한국항행학회 논문지 제14권 제6호 2010년 12월

자바 스레드와 네트워크 자원을 이용한 병렬처리

Thread-Level Parallelism using Java Thread and Network Resources

김태용*

Tae-Yong Kim*

요 약

본 논문에서는 초소형 정밀 마이크로 흐름센서를 설계하기 위해 Java 멀티스레드를 이용한 병렬 프로그래밍 기법을 도입하여 센서 모듈의 성능 분석과 개선이 가능한 병렬처리형 설계 툴을 개발하였다. 연산에 따른 기본 성능을 측정하기 위하여 열운송 방정식에 지배되는 포텐셜 문제를 두 개의 실험모델로 나누어 실험을 수행하였다. 시뮬레이션 결과 네트워크 PC의 수를 증가시키면 이와 비례하는 속도향상 특성이 나타났다. 따라서 본 연구에서 제안하는 병렬화 방안은 대규모 연산모델에도 적용 가능함을 확인하였다.

Abstract

In this paper, parallel programming technique by using Java Thread is introduced so as to develop parallel design tool to analyze the small micro flow sensor. To estimate computing time for Thread-level parallelism, the performances of two experimental models for potential problem subject to Thermal transfer equation are examined. As a result, if the number of network PC is increase, computing time for parallelism on network environment is enhanced to be almost n times. The micro sensor design tool based on distributed computing can be utilized to analyze a large scale problem.

Key words : Micro flow sensor, MCM, multi thread, parallelism

I. 서 론

유비쿼터스 컴퓨팅은 기기 지능화를 위하여 자동차, 전자 기기 등에 MPU를 내재화하는 단계에서 벗어나 환경, 사물, 인간간의 유기적이고 매듭이 없는 네트워크 구성을 통한 생활공간에서의 센서 및 컴퓨팅 기기 내재화의 단계로 발전하고 있다. 따라서 무선 센서네트워크 환경과 같은 분야에서 초소형 정밀 센서는 센서 노드와 결합하여 유비쿼터스 환경 구축을 위한 역할이 중요시되고 있다[1].

최근 실리콘 기판위에 반도체집적 공정과 같은 미세가공기술을 이용하여 정교한 마이크로 소자의 개발이 진행되고 있다. 이렇게 개발된 소자는 저전력, 다차원화, 다기능화, 지능화, 시스템화가 가능할 뿐만 아니라 매우 빠른 응답특성을 가져 유비쿼터스 환경 구축을 위한 활용도가 높아지고 있다.

예를 들어 산화물 반도체 감지막이 동작온도에 따라 감응특성을 가지는 마이크로 흐름센서는 센서 소

* 동서대학교 컴퓨터정보공학부
· 제1저자 (First Author) : 김태용
· 투고일자 : 2010년 12월 3일
· 심사(수정)일자 : 2010년 12월 3일 (수정일자 : 2010년 12월 22일)
· 게재일자 : 2010년 12월 30일

자 중앙에 위치한 히터에 전력을 공급하고 특정 흐름에 따라 소자 상면부에서의 열전달의 차이를 히터 양측에 놓인 감지막을 통하여 계측함으로써 센서로서의 기능을 부여할 수 있다. 이때 센서 소자의 온도 전달특성을 제어하여 정밀도가 높은 센서를 설계하기 위해서는 소자의 열전달 특성 및 소비전력 등의 물리적 특성을 계측기 등을 사용하여 측정하고 성능을 개선시키는 것은 시간과 경비의 증가로 많은 어려움이 따른다.

따라서 효율적으로 센서를 설계하는 방법으로서 실제 센서 모듈을 제작하지 않고, 온도특성과 같은 물리현상을 지배하는 기초(지배)방정식을 토대로 수치 모델을 수립하고 이를 해석적 또는 수치적으로 계산함으로써 이론 해를 구하고 실험결과와 비교하는 방법을 도입할 필요가 있다.

