병렬 처리 설계

자율주행 연구실 / 인턴 김시현

목차

GIST

병렬 처리

GIST

- 한 개의 cpu의 처리 속도가 너무 느리기 때문
- 여러 개의 작업을 동시에 수행함으로써 작업 속도를 높일 수 있다.

  1. 멀티 스레드
  2. 멀티 프로세스

- 오버헤드의 증가 및 메모리 사용률이 높아지는 단점이 존재

Python – GIL 정책

GIST

- Global Interpreter Lock
- Python 언어에서 자원을 보호하기위해 사용
- **하나의 프로세스** 안에 모든 자원의 락(Lock)을 글로벌(Global)하게 관리함으로써

  한번에 하나의 쓰레드만 자원을 컨트롤하여 동작 하도록 한다.

  - Ex. 두개의 쓰레드를 동시에 실행시켜도,

    결국 GIL 때문에 한번에 하나의 쓰레드만 실행하여 작업 시간이 비슷하게 나온다.

Python – GIL 정책

GIST

- GIL이 적용되는 것은 **cpu 동작**에서만
  - Ex. 계산 처리

- **I/O 작업**을 실행하는 동안에는 다른 쓰레드가 cpu 동작을 동시에 실행할 수 있다.
- → I/O작업은 병렬 처리 가능 !
  - Ex. 대량의 파일 처리

