

# Parallel Programming, Homework #3

## 1. 개요

이번 숙제는 주어진 순차 프로그램을 병렬 프로그래밍 기법으로 재구성하여 슈퍼 컴퓨터 (CrayT3E)에서 실행하여 그 결과를 제출 하는 것인데 MPI 기법과 HPF 두 가지 방법으로 실행하여 결과를 보는 것이다.

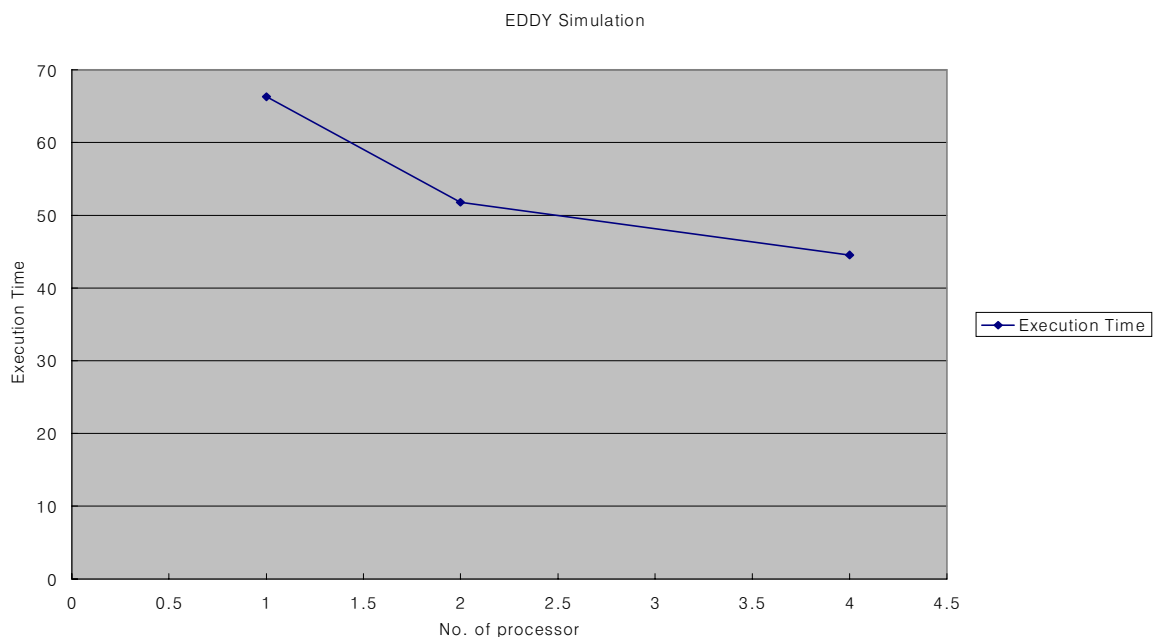
우선 HW#2 에서 분석된 결과를 토대로 시간을 최대한 많이 소모하는 Poission Equation 을 Gaussian iteration 기법으로 실행하는 부분이 프로그램의 성능에 가장 많은 부분을 차지 하므로 이분에 한해서 병렬 프로그램 기법을 적용해 보는 것이 매우 의미 있게 되고 그 실행한 결과를 보이면 된다.

주어진 프로그램은 F77 의 구문 규칙으로 되어 있으며 실행 결과는 매 iteration 마다 FSUM 값을 출력하고 있는데 E-14~15 의 범위의 값이 나온다. GAUSEI 는 널리 알려진 SOR 알고리즘에 해당한다.

