Journal of Digital Contents Society Vol. 10 No. 2 Jun. 2009(pp. 249-258)

PDA 클러스터 컴퓨팅을 활용한 대용량 모바일 소프트웨어 실행

민혜린*, 이종우**

요 약

최근 무선 인터넷 시장의 발전으로 모바일 단말기를 이용한 응용 개발이 늘어나고 있다. PDA 같은 모바일 장치는 유비쿼터스 컴퓨팅이라는 장점으로 인해 컴퓨팅을 요구하는 다양한 환경에서 필수적인 요소로 자리 잡고 있다. 본 논문의 목적은 PDA 클러스터 시스템을 이용해 PDA 단독으로는 실행시킬 수 없었던 대용량 소프트웨어를 PDA 상에서 실행시키는 것이다. 구체적인 구현 방법으로는 기존 워크스테이션 클러스터 컴퓨팅 엔진 JPVM을 PDA로 이식한 버전인 PDA-JPVM을 이용하였다. PDA 클러스터 상에서 병렬 응용 프로그램들을 실행시킨 결과, 이식된 PDA 클러스터 시스템을 이용해 대규모 소프트웨어를 PDA 상에서 실행시킬 수 있음을 확인하였으며, 아울러 그 성능 평가 결과도 보인다.

Running Large-scale Mobile Software using PDA Cluster Computing

Minhye Rhyn*, Jongwoo Lee**

Abstract

As wireless internet markets become larger than before, many mobile applications are also being developed actively. In this circumstances mobile devices such as cell phones, PDAs are playing an important role to satisfy the user's need of ubiquitous computing. Due to the hardware limitations, however, the mobile devices like PDA can not run large-scale softwares by itself. The main goal of this paper is to make large-scale applications runnable on PDA. To accomplish this, we used the PDA-JPVM cluster computing engine which has been already developed by us. We found out by running the applications and the performance evaluation that large-scale Java softwares can easily run on the hardware-limited PDA. And the performance evaluation results are also presented.

Keywords : Cluster Computing, PDA, Parallel Virtual Machine, Java Parallel Virtual Machine, Performance Evaluation

1. 서론

유비쿼터스 스마트홈 시대를 맞아 다양한 운영체제를 탑재한 PDA의 사용이 늘어나고 있다. 이에 따라 윈도우즈 모바일 PDA(이하 WM-PDA) 응용 개발을 위한 프로그래밍 환경에 대한 요구도 C/C++에서 JAVA까지 다양해지고 있다. 모바일 단말기는 때와 장소의 제약을 극복할 수 있다는 장점으로 인해 컴퓨팅을 요구하는 다양한 환경에서 필수적인 요소로 자리 잡고 있고, 아울러 무선 통신 서비스의 급증은 이러한 모바일 디바이스를 하나로 묶어주는 역할을 하고 있다. 이 같은 상황에도 불구하고 PDA의 하드웨어적 한계로 인해 PDA에서 실행시킬 수 있는 프로그램의 규모에는 제약이 있어 왔는데, 본 연구에서는 이런 PDA의 제약을 해소하여 대용량의 소프트웨어도 PDA에서 실행시키고자 하였다. PDA 클러스터 컴퓨팅 구현이 바로 이러한 본 논문의 목표를 달성하기 위한 최선의 방법이

※ 제일저자(First Author) : 민혜린
접수일:2009년 04월 14일, 완료일:2009년 06월 16일
* 숙명여대 멀티미디어과학과 석사과정
minuh82@hanmail.net
** 숙명여대 멀티미디어과학과 부교수(교신저자)
bigrain@sookmyung.ac.kr
▣ 본 연구는 2009년 서울산업통상진흥원의 전략산업혁신클러스터육성지원사업(10581C093104)으로 수행되었습니다.

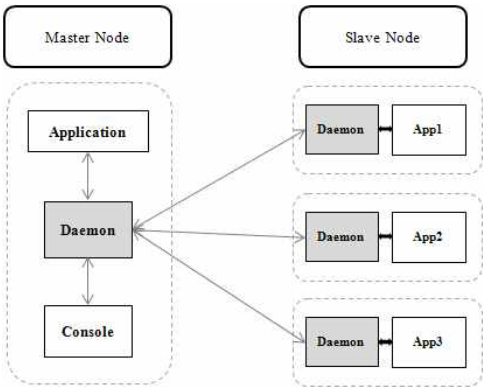
라고 판단한 본 연구진은 응용의 호환성까지 고려해 자바로 구현된 병렬 프로그래밍 엔진인 JPVM[1]을 이식 대상으로 선택하고 이를 PDA에 이식한 바 있으며[2], 그 과정에서 PDA에 자바 프로그램을 이식하는 것이 쉽지 않음을 발견하고 이를 해결하기 위한 방법 또한 이미 연구하였다[3].

이러한 기존 연구를 바탕으로 본 논문에서는 PDA 클러스터 시스템 상에서 대용량 소프트웨어를 실제로 실행시킬 수 있음을 보인다. 이를 위해 다양한 유형의 PDA(+PC) 클러스터 시스템을 구축하고 그 상에서 병렬 응용이 실행됨을 보인다. 아울러 성능평가 결과도 보인다.

