데이터 병렬 처리 및 작업 병렬 처리의 성능 분석

Data Parallelism and Task Parallelization Performance Analysis

지도교수 ooo

oo대학교

oo 공학과

o o o

## 서론

현대의 컴퓨터 아키텍처는 멀티코어 프로세서를 채택하여 성능을 극대화하는 방향으로 발전하고 있다. 멀티코어 환경에서는 병렬 처리를 통해 데이터 처리 속도를 크게 향상시킬 수 있으며, 이는 다양한 분야에서 중요한 이점을 제공한다. 특히, 데이터 병렬 처리와 작업 병렬 처리 기법은 이러한 성능 개선을 위해 자주 사용된다. 데이터 병렬 처리에서는 동일한 연산이 대규모 데이터 세트에 대해 수행되며, 작업 병렬 처리에서는 여러 개의 독립적인 작업이 동시에 수행된다. 본 연구의 목표는 이러한 두 가지 병렬 처리 방식의 성능을 비교하고, 스레드 및 프로세스 수의 증가가 처리 시간에 미치는 영향을 평가하는 것이다.

# 문제제기

병렬 처리를 효과적으로 구현하기 위해서는 적절한 방법과 구조를 선택하는 것이 중요하다. 데이터 병렬 처리와 작업 병렬 처리 모두 스레드와 프로세스의 개수에 따라 성능이 달라지며, 이로 인해 성능 향상 또는 저하가 발생할 수 있다. 특히, 자원 경쟁, 컨텍스트 전환 비용, 메모리 대역폭 제한 등의 요인은 스레드 또는 프로세스 수를 증가시키면서 처리 성능에 부정적인 영향을 미칠 수 있다. 이러한 문제를 해결하기 위해 본 연구에서는 다양한 스레드 및 프로세스 수를 조절하며 실제 성능을 측정하고 분석한다.

## 본론

# 솔루션 제안

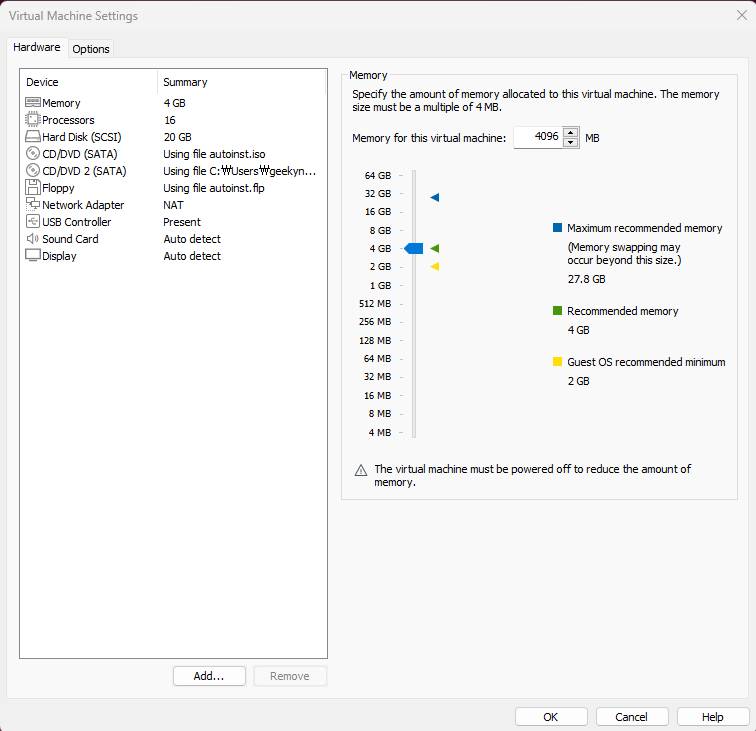
본 연구에서는 두 가지 주요 병렬 처리 기법을 비교하여 성능 최적화를 위한 방향을 제시한다. 첫 번째 기법은 데이터 병렬 처리로, 대규모 행렬 곱셈 연산을 통해 스레드 및 프로세스 수의 영향을 측정한다. 두 번째 기법은 작업 병렬 처리로, 여러 개의 텍스트 파일을 압축하는 작업을 수행하여 그 성능을 비교한다. 이러한 실험을 통해 데이터 병렬 처리와 작업 병렬 처리 각각의 최적화 기법을 평가하고, 최적의 병렬 처리 방법을 제안한다.

# 실험 및 검증

### 1. 실험 환경

본 실험은 GCC 컴파일러 기반 C언어를 사용하였으며, VMware 가상환경의22.04 LTS 버전 Ubuntu 운영체제에서 실시하였으며, 사양 설정은 그림1과 같다.

데이터 병렬 측정에 사용한 행렬의 크기는 1000x1000이며, 작업 병렬 측정에 사용한 파일은 다양한 크기의 텍스트 파일로 진행하였다.



그림

### 2. 실험 절차

데이터 병렬 처리:

두 개의 행렬 A와 B를 초기화한다. 각 요소는 랜덤한 값(0~9)으로 채워진다.

