효과적인 CPU 성능분석 기술

장한휘　　김동규　　김장우
포항공과대학교 고성능컴퓨팅연구실

연구의 필요성

- 소프트웨어 시뮬레이션을 통한 CPU 성능 측정은 많은 시간이 소요됨
- 카운터를 이용한 성능 측정은 부정확하며 원하는 정보를 제공하지 못함

연구 목표

- 이 연구에서는 빠르면서 정확한 새로운 CPU 성능측정 방법을 제시한다.
- 병목현상이 CPU 성능에 미치는 영향을 효율적으로 보여주기 위하여 사용자에게 CPI 스택을 제시한다.
- 컴퓨터 구조의 변화에 따른 성능변화를 효율적으로 보여준다.

연구 배경

- **CPI(Cycles Per Instructions)**
  > 어떤 응용 프로그램을 실행할 때 한 명령어를 수행하기 위해 얼마나 많은 싸이클이 소모되는지를 나타내는 수치
- **CPI 스택**
  > 주어진 CPI 수치를 각 병목현상으로 인하여 잃어버리는 수치로 나누어서 보여주는 막대 그래프
- **구간 분석법[1] 및 FMT[2]**
  > 명령어 실행시간이 큰 CPU 병목현상에 따라 구간으로 나누어진다는 이론 (**그림1**)
  > 전단 명목현상 (예: 명령어 캐시 미스, 분기 추측 실패)과 긴 말미 병목현상 (예: L2 자료 캐시 미스)에 의해 구간이 결정됨
  > **카운터를 이용하여 각 병목현상에 따라 명령어 실행이 멈추었을 때의 손실을 빠르게 계산함**
  > **병목현상들의 중첩효과를 설명하지 못하기 때문에** 짧은 말미 병목현상 (예: L1 자료 캐시 미스, 긴 연산 명령어)에 의한 성능 손실을 정확하게 계산하지 못함 (**그림2**)

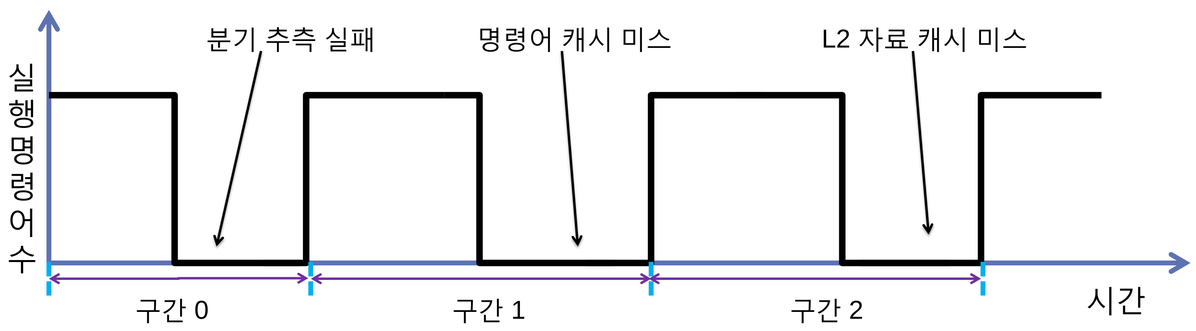

그림1. 구간 분석법

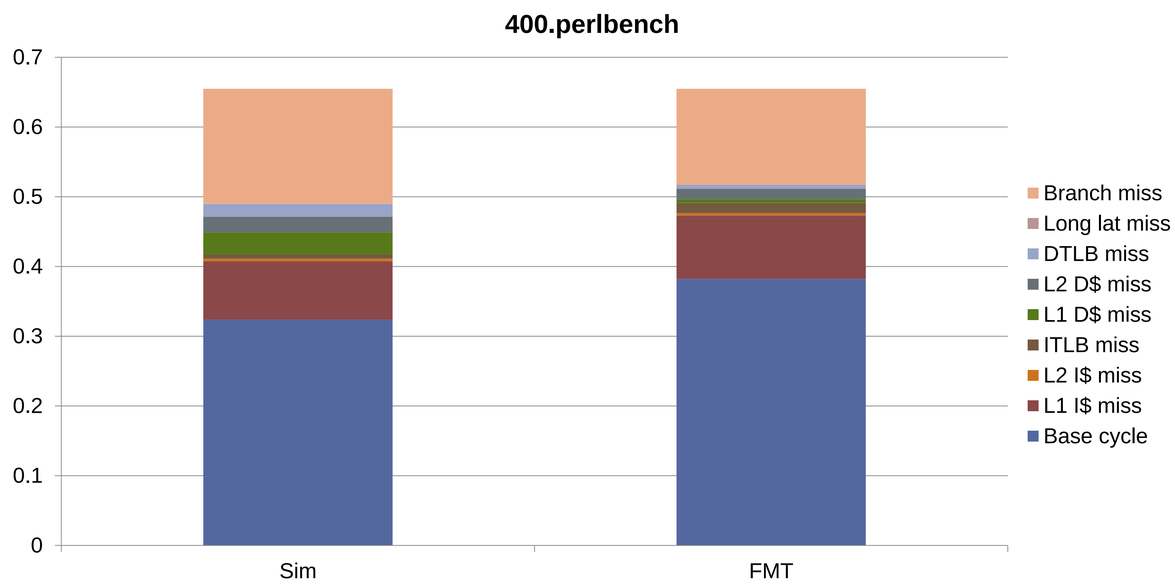

그림2. 시뮬레이션과 FMT 비교

- **의존 그래프를 이용한 상호작용 비용 계산[3]**
  > 여러 병목현상들이 독립적이지 않고 서로 연관이 있다는 이론
  > **의존 그래프를 이용하여 중첩효과를 설명(그림3)**
  - 하나이상의 병목현상을 제거 후 **임계경로**를 계산하여 CPI를 계산
  > **고려해야 할 경우의수가 지수 승에 비례($O(2^n)$)**