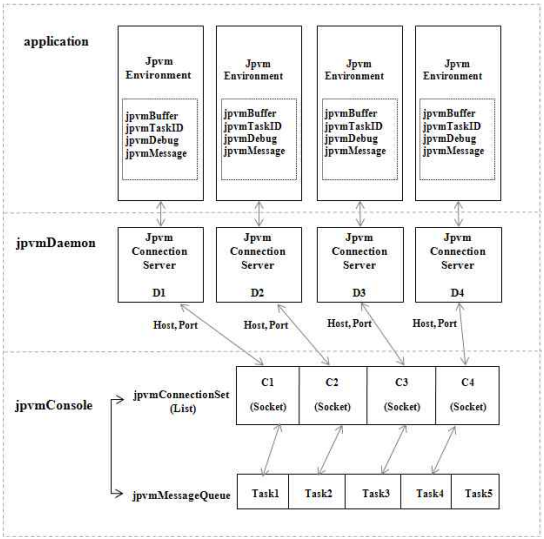
본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 기존 연구에 대해 살펴보고, 3장에서는 PDA 클러스터 상에서 실행시킨 벤치마크 프로그램들의 실행 모습을 보인다. 4장에서는 3장에서 실행시킨 병렬 벤치마크 프로그램들의 성능 비교 결과를 보이고, 5장에서는 결론과 향후 연구과제에 대해 논한다.

2. 기존 연구

JPVM[1]은 자바로 구현된 네트워크 병렬 컴퓨팅 시스템으로써, JPVM에서 제공하는 프로그래밍 인터페이스는 PVM[4]의 인터페이스와 비슷하지만 네트워크 병렬 프로그래밍을 위한 언어로 자바를 사용함으로써 자바의 플랫폼 간 호환성 장점을 최대한 활용하고 있다. TCP를 기반으로 구현된 JPVM 시스템 구조는 그림 1과 같다. JPVM에 참여하는 워크스테이션 노드들은 마스터 노드 하나와 여러 개의 슬레이브 노드로 나뉜다. 마스터는 모든 슬레이브 노드 상의 데몬들과 통신하며, 클러스터의 태스크 분배와 슬레이브 노드들을 관리하고 결과를 모아 출력하는 작업을 수행한다. 슬레이브 노드에서는 마스터 노드에서 태스크를 받아 작업을 수행하고 결과를 데몬을 통해 다시 마스터 노드에게 전달한다. JPVM의 좀 더 구체적인 흐름도를 살펴보면 그림 2와 같다. JPVM의 중요 흐름은 크게 데몬, 콘솔, 응용의 세 부분으로 나눌 수 있다. 콘솔은 호스트 추가 기능과 클러스터를 구성하는 호스트들의 상태정보 출력, 데몬의 강제 종료 등을 담당한다. 또한, 사용자가 콘솔을 통해 입력을 보내면 데몬에 소켓을 반환하고 메시지 큐를 이용하여 대기중인 태스크를 처리한다. 데몬은 JPVM에서 마스터 노드와 슬레이브 노드와 관련한 모든 연결 기능을 담당한다. 콘솔에서 추가한 호스트를 목록에 추가하여 클러스터를 재구성하고, 태스크를 생성하여 jpvmDaemonMessage- Tag에서 미리 정수로 정한 태그로 요청 내용을 구분하여 요청 기능을 호출하는 역할을 한다. 데몬은 JPVM의 메인(Main) 함수를 가지고 있으며, jpvmConnection 서버와 연결되어 jpvmEnvironment를 실행 호출하고 응용을 실행시킨다. 마지막으로, 응용은 PVM API를 이용해 작성되는데, JPVM에서 제공하는 PVM API는 jpvmEnvironment와 jpvmBuffer 클래스, 그리고 jpvmTaskId 클래스에 구현되어 있다.



(그림 1) JPVM 시스템 구조



(그림 2) JPVM의 시스템 흐름도

JPVM은 PVM 응용의 C, Fortran과 유사한 인터페이스를 지원하면서도 자바로 구현되어 있고, 응용프로그램도 자바로 작성되어야 한다. JPVM API는 PVM과 비슷하여 PVM에 익숙한 프로그래머가 자바환경에서 PVM 시스템을 구축하기 쉽도록 해준다. 동시에, JPVM은 PVM에서는 제공되지 않던 스레드 안정성, 태스크 당 다중 통신 종단점, 직접 메시지 루틴과 같은 새로운 특징도 제공한다. JPVM은 전부 자바로 구현되어 있어서 자바 가상 머신을 제공하는 플랫폼 사이에서 높은 이식성을 나타내는데, 이점이 바로 [2]에서 JPVM을 PDA 클러스터 컴퓨팅 시스템을 위한 기반 시스템으로 사용한 이유이다. 자바로 작성된 PDA 클러스터 컴퓨팅 응용이 실행 가능하다면 대규모 응용을 PDA에서 실행시킬 수 있음과 동시에 호환성이 뛰어난 PDA 응용도 작성할 수 있기 때문이다.

이외에도 많은 PC 또는 워크스테이션용 병렬 프로그래밍 엔진들이 있지만[5, 6] PDA 클러스터 컴퓨팅을 위해 모바일 단말기에 이식하기에는 모두 그 엔진 크기가 큰 것들이어서 이식 대상으로는 적절하지 않다고 할 수 있다.

기존 연구 중 PDA 상에서의 대용량 소프트웨어 실행에 관한 연구는 찾을 수 없었는데, 이는 PDA의 경우 소프트웨어적 해결책 보다는 하드웨어적 해결책을 더 선호하기 때문인 것으로 판단된다. 따라서 기존 연구 중 본 연구와 그 목표가 일치하지는 않지만 관련 있는 것들 중 모바일 클러스터 컴퓨팅에 관한 기존 연구를 소개하면 다음과 같다.

