Journal of the Korea Convergence Society
Vol. 7. No. 5, pp. 15-21, 2016 http://dx.doi.org/10.15207/JKCS.2016.7.5.015

퀀텀 에스프레소와 제온 파이 프로세서의 융합을 이용한 분산컴퓨팅 성능에 대한 연구

박영수[1], 박구락[2*], 김동현[3]
[1]공주대학교 컴퓨터공학과, [2]공주대학교 컴퓨터공학부, [3]우송대학교 IT융합학부

A Study of Distribute Computing Performance Using a Convergence of Xeon-Phi Processor and Quantum ESPRESSO

Young-Soo Park[1], Koo-Rack Park[2*], Dong-Hyun Kim[3]
[1]Dept. of Computer Engineering, Kongju National University
[2]Dept. of Computer Science & Engineering, Kongju National University
[3]Dept. of IT Convergence, Woosong University

요 약 최근 프로세서의 집적도는 급속도로 발전하고 있으나 클락 스피드는 증가하지 않는 대신에 프로세서 내의 코어 수가 늘어나고 있는 실정으로 프로그래밍 속도 향상을 위한 방법에 대한 연구가 필수적이라 할 수 있다. 이에 본 논문에서는 현재 연산 가속화를 위해 사용되는 매니 코어 프로세서의 대표적인 인텔 제온 파이의 성능 분석을 위하여 퀀텀 에스프레소를 활용하였다. 또한 제온 파이에서 MPI 실행시 랭크의 수를 변화시키면서 성능 벤치마킹을 수행하여 하드웨어적인 성능 특성을 연구하였다. 그 결과 물리 코어가 57개인 제온파이 프로세서의 하나의 코어 당 4개의 작업을 처리할 때 가장 좋은 성능을 나타내고 있으며, 물리 코어 하나에 MPI 랭크수를 4개 이상 확장하면 성능향상이 거의 일어나지 않는다. 이러한 융합 기술을 통하여 퀀텀 에스프레소의 성능 향상과 제온 파이의 하드웨어적인 특성을 확인할 수 있다.

• 주제어 : 융합, 병렬처리, 퀀텀 에스프레소, 제온 파이, 소프트웨어 최적화

Abstract Recently the degree of integration of processor and developed rapidly. However, clock speed is not increased, a situation that increases the number of cores in the processor. In this paper, we analyze the performance of a typical Intel Xeon Phi of many core process used for the current operation accelerate. Utilizing the Quantum ESPRESSO, which was calculated using the FFTW library. By varying the number of ranks in MPI when running the benchmarks the performance Xeon Phi. The result shows a good performance in the handling of four job on one physical core. However, four or more to expand the number of MPI Rank is degraded. Through this convergence it was found to improve the performance of Quantum ESPRESSO. It is possible to check the hardware characteristics of the Xeon Phi.

• Key Words : Convergence, Parallelism, Quantum ESPRESSO, Xeon Phi, Software Optimization

*Corresponding Author : 박구락(ecgrpark@kongju.ac.kr)
Received September 2, 2016 Revised October 6, 2016
Accepted October 20, 2016 Published October 31, 2016

1. 서론

최근 들어 프로세서의 집적도는 빠르게 발전하고 있으나 프로세서의 제어신호인 클락 스피드는 더 이상 증가하지 않고 있다. 이에 프로세서 내의 코어 수가 늘어나고 있는 상황에서 매니 코어 프로세서를 위한 많은 소프트웨어 연구가 요구되고 있으며[1], 현재 연산 가속화를 위해 사용되는 매니 코어 프로세서는 AMD의 GPU(Graphic Processor Unit) 들과 인텔의 제온 파이(Xeon-Phi) 와 같은 가속기가 대표적이라 할 수 있다.

또한 고속 연산을 필요로 하는 대량의 연산 분야에서 매니 코어 프로세서를 이용한 구현과 성능 분석, 개선은 고속 연산의 구현뿐만이 아니라 매니 코어 프로세서의 성능 이해와 미래 설계를 위한 필수적인 연구이다[2]. 이를 위하여 제온 파이의 아키텍처, 프로그래밍 속도 향상을 위한 방법에 대한 연구[3]와 효율적인 병렬화 컴퓨팅에 대한 연구[4], 생물정보학[5,6], 데이터베이스 운영[7], 최적화 프로그래밍[8]에 관한 연구가 활발히 이루어지고 있다.