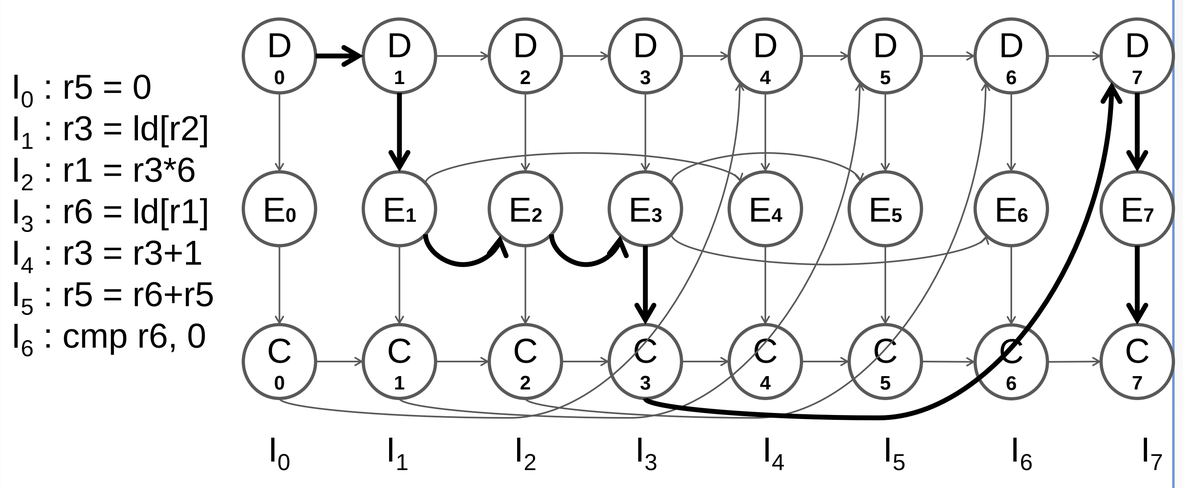

그림3. 의존 그래프

접근방법

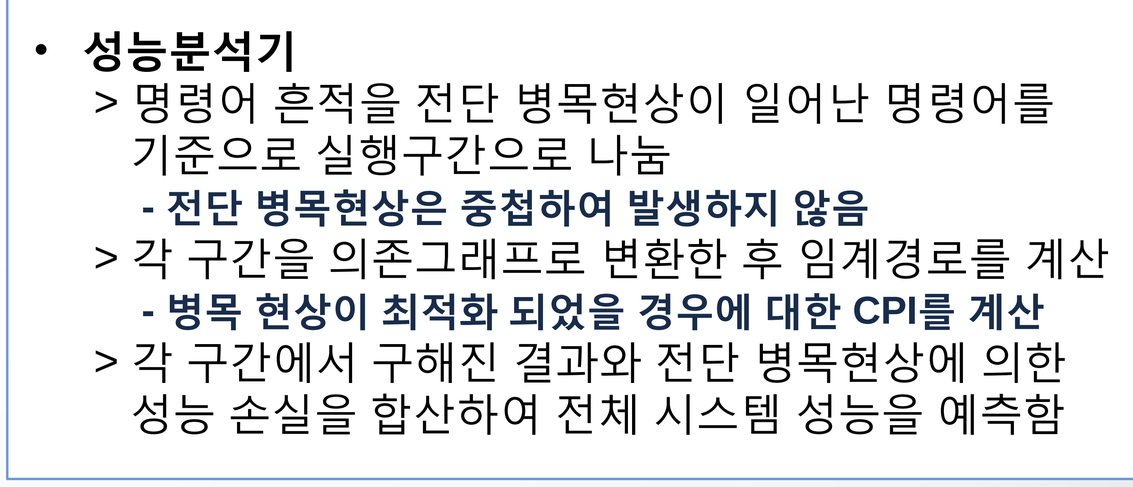

그림4. CPU 성능 분석 시스템

- **명령어 흔적**
  > 하드웨어 혹은 시뮬레이터에서 실행된 명령어들을 기록함
  > 각 명령어의 시간, 의존관계, 그리고 병목현상 정보를 포함하고 있음
- **성능분석기**
  > 명령어 흔적을 전단 병목현상이 일어난 명령어를 기준으로 실행구간으로 나눔
  - 전단 병목현상은 중첩하여 발생하지 않음
  > 각 구간을 의존그래프로 변환한 후 임계경로를 계산
  - 병목 현상이 최적화 되었을 경우에 대한 CPI를 계산
  > 각 구간에서 구해진 결과와 전단 병목현상에 의한 성능 손실을 합산하여 전체 시스템 성능을 예측함

참고문헌

[1] S. Eyerman, et al, “A performance counter architecture for computing accurate CPI components” ACM international Conference on Architectural Support for Programming Languages and operating Systems, 2006, p175-184
[2] S. Eyerman, et al, “A Mechanistic Performance Model for Superscalar Out-of-Order Processors” ACM Transactions on Computer Systems, 2009, p3:1~3:36
[3] Fields, Brian A., et al. "Interaction cost and shotgun profiling." ACM Transactions on Architecture and Code Optimization (TACO) 1.3 (2004): 272-304.