Moset[7]은 유무선 네트워크로 연결되는 클러스터로써 모바일 노드와 정적 노드를 통합하기 위해 운영체제 및 네트워크 프로토콜 수준에서 모바일 단말기가 갖는 제약과 이질성을 해결하고자 하였다. Moset에서는 모바일 호스트(MH)들과의 통신 신뢰도를 높이는 데에 그 주안점이 있다. 인접한 몇몇 모바일 호스트들을 묶어 CELL로 엮어두고 중재자를 거쳐 서로 통신하게 하고 있다. 데이터 전달의 신뢰도를 위해 모바일 단말기의 제약을 고려하여, Moset은 신뢰성 있는 멀티캐스트 프로토콜을 제안하고 있다. 따라서 모바일 노드와 정적 노드간의 통신 프로토콜에 초점을 맞춘 연구라고 할 수 있으므로, 모바일 클러스터 컴퓨팅을 위한 런타임 라이브러리를 활용하는 본 논문과는 크게 관련이 없다고 할 수 있다.

하이브리드 클러스터 컴퓨팅[8]에서는 유무선으로 연결된 모바일 장치와 정적 컴퓨터로 구성되는 클러스터를 하이브리드 클러스터라고 칭하고 있는데, 하이브리드 클러스터를 구성하기 위한 자바 모바일 객체와 모바일 IP 시스템을 제안하고 있다. 전체 시스템은, “컴퓨터와 하이브리드 클러스터 컴퓨팅을 지원하는 네트워크”와 “무선 모바일 IP와 모바일 객체를 지원하는 클러스터 미들웨어”, 그리고 “병렬 분산 응용프로그램”으로 구성된다. 이 연구는 이 세 부분 중 클러스터 미들웨어를 중점으로 설명하고 있고 모바일 장치로는 휴대폰이 아닌 노트북을 대상으로 하고 있다. 따라서 이 연구도 역시 본 논문과는 초점이 다르다고 하겠다.

한편 본 연구진은 [3]에서 WM-PDA로의 J2SE 자바 프로그램 기법에 대해 연구했으며, 이를 기반으로 J2SE 자바 코드로 코딩되어 있는 JPVM을 PDA로 이식할 수 있다고 판단했고, 실제로 [2]를 통해 JPVM을 PDA로 이식하는 데에 성공하였다. [2]에서 기술한 이식 과정 중 수정된 부분도 있고 추가된 부분도 있지만 본 연구는 PDA-JPVM의 최종 구현 버전을 활용하는 것으로부터 시작된 것이므로 여기서는 언급하지 않기로 한다. 본 논문은 [2, 3]의 연장선상에 있는 후속 논문으로써 본 연구진에 의해 기 개발된 PD

A-JPVM을 이용해 PDA 상에서도 대용량 응용 프로그램들이 실행될 수 있음을 보이는 것을 그 목적으로 하고 있다.

3. PDA 클러스터 상에서의 대용량 소프트웨어 실행

이번 장에서는 본 연구진이 [2, 3]에서 이식·구현한 PDA-JPVM을 이용하여 병렬처리를 요구하는 벤치마크 프로그램들을 테스트함으로써 PDA 클러스터 상에서 대용량 소프트웨어가 실행될 수 있음을 보인다. 또한 다양한 기기와의 범용성을 검증하기 위해서 마스터 노드로 PC가 참여하고 슬레이브로 PDA가 참여하는 이기종 플랫폼 클러스터에서의 병렬 처리 데모도 수행하여 성능을 측정한다.

3.1 벤치마크 프로그램 소개

PDA 클러스터 상에서 실행할 병렬 프로그램은 JPVM의 패키지 안에 들어있던 예제들과 본 논문에서 별도로 작성한 응용들로 구성되는데 이들에 대해서 간단히 설명하면 다음과 같다.

첫 번째 프로그램인 **Hello**는 클러스터 네트워크에 있는 마스터와 슬레이브들 간의 메시지 전달을 알아보기 위한 간단한 벤치마크이다. 간단한 숫자 태그를 각 노드 간에 제대로 주고받는지 확인하기 위한 일종의 핑(ping) 테스트이다. 동작 방식은 다음과 같다. 클러스터 내 각 노드에 데몬이 띄워진 상태에서 마스터가 슬레이브 노드들에게 "Hello"라는 메시지를 전달하면, 슬레이브 노드들은 마스터로 응답 메시지를 전달한다. 이러한 에코 형태의 테스트를 통해 Hello는 JPVM의 통신 환경을 점검하고 정상 작동 여부를 확인하는 역할을 한다.

두 번째 벤치마크는 **Mat_Mult**로 간단한 행렬 곱셈을 병렬로 처리하는 프로그램이다. 일반적으로 N * N으로 구성된 행렬 곱셈은 다음과 같은 형태이다.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} \quad \text{수식 (1)}$$

$$c_{i,j} = a_{i,j}b_{i,j} + a_{i+1,j}b_{i,j+1} + \cdots a_{n,j}b_{i,n}$$

수식 (1)과 같이 N * N의 행렬 곱셈 연산을 수행할 때 하나의 결과 행렬의 요소를 계산하기 위해서는 N번의 곱셈과 N-1번의 덧셈이 필요하다. 따라서 전체 결과 행렬을 구하기 위해서는 N^3번의 곱셈과 $N^2(N-1)$번의 덧셈이 필요하므로 $O(n^3)$의 연산 시간이 요구된다. 본 벤치마크에서는 PDA-JPVM을 사용하여 K개의 태스크 노드를 할당하고 각각의 노드에 일정한 크기의 결과 행렬 연산을 분배한다. 각각의 노드는 N^3/K번의 곱셈과 $N^2(N-1)/K$번의 덧셈을 처리하여 총 연산 시간을 줄일 수 있으므로 병렬 처리의 특성을 잘 파악할 수 있다.