그러나 시제품을 제작하기 전에 다양한 물리적 상황과 환경을 고려할 경우 수치 해석적 접근 방법은 유용하다고 볼 수 있으나, 수치 모델링을 위한 계산 공간이 커져 일반 연산과정을 통한 수치 시뮬레이션에 시간과 비용이 증대하는 문제점을 안고 있다. 이를 극복하기 위한 대안으로서 분산 병렬 컴퓨팅의 한 방법으로 알려진 그리드 컴퓨팅(Grid Computing)을 도입하면 원거리 통신망(WAN)으로 연결된 원격지 유휴 PC자원을 활용하는 것이 가능하다. 그러나 이 경우에도 원격지 유휴 PC자원을 활용하기 위한 프레임워크가 구축되어 있을 필요가 있어 적용상의 많은 문제가 예상된다[2].

현재 인터넷 및 인트라넷의 보급이 확산되면서 네트워크는 컴퓨팅 환경의 기반을 이루게 되었고 이기종 컴퓨터간 분산 컴퓨팅이 가능해지고 있다. 그러나 이기종 컴퓨터간의 분산 컴퓨팅은 각 컴퓨팅 환경을 구성하는 플랫폼의 다양성 때문에 매우 복잡하고 어려운 작업으로 여겨지고 있다. 관련 업체 및 표준화 단체에서 분산 컴퓨팅을 위한 표준을 제안하고 있지만 분산 응용 프로그램의 요구가 점차 다양화됨에 따라 보다 일관적이고 플랫폼에 독립적인 개발 환경을 필요로 하게 되었다.

Java 언어가 등장하면서 분산 응용프로그램 개발자들은 이러한 개발 환경을 제공 받을 수 있는 기회를 갖게 되었다. Java언어가 제공하는 엔터프라이즈 플랫폼은 분산 컴퓨팅을 위한 다양한 API를 제공하며 응용 프로그램의 생산성 향상 및 플랫폼 독립성을 제공한다[3,4].

본 연구에서는 Java 언어에서 제공하는 스레드(Thread)를 기반으로 원격지에 있는 유휴 PC 자원을 활용하여 연산 수행에 필요한 태스크를 공유하여 분산 컴퓨팅이 가능한 병렬처리 방안을 제시하고 그 유효성을 검증한다.

Ⅱ. 병렬처리 구축방안

2-1 문제 정의 및 계산 알고리즘

마이크로 센서 특성을 해석하기 위해 적용 가능한 수치해석 방법으로는 유한요소법(FEM), 경계요소법(BEM), 유한차분법(FDM) 등이 보편화되어 있다[3]. 또 다른 방법으로 먼저 지배방정식을 유한차분법을 이용하여 지배방정식을 이산화시키고 이를 통계적 수법에 해당하는 MCM(Monte Carlo Method)을 적용하여 마이크로 소자의 열전달 특성을 효과적으로 해석하는 예도 늘고 있다[5,6].

MCM은 난수를 이용해서 시스템 입자의 이동 여부를 결정하고 계속해서 미시적인 상태를 작성하여 가는 수법으로서 주로 평형 통계역학, 즉 열역학적 평형상태에 있는 시스템에 대한 시뮬레이션 방법으로 재조명 받고 있다. 또한 MCM은 확률계산을 동반하여 적분 등의 계산을 난수를 이용해서 근사적으로 수행하는 방법으로 사용되기도 한다. 이 경우 근사오차가 발생하며, 계산의 효율성이 반드시 높다고 할 수 없으나 범용성 측면에서 실장이 간단하다는 이점도 있다.

