OpenMP

Lab 2

실습 결과물 제출 안내

- **LMS을 통해 과제 제출**
- **제출 내용**
  - **작성한 소스코드**
    - 소스코드 원본 및 실행 파일 (압축하여 제출)
      - 헤더파일 및 소스코드(예, .c .cpp .h .hpp 등)와 실행 파일 만 포함할 것
      - 프로젝트 설정, DB, 오브젝트 파일 등 부수 파일 제출 금지
    - 소스코드를 모두 모은 한글 (or MS Word) 파일

    * 위 두가지 형태 모두 제출해야 함
  - **보고서**
    - 실행 결과 캡쳐 및 결과에 대한 간단한 설명 및 소감

* 소스코드 모음 한글 파일 및 보고서는 압축하지 않고 개별 파일로 제출 할 것

실습 결과물 제출 안내

- **제출 기한: 3/26(일) 23:59**

- **Copy 허용 (1.5점 인정)**
  - 과제 보고서에 Copy 했음을 반드시 명시 할 것
    - 명시 없이 Copy 및 적발 시, 최정 학점 한단계 하락 적용
  - 단, 직접 3번 typing 하는 영상 제출
    - Youtube 업로드 후, 링크 제출

- **감점 사항**
  - 지각 제출: -0.5점/일
  - 보고서 미제출: 0점 처리
    - 소감 미 포함 시, 보고서 불인정

Lab. 2-1

Matrix x Vector

Feel the Power of Parallel Computing

Matrix x Vector

- 행렬과 벡터의 곱을 구하는 프로그램 작성

a_{00}	a_{01}	$\cdots$	$a_{0,n-1}$
a_{10}	a_{11}	$\cdots$	$a_{1,n-1}$
$\vdots$	$\vdots$		$\vdots$
a_{i0}	a_{i1}	$\cdots$	$a_{i,n-1}$
$\vdots$	$\vdots$		$\vdots$
$a_{m-1,0}$	$a_{m-1,1}$	$\cdots$	$a_{m-1,n-1}$

x_0
x_1
$\vdots$
x_{n-1}

$=$

y_0
y_1
$\vdots$
$y_i = a_{i0}x_0 + a_{i1}x_1 + \cdots a_{i,n-1}x_{n-1}$
$\vdots$
y_{m-1}

[Image from Introduction to Parallel Programming, Pacheco]

- 실습 목표
  - Parallel construct, work-sharing construct 활용하기
  - 병렬처리에서 Read와 Write operation의 차이 알기
  - 병렬처리의 힘을 느껴 보기

Matrix x Vector

- **입력**
  - Matrix **A**의 크기 (row, column), Vector **b**의 길이 (line 10-11)
  - 입력 받은 크기에 맞추어 **A**, **b가 생성 됨**
    - Float type, 값은 random