세 번째 **Mandelbrot set**은[9] 브누아 만델브로가 고안한 프랙탈(fractal)의 일종으로 비트 매핑된 이미지를 프로세싱하는데 사용된다. 만델브로 집합은 다음 점화식으로 정의된 수열이 발산하지 않는 성질을 갖도록 하는 복소수 c의 집합으로 정의된다.

$$\begin{aligned} Z_0 &= 0 \quad (\text{단}, Z_n\text{은 복소수}) \\ Z_{n+1} &= Z_n^2 + c \end{aligned} \quad \text{수식 (2)}$$

수식 (2)에서 $Z_n = x_n + iy_n$, $c = a + ib$ 의 복소수로 구성되어 있다면, Z_{n+1}은 다음과 같이 계산할 수 있다.

$$\begin{aligned} Z_{n+1} &= x_{n+1} + iy_{n+1} \\ x_{n+1} &= x_n^2 - y_n^2 + a \\ y_{n+1} &= 2x_ny_n + b \quad (\text{단}, x_n,\ y_n,\ a,\ b\text{는 실수}) \end{aligned} \quad \text{수식 (3)}$$

만델브로 집합을 실제로 컴퓨터에서 표현할 때에는 수식 (3)에 주어진 점화식에 따라 Z_n을 계산하면서 수열이 발산하는지 그렇지 않은지를 대수적으로 검사하게 된다. Z_n의 절대 값이 Θ보다 크다면(즉, $x_n^2 + y_n^2 < \theta^2$ 이라면) Z_n은 발산한다고 할 수 있으며, 이 때 c는 만델브로 집합에 속해 있지 않는다. 여기서 Θ는 발산하는 수열의 계산의 범위를 정해주는 경계 값이다. 반면 만델브로 집합 안에 속해 있는 점의 경우 Z_n은 발산하지 않으므로 무한히 계산해도 끝나지 않으므로 일정한 쓰레드를 이용하여 계산을 멈추게 구성한다. 본 벤치마크에서는 이러한 만델브로 집합을 구성하기 위한 좌표 영역을 각각의 태스크에 할당하고 계산함으로써 병렬 수행의 성능을 측정한다.

3.2 PC 클러스터 상에서의 실행 모습

PDA 클러스터에서 실행했을 경우와의 비교를 위해 먼저 PC들로만 구성된 기존 JPVM 클러스터 상에서 벤치마크 프로그램들이 어떻게 실행되는지 그 실행 모습들을 살펴보도록 하자.

3.2.1 Hello 실행결과

(그림 3)은 Hello를 총 4대의 PC로 구성된 JPVM 클러스터 상에서 실행시킨 결과 화면이다. 4개 노드간의 메시지 전달이 원활히 이루어진 것을 확인할 수 있다.

```
C:₩jpvm>java hello
Task Id: NODE1, port #3914
Worker tasks:
        NODE1, port #3916
        NODE1, port #3917
        NODE2, port #1308
        NODE2, port #1311
Got message tag 12345 from NODE1, port #3916
Received: Hello from jpvm task, id: NODE1, port #3916
Got message tag 12345 from NODE1, port #3917
Received: Hello from jpvm task, id: NODE1, port #3917
Got message tag 12345 from NODE2, port #1308
Received: Hello from jpvm task, id: NODE2, port #1308
Got message tag 12345 from NODE2, port #1311
Received: Hello from jpvm task, id: NODE2, port #1311

C:₩jpvm>_
```

(그림 3) PC 클러스터 상에서의 Hello 실행결과

3.2.2 Mat_Mult 실행결과

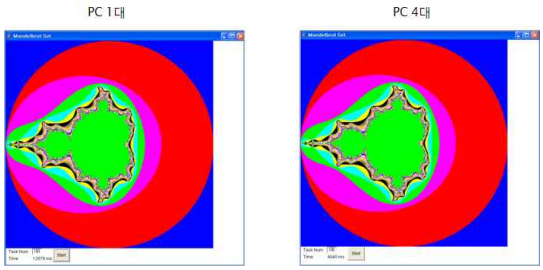
(그림 4)는 4-노드 PC 클러스터 상에서 Mat_Mult를 다양한 입력으로 실행한 결과화면이다. 나중에 PDA에서 실행된 결과와의 비교를 위해 여러 인자 값을 주어 실행하고 그 실행 시간을 측정하였다.

```
C:₩jpvm>java mat_mult 4 2
sss
2x2 matrix multiply, 4 tasks
Total time: 812.0 (mult: 390.0)

C:₩jpvm>java mat_mult 9 3
sss
3x3 matrix multiply, 9 tasks
Total time: 1187.0 (mult: 421.0)

C:₩jpvm>java mat_mult 16 4
sss
4x4 matrix multiply, 16 tasks
Total time: 2093.0 (mult: 672.0)

C:₩jpvm>java mat_mult 25 5
sss
5x5 matrix multiply, 25 tasks
Total time: 3094.0 (mult: 750.0)
```

(그림 4) PC 클러스터 상에서의 Mat_Mult 실행결과

3.2.3 Mandelbrot set 실행 결과

마스터 노드에서 반복 계산하여 600*600 크기의 만델브로 집합 이미지를 생성한다. (그림 5)는 PC 1대에서 만델브로 집합을 실행한 결과와 4-노드 PC 클러스터 상에서의 실행 결과 화면을 보이고 있다. 당연한 이야기지만 이 두 결과 화면은 정확히 일치함을 알 수 있다.



