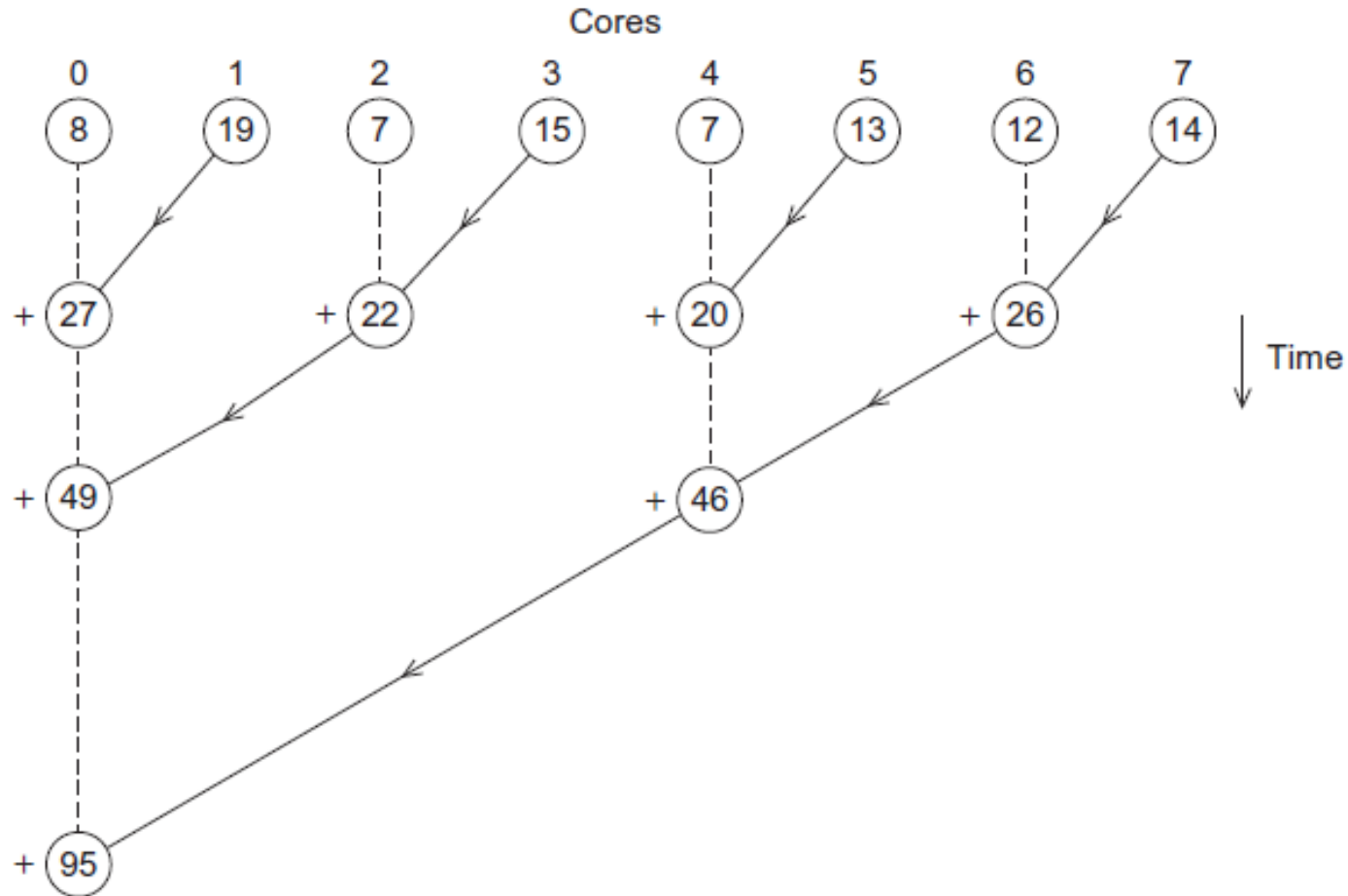


Lecture 6-1

Reduction Clause

Introduction to OpenMP

Example – Improved Algorithm



// Total computing time = ?