일반적으로 고성능 마이크로 흐름센서 설계용 수치 시뮬레이션 툴을 개발하기 위해서는 일반적으로 열전도, 대류, 복사 등의 특성을 고려한 열운송방정식을 토대로 수치 모델을 구축할 필요가 있다. 센서의 물리적 특성을 고려하여 지배 방정식을 이산화 방정식으로 변환하기 위해서 기본적으로 유한차분법을 도입하여 사전에 정식화를 수립하고 다음 단계에서는 통계 및 확률적인 개념에서의 새로운 이산화 방정

식을 도출할 필요가 있다.

일반적으로 열전달 문제를 해석하기 위해서는 다음과 같이 열전도 및 대류가 평형을 이루는 정상상태에서의 열운송 방정식(Thermal Transfer Equation)을 고려할 수 있다. 이 방정식은 필요한 경우 3차원 모델링을 위한 보다 일반화된 방정식으로 표현 가능하다[1,5].

$$u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}=\alpha\left(\frac{\partial^2 T}{\partial^2 x}+\frac{\partial^2 T}{\partial^2 y}\right) \tag{1}$$

위 식에서 $(u,\ v)$는 대류 전달속도, α는 열전달확산계수를 의미한다. 식 (1)에 대해서 좌변의 편미분 항들은 후진차분을, 우변의 항들에 대해서는 중앙차분을 적용하여 정리하면 다음과 같이 온도분포를 나타내는 차분 방정식으로 표현된다.

$$T(x,y)=p_{x+}T(x+\Delta,y)+p_{x-}T(x-\Delta,y)+p_{y+}T(x,y+\Delta)+p_{y-}T(x,y-\Delta) \tag{2}$$

여기서 각 격자점은 등간격으로 분할하는 것으로 가정하여 $\Delta=\Delta x=\Delta y$로 두었다. 식 (2)에서 p_*는 차분 방정식의 계수로서 다음과 같이 정리된다.

$$p_{x+}=p_{y+}=\frac{1}{C_x+C_y+4} \tag{3a}$$

$$p_{x-}=\frac{1+C_x}{C_x+C_y+4},\quad p_{y-}=\frac{1+C_y}{C_x+C_y+4} \tag{3b}$$

$$C_x=\frac{u\Delta}{\alpha},\quad C_y=\frac{v\Delta}{\alpha} \tag{3c}$$

여기서 식 (3)은 입자가 각각 (x+△, y), (x-△, y), (x, y+△), (x, y-△) 방향으로 이동할 확률을 나타내고 있다. 즉 p_{x+}, p_{x-}, p_{y+}, p_{y-}는 시작점(일반적으로 공간내 구하고자하는 포텐셜)에서 주변 경계점으로 이동할 통계학적인 확률해석을 의미한다. 따라서 각 입자의 이동이 경계면에 도달할 때의 모든 포텐셜의 합을 평균으로 취하면 주어진 경계조건을 만족하는 온도분포를 다음과 같이 계산할 수 있다.

$$T(x,y)=\frac{1}{N}\sum_{i=1}^{N}T_p(i) \tag{4}$$

여기서 N은 입자가 경계면까지 도달한 횟수를 나타내며 수치해가 수렴하기 위해서는 워킹(walking) 수가 많을수록 보장된다.

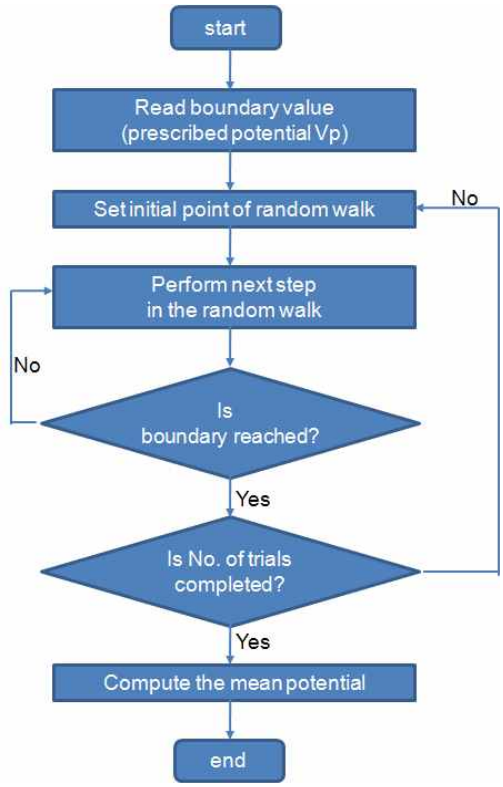

그림 1. 계산 알고리즘
Fig. 1. Computing algorithm.