(그림 5) PC 클러스터 상에서의 Mandelbrot set 실행결과

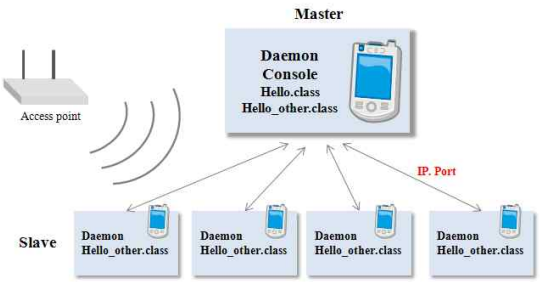
Mandelbrot set은 복정밀도 연산을 병렬 처리하기 때문에 단정밀도 연산인 mat_mult보다 그 계산양이 훨씬 많다고 할 수 있는데, PC의 CPU에는 FPU(Floating Point Unit)가 있어서 수학 계산을 빠르게 할 수 있어 병렬 처리까지 가세하면 손쉽게 그 결과를 볼 수 있다. 그러나 FPU는 x86계열의 PC에는 보조처리기 형태로 장착되어 있지만 PDA에는 장착되어 있지 않다. PDA는 ARM이나 MIPS 계열의 CPU를 사용[10], 32Bit 정수 연산만 지원하며 배정밀도(double) 연산 같은 경우는 일반적으로 PC 보다 40~100배 정도 느린 성능이 나온다[11]. 따라서 PDA 클러스터 상에서의 만델브로 집합 실행 성능은 이 같은 근본적인 성능 차이를 감안하고 비교해야 할 것이다.

3.3 PDA 클러스터 상에서의 실행 모습

3.3.1 Hello 실행결과

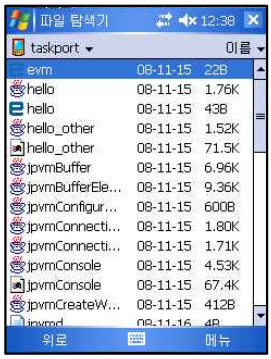
PDA 클러스터를 구성한 모습은 그림 6과 같다. 마스터 PDA에는 데몬 스레드와 콘솔 스레드를 Eve[12] JVM으로 실행시키고, 응용 프로그램인 Hello 클래스는 PJava[13] JVM으로 실행시킨다. 이렇게 마스터 내에서 여러 버전의 자바 가상 머신을 사용한 이유는, 이 방법이 PDA에 J2SE 자바 코드를 이식하기 위해 개발한 기법 중 하나 이기 때문이다. 이에 대한 자세한 사

항은 [3]을 참조하기 바란다.

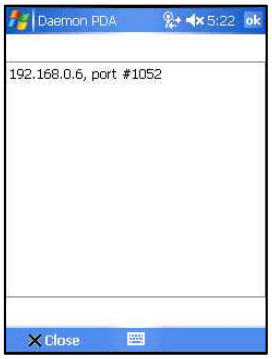


(그림 6) Hello 실행을 위한 PDA 클러스터 구성

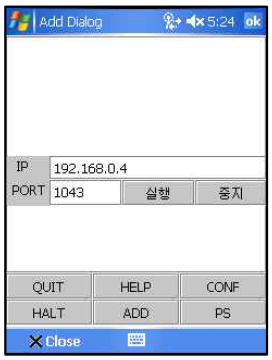
PDA 클러스터를 구성하려면 우선 (그림 7)의 taskport 디렉터리에 있는 파일 중 Eve로 구현한 jpvmDaemon을 실행한다(그림 8). 데몬이 실행된 후 jpvmConsole을 실행하고 (그림 9)와 같이 ADD를 눌러 다른 슬레이브 노드에서 실행 중인 데몬들의 IP 주소와 포트번호를 입력한 후 실행 버튼을 누른다. 이러한 방식으로 (그림 10)과 같이 계속 노드를 추가하면 (그림 11)과 같이 되며, 파일탐색기로 taskport의 Hello를 실행하면 (그림 12)와 같은 결과화면을 볼 수 있다.



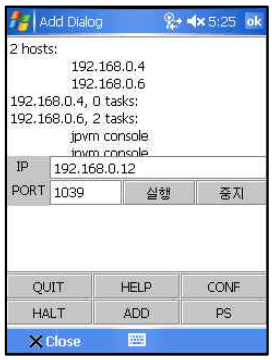
(그림 70 taskport 디렉터리



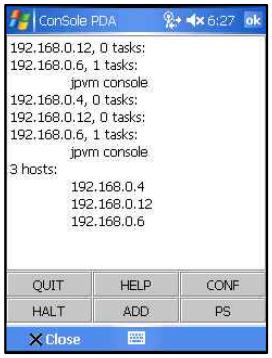
(그림 8) 데몬 실행



(그림 9) 콘솔 Add 실행



(그림 10) 슬레이브 추가

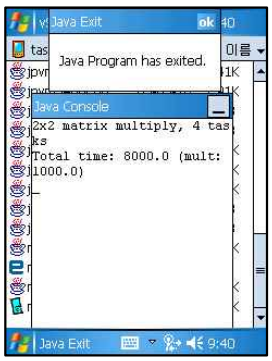


(그림 11) 3개의 호스트 참여

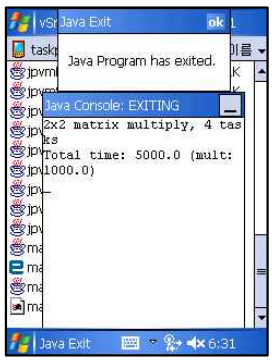
(그림 12) 예제 실행

3.3.2 Mat_Mult 실행결과

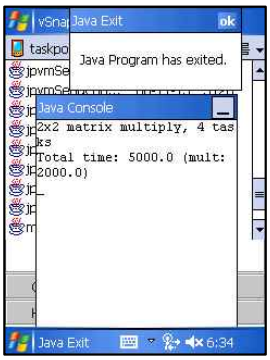
PDA 클러스터에 참여한 PDA 수를 1대에서 3대까지 증가시켜 가며 실험하였으며, (그림 13) ~ (그림 15)가 그 실행 결과를 보이고 있다.