Performance Analysis

- 3072개의 값을 계산하고 전체 합을 구할 때
- Naïve algorithm

# of cores	8	16	32	64	128	256	512	1024
값 계산 시간	384	192	96	48	24	12	6	3
합 계산 시간	7	15	31	63	127	255	511	1023
총 시간	391	207	127	111	151	267	517	1026

- Improved algorithm

# of cores	8	16	32	64	128	256	512	1024
값 계산 시간	384	192	96	48	24	12	6	3
합 계산 시간	3	4	5	6	7	8	9	10
총 시간	387	196	101	54	31	20	15	13

Reduction Clause

reduction (operation:var_list)

- 병렬처리 결과값을 하나로 취합 할 때 사용
- var_list의 변수는 shared variable
 - 각 스레드에 private 변수로 생성 및 초기화
 - 초기화 값은 연산의 종류에 따라 결정 됨
 - 병렬처리 수행 후, 최종 취합된 값을 공유변수에 저장

Reduction Clause

reduction (operation:var_list)

- 지원 연산자 및 초기화 값

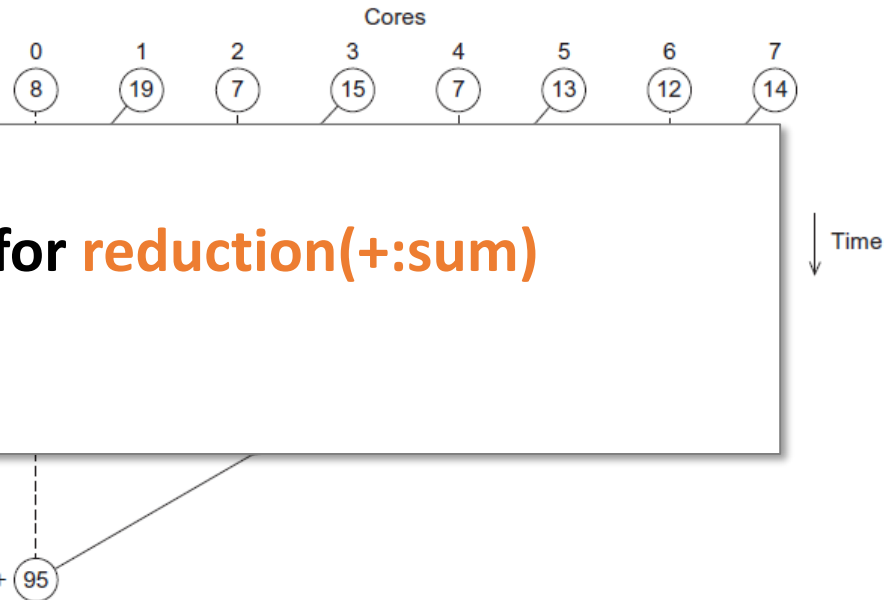
Operator	Data Types	초기값
+	integer, floating point	0
*	integer, floating point	1
-	integer, floating point	0
&	integer	all bits on
	integer	0
^	integer	0
&&	integer	1
	integer	0

Global Sum Example

- Global Sum

```
sum = 0;
for (int i = 0 ; i < n ; i++)
    sum += a[i];
```

- Parallel global sum



- Quick Lab.

Trapezoidal Rule Example

```
sum = 0;
#pragma omp parallel for num_threads(4) reduction(+:sum)
for (int i = 0; i < n - 1; i++)
{
    double x_i = a + h * i;
    double x_j = a + h * (i + 1);
    double d = (f(x_i) + f(x_j)) / 2.0;

    //#pragma omp critical
    //{
        sum += d*h;
    //}
}
```

Lecture 6-2

Scheduling Clause

Introduction to OpenMP

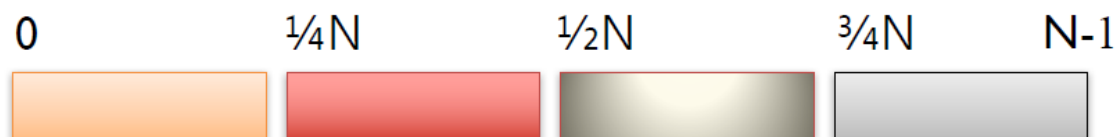
Scheduling Clause

`#pragma omp parallel for schedule(type, chunk_size)`

- For loop의 병렬화에 대한 일 분배 방법을 지정
- Types
 - **static**
 - Chunk 단위로 나누어서 Round-robin 방식으로 분배
 - **dynamic**
 - Idle 상태의 thread에게 chunk 크기의 일 할당
 - **guided**
 - dynamic 과 유사한 방법
 - Chunk 크기를 exponential 하게 줄여가며 할당
 - **runtime**
 - 환경변수 OMP_SCHEDULE 값에 따라 runtime에 결정 됨

Static Scheduling

- Static schedule on iteration space



- Example
 - 12 iterations(0, 1, ..., 11) and 3 threads

```
schedule(static,1)
```

