유전알고리즘을 이용한 병렬프로그래밍

2017270986 황창국

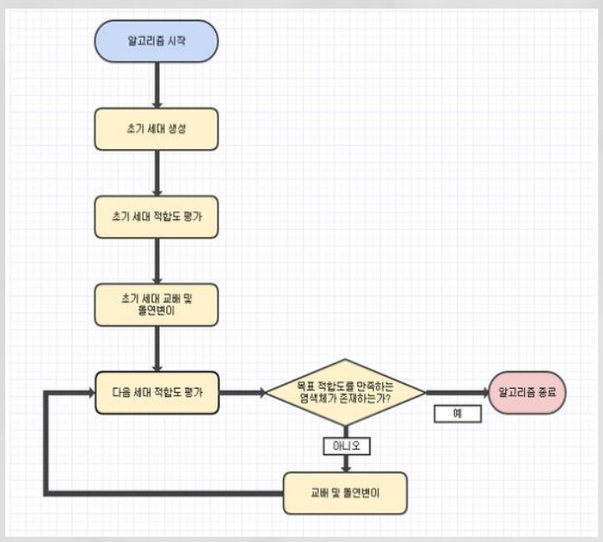
*용어 정리

- - 염색체(Chromosome) : 유전정보를 담고 있는 생물학적인 집합을 연속된 문자열로 추상화한 것.
- - 유전자(Gene) : 염색체를 구성하는 요소, 예를 들어 염색체가 ABC라면 유전자는 A 또는 B 또는 C를 뜻한다.
- - 교차(Crossover) : 두 개의 유전자가 각각의 유전자를 조합하여 새로운 염색체를 생성하는 연산.
- - 돌연변이(Mutation) : 교차연산 이후, 확률적으로 유전자의 정보가 바뀌는 것, 생물학적인 돌연변이와 같음.
- - 자손(Offspring) : 이전 세대의 염색체로부터 교차, 돌연변이 연산을 통해 생성된 다음 세대 염색체.

*유전 알고리즘이란?

- 1960년대 독일 대학원생인 Rechenberg 와 Schwefel에 의 해 제안현재 해에 정규 분포로 얻은 값을 합하는 것으로 유전 알고리즘의 변이 연산에 해당, 지역 최적점에서 탈출할 수 있도록 해줌.
- 기존 미분에 기반한 최적화 알고리즘과의 차이점초기 해를 여러 개 생성하여 이것을 조금씩 개선

*Flow Chart



*CPU로 구현방법

1.

- Initialize함수는문자형배열A와정수형변수nLength를매개변수로하는함수로 문자열(유전자)를생성하고배열에저장
- (nLength*MAX)개의 아스키코드에 따른 알파벳, 공백(‘ ’)문자를 랜덤으로 생성한다.
- 생성된 랜덤문자는 배열 A에 저장한다.

2.

- Evaluation함수는 유전자(문자열)를 평가, 즉 점수를 매기는 단계
- A 배열이 행이 MAX이고 열이 nLength인 행렬을 1차원 배열로 나타냈다고 생각했을 때 A의 문자와 입력된 문자가 같으면 nSum값을 증가
- nSum[i]는 A배열의 i번째 행의 랜덤 문자열의 점수를 평가한 것

3.

- Selection함수는 우성 유전자를 선택하는, 즉 높은 점수로 평가받은 문자열을 선택하는 단계
- 샘플로 추출할 유전자의 개수 SAM만큼의 nSamList를 생성하고 nSum의 값을 내림차순으로 SAM개 만큼 저장
- nSamList의 인덱스에 해당하는 배열A의 문자를 배열 ABackUp에 저장
- 이 과정을 통해 우수한 유전자(점수가 높은)가 후대로 생존

4.

- Replace함수는 부모 유전자를 통해 자손 유전자를 생성하는 단계
- 정수 부(F)와 모(M)는 부,모로 선택될 문자열 행 랜덤하게 결정
- 정수 nCut은 부모로 부터 얼만큼의 유전자를 물려받을지 랜덤하게 결정
- 새로운 자손은 F유전자에서 좌측 nCut만큼, M유전자에서 우측 (nLength-nCut)만큼의 문자열을 물려받고 A배열에 저장

5.

- Main함수를 통한 알고리즘의 실행
- 필요한 배열 할당 및 목표 문자열(Input Sentence)입력
- 초기 유전자를 Initialize함수를 통해 초기화
- 초기화 후 무한루프를 통해 Evalution, Selection, Replace의 단계를 거치고 1000번째 실행 단위로 중간 결과 출력
- 입력 문자열과 동일한 결과가 나오면 최종 결과를 출력하고 반복문 종료

*GPU로 구현방법

1.

- TILE_WIDTH는 32로 설정(한 블록의 크기 : 32*32)
- 쓰레드 인덱스를 i와 j로 설정
- CPU에서 반복문을 통해 순차적으로 생성한 랜덤 문자를 병렬적으로 생성하고 배열 A에 저장

2.