(그림 13) PDA 1대에서의 Mat_Mult 실행 결과



(그림 14) PDA 2대 클러스터에서의 Mat_Mult 실행 결과



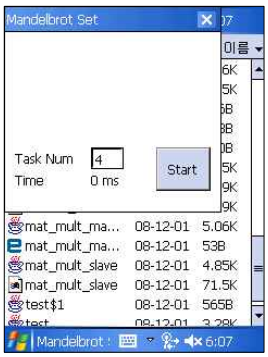
(그림 15) PDA 3대 클러스터에서의 Mat_Mult 실행 결과

이 결과에서 알 수 있는 점은 당연히 PDA 클러스터 상에서의 성능이 PC 클러스터 상에서의 성능보다 크게 떨어지긴 하지만, (그림 13)의 1대의 PDA에서 계산한 결과 8초의 시간이 걸렸는데 (그림 15)의 PDA 노드 3개가 참여한 측정 시간이 5초까지 줄어든다는 점이다. 이 같은 성능에 대한 분석은 다음 절에서 자세히 다루도록 한다.

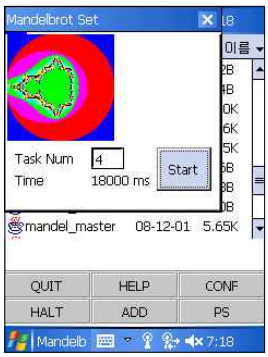
3.3.3 Mandelbrot Set 실행 결과

PC 클러스터에서 실행시켰던 Mandelbrot set을 PDA용으로 이식하여 실행시켰다. 각 PDA마다 데몬을 띄우고 콘솔에서 ADD 버튼을 눌러 호스트를 추가한다. 마스터 PDA에서는 /taskport 폴더에 있는 **mandel-master**를 실행한다. 각 슬레이브 노드에서는 **mandel-slave**를 실행시킨다. 마스터는 슬레이브들과 병행으로 계산을 수행한 후 결과를 모아 출력할 것이다.

(그림 16)은 실행하기 직전 화면을 보이고 있다. 작업 태스크의 개수를 넣어 반복 계산하는 횟수를 늘릴 수 있다. Start를 누르면 마스터 노드는 100*100 크기의 만델브로 집합 이미지를 생성한다. 결과는 (그림 17)과 같으며 최종 프랙탈 이미지뿐만 아니라 총 소요 시간도 출력한다.



(그림 16) PDA 클러스터에서 실행한 만델브로 집합 시작화면



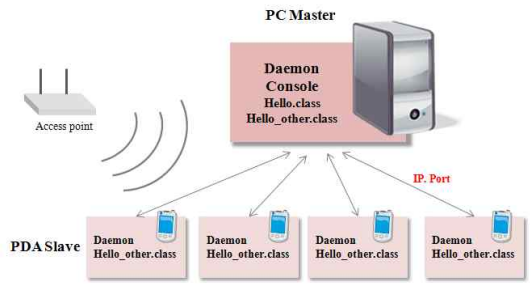
(그림 17) PDA 클러스터에서 실행한 만델브로 집합 결과화면

3.4 PC+PDA 클러스터 상에서의 실행 모습

PC를 마스터 노드로 하고 PDA를 슬레이브로 하는 클러스터를 구성한다. PC와 PDA라는 이기종간의 클러스터를 구현한 것은 자바로 구현된 PDA-JPVM을 여러 기기에 범용으로 이식할 수 있다는 것을 보여주는 좋은 예이다. 또한 PDA의 부족한 계산 처리능력이 PC가 마스터로 클러스터로 구성되면 프로세스 호출 시간이 단축되므로 PDA로만 구성했을 때 보다 좋은 결과를 기대할 수 있다.

3.4.1 Hello 실행 결과

Hello를 PC와 PDA가 혼합된 클러스터 상에서 실행시키기 위해서는 (그림 18)과 같이 마스터인 PC에 **Hello**와 데몬 스레드, 콘솔 스레드를 실행해야하며, 슬레이브 노드인 PDA들에서는 데몬 스레드와 **Hello_other**를 실행시켜야 한다. (그림 19)는 위와 같은 클러스터 구성 하에서 Hello 응용 프로그램을 실행한 결과 화면이다. PC와 PDA 간의 메시지 전달 결과가 정상적으로 동작했음을 확인할 수 있다.



(그림 18) PC와 PDA가 혼합된 클러스터 구성 모습

```
C:₩EveWork₩jpvmJava₩jpvm>java hello
Task Id: 192.168.0.2, port #4085
Worker tasks:
        192.168.0.6, port #1112
        192.168.0.4, port #1037
        192.168.0.12, port #1068
        192.168.0.2, port #4087
Got message tag 12345 from 192.168.0.6, port #1112
Received: Hello from jpvm task, id: 192.168.0.6, port #1112
Got message tag 12345 from 192.168.0.4, port #1037
Received: Hello from jpvm task, id: 192.168.0.4, port #1037
Got message tag 12345 from 192.168.0.12, port #1068
Received: Hello from jpvm task, id: 192.168.0.12, port #1068
Got message tag 12345 from 192.168.0.2, port #4087
Received: Hello from jpvm task, id: 192.168.0.2, port #4087
```

(그림 19) PC와 PDA 클러스터 - Hello 결과화면

3.4.2 Mat_Mult 실행결과

(그림 20)은 PC 1대로만 Mat_Mult를 실행했을 때 모습이고, (그림 21)은 (그림 18)과 같은 클러스터 환경에서 실행한 결과를 보이고 있다. 비록 컴퓨팅 능력이 떨어지는 PDA가 슬레이브로 참여했지만 성능이 월등히 좋아졌음을 알 수 있다.

```
C:₩EveWork₩jpvmJava₩jpvm>java mat_mult_master
2x2 matrix multiply, 4 tasks
Total time: 8672.0 (mult: 3390.0)
```