제온 파이 3120A는 22 나노 공정 라인에서 생산된 매니 코어 프로세서로서 하이앤드급의 GPU의 성능과 유사하지만 X86 CPU 타입의 명령어 세트를 가지고 있고, 57~61 코어가 동작하며, 약 1.1GHz의 클락 스피드로 프로세스가 처리된다. 또한 각각의 코어는 512bits FMA(Fused Multiply Add) 벡터 유닛과 4 스레드 SMT(Simultaneous Multi-Threading)를 가지고 있어서 결국 240개의 동작 가능한 스레드를 보유하고 있다.

이러한 제온 파이는 수치 가속 플랫폼으로서 수치해석에 많이 사용되고 있는 유한차분법인 FDTD(Finite Difference Time-Domain)[9,10,11,12], 연산 대상을 유한개의 요소로 분할하여 각각의 영역에 대하여 계산을 수행하는 연산 방법인 FEM(Finite Elements Method)을 가속화하는데 사용되고 있다[13].

이에 본 논문에서는 인텔 제온 파이를 대상으로 오픈소스인 퀀텀 에스프레소(QE: Quantum ESPRESSO)의 주요 알고리즘인 퓨리에 변환(FFTW: Fastest Fourier Transform in the West)을 수행하여, 제온 파이 프로세서에서 퓨리에 변환의 성능을 벤치마킹하고, FFT 및 제온 파이의 성능 특성을 분석하여, 클락 스피드, 코어 수, 벡터 사이즈가 성능에 어떠한 영향을 미치는지에 대한 성능 분석 방법을 제안한다.

2. 관련 연구

2.1 퀀텀 에스프레소

QE는 나노 크기의 전자 구조 계산 및 자료 모델링을 위한 오픈 소스 코드의 통합 제품군이며, 밀도 함수 이론, 평면파 및 가상의 위치 에너지에 기초한다[14].

QE의 FFTW Ver. 2는 FFTW3 라이브러리가 지원되고[2], QE의 병렬화는 MPI(Message Passing Interface)와 공유 메모리 다중 처리 프로그래밍 API인OpenMP(Open Multi-Processing)기술을 지원한다. 이에 더해 QE의 내부 FFTW 라이브러리는 스레드 레벨 병렬처리와 분산처리 기술을 동시에 구현하는 하이브리드(MPI와 OpenMP 동시 지원) 의 기술을 사용하여 병렬화 기술에 사용된다.

퓨리에 변환은 QE 수행 계산 시 큰 부분에 사용되기 때문에, FFT 연산 성능은 QE 제품군의 성능에 큰 영향을 미친다. 또한 FFTW는 이산 퓨리에 변환(DFT: Discrete Fourier Transform)의 컴퓨터 라이브러리 버전이며, 가장 빠른 알고리즘으로 알려져 있다.

대부분의 주요 하드웨어 플랫폼은 그에 상응하는 최적화 된 수치 라이브러리와 함께 제공하는데, 라이브러리들은 이미 FFT 계산을 위한 루틴을 포함하고 있으며, IBM의 ESSL(Engineering and Scientific Software Library), 인텔의 MKL(Math Kernel Libraries) 등은 모두 이러한 라이브러들이며 QE 역시 이 라이브러리들이 지원된다.

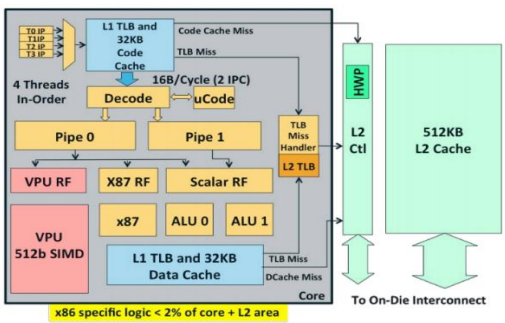
2.2 제온 파이

많은 범용 애플리케이션에 적합한 대용량 병렬 프로세서와 GPU의 성장으로, 인텔은 제온 파이라는 이름으로 다중 통합 코어 아키텍처를 기반으로 제품 라인을 출시했다. 제온 파이 코프로세서는 대칭 멀티 프로세서로서, PCI 익스프레스(Peripheral Component Interconnect Express)를 통해 호스트 시스템에 연결한다는 점에서 GPU와 유사하고, μOS 리눅스 운영체제를 가지고 있다. 그러나 제온 파이는 독립된 프로세서로서 사용되지만, 호스트 시스템이 있어야만 동작 하도록 설계 되었다.

