.658	7.023	6.395
Maximum Cell Area($10^5 m^2$)	82.390	75.716	65.771	48.811
Minimum Cell Area($10^5 m^2$)	0.868	0.700	0.548	0.443



(a) Highest Water Level at Right Banks



(b) Highest Water Level at Left Banks



(c) Highest Water Level at Gauges

Fig. 10. Comparison Between Measured Data and Numerical Results in each Case

Case 마다 다소 차이를 보여주고 있으나 현장수위 및 실측 수위와 대체로 일치된 결과를 보여주고 있다.

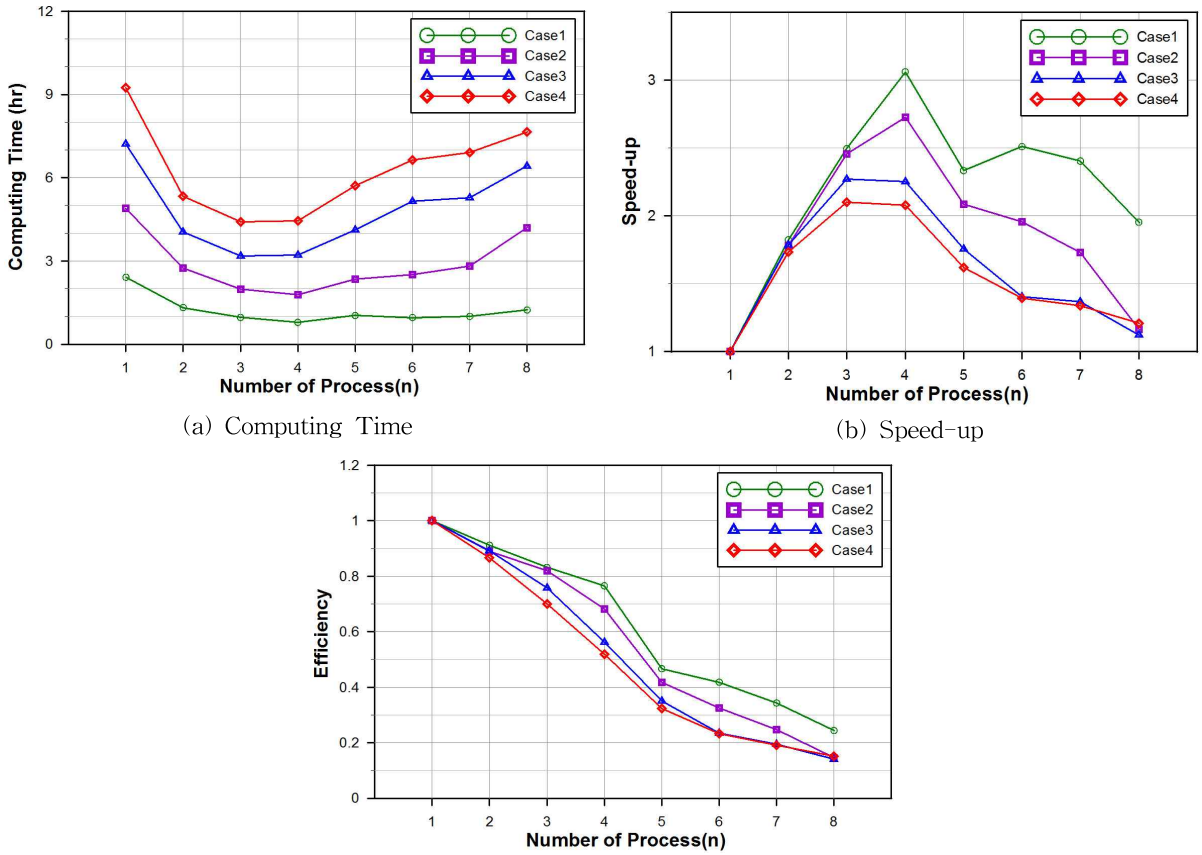
큰 규모의 입력자료에 대한 병렬화 모형의 계산속도에 대한 개선정도 확인하기 위해 L자형 실험하도와 마찬가지로 총 8개의 프로세스에 대해 스레드 개수를 증가시키면서 모의를 실시하였다. 앞서 Eqs. (9) and (10)에 설명한 성능향상도와 효율성을 기존 모형에 대한 계산속도의 개선을 확인하기 위한 지표로 사용하였고, 입력자료의 규모에 대한 계산속도의 개선 정도를 파악하기 위해 격자수를 기준으로 분류된 4가지 Case에 대해 성능향상도 및 효율성을 검토하였다.