## 2. 실행 결과

### 2.1 MPI 적용

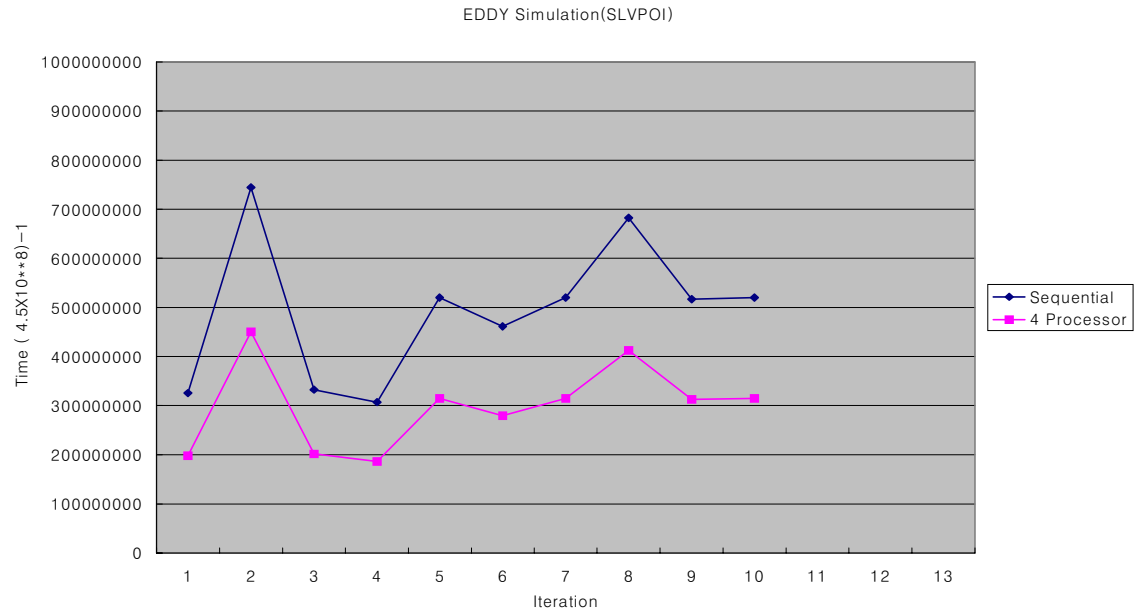
GAUSEI 부분의 계산하는 첫번째 DO Loop 에는 decomposed array 의 boundary 마다 MPI\_send/MPI\_receive 를 사용하여 서로 교환하는 방법으로 변경하였을 때에는 정확한 결과를 만들지 못해서 비교할 수가 없다고 판단되었고 두 번째 Loop 에 해당하는 오차의 범위를 체크하고 total 을 계산하는 부분에 MPI\_ALLREDUCE 를 적용한 때에는 (그림 1)과 같이 정확한 결과를 도출하면서 약간의 개선이 있었다.



( 그림 1 ) MPI 프로그램에서 Processor 수에 따른 수행 시간 비교

이번 프로그램은 SOR 의 성격이 매 반복마다 바로 이전의 결과를 사용하면서 반드시 상하 좌우의 결과를 사용하므로 사실상 병렬화로 분산 시키는 것이 용이 하지 않았다. 따라서 정확한 결과를 만들어 내면서 실행 시간을 단축하기 위해서는 매 수행