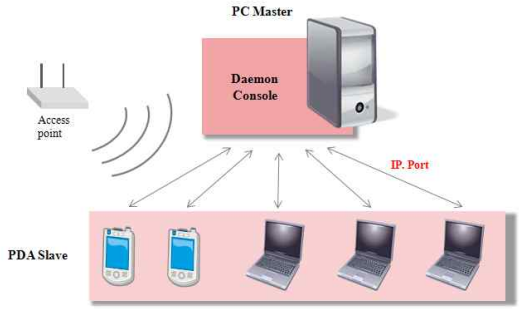
(그림 20) PC 한 대에서 실행한 Mat_Mult 결과 화면

```
C:₩EveWork₩jpvmJava₩jpvm>java mat_mult_master
2x2 matrix multiply, 4 tasks
Total time: 4735.0 (mult: 704.0)
```

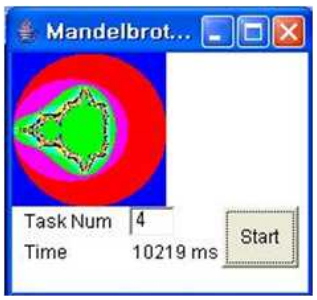
(그림 21) PC+PDA 클러스터에서 실행한 Mat_Mult 결과화면

3.4.3 Mandelbrot Set 실행 결과

Mandelbrot set 응용을 실행하기 위해 (그림 22)와 같이 데스크탑 1대와 노트북 3대, PDA 2대로 클러스터 환경을 구성하였다. 태스크 4개로 결과 이미지를 출력하는데 걸린 시간은 10.219초였는데(그림 23), PC로만 구성된 클러스터에서의 결과보다 현저히 떨어지지만 PDA 만으로 구성한 클러스터보다는 2초 정도 빠르다는 것을 확인 할 수 있었다.



(그림 22) PC+노트북+PDA로 구성된 클러스터



(그림 23) PC+노트북+PDA 클러스터 상에서의 만델브로 집합 실행 결과 화면

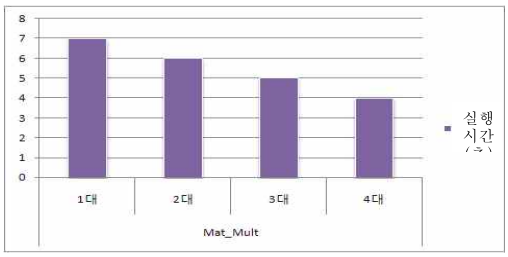
4. 성능 평가 및 분석

PC 클러스터 상에서의 벤치마크 성능은 본 논문의 관심 사항이 아니므로 생략하고, 이번 절에서는 PDA 클러스와 PC+PDA 클러스터 상에서의 성능을 비교하도록 한다. 단, 단순 정상 동작 유무 확인 병렬 프로그램인 Hello는 성능 분석에서 제외하였다.

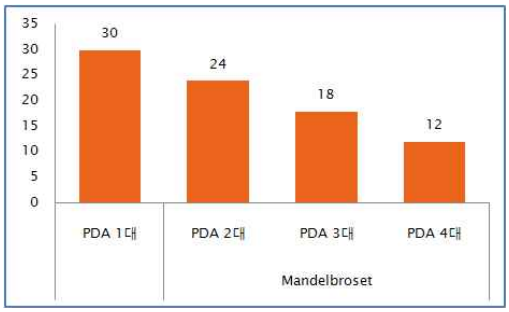
4.1 PDA 클러스터

(그림 24)는 PDA 클러스터 상에서 Mat_Mult 응용을 노드 수를 증가시켜가며 실행시킨 결과를 보이고 있다. PDA 노드 수가 증가함에 따라 실행 시간이 짧아진 다는 것을 확인할 수 있다. 노드 수가 2배가 되었다고 해서 성능도 두 배가 되지는 않는데, 이는 PDA 노드들 간의 통신 오

버헤드가 크기 때문인 것으로 판단된다. 한편 (그림 25)는 Mandelbrot set을 PDA 클러스터 상에서 실행시킨 결과를 보이고 있다. 단일 노드일 때 30초 걸리던 것이 PDA 노드 수가 늘어남에 따라 차츰 수용할만한 시간 안에 실행이 완료되고 있음을 알 수 있다.



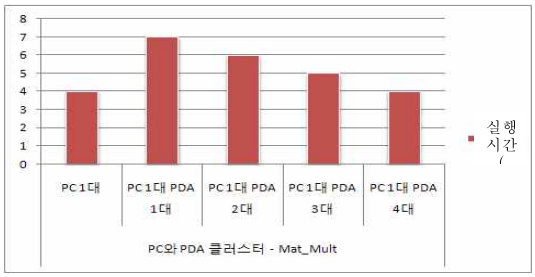
(그림 24) PDA 클러스터 상에서의 Mat_Mult 시간 측정 결과



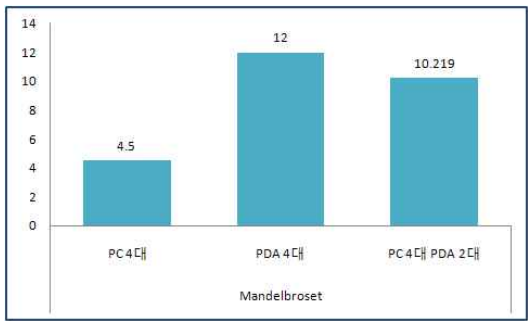
(그림 25) PDA 클러스터 상에서의 Mandelbort set 시간 측정 결과

4.2 PC+PDA 클러스터

(그림 26)은 Mat_Mult를 PC+PDA 클러스터 상에서 실행한 결과를 보이고 있다. PC 한 대로 JPVM 어플리케이션을 실행했을 때보다 PDA 한 대를 추가하면 계산 결과 시간이 좀 더 오래 걸린다는 것을 볼 수 있는데, 이는 PDA의 프로세스 호출 시간과 GUI 화면을 띄우는 시간, 응용을 인보킹하는 데에 걸리는 시간이 등이 필요하기 때문이다. 하지만 클러스터에 PDA 노드들을 지속적으로 추가하면 계산 시간이 단축되는 것을 확인할 수 있다. 또한, 그림 27은 PC 클러스터와 PDA 클러스터, 그리고 PC+PDA 클러스터 상에서 Mandelbrot set을 각각 실행시킨 결과를 비교한 것이다. PDA 클러스터 상에서도 수용 가능한 성능을 보이고 있음을 알 수 있다.