제온 파이는 현대의 x86 CPU 에 비해 훨씬 단순하고 그에 따라 더욱 작아진 코어와 단위면적당 더 적은 열을 방출한다. 또한 강력한 512-bit SIMD(Single Instruction Multiple Data) 벡터 유닛이 CPU 코어에 내장되어 있다[15]. 따라서 제온 파이는 단순화된 각 물리 코어의 각 프로세서는 57~60 코어, 64 비트의 x86 명령어를 지원하

고, 약 1.1GHz의 클락 스피드를 가지고 있다. 물리 코어 수는 모델과 제품에 따라 조금씩 다르며, 본 논문에서 사용한 제온 파이는 3120 시리즈(rev 11)을 사용하였으며, CPU의 물리 코어는 57개이다.

제온 파이는 물리 코어 하나 당 4개씩 총 57*4(=228)개 이상의 논리 코어를 제공한다. 이는 4-way 하이퍼 쓰레딩을 지원한다는 의미를 가지고 있으며, 각각의 코어는 고속의 양방향 링 구조로 연결되어 있고, 30MB 의 L2 캐시를 공유하고 있다. 또한 제온 파이는 6~16GB 용량의 GDDR5 메모리를 내장하고 있다. 그리고 인텔 MCI(IMCI) 로 불리는 인텔의 새로운 명령어 세트를 지원한다. 다음의 [FIg. 1]은 제온 파이의 하드웨어 시스템 아키텍처이다.



[Fig. 1] System Architecture

2.3 인텔 MPI

MPI는 여러 병렬 프로그래밍 개발 모델 중 메시지 파싱 모델의 표준이라 할 수 있다. 하나의 작업을 다수의 분산 메모리 프로세스들에게 나누어 실행하게 하는 병렬 연산에서는 프로세스들 간의 통신이 필요하다.

메시지 파싱 모델은 각자의 분산 메모리를 프로세스별로 따로 가지는 분산 시스템 환경에서 프로세스들 사이의 통신을 메시지들의 교환으로 구현하는 프로그래밍 모델을 의미한다. 따라서 메시지 파싱 모델은 프로세스들 간에 메모리 공간을 공유하지 않으며, 하나의 프로세스가 다른 프로세스의 메모리에 직접 접근 할 수 없다. 인텔 MPI 라이브러리 5.0은 인텔 아키텍처에서 애플리케이션의 속도를 향상시키기 위한 라이브러리로서, 특히 인텔 아키텍처 및 제온 파이에서 최적화 되어있다. 본 논문에서는 제온 파이에 QE를 MPI 인터페이스에 적용하기 위하여 인텔 MPI 라이브러리 5.0을 적용하였다.

3. 연구 방법 및 결과

3.1 소프트웨어 구현 환경

QE를 제온 파이에 이식을 하기 위한 소프트웨어 환경 설정 과정은 다음과 같다. 첫째, 제온 파이 자체에 대한 컴파일 환경이 지원 되지 않으며, 크로스 컴파일을 호스트 머신에서 수행해야 한다. 호스트 머신에서 제온 파이 프로세서용 실행 바이너리 파일을 생성하여, 파일을 옮기는 과정이 필요하다. 둘째, 현재는 인텔 컴파일러만이 제온 파이i 크로스 컴파일 환경을 지원하고 있다. 따라서 인텔 Parallel Studio 2015에서 컴파일 환경을 구축하였고, QE를 제온 파이i 프로세서용 바이너리를 생성하여 성능 분석을 수행하였다.

다음의 <Table 1>은 실험을 위한 하드웨어 및 소프트웨어 구현 환경 이다.