- **출력**
  - 결과 검증 결과
    - 직렬처리 결과와 병렬처리 결과 비교
  - 직렬처리 시간
  - 병렬처리 시간

Matrix x Vector

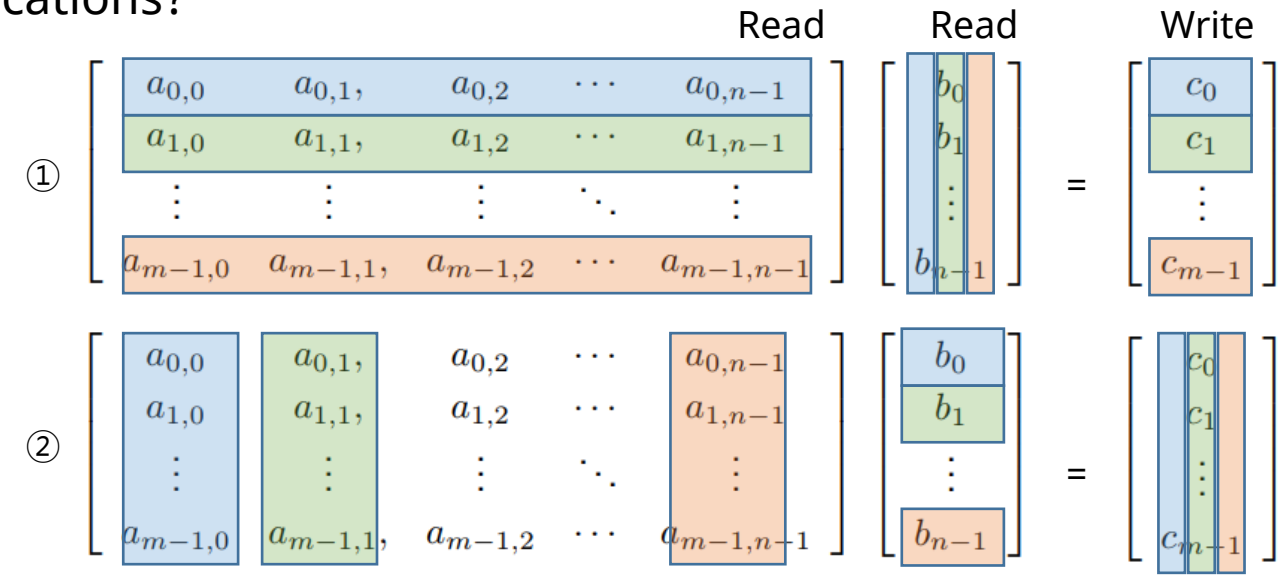
- **작성할 코드**
  1. Serial algorithm (line 30-38)
  2. Parallel algorithm (line 43-50)
  3. 결과 검증 코드 (line 54-67)
     - 결과가 정상 일 사, **isCorrect** flag를 true로 변경

Matrix x Vector

- **보고서에 포함 할 내용**
  - 행렬과 벡터의 크기를 변화시키며 성능 비교/분석
    - 최대 크기는 A = 10,000 by 10,000, b = 10,000 by 1 이상까지
      - 예) 1000, 2000, ..., 10,000
  - 스레드의 수를 변화시키며 성능 비교/분석
    - 예) 1, 2, 4, 8, 16

Matrix x Vector – Hints

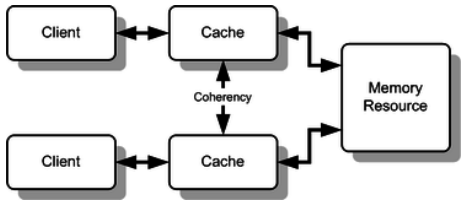
- **How to decompose the work?**
  - Row-wise? Column-wise?
  - Communications?



Matrix x Vector – Hints

- **같은 변수를 여러 스레드가 Read하는 경우,**
  - 여러 thread가 읽어도 문제가 되지 않음
    - 데이터를 수정 하지 않는다

- **같은 변수에 여러 스레드가 Write하는 경우,**
  - **Synchronization이 필요**
    - 여러 thread가 동시에 수정 → 잘못된 데이터 생성
  - **Cache coherency issue** → 성능 저하 (bottleneck)
    - https://en.wikipedia.org/wiki/Cache_coherence