Table 8에 각 Case에 대한 프로세스 개수별 계산시간과 성능향상도, 그리고 효율을 제시하였다. 동일한 결과로써 계산시간, 성능향상도, 그리고 효율에 대한 각 Case 별 비교결과를 Fig. 11에 도시하였다. Fig. 11(a)에서 보는 바와 같이 Case 별로 격자수가 증가함에 따라 계산시간도 함께 증가하는 일반적인 양상을 나타냄을 확인할 수 있다. 각 Case에 대해서 프로세스의 수가 증가됨에 따라 계산시간도 함께 단축되지만 L자형 실험하도의 계산결과와 마찬가지로 특정 수 이상의 프로세스 증가는 오히려 계산시간을 증가시키는 결과를 유발하였다. Case 1과 Case 2의 경우는 앞서 검토된 작은 규모의 입력자료의 분석결과와 마찬가지로 5개 이상의 프로세스를 사용하는 경우 계산속도가 증가되는 현상을 나타내었으며, 입력자료의 규모가 더 큰 Case 3과 Case 4의 경우는 4개의 프로세스를 사용하는 경우에도 계산속도가 증가되는 현상을 보여주고 있다. 이러한 현상은 앞서 설명한 바와 같이 통신부하가 발생한 원인 때문인 것으로 판단된다. 즉, 입력자료의 규모가 더 커질수록 전달해야 하는 데이터의 양이 더 늘어나기 때문에 계산속도의 개선 효과가 점점 더 감소하여 Case 3과 Case 4의 경우 4개의 프로세스를 사용하는 경우에도 계산속도가 증가되는 현상이 발생한 것으로 생각된다.

각 Case별 계산시간에 대한 성능향상도 및 효율성은 Figs. 11(b) and 11(c)에 제시된 바와 같이 2개의 프로세스를 사용하는 경우에는 거의 비슷한 정도를 나타내고 있으나, 3개 이상의 프로세스를 사용하는 경우는 입력자료의 규모가 증가함에 따라 점점 감소하고 있음을 확인할 수 있다. 또한 입력자료의 규모가 가장 작은 Case 1에서

Table 8. Computing Time in each Case

Number of Process	Case 1			Case 2		
	Computing Time (hr)	Speed-up	Efficiency	Computing Time (hr)	Speed-up	Efficiency
1	2.416	1.000	1.000	4.894	1.000	1.000
2	1.325	1.823	0.912	2.746	1.782	0.891
3	0.968	2.495	0.832	1.992	2.457	0.819
4	0.789	3.062	0.765	1.794	2.729	0.682
5	1.035	2.334	0.467	2.344	2.088	0.418
6	0.962	2.510	0.418	2.501	1.957	0.326
7	1.005	2.404	0.343	2.829	1.730	0.247
8	1.236	1.954	0.244	4.199	1.166	0.146
Number of Process	Case 3			Case 4		
	Computing Time (hr)	Speed-up	Efficiency	Computing Time (hr)	Speed-up	Efficiency
1	7.227	1.000	1.000	9.250	1.000	1.000
2	4.049	1.785	0.892	5.335	1.734	0.867
3	3.182	2.272	0.757	4.404	2.100	0.700
4	3.207	2.253	0.563	4.447	2.080	0.520
5	4.117	1.756	0.351	5.716	1.618	0.324
6	5.145	1.405	0.234	6.641	1.393	0.232
7	5.286	1.367	0.195	6.914	1.338	0.191
8	6.427	1.124	0.141	7.646	1.210	0.151



(a) Computing Time

(b) Speed-up

(c) Efficiency

Fig. 11. Comparison of Computing Time, Speed-up, and Efficiency in each Case

는 프로세스 개수를 4개까지 증가함에 따라 성능향상도가 선형적으로 증가되는 것을 확인할 수 있으나, 입력자료의 규모가 증가됨에 따라 성능향상도의 증가폭이 점점 감소되고 있는 것으로 보아, 큰 규모의 입력자료는 프로세스의 개수를 증가시키더라도 계산시간에 대한 큰 성능향상을 기대할 수 없다는 것을 확인할 수 있다.