<Table 1> Implementation Environment

Division	Contents
HOST	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz, 32G Memory
Coprocessor	Intel Xeon-Phi Coprocessor 3120 Series (rev 11)
OS	Linux Xeon-Phi Host 2.6.32-431.el6.x86_64 #1 SMP Fri Nov 21 02:14:10 UTC 2013 x86_64 GNU/Linux
Compiler	Intel Parallel Studio 2015 update 2 (cluster edition)
Coprocessor Driver	(Many Core software stack) : MPSS 4.3.3

3.2 구현 준비

MPI 애플리케이션을 처음으로 실행하기 위해 MPI 관련 라이브러리와 실행 유틸리티들을 준비된 시스템의 제온 파이 프로세서들로 각각 복사하는 과정이 필요하다.

```
# sudo scp ${Intel mpi directory}/mic/bin/* mic0:/bin/
mpiexec 100% 1161KB 1.0MB/s 00:00
pmi_proxy 100% 817 KB 800.4KB/s 00:00
...
# sudo scp ${Intel mpi directory}/mic/lib/* mic0:/lib64/
libmpi.so.4.1 100% 4381KB 4.3MB/s 00:00
libmpigf.so.4.1 100% 323KB 320.8KB/s 00:00
libmpigc4.so.4.1 100% 185KB 175.2KB/s 00:00
...
# sudo scp ${Intel compiler directory}/lib/mic/*
mic0:/lib64/
libimf.so 100% 2517KB 2.5MB/s 00:01
libsvml.so 100% 4884KB 4.8MB/s 00:01
libintlc.so.5 100% 129KB 129.1KB/s 00:00
```

[Fig. 2] Coping Intel MPI Library to Xeon Phi

다음의 [Fig. 2]는 복사 과정을 나타낸 것으로, 구현 시스템의 모든 프로세서 들은 각각 유일한 IP 어드레스를 가져야 한다. 첫 번째 프로세서는 MIC0 호스트 네임 또는 IP 어드레스로 접근 가능하고, 마찬가지로 두 번째 프로세서는 MIC1 또는 IP 어드레스로 접근 할 수 있다.

3.3 제온 파이 포팅

QE 배포는 구성 요소의 핵심 세트 및 고급 작업을 수행하는 플러그인 세트, 플러스 핵심 구성 요소와 상호 운용할 수 있도록 설계된 패키지로 구성되어 있다.

다음의 [Fig. 3]은 QE를 제온 파이에 포팅 하는 과정으로 크게 7단계로 구분할 수 있다.

[Fig. 3] Xeon Phi Porting Process

QE 코드는 오픈 소스 배포판으로 사용할 수 있으며, 인텔 제온 파이 프로세서에서 QE 코드 버전 5.1.1을 실행하려면 빌드 프로세스 없이 소스를 최소한으로 수정하는 것이 필요하다. 인텔 제온 파이 프로세서 지원은 인텔 MKL 자동 오프로드 기능을 통해 이루어진다. 인텔 MKL 버전 11.1.3 이상을 사용하고 있는지 확인하고, 다운로드 하여 사용할 수 있다.

첫째, Quantum ESPRESSO의 압축된 소스 코드를 다음의 [FIg. 4]와 같은 명령으로 해제한다.

```
tar xvfz espresso-5.2.1.tar.gz
```

[Fig. 4] Extract Quantum ESPRESSO source code

둘째, 인텔 컴파일러와 MPI* 소프트웨어 환경을 다음의 [Fig. 5]과 같이 읽어온다.

```
source
/opt/Intel/composer_xe_2013.2.146/bin/compilervars.sh
Intel64
source    /opt/Intel/impi/4.1.1/bin64/mpivars.sh
```