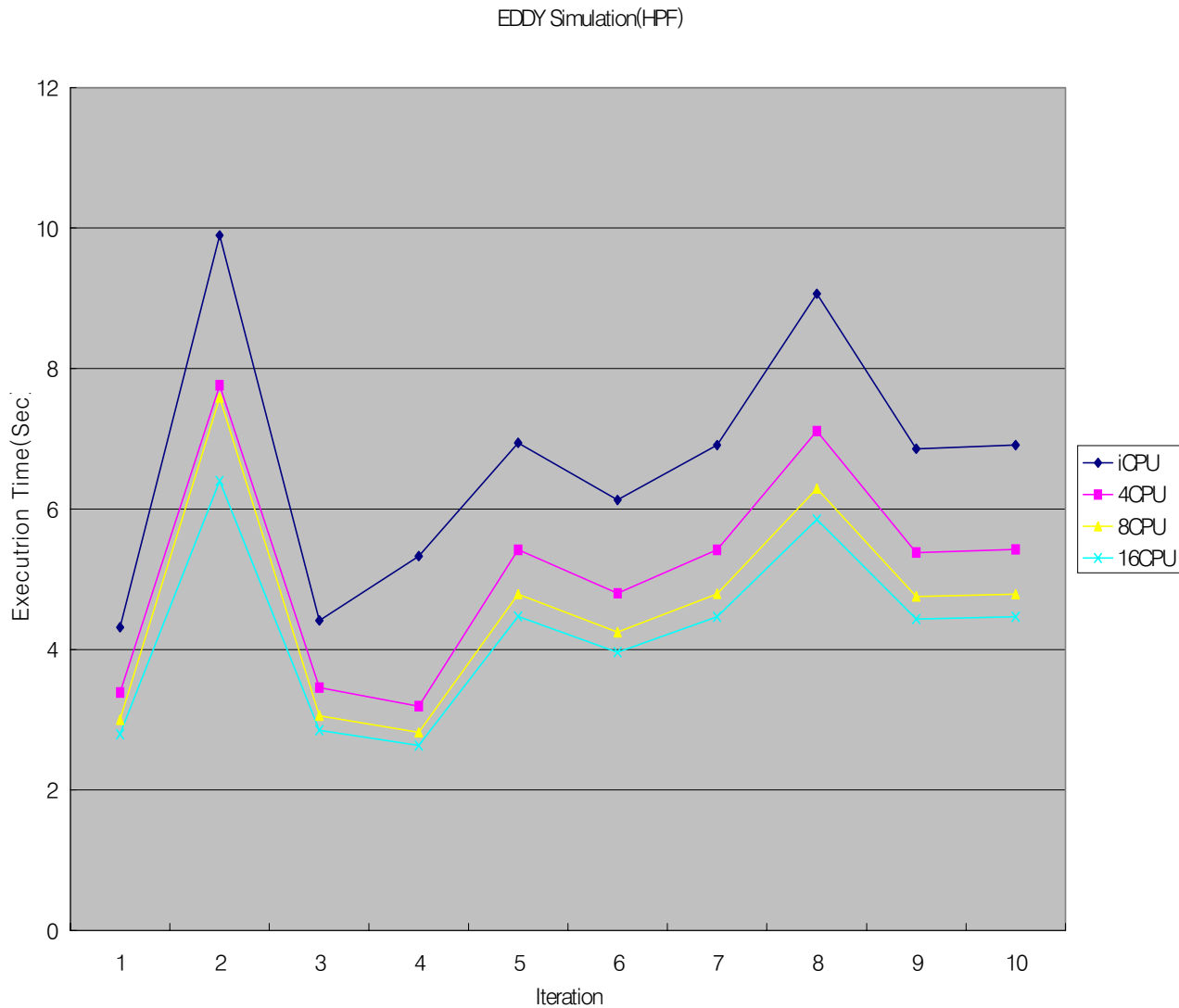
마다 정확한 결과를 내면서 보다 빨리 수렴하는 것이 가능한지 의문이다. 따라서 그림 2 와 같이 Trace 를 하여 개선 폭의 변화를 알아 보았다.



(그림 2) 10 번의 수행 Step 마다의 1CPU Vs 4 CPU 의 실행 시간 추적

## 2.2 HPF 적용

- 1) 순차 프로그램을 그대로 두고 data parallelization 만을 위하여 HPF의 Directive 를 추가하는 것 만으로는 마찬가지로 결과가 상이 하거나 개선되지 않았다. 따라서 두번째 loop 에 functional parallelism 정도의 효과를 보도록 개선하였다.



추가한 HPF DIRECTIVE 는 3 line 으로 아래와 같다.

```
!HPF$ PROCESSORS P(16)
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO P :: FARRAY,PHI,DIF
!HPF$ INDEPENDENT, REDUCTION(TDIF,TOTAL)
```

- 2) 순차 SOR 의 알고리즘을 변형하여 병렬 SOR 알고리즘이 되도록 Red/Black 기법에 의한 SOR 로 재구성하여 실행 결과를 알아 보았다.