식 (2)를 기초하여 MCM을 적용한 계산 알고리즘의 흐름도는 그림 1과 같다.

2-2 스레드를 이용한 분산 컴퓨팅

병렬처리 프로그램(Parallel Program)은 메모리 모델로 분류하면 크게 공유 메모리 모델(Thread model)과 분산 메모리 모델(Message passing model)로 분류할 수 있다.

공유 메모리 모델의 경우, 복수의 프로세스 및 스레드가 실행의 주체가 되고 서로 통신 및 동기를 맞춰가며 계산이 수행되는 모델을 의미한다. 그러나 스

레드간 실행순서(동기화)를 프로그램이 보증하지 않는 경우에는 원하는 결과를 얻지 못하는 단점이 있다.

메시지 패싱 모델(Message Passing Model)에서는 복수의 프로세스가 메모리 공간을 독립적으로 가지면서 연산을 수행할 수 있다. 프로세스 사이에서 데이터 통신의 필요가 발생하면 송신측(sender)과 수신측(receiver) 사이에서 서로 통신 채널을 연다. 이때 쌍방의 준비가 되었다는 것을 동시에 확인 가능하므로 특별한 동기화 방안을 고려하지 않아도 된다. 단, 고속화를 의도한 비동기 통신을 수행하므로 이 경우에는 동기를 확인할 필요가 있다.

본 연구에서는 Java 언어에서 제공하는 스레드를 활용하여 메시지 패싱 모델을 채용하여 연산에 필요한 자원을 네트워크에 존재하는 유휴 PC 자원으로 작업요청을 하고 그 결과를 취합하여 고속으로 연산을 수행할 수 있는 병렬처리 플랫폼 구축을 목적으로 한다.

그림 1의 알고리즘을 토대로 병렬화를 시도하기 위해서는 문제의 초기화를 제외한 나머지 축차연산 과정을 하나의 작업 태스크 영역으로 분리하고 이를 원격지에 있는 네트워크 자원에 연산 요청을 하여 그 결과를 받아오는 것이 바람직하다. 따라서 각 작업단위의 규모와 분할방법에 따라서 연산의 규모가 결정될 수 있으며 작업의 병렬화를 통한 고속화 연산이 가능하다.

본 연구에서는 그림 2와 같은 계산 영역을 대상으로 외부 포텐셜이 경계조건으로 주어져 있는 포텐셜 문제를 고려하여 병렬화를 시도하였다. 그리고 문제의 단순화를 위해 식 (1)에서 흐름에 기여하는 속도항 u와 v는 무시하고 정상상태에서의 포텐셜 문제로 제한하였다.

그림 2의 영역은 포텐셜 계산을 위해 내점(internal point) 단위로 이산화를 수행하고 각 행 단위로 작업 태스크를 정의하여 네트워크 자원들에게 작업 요청을 할 수 있다. 네트워크에 존재하는 PC 자원들은 이들 작업 태스크를 기본 단위로 요청받고 그림 1에 주어진 계산 알고리즘을 수행하여 작업결과를 처리하여 마스터 PC로 반환하게 된다. 여기서 마스터 PC의 역할은 네트워크 PC 자원들에게 작업을 분배하고 이를 취합하는 역할을 수행하게 된다. 또한 네트워크에 있는 PC 자원들은 별도의 스레드 단위로 대기하고 있다가 마스터 PC의 작업요청에 응답하는 클라이언트 역할을 담당하게 된다. 이러한 병렬화를 과정을 나타내면 그림 3과 같다.