설계 코드

GIST

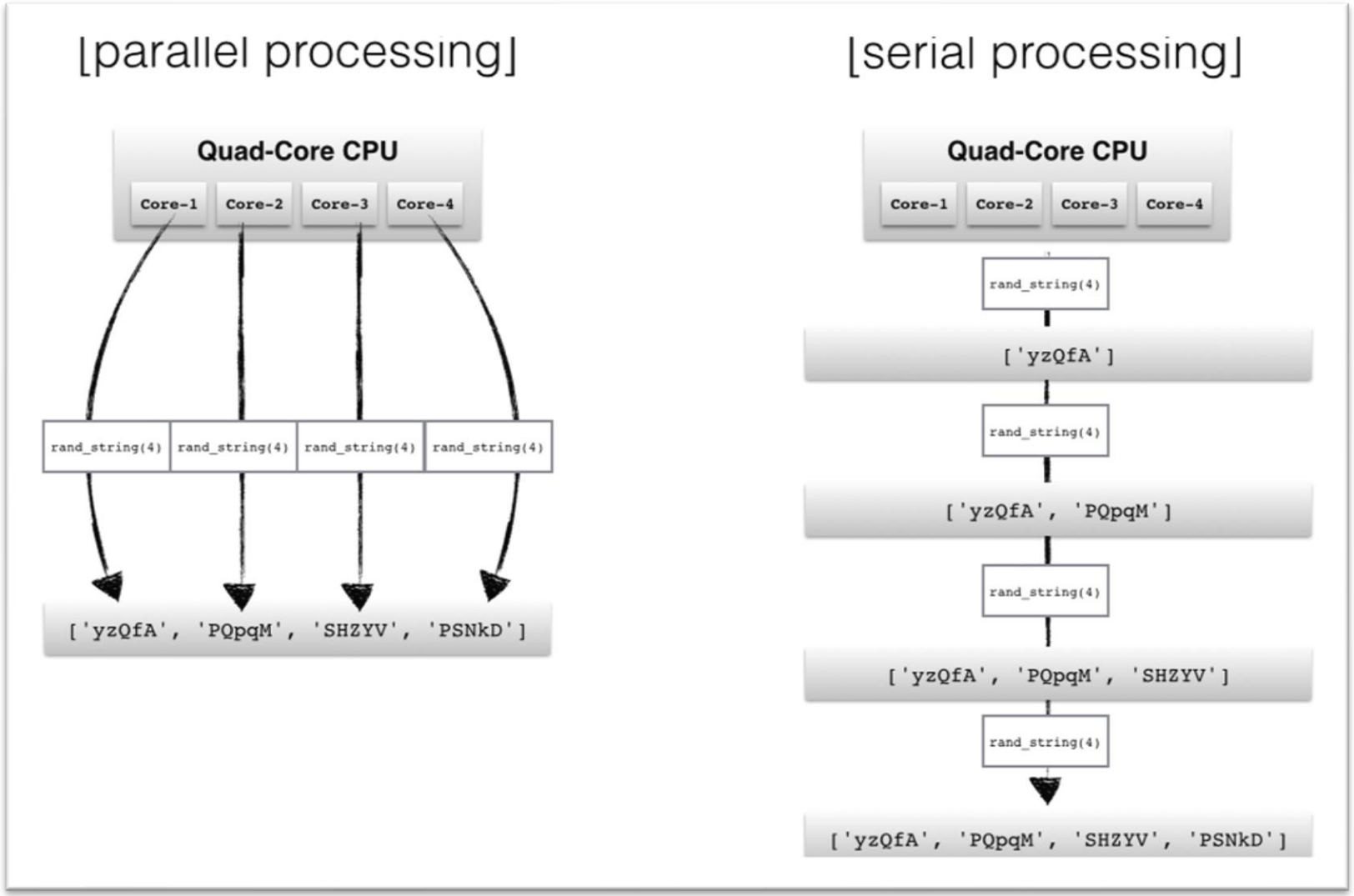
A. 싱글 프로세스

B. 멀티 스레드

C. 멀티 프로세스

- Pool
- Process

D. CPU 개수 확인



깃허브 주소 : https://github.com/ssihh/GIST-Intern

A. 싱글 프로세스

GIST

```python
import time
import shutil  # 셸 유티리티
import os

path = './test7GB/'  # 경로 설정
files = os.listdir(path)  # listdir() 해당 경로의 파일을 리스트로 반환


# 파일 복사 함수
def copy_test(partial_works):
    for w in partial_works:
        shutil.copy(path + files[w], './testclone/' + files[w])


# 싱글 프로세싱
if __name__ == "__main__":
    start_time = time.time()
    copy_test(range(2300))  # 2300장 복사
    print("--- elapsed time %s seconds ---" % (time.time() - start_time))

```

- time – 작업 시간 측정
- shutil – 데이터 복사/이동
- os – 파일 목록을 리스트로 반환

B. 멀티 스레드

GIST

```python
import threading

path = './test7GB/'  # 경로 설정
files = os.listdir(path)  # listdir() 해당 경로의 파일을 리스트로 반환


def copy_test(partial_works):  # 파일 복사 함수
    for w in partial_works:
        shutil.copy(path + files[w], './testclone/' + files[w])


# 스레드 개수와 스레드 리스트
thread_count = 10
threads = []

if __name__ == "__main__":
    start_time = time.time()
    # 새로운 스레드 생성/실행 후 스레드 리스트에 추가
    for i in range(thread_count):
        thread = threading.Thread(target=copy_test, args=(range(i * 230, (i + 1) * 230),), )
        thread.start()
        threads.append(thread)
    # 메인 스레드는 각 스레드의 직업이 모두 끝날 때 까지 대기
    for thread in threads:
        thread.join()
    print("--- elapsed time %s seconds ---" % (time.time() - start_time))
```

- threading - 스레드 사용
- thread.join - 다른 스레드의 실행이 끝날 때까지 join() 함수를 호출한 스레드(메인 스레드)는 대기한다.

C. 멀티 프로세스 - Pool 코드

GIST

```python
import multiprocessing

# 파일복사
path = './test5.66GB/'
clone = './testclone/'
files = os.listdir(path)  # listdir() 해당 경로의 파일을 리스트로 반환


def copy_test(partial_works):  # 파일 복사 함수
    print("value %s is in PID : %s" % (partial_works[0], os.getpid()))  # 프로세스 아이디
    for w in partial_works:
        shutil.copy(path + files[w], clone + files[w])


list = []
for i in range(10):
    list.append(range(i * 230, (i + 1) * 230))


if __name__ == "__main__":
    start_time = time.time()
    # 멀티프로세싱(Pool)
    pool = multiprocessing.Pool(processes=5)
    pool.map(copy_test, list)
    pool.close()
    pool.join()
    print("--- elapsed time %s seconds ---" % (time.time() - start_time))
```

- multiprocessing - 멀티 프로세싱을 위한 모듈 임포트
- Pool - 프로그램 내의 여러 자식 프로세스를 쉽게 실행하고 풀에서 작업자를 선택할 수 있다.
- (Processes = 5) - 5개의 프로세스로 처리함.
- map - 병렬 처리할 함수와 그 함수의 매개변수를 넘겨준다.

C. 멀티 프로세스 - Pool 결과

GIST

MPpool

```
C:\Users\User\anaconda3\python.exe C:/Users/User/PycharmProjects/pythonProject/dsttest/MPpool.py
value 0 is in PID : 2176
value 230 is in PID : 5432
value 460 is in PID : 11380
value 690 is in PID : 8292
value 920 is in PID : 4080
value 1150 is in PID : 4080
value 1380 is in PID : 11380
value 1610 is in PID : 8292
value 1840 is in PID : 8292
value 2070 is in PID : 8292
--- elapsed time 22.60959005355835 seconds ---
```

- PID - 각기 다른 5개의 프로세스 id를 사용하는 것을 확인할 수 있다.