[Fig. 5] Read the Software Environment

셋째, 인텔 컴파일러 초기 컴파일 환경 구성과 설정을 위하여 다음의 [Fig. 6]와 같이 설정한다.

```
export FC=mpiifort
export F90=$FC
export MPIF90=$FC
export F77=$FC
export FFLAGS=$FCFLAGS
export FCFLAGS="-O3 -mmic -xAVX -fno-alias
      -ansi-alias -g -mkl -L$MKLROOT/include/fftw"
export CC=mpiicc
export CPP="icc -E"
export CFLAGS=$FCFLAGS
export BLAS_LIBS="-lmkl_cdft_Core  -lmkl_Intel_lp64
      -lmkl_blas95_lp64 -lmkl_Core -lmkl_Intel_thread
      -lpthread    -lm"
export LAPACK_LIBS="-lmkl_lapack95_lp64
      -lmkl_blacs_Intelmpi_lp64 "
export SCALAPACK_LIBS="-lmkl_scalapack_lp64"
export FFT_LIBS="-L$MKLROOT/Intel64"
```

[Fig. 6] Read the QE configuration environment

넷째, 하이브리드 MPI+OpenMP 모드, FFTW3, 스칼라 팩의 설치, DFLAGS 및 패키지 디렉토리 설정을 위한 시스템 파일을 다음의 [Fig. 7]과 같이 설정한다.

```
CFLAGS  = -O3 -mmic  -fno-alias -mkl -ansi-alias
          -L/opt/인텔/composer_xe_2015.1.133/mkl
          /include/fftw    $(DFLAGS) $(IFLAGS)
F90FLAGS = $(FFLAGS) -fpp  -nomodule -openmp
          $(FDFLAGS) $(IFLAGS) $(MODFLAGS)
FFLAGS = -O3 -mmic  -mkl  -fno-alias -ansi-alias
         -L/opt/인텔/composer_xe_2015.1.133/mkl/
         include/fftw

LD          = mpiifort
LDFLAGS  = -openmp -mmic -L/opt/Intel/composer_
           xe_2015.1.133/mkl/lib/mic
LD_LIBS    =

BLAS_LIBS = -lmkl_Intel_lp64 -lmkl_blas95_lp64 -lmkl
            _cdft_Core -lmkl_Intel_thread -lmkl
            _Core -lpthread -lm
BLAS_LIBS_SWITCH = external

LAPACK_LIBS= -lmkl_blacs_Intelmpi_lp64 -lmkl_lapack
             95_lp64
LAPACK_LIBS_SWITCH = external

ELPA_LIBS_SWITCH = disabled
SCALAPACK_LIBS = -lmkl_scalapack_lp64
```

[Fig. 7] Edit Make.sys File

다섯째, 인텔 MKL 자동 오프로드를 위한 멀티 스레드 버전 MKL 빌드를 한다.

여섯째, Xeon-Phi 프로세서용 실행 바이너리를 복사한다.

마지막으로 Pw.x 샘플 MPI를 다음의 [Fig. 8]과 같이 실행한다. (FFT Dimensions: (18, 18, 72))

```
(mpirun -n 2 -host mic0 /tmp/pw.x -igraphene.scf.in
 -o 250.out )   >> fftw_result.out 2>&1
```

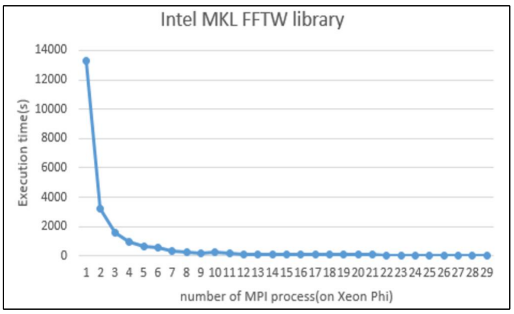
[Fig. 8] Sample MPI Execution

3.4 성능 평가

QE FFTW3의 구현에 모든 하이브리드 버전은 대부분의 경우에 하이브리드 모드에서 사용되는 내부 FFTW 라이브러리보다 인텔 MKL 라이브러리의 FFT 함수를 사용 하였을 때 더 나은 성능을 보이고 있다. 또한 분산 병렬 컴퓨팅 MPI와 스레드 레벨 병렬화 도구인 OpenMP를 동시에 사용한 하이브리드 적용 시 순수 MPI만을 사용하여 컴파일 했을 때보다 좋은 성능을 보였다. 제온 파이 프로세서의 물리 코어 개수인 57개의 개수에 맞게 MPI 랭크 수를 부여하고 MPI 랭크 수를 물리 코어 수의 배수로 증가하여 성능과 실행 시간을 측정 하였다.