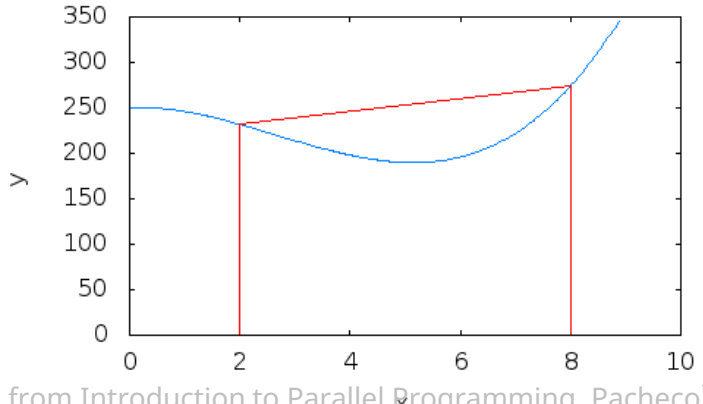
- **같은 변수에 대한 동시 write를 피해야 함**

Lab. 2-2

Trapezoidal Rule

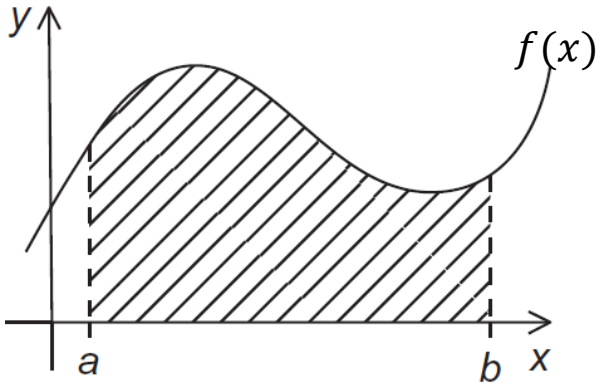
An Interesting Example!

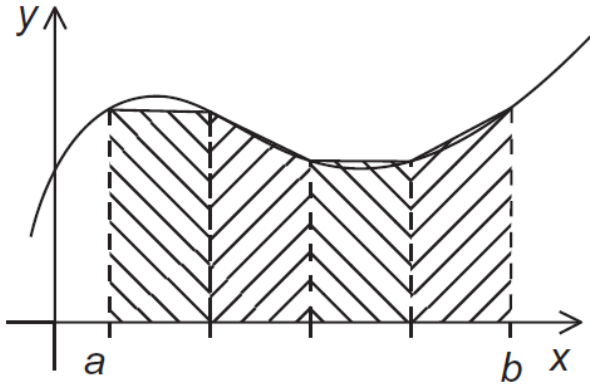
DIY from scratch



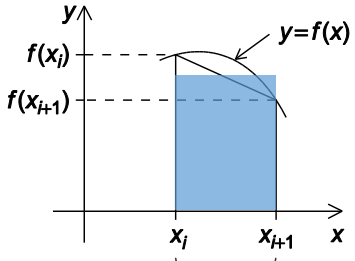
[Image from Introduction to Parallel Programming, Pacheco]

Trapezoidal Rule





[Image from Introduction to Parallel Programming, Pacheco]



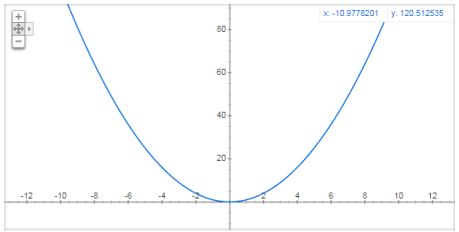
= (b-a)/n

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} * h$$

Trapezoidal Rule

- 사다리꼴 규칙을 이용하여 정적분의 근사 값 구하기 (수치적분)
- 목표 함수: $f(x) = x * x$

x*x 그래프



[Image from wolframalpha.com]

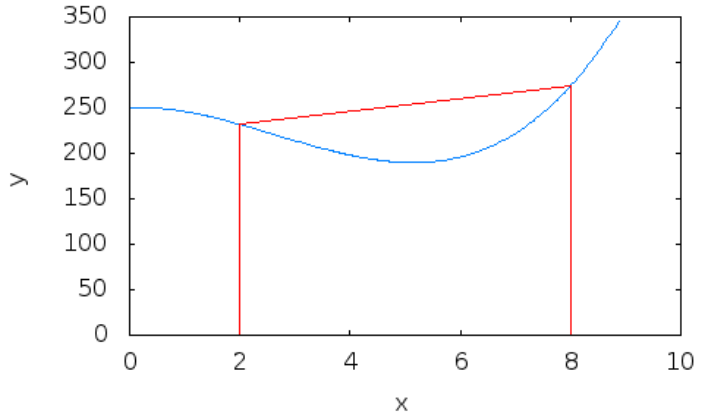
- 입력(프로그램 인자) : a, b, n
  - [a, b] 구간을 n개로 나누어 계산
- 출력
  - Serial & Parallel algorithm 의 처리 시간 및 계산 결과