- Replace는 행 단위로 실행되기 때문에 쓰레드 인덱스 x인 i만 설정
- CPU에서 반복문을 통해 0부터 MAX행까지 순차적으로 생성하던 nCut, F, M을 병렬적으로 설정
- 자손 문자열을 배열 A에 저장

- 3.
- Selection함수는 우성 유전자를 선택하는, 즉 높은 점수로 평가받은 문자열을 선택하는 단계
- 샘플로 추출할 유전자의 개수 SAM만큼의 nSamList를 생성하고 nSum의 값을 내림차순으로 SAM개 만큼 저장
- nSamList의 인덱스에 해당하는 배열A의 문자를 배열 ABackUp에 저장
- 이 과정을 통해 우수한 유전자(점수가 높은)가 후대로 생존

*돌연변이 해결방법(CPU)

```
void Mutation(char* _A, int _nLength) {
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < _nLength; j++) {
            if (rand() % 1000 == 0) {
                bool finding = true;
                int RndAsc = 0;
                while (finding) {
                    RndAsc = rand() % ((123 - 65) + 1) + 65;
                    if (RndAsc == 123) RndAsc = 32;
                    if ((RndAsc <= 90) || (RndAsc >= 97)) finding = false;
                }
                _A[i * _nLength + j] = char(RndAsc);
            }
        }
    }
}
```

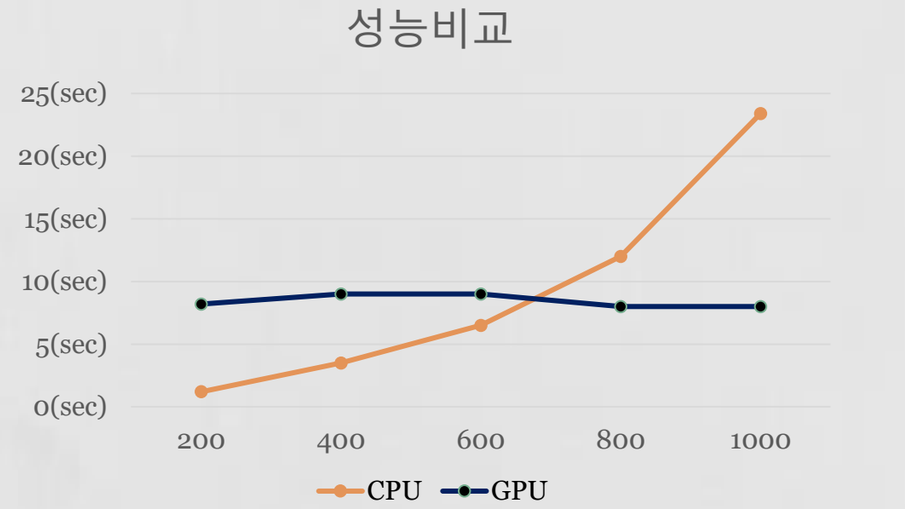
- Mutation 함수는 유전자(문자열)에 돌연변이를 발생시키는 함수
- 유전자(문자열)가 저장된 배열 A에 0.1%의 확률로 랜덤 문자로 변경하여 돌연변이 발생
- Initialize함수에서 입력 문자열과 일치하지 않는 문자가 생성되어 결과가 무한하게 문자열이 일치하지 않는 문제를 돌연변이를 통해 해결

*돌연변이 해결방법(GPU)

```
__global__ void Mutation(char* _A, int _nLength) {
    int i = blockIdx.x *blockDim.x + threadIdx.x;
    int j = blockIdx.y *blockDim.y + threadIdx.y;
    if (i < MAX && j < _nLength) {
        if ((clock() + blockIdx.x - blockDim.y * threadIdx.x) % 1000 == 0) {
            bool finding = true;
            int RndAsc = 0;
            while (finding) {
                RndAsc = (clock() + blockIdx.y - blockDim.x * threadIdx.y) % ((123 - 65) + 1) + 65;
                if (RndAsc == 123) RndAsc = 32;
                if ((RndAsc <= 90) || (RndAsc >= 97)) finding = false;
            }
            _A[i * _nLength + j] = char(RndAsc);
        }
    }
}
```

- x,y 쓰레드 인덱스를 i, j 저장
- CPU에서 반복문을 통해 순차적으로 발생시키던 돌연변이 문자를 병렬적으로 생성

*CPU와 GPU간의 성능비교



*CPU : i5-8500,
*GPU : **RTX 2060 WINDFORCE OC D6 6GB** 를 사용하였음.

1. CPU는 데이터가 선형적으로 증가함에 따라 소요시간은 지수적으로 증가한다.
2. GPU는 데이터의 수가 쓰레드(CUDA 코어)의 수보다 많아 지기 전까지는 거의 일정한 시간이 소요하게 된다.