MPI 랭크 수의 증가에 따라 성능이 증가 하였으나 어느 시점 이후로는 MPI 랭크 수를 늘려도 성능이 개선되지 않고 수렴 하였다. 이는 병렬처리 프로세스는 계속 증가하지만 어느 한계점 이상으로는 MPI 랭크 프로세스들 간에 통신 병목이 생기는 것을 알 수 있었다.

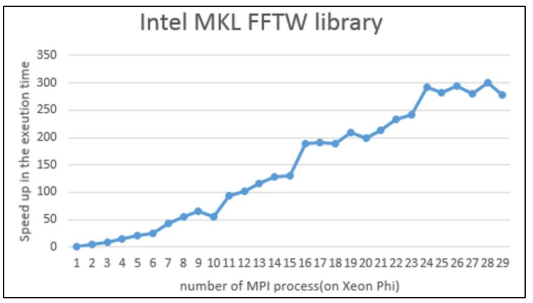
다음의 [Fig. 9]은 MPI 랭크 수를 제온 파이 물리 코어의 배수로 증가시켜 측정한 결과이다.



[Fig. 9] Measurement Results(1)

MPI 랭크 수를 물리 코어의 8배 이상 주었을 때는 더 이상 실행 시간이 단축되지 않음을 알 수 있다.

다음의 [Fig. 10]는 MPI 랭크 수가 증가함에 따라 실행 시간의 성능이 계속 증가 하지만 MPI 랭크 수를 물리 코어의 24배 이상 주면 오히려 성능이 감소하기 시작 하고 있다.



[Fig. 10] Measurement Results(2)

4. 결론 및 향후 연구방향

인텔 MKL FFTW 라이브러리를 지원하기 위해 QE의 라이브러리 확장하면서 시도하였다. 인텔 MKL FFTW 라이브러리를 QE 포팅하고 MPI 랭크 숫자를 늘려가며 QE 성능을 벤치마킹 하였다. 인텔 제온 파이는 고성능 병렬 프로그램에 적합한 하드웨어 장비로서, 성능 분석 결과 MPI 랭크를 물리 코어의 4배수(57*4=228) 까지는 거의 선형으로 성능이 늘어남을 볼 수 있었다. 이는 제온 파이의 아키텍처가 4 Way 하이퍼 스레딩을 지원하므로, 논리 코어가 물리 코어 하나에 4개씩 존재하여, 실제 계산 성능도 물리 코어 하나당 4개의 작업을 처리 할 때 좋은 성능을 나타낸다. 그리고 물리 코어 당 4배수 이상의 작업을 부여하면 실제 성능은 늘어나지 않고 수렴하는 것을 볼 수 있다. 실제 제온 파이는 물리 코어 하나당 4개의 입력 포트를 가지고 있고, 이 입력 포트 하나는 하나의 작업을 입력 받는 역할을 하여 결국 논리 코어를 물리 코어 하나당 4개의 작업을 동시에 처리 할 수 있도록 한다.

따라서 이것은 인텔 제온 파이의 하드웨어 특성에 기인 한 것으로 볼 수 있다. 물리 코어 하나 당 MPI 랭크 수를 4개 이상 계속 늘리면 성능이 늘어나는 속도 가 느리긴 하지만, 어느 정도까지는 계속 늘어나다가 24개 이상으로 가면 오히려 성능이 저하됨을 알 수 있다. 이는 작업을 병렬로 처리하는 성능 이득 보다 작업을 분배하

고 MPI 랭크 들 간에 통신 하는 비용이 더 많아서 오히려 성능이 더 저하됨을 알 수 있다. 향후 잘 알려진 MKL FFTW, FFTW3, 인텔 IPP, Numutils, Kiss-fft 등 잘 알려진 FFTW 라이브러리들을 비교 분석하여 제온 파이의 성능 특성 분석에 대한 연구가 계속되어야 할 것이다.

REFERENCES

[1] Asanovic, Krste, et al. "The Landscape of Parallel Computing Research: A View from Berkeley", Technical Report UCB/EECS-2006-183, EECS, Department, University of California, Berkeley, 2006

[2] H. J. Lee, E. J. Im, "SpMV on Xeon-Phi", Proceedings of the KIISE, pp. 42-44, 2014.