/Lab2-1.exe -1 1 1048576
f(x) = x * x
range = (-1.000000, 1.000000), n = 1048576
[Serial] area = 0.666665
[Parallel] area = 0.666665

[Serial] : 1036.22770 ms (1036.22770 ms)
[Parallel] : 351.47470 ms (351.47470 ms)

Trapezoidal Rule

- **Tasks**
  1. Serial algorithm 구현
  2. Parallel algorithm 구현
  3. 두 알고리즘의 성능 비교
     - a, b, n을 조절해가며 성능 비교/분석
       - 예) a = 0, b = 1024, n = 1073741824

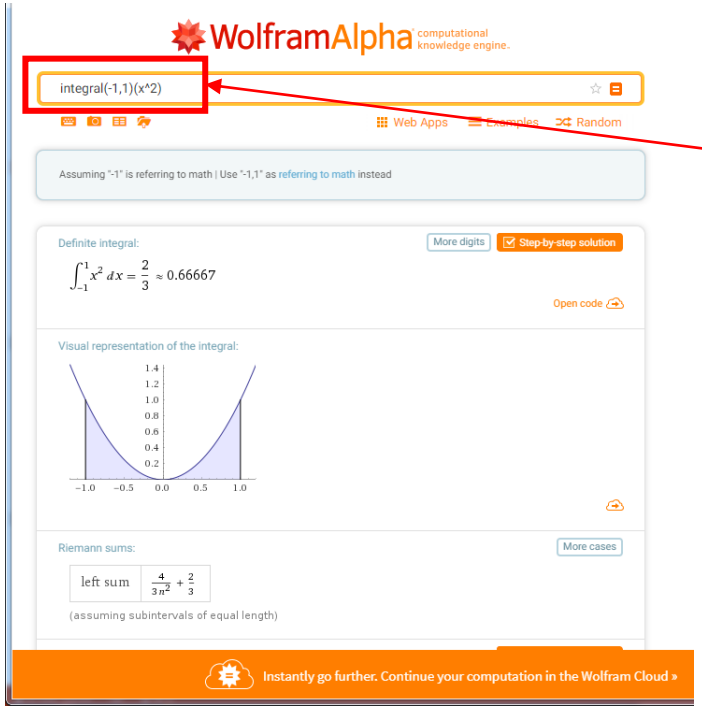


[Image from Introduction to Parallel Programming, Pacheco]