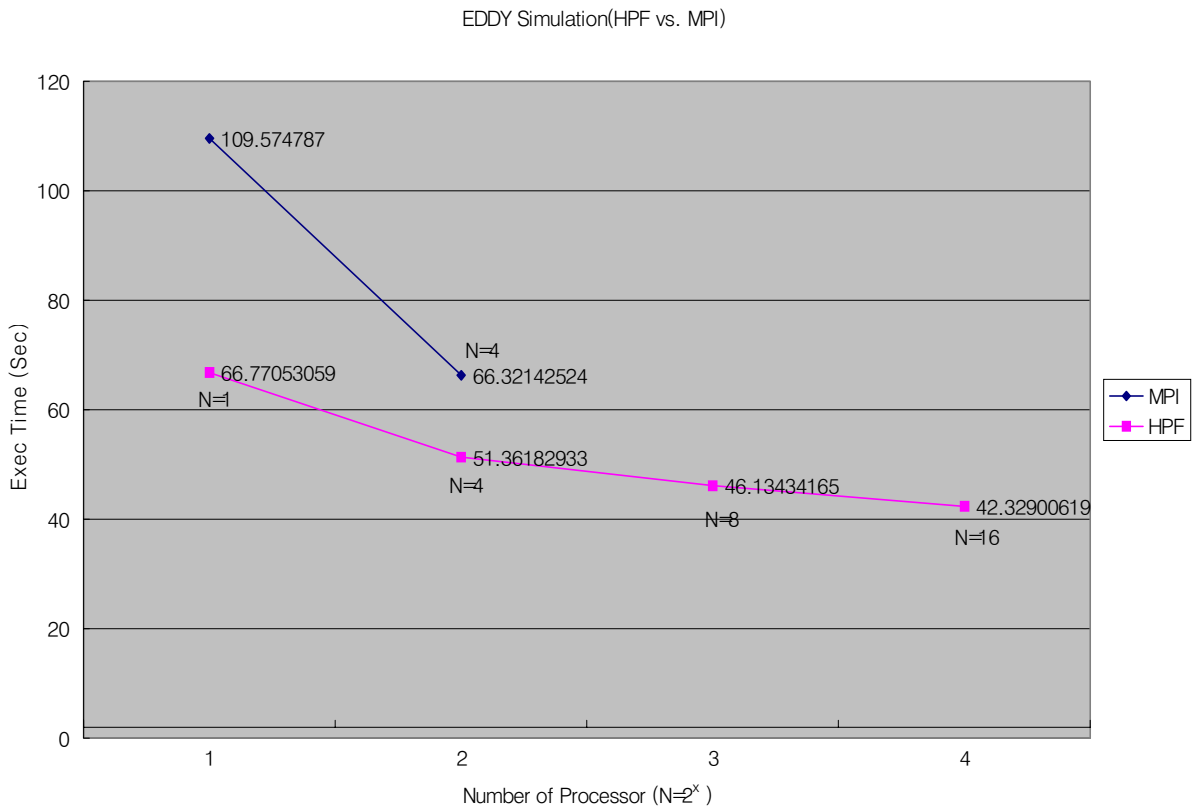
우선 F77 구문을 F90 and HPF 구문으로 변경하고 Red/Black SOR 로 반복 계산 부

분을 변경하였다. (별첨 1 참조) **EDDY의 GAUSEI 만 발췌한 Sample 프로그램에서는 기존의 Sequential SOR 보다 빠르게 수렴하였다. 동일하게 1 CPU 에서도 3 배 이상 빠름**을 확인하고 EDDY Simulation 에 적용한 결과는 동일한 Boundary Check 와 Iteration tolerancy 로서는 첫번째 iteration 을 제외하고는 수렴하지 않았다. 즉 정확한 결과 오차 범위에 따라 실행 속도 개선이 확실한 방법이나 본 속제에서는 동일 결과를 산출하면서 개선 되는 것을 확인할 수 없었다. HPF 의 경우 추가한 directive 는 다음과 같다.

```
!HPF$ PROCESSORS p(4,4)
!HPF$ ALIGN PHI(:, :, :) WITH O_PHI(:, :, :)
!HPF$ DISTRIBUTE O_PHI(BLOCK, *, BLOCK) ONTO P
```

따라서 개선한 알고리즘만 제시하기로 하였다.(별첨 2 NEW GAUSEI subroutine 참조)

### 3. HPF Vs. MPI in GAUSEI



(그림 1)과 (그림 2)를 종합하여 MPI 와 HPF 를 동일 조건으로 프로그램 한 후에 실행 결과는 (그림 3)과 같이 HPF 가 디버깅은 어려우나 MPI 보다 좋은 결과를 보였다.

기타 Red/Black SOR 과 기존의 SOR 모두에 적용하면 HPF 의 경우 CPU 를 많이 사용하면 반대로 실행 시간이 보다 많이 걸렸다.

### 3. 결론

병렬 프로그래밍은 디버깅이 어렵고 `data parallelism`을 위해서 그 응용 프로그램의 모든 것을 속속들이 잘 알 필요가 있었다. 그리고 병렬 알고리즘을 도입해서 개선해야 할 것과 단순히 프로그램 환경만 교체하여 실행 속도를 개선하는 것을 나누어서 따로이 심도있게 할 필요가 있다. 본 프로그램과 같은 경우는 들인 시간에 비하여 개선의 효과가 매우 적었다.