이와 같은 연구 결과를 통해, 병렬화 된 프로그램을 이용하여 계산시간에 대한 성능향상을 얻기 위해서는 입력데이터의 규모와 각 프로세스간에 전달되어야 하는 데이터의 양이 중요하게 고려되어야 하는 요소임을 알 수 있다. 즉, 지나치게 큰 규모의 입력자료에서는 프로세스의 증가가 오히려 계산속도에 대한 성능의 저하를 불러 올 수 있으므로, 적절한 입력자료의 규모와 프로세스 개수의 선정이 병렬화를 통한 성능이득을 위한 필수 고려조건임을 확인할 수 있다. 본 연구모형의 경우 5개 이상의 프로세스의 사용은 계산속도에 대한 성능향상에 큰 이득을 얻을 수 없다는 점과, Case 3과 Case 4의 경우 4개의 프로세스를 사용하는 경우에도 성능향상도가 크게 저하될 수 있다는 점을 고려하여 입력자료를 구축해야 할 것이다.

다만 작은 크기의 격자가 큰 크기의 격자보다 더 자세한 지형정보를 반영하여 정확한 계산결과를 얻을 수 있다는 일반적 개념을 고려하였을 때, 통신부하에 의해 지나친 성능저하가 발생하지 않는 범위에서 적절한 규모의 입력자료를 구축하는 것이 계산결과에 대한 정확도와 계산시간에 대한 성능향상을 동시에 기대하기 위한 필요충분조건이라고 볼 수 있다. 병렬코드영역을 증가시키고 통신부하를 최소화 할 수 있는 프로그램의 개선이 추가로 이루어진다면, 더 많은 프로세스 개수와 더 큰 규모의 입력자료에서도 효율적인 성능이득을 얻을 수 있을 것으로 기대된다.

5. 결 론

본 연구에서는 삼각형 및 사각형 혼합격자에 적용가능하도록 개발되어 정확도와 적용성이 입증된 바 있는 2차원 유한체적모형의 계산속도를 개선하고자 코어 수의 제약에 자유로운 MPI 기법을 이용하여 프로그램을 병렬화하였다. 병렬화된 개발모형을 실험자료가 존재하는 2차원 실험하도와 실제 댐 붕괴사상에 대해 적용함으로써 기존

모형과의 계산결과에 대한 일치성을 검증하고 계산속도에 대한 성능향상도를 확인하였다. 본 연구를 통해서 얻은 주요 결과는 다음과 같다.

1) 기 개발된 프로그램의 병렬화를 위해 프로그램 내의 병렬화 가능 영역과 메시지 패싱이 필요한 영역을 검토한 후, 계산속도에 대해 성능향상을 가장 크게 얻을 수 있는 흐름률 계산영역과 계산시간간격 계산영역에 대해 논 블록킹 점대점통신을 이용하여 프로그램의 병렬화를 수행하였다.

2) 병렬처리된 모형의 기존 모형에 대한 일치성 및 계산시간 개선정도를 확인하기 위해 실측치가 존재하는 L자형 실험하도에 대해 모형을 적용한 결과, 병렬화 작업과정에서 데이터의 효율적 처리를 위한 변수형 수정과 데이터 의존성을 배제하고자 최적화가 이루어진 부분 때문에 기존모형과의 계산결과에 미세한 오차가 존재하였으나 대체로 높은 일치성을 보여주었다. 또한 계산시간에 대한 성능향상도와 효율성을 검토한 결과, 프로세스간 전달되는 데이터양의 증가로 발생되는 통신부하로 인해 5개 이상의 프로세스 사용은 원하는 성능이득을 구현하지 못하였고, 효율적인 성능향상을 위해 4개의 프로세스를 이용하는 것이 가장 적절함을 알 수 있었다.