C. 멀티 프로세스 - Process 코드

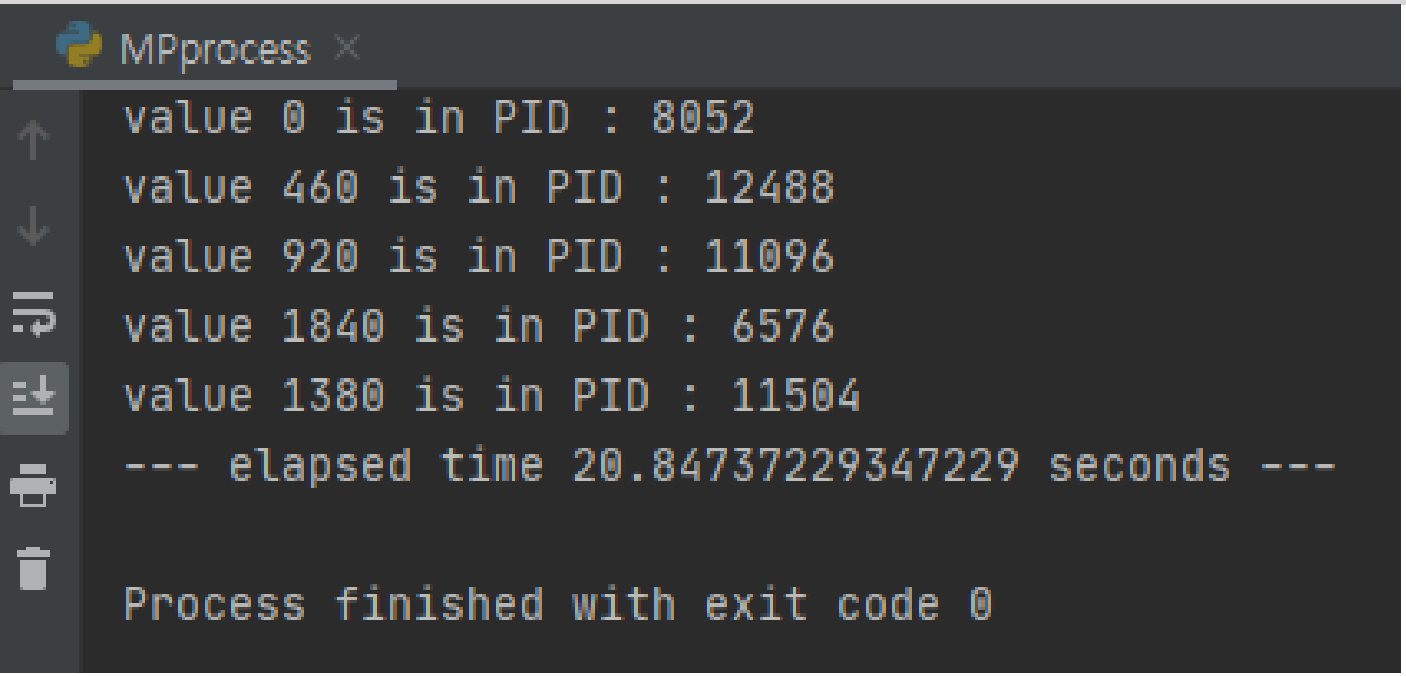
GIST

```python
from multiprocessing import Process

# 파일복사
path = './test5.66GB/'
files = os.listdir(path)  # listdir() 해당 경로의 파일을 리스트로 반환


def copy_test(partial_works):  # 복사
    print("value %s is in PID : %s" % (partial_works[0], os.getpid()))  # 프로세스 아이디
    for w in partial_works:
        shutil.copy(path + files[w], './testclone/' + files[w])


list = []
for i in range(0, 10, 2):
    list.append(range(i * 230, (i + 2) * 230))
procs = []

if __name__ == "__main__":
    start_time = time.time()
    # 멀티 프로세스 - process
    for i in list:
        proc = Process(target=copy_test, args=(i,))
        procs.append(proc)
        proc.start()
    for proc in procs:
        proc.join()
    print("--- elapsed time %s seconds ---" % (time.time() - start_time))
```