[3] Yang, Xiaoling, and Wenhua Yu. "Phi Coprocessor Acceleration Techniques for Computational Electromagnetics Methods", Applied Computational Electromagnetics Society Journal, Vol. 29, Issue 12, 2014.

[4] Heinecke A, Vaidyanathan K, Smelyanskiy M, et al. "Design and implementation of the linpack benchmark or single and multi-node systems based on intel xeon Phi coprocessor", Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on. IEEE, pp. 126 - 137, 2013.

[5] Liu Y, Maskell DL, Schmidt B. "CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units", BMC Research Notes, 2, 73, 2009.

[6] Lan H, Liu W, Schmidt B, et al. "Accelerating large-scale biological database search on Xeon Phi-based neo-heterogeneous architectures", Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on. IEEE, pp. 503 - 510, 2015.

[7] Lu M, Zhang L, Huynh HP, et al. "Optimizing the mapreduce framework on intel xeon phi coprocessor", Big Data, 2013 IEEE International Conference on. IEEE, pp. 125 - 130, 2013.

[8] M. Bernaschi, M. Bisson, and F. Salvadore, "Multi-Kepler GPU vs. multi-Intel MIC for spin systems simulations", Computer Physics Communications, vol. 185, no. 10, pp. 2495 - 503, 2014.

[9] A. Taflove and S. Hagness, "Computational electromagnetics: the finite-difference timedomain method", 3rd ed., Artech House, Norwood, MA, 2005.

[10] W. Yu, X. Yang, Y. Liu, et al., "Parallel finite difference time-domain method", Artech House, Norwood, MA, 2006.

[11] W. Yu, X. Yang, and W. Li, "VALU, AVX, GPU acceleration techniques for parallel finite difference time domain methods", SciTech Publisher Inc., Raleigh, NC, 2013.

[12] A. Elsherbeni and V. Demir, "The finite difference time domain method for electromagnetics: with MATLAB simulations", SciTech Publisher Inc., Raleigh, NC, 2009.

[13] J. M. Jin, "The finite element method in electromagnetics", (2nd edition), New York: John Wiley & Sons, 2002.

[14] M. Frigo, S. G. Johnson, "The Design and Implementation of FFTW3", Proceedings of the IEEE 93(2), pp. 216-231, 2005.

[15] Lan, Haidong, et al., "Parallel algorithms for large-scale biological sequence alignment on Xeon-Phi based clusters", IEEE International Conference on Bioinformatics and Biomedicine 2015 Washington, DC, USA. pp. 9-12, 2015.

저자소개

박 영 수(Young-Soo Park) [정회원]

- 1999년 2월 : 충남대학교 물리학과 (이학사)
- 2001년 2월 : 충남대학교 대학원 물리학과(이학석사)
- 2014년 3월 ~현재 : 공주대학교 대학원 컴퓨터공학과 박사과정
- 2014년 3월 ~ 현재 : 대한컨설팅 그룹 이사

<관심분야> : 고성능컴퓨팅, 소프트웨어 병렬화/최적화

박 구 락(Koo-Rack Park) [정회원]

- 1986년 2월 : 중앙대학교 전기공학과(공학사)
- 1988년 2월 : 숭실대학교 전자계산학과(공학석사)
- 2000년 2월 : 경기대학교 전자계산학과(이학박사)
- 1991년 4월 ~ 현재 : 공주대학교 컴퓨터공학부 교수

<관심분야> : 경영정보, 정보통신, 고성능컴퓨팅, 전자상거래

김 동 현(Dong-Hyun Kim) [정회원]

- 1986년 2월 : 중앙대학교 물리학과(이학사)
- 2005년 2월 : 공주대학교 컴퓨터멀티미디어공학과(공학석사)
- 2010년 2월 : 공주대학교 컴퓨터공학과(공학박사)
- 2016년 3월 ~ 현재 : 우송대학교 IT융합학부 겸임교수

<관심분야> : 영상처리, 지리정보, 시뮬레이션, 고성능컴퓨팅