3) 큰 규모의 입력자료에 대한 병렬화 모형의 적용성과 입력자료의 규모에 따른 성능향상도 및 효율성을 검토하기 위해, Malpasse 댐 붕괴 사상에 대해 모형을 적용한 결과, 입력자료의 규모가 작을수록 프로세스의 적용 개수를 4개까지 증가시킴에 따라 선형적으로 증가되는 성능향상도를 나타내었으나, 그 이상의 프로세스 증가는 통신부하로 인해 원하는 성능이득을 볼 수 없었다. 또한 입력자료의 규모가 클수록 성능향상도 및 효율성이 더 감소하는 것으로 보아, 전달되어야 하는 데이터의 증가에 따른 통신부하 현상이 더 크게 나타남을 확인할 수 있었다.

4) 계산결과에 대한 정확도와 원하는 정도의 계산시간에 대한 성능향상을 얻기 위해 적절한 규모의 입력자료를 구축하는 것이 정확한 지형정보를 반영하면서 지나친 통신부하에 의한 성능저하를 피하기 위한 필수요인임을 확인하였고, 더 많은 프로세스 개수와 큰 규모의 입력조건에 대한 효율적인 성능이득을 얻기 위해 병렬코드 영역의 증가 및 통신부하를 최소화 할 수 있는 프로그램의 개선이 이루어 질 수 있도록 연구가 지속되어야 할 것으로 판단된다.

감사의 글

본 연구는 KISTI 슈퍼컴퓨팅센터의 기술지원을 받아 수행되었고, 소방방재청자연재해저감기술개발사업단(자연피해예측및저감연구개발사업)의 지원으로 수행한 '침수재해 경감 표준모델 개발 및 관리기술 고도화' [NEMA-자연-2014-75]과제의 성과입니다.

References

Alcrudo, F., and Garcia-Navarro, P. (1993). "A high-resolution Godunov-type scheme in finite volumes for the 2D shallow-water equations." *International Journal for Numerical Methods in Fluids, John Wiley & Sons, Inc.*, Vol. 16, No. 6, pp. 489-505.

Andrews, G.R. (2000). *Foundations of multithreaded, parallel, and distributed programming.* Addison Wesley, United States, pp. 1-9.

Begnudelli, L., Sanders, B., and Bradford, S. (2008). "Adaptive Godunov-based model for flood simulation." *Journal of Hydraulic Engineering*, Vol. 134, No. 6, pp. 714-725.

Brufau, P., Garcia-Navarro, P., and Vazquez-Cendon, M.E. (2004). "Zero mass error using unsteady wetting drying conditions in shallow flows over dry irregular topography." *International Journal for Numerical Methods in Fluids, John Wiley & Sons, Inc.*, Vol. 45, No. 10, pp. 1047-1082.

Caleffi, V., Valiani, A., and Zanni, A. (2003). "Finite volume method for simulating extreme flood events in natural channels." *Journal of Hyraulic Research, Talor & Francis*, Vol. 41, No. 2, pp. 167-177.

Chippada, S., Dawson, C.N., Martinez, M.L., and Wheeler, M.F. (1998). "A Godunov-type finite volume method for the system of shallow water equations." *Computer Methods in Applied Mechanics and Engineering, Elsevier Science*, Vol. 151, No. 1-2, pp. 105-129.

Goutal N. (1999). "The Malpasset dam failure. An overview and test case definition." The Proceeding of the 4th CADAM Meeting, Zaragoza, Spain, pp. 1-7.

Jung, S.Y., Park, J.H., Hur, Y.T., and Jung, K.S. (2010). "Application of MPI technique for distributed rainfall-runoff model" *Journal of Korea Water Resources*

Association, KWRA, Vol. 43, No. 8, pp. 747-755.

Kim, B.H., Han, K.Y., and Kim, J.S. (2009). "Handling method for flux and source terms using unsplit scheme" *Journal of Korea Water Resources Association*, KWRA, Vol. 42, No. 12, pp. 1079-1089.