(그림 26) PC+PDA 클러스터 상에서의 Mat_Mult 시간 측정 결과



(그림 27) 각 클러스터를 비교한 Mandelbrot set 시간 측정 결과

5. 결론

본 논문에서는 컴퓨팅 능력이 부족한 모바일 디바이스의 한계를 극복하기 위해 플랫폼 이식성이 높은 Java로 구현된 JPVM을 이용하여 PDA 클러스터를 구현하였다. 본 논문이 가지는 연구 성과는 다음과 같다. 첫째, JPVM을 기반 시스템으로 하여 WM-PDA들로 구성된 PDA 클러스터 시스템은 어떠한 모바일 디바이스에도 이식이 가능하다. 이를 사용하면 PC와 PDA간의 또는 다른 윈도우 모바일이 탑재된 모든 기기 사이의 클러스터링이 가능하므로 유비쿼터스 시대를 맞아 모바일 기기들의 컴퓨팅 능력을 향상시킬 수 있다. 둘째, WM-PDA에서 계산량이 많은 대규모의 어플리케이션을 실행시킬 수 있다는 연구 성과를 이루어 WM-PDA의 컴퓨팅 능력을 높일 수 있었다.

향후 연구과제로는 기존의 JPVM은 태스크를 일정하게 분배하기 때문에 본 논문처럼 성능이 다른 노드들로 구성된 이질적 클러스터인 경우 노드의 컴퓨팅 능력에 따라 태스크를 각각 다르

게 분배해 줄 수 있는 Adaptive JPVM을 연구할 계획이다. 또한 현재 구현상의 미비로 인해 그 실행 결과를 보여주지 못한 PDA(마스터)+PC(슬레이브) 클러스터 구성도 빠른 시일 내에 완성해야 할 과제다.

참고문헌

[1] Adam J. Ferrari, "JPVM: Network Parallel Computing in Java," Technical Report: CS-97-29, 1997.

[2] 민혜린, 이종우. PDA 클러스터 컴퓨팅을 위한 JPVM 구현, 한국디지털콘텐츠학회 학술발표대회 논문집, pp. 337~381. 2008.

[3] 민혜린, 이종우. Windows Mobile PDA로의 JAVA 프로그램 이식 기법, 이식 기법, 한국멀티미디어학회 추계학술발표대회논문집, Vol.9, No.1, pp. 234~238. 2008.

[4] A. Geist, A. Beuelin, J.Dongarra, W.Jiang,R.Manch, and V. Sunderamm, "PVM 3 User's Guide and Reference manual," Oak Ridge National Laboratory, Oak Ridge, Tennesee 37831, Sept. 1994.

[5] The Message Passing Interface(MPI) standard, http://www-unix.mcs.anl.gov/mpi

[6] Adam J. Ferrari. TPVM: Distributed Concurrent Computing with Lightweight Processes, In Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing, Aug. 1995.

[7] M.A Maluk Mohamed, A. Vijay Srinivas, D. Janakiram, "moset:An anonymous remote mobile cluster computing paradigm," Journal of Parallel and Distributed Computing, 2005

[8] L. Cheng, A. Wanchoo, I. Marsic, "Hybrid Cluster Computing With Mobile Objects," The Fourth International Conference/Exhibition on Volume: 2, 2000, Page(s): 909 -914 vol.2

[9] http://en.wikipedia.org/wiki/Mandelbrot_set

[10] T. Lindholm and F. Yellin, "The JavaTM Virtual Machine Specification," Addison-Wesley, 1996.

[11] David Seal, "ARM Architecture Reference Manual," Addison-Wesley, 1996.

[12] Michael L Brereton, "Eve Application Development," May, 2008. http://www.EweSoft.com

[13] 정 동신, "Personal Java를 이용한 EPOC PDA Java Programming," (주)엔슬래시닷컴 개발팀, May, 2001.

민 혜 린

2007년 : 숙명여자대학교 정보과학부 멀티미디어과학전공(이학사)

2009년 : 숙명여자대학교 멀티미디어과학과(석사)

관심분야 : Mobile System Software, Embedded System Software, Mobile Computing

이 종 우

1990년 : 서울대학교 컴퓨터공학과(학사)

1992년 : 서울대학교 컴퓨터공학과 대학원(석사)

1996년 : 서울대학교 컴퓨터공학과 대학원(박사)

1996년~1998년 : 현대전자㈜ 정보시스템사업본부 파장

1998년~1999년 : 현대정보기술㈜ 책임연구원

1999년~2002년 : 한림대학교 정보통신공학부 조교수

2002년~2003년 : 광운대학교 컴퓨터공학부 조교수

2003년~2004년 : 아이닉스소프트㈜ 개발이사

2004년~현 재 : 숙명여자대학교 정보과학부 멀티미디어과학전공 부교수

2008년 : 뉴욕주립대 스토니브룩 Research Scholar

관심분야 : Mobile System Software, Storage Systems, Computational Finance, Cluster Computing, Parallel and Distributed Operating Systems, Embedded System Software