- multiprocessing - 멀티 프로세싱을 위한 모듈 임포트

C. 멀티 프로세스 - Process 결과

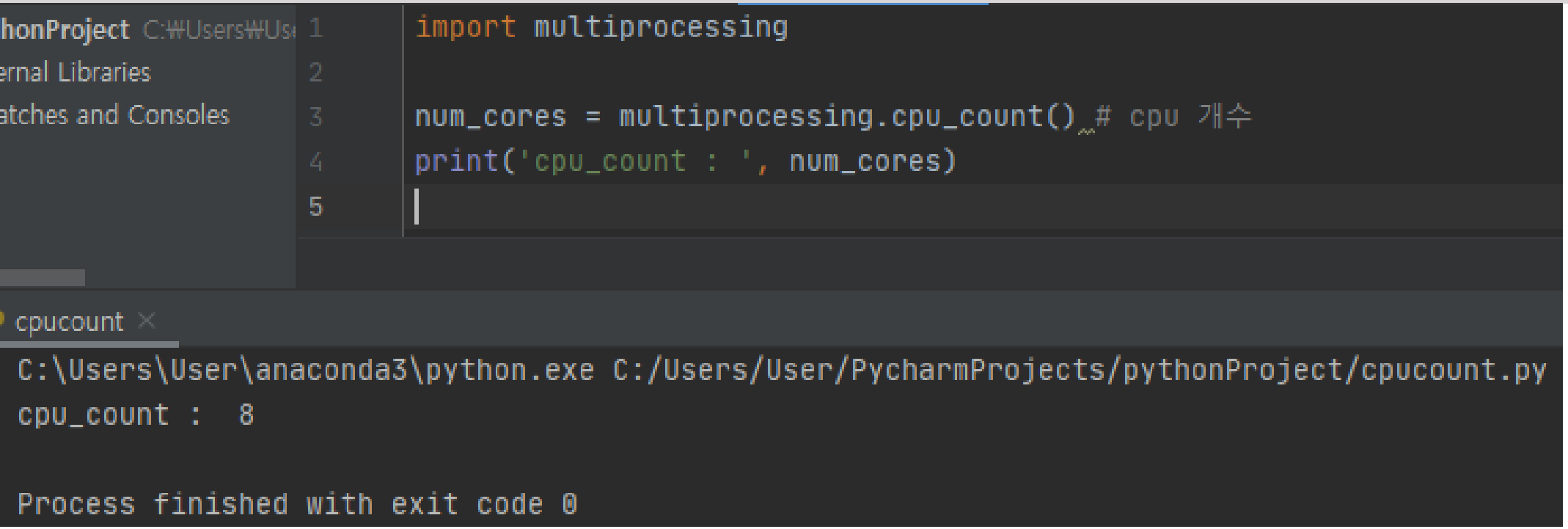
GIST



- PID - 각기 다른 5개의 프로세스 id를 사용하는 것을 확인할 수 있다.

D. Cpu 개수 확인

GIST



- cpu_count : 8 - cpu개수를 확인하여 멀티 프로세싱에 활용

멀티 프로세스 코드 사용법

- Pool -

멀티 프로세스 – Pool 사용법 (1)

GIST

```
if __name__ == "__main__":
        pool = multiprocessing.Pool(processes=4[사용할 프로세스 개수])
        pool.map([함수명]textset, [매개변수]label_path_multi)
        pool.close()
        pool.join()
```

매개변수가 한 개인 함수 예시

```
def textset(Label_train):
        print("value %s is in PID : %s" % (Label_train[0], os.getpid())) # PID 출력
        for file in Label_train:
                shutil.move(original_Dir + file, new_Dir + file)
```

멀티 프로세스 – Pool 사용법 (2)

GIST

```
if __name__ == "__main__":
        pool = multiprocessing.Pool(processes=4[사용할 프로세스 개수])
        pool.map([함수명]textset, [매개변수]label_path_multi)
        pool.close()
        pool.join()
```

매개변수 리스트

```
label_path_list1 = label_path_list[:int(len(label_path_list) / 4)]
label_path_list2 = label_path_list[int(len(label_path_list) / 4):2 * int(len(label_path_list) / 4)]
label_path_list3 = label_path_list[2 * int(len(label_path_list) / 4):3 * int(len(label_path_list) / 4)]
label_path_list4 = label_path_list[3 * int(len(label_path_list) / 4):]
label_path_multi = [label_path_list1, label_path_list2, label_path_list3, label_path_list4]
```

멀티 프로세스 – Pool 사용법 (3)

GIST

```
if __name__ == "__main__":
        pool = multiprocessing.Pool(processes=4[사용할 프로세스 개수])
        pool.starmap([함수명]copy_test, [매개변수]zip(list, repeat(path), repeat(files)))
        pool.close()
        pool.join()
```

매개변수가 여러 개일 때
① zip 과 repeat 사용
② 기존의 map을 starmap으로 변경

매개변수가 여러 개인 함수 예시

```
def copy_test(partial_works, path, files): # 복사
        for w in partial_works:
                shutil.copy(path+files[w], clone+files[w])
```

감사합니다

End