Thread 0 : 0,3,6,9

Thread 1 : 1,4,7,10

Thread 2 : 2,5,8,11

```
schedule(static,2)
```

Thread 0 : 0,1,6,7

Thread 1 : 2,3,8,9

Thread 2 : 4,5,10,11

```
schedule(static,4)
```

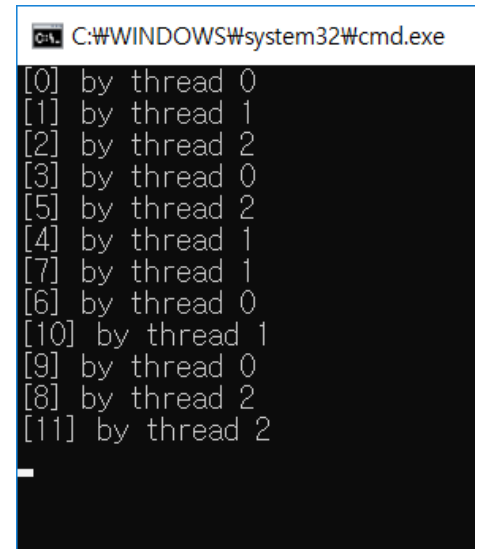
Thread 0 : 0,1,2,3

Thread 1 : 4,5,6,7

Thread 2 : 8,9,10,11

Static Scheduling (example)

```
void main(void)
{
    #pragma omp parallel for schedule(static, 1) num_threads(3)
    for (int i = 0 ; i < 12 ; i++)
    {
        int tID = omp_get_thread_num();
        printf("[%d] by thread %d\n", i, tID);
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
[0] by thread 0
[1] by thread 1
[2] by thread 2
[3] by thread 0
[4] by thread 1
[5] by thread 2
[6] by thread 0
[7] by thread 1
[8] by thread 2
[9] by thread 0
[10] by thread 1
[11] by thread 2
```

- Quick Lab.

Dynamic Scheduling

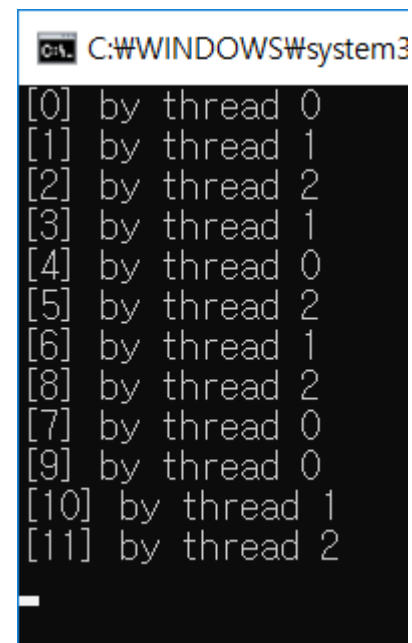
- **Dynamic** schedule on iteration space (master/worker)



- Chunk 크기로 일을 나누어 놓음
 - 각각의 스레드는 하나의 chunk를 할당 받음
 - 스레드는 할당 받은 chunk 처리를 끝내면, 다른 chunk를 요청
-
- Default chunk size = 1

Dynamic Scheduling (example)

```
void main(void)
{
    #pragma omp parallel for schedule(dynamic, 1) num_threads(3)
    for (int i = 0 ; i < 12 ; i++)
    {
        int tID = omp_get_thread_num();
        printf("[%d] by thread %d\n", i, tID);
        Sleep(1);
    }
}
```

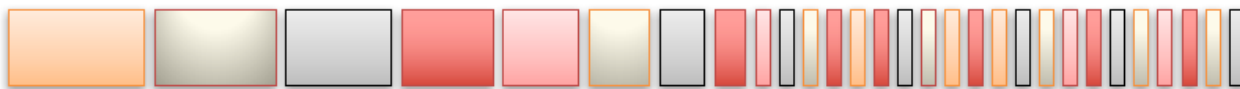


```
C:\WINDOWS\system32
[0] by thread 0
[1] by thread 1
[2] by thread 2
[3] by thread 1
[4] by thread 0
[5] by thread 2
[6] by thread 1
[7] by thread 2
[8] by thread 0
[9] by thread 0
[10] by thread 1
[11] by thread 2
```

- Quick Lab.