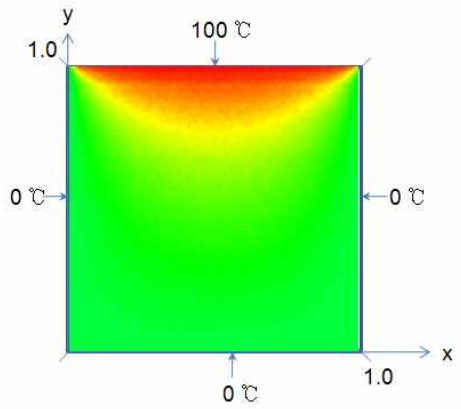

그림 2. 병렬화 대상 포텐셜 문제
Fig. 2. Potential problem for distributed computing.

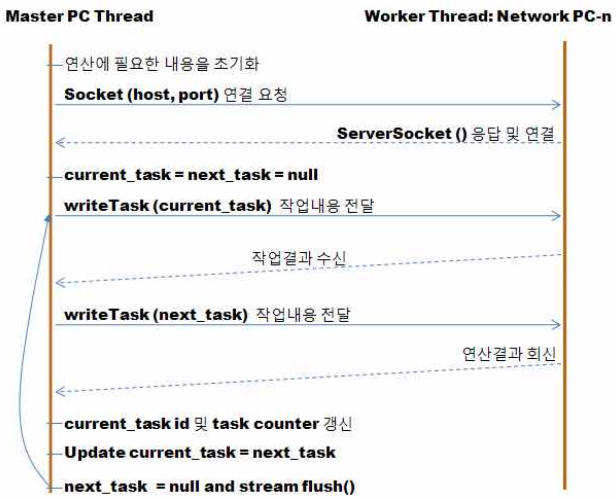

그림 3. 네트워크 기반 작업 분배 및 처리과정
Fig. 3. Task distribution and processing on remote networking.

Ⅲ. 성능 시뮬레이션 및 실험 결과

최근 멀티코어 프로세서들이 범용 PC 뿐만 아니라 임베디드 시스템에 탑재되어 그 사용이 보편화되고 있다. 본 연구에서 제시하는 병렬화 방안에 대한 성능을 시험하기 위해서 그림 2에 주어진 포텐셜 문

제를 대상으로 실험환경을 두 가지로 제한하였다.

첫째, 2-코어 PC에서 네트워크의 클라이언트 PC들의 역할을 멀티 스레드 환경으로 대체하여 실험을 수행하였다. 이때 연산에 필요한 스레드의 수에 따른 연산수행 시간을 측정하여 스레드 수의 증가에 따른 속도향상비를 측정하여 단일 PC에서의 병렬화 가능성을 확인하였다

두 번째 실험환경은 네트워크에 필요한 PC 자원들이 존재하고 마스터 PC에서 분배하는 각 태스크의 요청작업을 클라이언트 PC들이 담당하여 처리하고 그 결과를 반환하도록 하였다. 단, 네트워크에 존재하는 모든 PC들은 동일 네트워크 그룹으로 설정되어 있는 것으로 가정하였다.

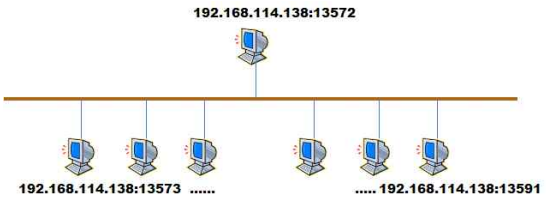

그림 4. 단일 PC에서의 실험모델
Fig. 4. Experimental model on single PC.