행렬 곱셈을 수행하기 위해 스레드를 활용한다. 스레드 수는 1, 2, 4, 8, 16으로 설정하고, 각 경우에 대해 실행 시간을 측정한다.

스레드 기반의 행렬 곱셈이 끝난 후, 동일한 방법으로 프로세스를 사용하여 행렬 곱셈을 수행한다. 프로세스 수 역시 1, 2, 4, 8, 16으로 설정하고, 처리 시간을 측정한다.

작업 병렬 처리:

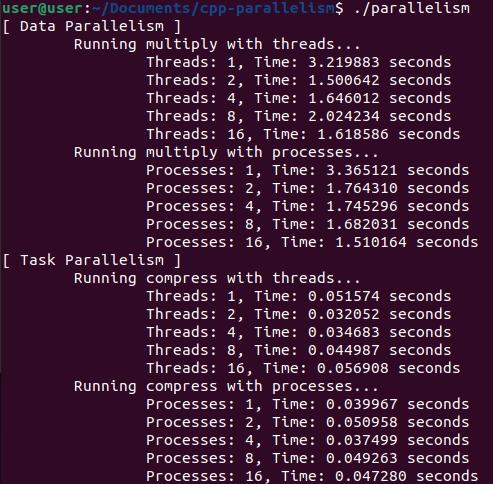
여러 개의 텍스트 파일을 준비하고, 각 파일에 대해 gzip 압축 작업을 수행한다.

스레드를 사용하여 파일 압축을 수행하고, 처리 시간을 측정한다. 스레드 수는 데이터 병렬 처리와 동일하게 설정한다.

프로세스를 사용하여 파일 압축 작업을 반복하고, 처리 시간을 측정한다.

### 3. 처리 시간 측정

clock() 함수를 사용하여 각 스레드 및 프로세스의 처리 시간을 측정하였으나, 너무 짧은 시간의 결과를 보여주는 구간이 있었기에 clock\_gettime() 함수를 대신 사용하였으며, 각 실험의 결과는 그림2와 같다.



그림

### 4. 성능 분석

행렬 곱셈:

스레드를 사용한 행렬 곱셈에서 스레드 수가 1에서 16으로 증가함에 따라 처리 시간은 감소하는 경향을 보였다. 이는 각 스레드가 행렬의 일부를 동시에 처리함으로써 작업을 병렬화한 결과로 해석된다. 하지만 8개 스레드 이상에서는 성능 향상이 둔화되었고, 이는 스레드 간의 컨텍스트 전환 비용이 증가하고, CPU의 메모리 대역폭이 포화되었기 때문일 가능성이 크다.

프로세스를 사용한 경우, 프로세스 수를 늘리면서도 전반적으로 성능이 향상되었다. 그러나, 프로세스 간의 메모리 공유가 불가능하므로, 각 프로세스가 필요한 메모리를 별도로 할당받아야 하는 점이 성능 저하의 원인이 될 수 있다.파일 압축:

파일 압축 작업에서 스레드 수가 증가할 때, 4개 스레드까지는 처리 시간이 줄어드는 경향이 있었다. 그러나, 8개 스레드 이상에서는 압축 작업의 특성상 파일 I/O 병목 현상으로 인해 성능이 감소하였다. 특히, I/O 대기 시간이 늘어나면서 스레드가 대기하는 시간이 증가했음을 확인할 수 있다.

프로세스 기반의 파일 압축에서도 유사한 패턴이 나타났다. 프로세스 수를 증가시킬수록 성능이 개선되다가 8개 이상의 프로세스에서는 I/O 대기 시간이 증가하며 성능이 감소하는 경향을 보였다.

### 5. 그래프

스레드 및 프로세스 수에 따른 데이터 병렬 처리 시간의 관계를 나타내는 그래프는 그림 3과 같다. X축은 프로세스의 개수, Y축은 시간을 나타낸다.

그림 3

스레드 및 프로세스 수에 따른 작업 병렬 처리 시간의 관계를 나타내는 그래프는 그림 4와 같다. X축은 프로세스의 개수, Y축은 시간을 나타낸다.

그림 4

## 결론

본 연구에서는 데이터 병렬 처리 및 작업 병렬 처리의 성능을 비교하기 위해 스레드와 프로세스를 사용하여 실험을 수행하였다. 실험 결과, 스레드 및 프로세스 수를 증가시킬수록 처리 시간이 감소하는 경향을 보였으나, 일정 수 이상에서는 성능이 정체되거나 감소하는 경향이 나타났다. 이러한 결과는 멀티코어 시스템에서 병렬 처리를 최적화하기 위해서는 적절한 스레드 및 프로세스 수의 선택이 필요함을 시사한다.

향후 연구에서는 다양한 작업 유형 및 하드웨어 환경에 대한 추가적인 성능 분석이 필요하며, 또한 스레드 및 프로세스 간의 메모리 관리 및 자원 할당 전략에 대한 연구도 중요할 것으로 보인다. 이를 통해 더욱 효율적인 병렬 처리 방법을 제시할 수 있을 것이다.