Guided Scheduling

- **Guided** schedule on iteration space (master/worker)

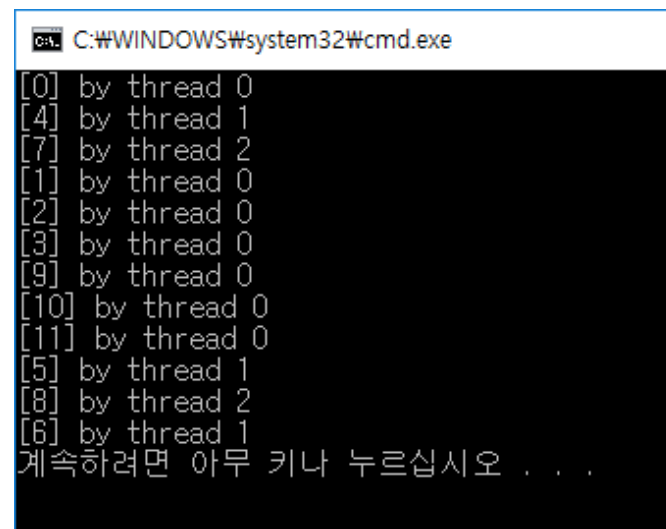


- Initial chunk size = # of iteration / # of threads
- Idle 상태의 thread는 새로운 chunk를 요청함
- 다음 chunk를 요청하면, 감소된 크기의 chunk를 할당
 - Exponentially decreased
- Chunk의 크기는 지정된 chunk_size까지 감소함

Guided Scheduling (example)

```
void main(void)
{
    #pragma omp parallel for schedule(guided, 1) num_threads(3)
    for (int i = 0 ; i < 12 ; i++)
    {
        int tID = omp_get_thread_num();
        printf("[%d] by thread %d\n", i, tID);
        Sleep(1);
    }
}
```

- Quick Lab.



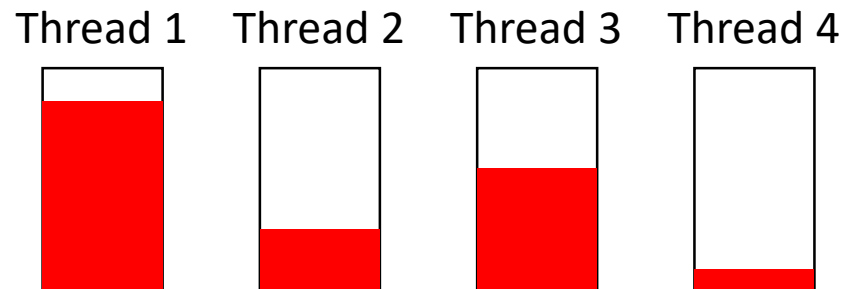
```
C:\WINDOWS\system32\cmd.exe
[0] by thread 0
[4] by thread 1
[7] by thread 2
[1] by thread 0
[2] by thread 0
[3] by thread 0
[9] by thread 0
[10] by thread 0
[11] by thread 0
[5] by thread 1
[8] by thread 2
[6] by thread 1
계속하려면 아무 키나 누르십시오 . . .
```

Work Scheduling / Distribution

- 목적

- 모든 computing resource (e.g., threads)들이 동일한 양의 일을 하도록 하는 것
- **Workload balance**

- Why?



Scheduling Methods (Centralized)

- **Static allocation**

- 일을 시작하기 전에 고정된 수 만큼 일 분배
- 각각의 일(work unit)의 크기가 일정할 때 사용
 - 또는 일의 양을 미리 알고 있을 때
- 구현이 간단
- Low overhead
- 일의 양을 정확히 알기 어려움
- 스레드간 연산량 균형(load balance)을 맞추기 힘들

Scheduling Methods (Centralized)

- **Dynamic allocation**

- 스레드들의 상황에 따라, 일을 동적으로 분배
 - Master : Slave 요청에 따른 일 할당
 - Slaves : Master에게 일을 할당 받아서 일을 처리
- 각각의 일의 크기가 다를 때 (예측 할 수 없을 때) 사용
- Good for load balance
- Master thread는 일 처리를 못함 (일 분배에 전념)
- High overhead for work distribution
 - Request and allocation
 - Chunk size에 영향을 많이 받음

Scheduling Methods (decentralized)

- **Work stealing method**

- 최초에 static하게 일을 분배
- 스레드가 자신의 일을 모두 끝내면,
다른 스레드의 일을 빼앗아 옴
- Static 과 Dynamic allocation 방법의 장점을 활용
- 모든 thread가 일을 처리 할 수 있음
- 구현이 복잡
- Work stealing overhead
 - Chunk size에 영향을 많이 받음

Advanced Scheduling Algorithm

- **Optimization-based scheduling algorithm**

Lecture 6-3

Nested Parallelism

Introduction to OpenMP

Nested Parallelism

- Parallel region 안에서 Parallel region을 만드는 것
 - Recursive한 parallel algorithm을 만들 수 있음