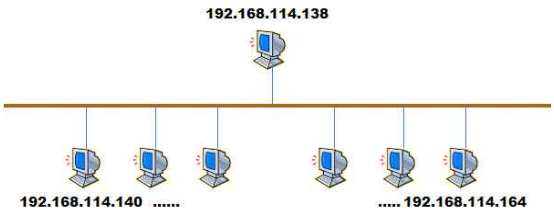

그림 5. 네트워크 기반 실험모델
Fig. 5. Experimental model on remote networking.

각 실험에서 측정된 속도향상비는 다음과 같이 정의하였다. 여기서 t_0는 기준 평가시간으로서 단일 PC에서 모든 작업을 하나의 독립된 스레드환경에서 처리하는데 소요된 시간을 의미한다. 또한 t_m은 별도의 멀티스레드(네트워크에 존재하는 클라이언트도 포함)의 수에 의존하여 평가된 시간을 의미한다.

$$\text{속도향상} = t_m / t_0 \tag{5}$$

위에서 언급한 두 가지 시나리오에 따른 실험에서 요구되는 네트워크 및 PC 사용환경은 표 1과 같다.

표 1. 실험환경 및 네트워크
Table 1. Experimental environment and network.

OS	Microsoft Windows XP Professional SP3
CPU	Intel(R) Core(TM)2 Duo CPU E6550 @2.33GHz
RAM	2GB
개발 툴	JDK(Java 6.0)
네트워크	T3 기반(외부 라우팅 없음)
사용 PC	26대
이산화	100 × 100

3-1 단일 PC상에서의 성능측정

실험에 사용한 단일 PC는 2-코어 멀티프로세서를 내장하고 있다. 그림 4에서 알 수 있듯이, 실험을 위하여 마스터 PC와 네트워크 자원을 네트워크의 포트번호 단위로 구분하여 네트워크를 경유하지 않은 상태에서 작업 스레드를 발생시켜 병렬화를 대신하였다.

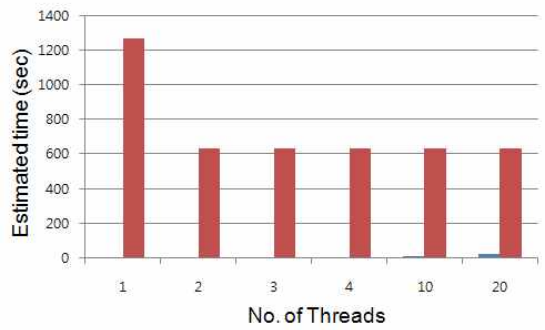

그림 6. 단일 PC에서의 평가시간 측정 결과
Fig. 6. Estimated time on single PC.

실험에서 멀티스레드를 고려하지 않고 단독으로 계산을 수행하여 평가된 시간은 1266.515초가 소요되었다. 이를 기준으로 둘 이상의 멀티스레드를 이용하여 연산을 수행한 결과는 그림 6에 나타내었다. 여기에서 알 수 있듯이 멀티스레드를 둘 이상 발생시켜 연산을 수행하더라도 기준 평가시간의 절반에 해당하는 633.61초 정도 소요되어 병렬화의 효과는 약 2배 정도로서, 이는 PC의 CPU가 2-코어인 점을 감안할 때 두 개의 CPU 코어를 모두 사용하여 얻어진 효과로 판단된다. 즉 계산을 단독으로 수행할 경우는

하나의 CPU 코어만을 사용하는 것으로 추정되며 단일 PC상에서 아무리 스레드의 수를 증가시켜도 병렬화에 따른 속도 향상이 기대하기 어렵다는 점을 시사하고 있다.

3-2 네트워크 PC를 이용한 성능측정

첫 번째 실험모델과 마찬가지로 이번에는 마스터 PC가 작업할당을 담당하여 네트워크에 존재하는 유휴 PC들이 각 태스크를 처리하도록 하여 실험을 수행하였다(그림 5 참조). 계산에 참여하는 네트워크의 PC들의 수를 증가시켜가며 연산속도를 측정한 결과는 그림 7에 나타내었다.