Trapezoidal Rule

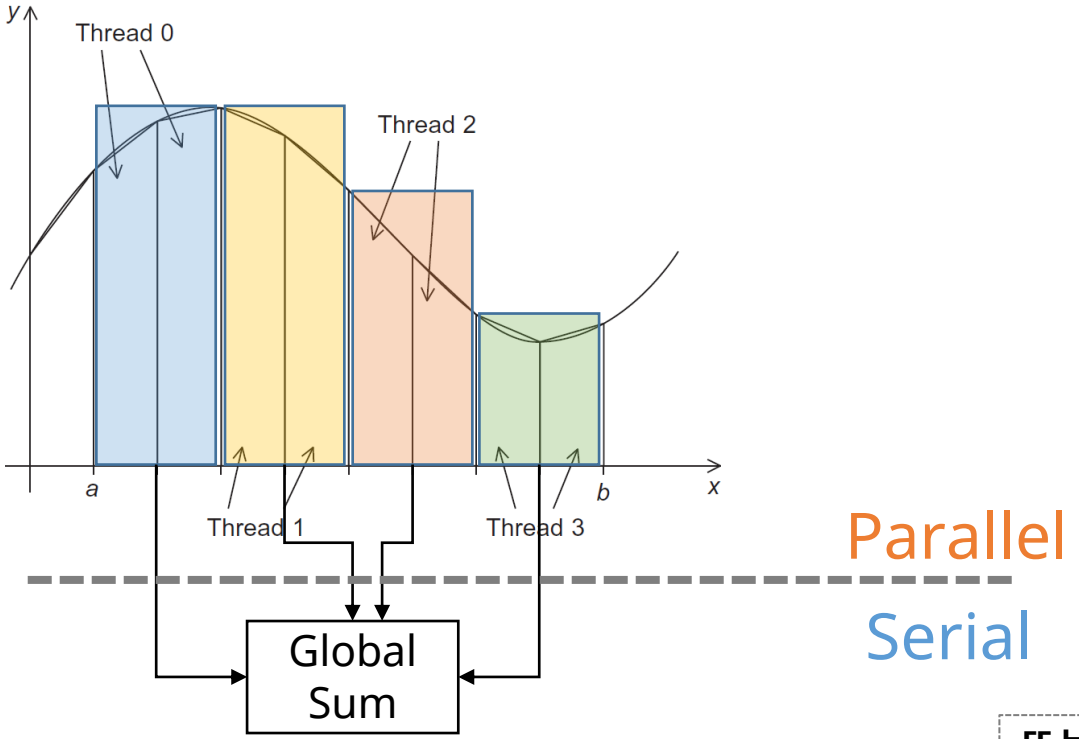
- 결과 검증

https://www.wolframalpha.com/



Integral(a,b)(x^2)

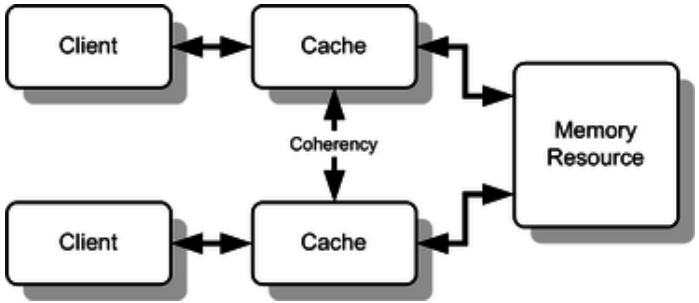
Trapezoidal Rule – Hint



또는,
동기화 지시어 사용 ➔ **Next class!**

Trapezoidal Rule – Hint

- **Cache coherency issue → 성능 저하 (bottleneck)**
  - https://en.wikipedia.org/wiki/Cache_coherence



DS_Timer 사용법

- **관련 파일**
  - DS_timer.h, DS_timer.cpp, DS_definition.h
- **Timer 생성**
  - `DS_timer timer(사용할 타이머의 수);`
- **Timer on/off**
  - timer.onTimer(TimerID); / timer.offTimer(TimerID);
    - TimerID는 0번 부터 시작
    - On과 off는 반드시 매칭 되어야 됨
      - on → off → on → off (O), on → on → off (X),
- **측정 결과 출력**
  - timer.printTimer();
- **Timer 이름 설정 (Optional)**
  - timer.setTimerName(timerID, (char*)"Name");

```
*       DS_timer Report       *
* The number of timer = 7, counter = 7
**** Timer report ****
Timer 0 : 1.03000 ms (1.03000 ms)
Timer 1 : 0.54840 ms (0.54840 ms)
Timer 2 : 23.20120 ms (23.20120 ms)
**** Counter report ****
*       End of the report     *
```

Bonus Points!

- **Lab 2-2를 해설하는 영상을 제작해서 제출 시, 보너스 포인트(기타 점수 영역) +3점**
  - 영상은 소스코드만으로 설명 하는 것이 아닌, 슬라이드를 만들어 설명해야 함
    - 영상 길이는 5분 이상
    - 제출된 영상은 온라인에 게시 및 다른 수강생들에게 공유될 예정

- **제출 방법**
  - 영상 원본을 조교에게 이메일로 제출
  - 기한: ~3/24(금) 23:59

Q & A

[그림 출처: illustAC (link)]

이미지 출처

- **본 슬라이드에 사용된 이미지들은,**
  - 본 과목의 주재인 창의적 공학설계(김은경 지음, 한빛미디어) 및
    - 도서 링크
  - 다음 출처로 부터 가져 왔으며, 상업적 사용 및 출처 표시 제한이 없는 이미지만 사용 했습니다
    - Pixarbay
    - illustAC

Pixabay로부터 입수된 Peggy und Marco Lachmann-Anke님의 이미지입니다.