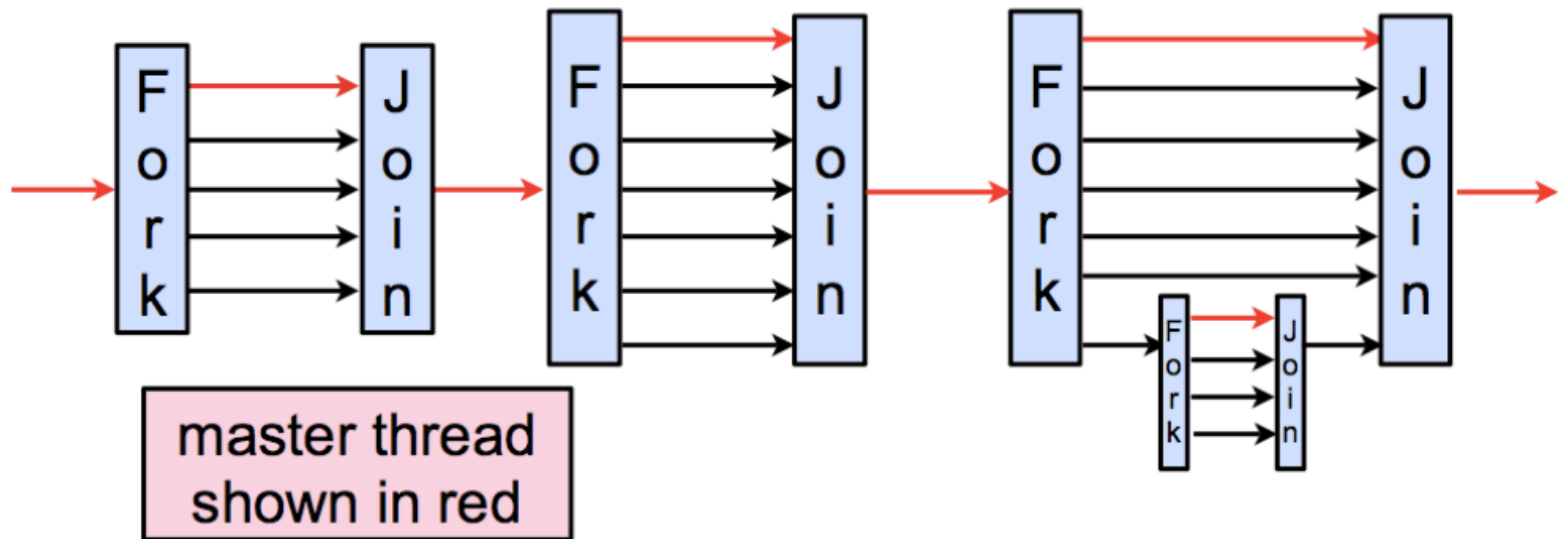


Image Source: John Mellor-Crummey's Lecture Notes

Nested Parallelism

- **Parallel region 안에서 Parallel region을 만드는 것**
 - Recursive한 parallel algorithm을 만들 수 있음
- **omp_set_nested(int) 함수를 통해 on/off**
 - 인자가 0이면 off, 양수면 on
 - Default = false
 - `int omp_get_nested(void)`
 - 현재 nested parallelism 지원 여부 확인

Nested Parallel Example

```

omp_set_nested(1);

printf("Nested parallelism : %s\n"
      , omp_get_nested() ? "On" : "Off");

#pragma omp parallel num_threads(4)
{
    int parentID = omp_get_thread_num();
    printf("Lv 1 - Thread %d\n", parentID);

    #pragma omp parallel num_threads(2)
    {
        printf("\tLv 2 - Thread %d of %d\n"
              , omp_get_thread_num(), parentID);
    }
}

```

C:\WINDOWS\system32\cmd.exe

```

Nested parallelism : On
Lv 1 - Thread 0
Lv 1 - Thread 1
    Lv 2 - Thread 1 of 0
    Lv 2 - Thread 0 of 1
    Lv 2 - Thread 1 of 1
    Lv 2 - Thread 0 of 0
Lv 1 - Thread 3
Lv 1 - Thread 2
    Lv 2 - Thread 0 of 3
    Lv 2 - Thread 1 of 3
    Lv 2 - Thread 1 of 2
    Lv 2 - Thread 0 of 2
계속하려면 아무 키나 누르십시오 . . .

```


Congratulation!



안녕하세요!