결과에서 알 수 있듯이, 네트워크의 PC 수가 증가하면 연산시간이 비례하여 속도향상에 기여하는 것을 볼 수 있다. 네트워크 PC를 25대 이용한 경우는 속도향상비가 23.62배였으며, 이상적인 경우라면 25배 속도향상이 기대되지만 마스터 PC와 클라이언트 PC 사이에서 제어용 패킷의 처리에 일부 시간이 사용되었음을 의미한다.

또한 할당된 작업을 처리하는 알고리즘의 연산과정은 대부분 축차연산과정으로 구성되어 있어 충분한 병렬화의 효과가 나타난 것으로 판단된다. 따라서 본 연구에서 제안한 병렬화 방법을 통하여 규모가 큰 계산모델을 대상으로 적용하여도 충분한 병렬화가 기대된다.

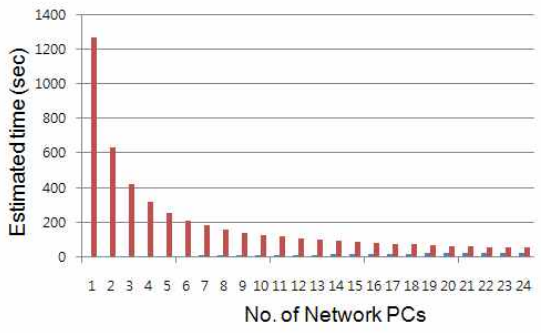

그림 7. 단일 PC에서의 평가시간 측정 결과
Fig. 7. Estimated time on single PC.

Ⅴ. 결 론

본 논문은 마이크로 흐름센서를 설계하기 위하여 열운송 방정식을 MCM법을 적용하여 계산하기 위한 병렬화 방안을 제시하였다. 병렬화를 위하여 자바 멀티 스레드를 이용하여 메시지 패싱 모델을 구현하고 자체 네트워크 프로토콜에 의한 작업할당과 분담처리를 통하여 네트워크에 존재하는 유휴 PC 자원을 활용하고, 네트워크 PC의 수 n을 증가시키는 것만큼 병렬화에 따른 속도 향상이 약 n배에 가까운 선형적인 특성을 보였다. 이를 통하여 연산규모가 큰 계산모델에 적용 가능함을 보였다.

향후 연구과제로서 실제 네트워크 환경을 고려하여 동일 네트워크 그룹으로 형성되어 있지 않더라도 외부 네트워크에 존재하는 PC 자원을 이용한 병렬화에 대한 성능분석과 마이크로 흐름센서 모델에 대한 대규모 연산모델 적용을 검토할 예정이다.

참 고 문 헌

[1] Wan-Young Chung and Tae Yong Kim, "A 2-Dimensional Micro Flow Sensor Properties," *Rare Metal Materials and Engineering, vol. 35, no. z2*, pp. 590-594, December 2006.
[2] http://www.koreaathome.org/(코리아앳홈)
[3] http://www.oracle.com/technetwork/java/index.html.
[4] 강철구 역, 멀티코어를 100% 활용하는 자바 병렬 프로그래밍, *에이콘*, 2008.
[5] J. P. Holman, Heat Transfer 7th ed., McGRAW-HILL BOOK COMPANY, 1992.
[6] S. Kamiyama and A. Satoh, Monte Carlo Simulation, Asakura Pub.(Japanese ed.), 1997.

김 태 용 (金泰用)

1993년 2월 : 부경대학교 전자공학과 (공학사)
1997년 3월 : 오카야마대학 전기전자공학과(공학석사)
2001년 3월 : 오카야마대학 자연과학연구과(공학박사)
2002년 3월~현재 : 동서대학교 컴퓨터정보공학부 교수
관심분야 : 무선통신, 수치해석 응용, 마이크로 센서, 센서네트워크