Kim, B.H., Han, K.Y., and Son, A.L. (2011). "Development of two-dimensional finite volume model applicable to mixed meshes." *Journal of Korea Water Resources Association*, KWRA, Vol. 44, No. 2, pp. 109-123.

Lee, H.S., Kim. J.H., Lee, S.W., and Lee, S. (2010). *A time for multi-core parallel programming.* Adbooks, pp. 40-41.

Liang, D., Lin, B., and Falconer, R.A. (2007). "An boundary-fitted numerical model for flood routing with shock-capturing capability." *Journal of Hydrology, Elsevier Science*, Vol. 332, No. 3-4, pp. 477-486.

Neal, J., Fewtrell, T., and Trigg, M. (2009). "Parallelisation of storage cell flood models using OpenMP" *Environmental Modelling & Software, Elsevier Science*, Vol. 24, No. 7, pp. 872-877.

Neal, J.C., Fewtrell, T.J., Bates, P.D., and Wright, N.G. (2010). "A comparison of three parallelisation methods for 2D flood inundation models." *Environmental Modelling & Software, Elsevier Science*, Vol. 25. No. 4, pp. 398-411.

Pacheco, P.S. (1997). *Parallel programming with MPI, Morgan Kaufmann*, San Francisco, C.A., pp. 11-39, 249-250

Park. N.S., Hong, S.H., and Shim, M.G. (2003). "Development of optimal pumping model for coastal region using genetic algorithms and parallel processing." *Journal of Korea Society of Civil Engineers*, KSCE, Vol. 23, No. 5B, pp. 397-403.

Pau, J.C., and Sanders, B.F. (2006). "Performance of parallel implementations of an explicit finite-volume shallow-water model." *Journal of Computing in Civil Engineering*, ASCE, Vol. 20, No. 2, pp. 99-110.

Paz, A.R., Collischonn, W., Tucci, C.E.M., and Padovani, C.R.P. (2011). "Large-scale modelling of channel flow and floodplain inundation dynamics and its application to the Pantanal (Brazil)." *Hydrological Processes, Wiley-Blackwell*, Vol. 25, No. 9, pp. 1498-1516.

Petaccia, G. (2003). *Propagazione di onde a fronte ripido per rottura di sbarramenti in alvei naturali.* Ph.D. dissertation, University of Pavia, Italy.

Sanders, B.F., Schubert, J.E., and Detwiler, R.L. (2010). "ParBreZo: A parallel, unstructured grid, Godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale." *Advances in Water Resources, Elsevier Science*, Vol. 33, No. 12, pp. 1456-1467.

Soares-Frazão, S., and Zech, Y. (1999). *Effects of a sharp bend on dam-break flow.* Proceedings 28th Congress of IAHR, Graz, Austria.

Valiani, A., Caleffi, V., and Zanni, A. (2002). "Case study: malpasset dam-break simulation using a two-dimensional finite volume method." *Journal of Hydraulic Engineering*, ASCE, Vol. 128, No. 5, pp. 460-472.

Wang, Y., Wang, H., Lei, X., Jiang Y., and Song X. (2011). "Flood simulation using parallel genetic algorithm integrated wavelet neural networks" *Neurocomputing, Elsevier Science*, Vol. 74, No. 17, pp. 2734-2744.

Wikipedia the Free Encyclopedia. (2014). *Multi-core processor*. Wikipedia Foundation, Inc., Web, 26 May 2014.

Yoon, T.H., and Kang, S.K. (2004). "Finite volume model for two-dimensional shallow water flows on unstructured grids." *Journal of Hydraulic Engineering*, ASCE, Vol. 130, No. 7, pp. 678-688.

Yu, D. (2010). "Parallelization of a two-dimensional flood inundation model based on domain decomposition" *Environmental Modelling & Software, Elsevier Science*, Vol. 25, No. 8, pp. 935-945.

논문번호: 14-013	접수: 2014.01.29
수정일자: 2014.06.05/06.12	심사완료: 2014.06.12