The Journal of the Convergence on Culture Technology (JCCT)
Vol. 8, No. 2, pp.361-366, March 31, 2022. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2022.8.2.361

JCCT 2022-3-47

GO언어를 이용한 대용량 데이터 리스트의 동시성 처리 비교

Concurrency processing comparison of large data list using GO language

이요셉*, 임영환**

Yoseb Lee*, Young-Han Lim**

요 약 대용량 데이터를 처리 하는 방식은 여러 가지가 있다. 처리 방법에 따라서 대용량 데이터 리스트를 만드는데 처리속도가 많은 차이를 두고 있다. 대표적으로 대용량 데이터 리스트를 만들기 위해서 대용량 데이터를 정규화 된 쿼리로 만들고 만들어 낸 쿼리의 결과를 List Map 담아두고 출력 가능한 형태로 변환 한다. 이러한 과정은 단계별로 처리 속도가 저하되는 원인으로 발생된다. 만들어 낸 쿼리의 결과를 List Map으로 담는 과정에서 데이터의 형태별로 저장되는 형식 다르기 때문에 처리속도의 차이가 나타난다. GO언의 동시성 처리를 통해서 기존에 처리속도의 차이가 발생되던 문제를 해결하고자 한다. 즉 GO언의 동시성 처리 결과가 기존의 List Map에 담는 형식과 동시성을 사용하여 처리하는 방식의 대용량 데이터 리스트 처리 방식이 얼마나 차이가 나고 어떠한 방식으로 진행되는지를 제공하여 보다 빠른 처리를 할 수 있도록 비교 한다.

주요어 : GO언어, 리스트맵, 동시성, 대용량 데이터

Abstract There are several ways to process large amounts of data. Depending on the processing method, there is a big difference in processing speed to create a large data list. Typically, to make a large data list, large data is converted into a normalized query, and the result of the query is stored in a List Map and converted into a printable form. This process occurs as a cause of lowering the processing speed step by step. In the process of storing the results of the created query as a List Map, the processing speed differs because the data is stored in a different format for each type of data. Through the simultaneous processing of GO language, we want to solve the problem of the existing difference in processing speed. In other words, it compares the results of GO language concurrency processing by providing how different and how it proceeds between the format contained in the existing List Map and the method of processing using concurrency in large data lists for faster processing. do.

Key words : GO language, List Map, Concurrency, Large Data

*정회원, 숭실대학교 미디어학과 박사과정 (제1저자 교신저자)
**정회원, 숭실대학교 미디어학과 교수 (참여저자)
접수일:2022년 2월 14일, 수정완료일: 2022년 3월 1일
게재확정일: 2022년 3월 8일

Received: February 14, 2022 / Revised: March 1, 2022
Accepted: March 8, 2022
*Corresponding Author: hehe82@nate.com
Dept. of Media, Soongsil Univ, Korea

Ⅰ. 서 론

대용량 데이터를 저장하는 데이터베이스는 여러 종류가 있다. PostgreSQL, MongoDB, MariaDB 이러한 데이터 베이스 들이 존재한다. 대용량 데이터베이스의 빅데이터 처리를 하기 위해서는 여러 가지 개발 언어로 진행이 된다. JAVA, PHP, Python 등 개발 언어를 통해서 데이터베이스 안에 있는 빅데이터를 처리를 한다. 빅데이터를 제공하는 방식으로 XML(eXtensible Markup Language), JSON(JavaScript Object Notation) 의 방식으로 표현 되어 여러 형태 서비스에서 활용 할 수 있도록 개발되어 제공 되고 있다. 이와 같은 방식으로 제공되기 위해서 대용량 데이터 처리의 속도가 중요하다. 또한 빅데이터를 처리하기 때문에 제공되는 속도에 따라서 서비스의 만족도가 달라지기 때문에 데이터 처리를 신속하게 처리 하는 것이 중요하다.[5]

빅데이터 데이터베이스는 PostgreSQL를 사용하여 출력하고자하는 데이터 23,698건의 데이터를 리스트화 하여 제공하고자 한다.

개발언어 두 가지를 이용하여 개발 방식 및 처리 속도를 비교하고자 한다.

첫 번째 빅데이터를 처리하는 언어는 Python을 이용하려고 한다. Python의 NumPy를 이용하여 빅데이터 리스트를 처리 하려고 한다. 처리를 하기 위해서는 NumPy 모듈의 List Map을 이용하여 데이터를 만들고 JSON 모듈을 통해서 리스트를 출력하려고 한다.

Go 언어는 GO 언어에서 제공하는 List Map를 제공하는 함수로 진행하려고 한다. 제공되는 함수에 동시성을 연결하여 대용량 데이터 처리 후 같은 모듈을 사용하여 JSON으로 출력 하여 속도를 비교 한다.

본 논문에서는 Go언어의 동시성을 이용하여 기존의 대용량 데이터 리스트의 처리 방식을 비교하고 처리 되는 속도를 비교 하고자 한다.

Ⅱ. Python 대용량 데이터 리스트 처리

이 절에서는 Pyhon에서 대용량 데이터를 어떻게 연결하고 List화 하여 JSON로 출력하는지 소개한다.

1. PostgreSQL 대용량 데이터 연결

PostgreSQL를 Python에 연결하기 위해서는 psql이라는 driver(adapter)를 이용해야 한다. psql을 Python과 함께 가장 많이 사용하고 있는 dirver는 psycopg2로 연결하여 데이터를 연동한다.[1]

Psycopg2 패키지가 설치가 되고 Connect 메서드를 이용 하면 Cursor가 생성이 된다. Cursor를 통해서 인스턴스가 생성되면 데이터 조작을 진행 할 수 있다.

데이터베이스 연동 하는 방법은 4가지로 나눌 수 있다. 첫 번째로는 연결(Connection), 두 번째로는 변경(insert, delete, update), 세 번째로는 조회(select), 네 번째로는 해제(Close) 이러한 네 가지 방법을 통해서 데이터를 조작 할 수 있다. 우리는 연결, 조회, 해제 이 세 가지만을 이용하여 데이터베이스를 연결하고 데이터를 조작하려고 한다.

연결된 데이터베이스에 Cursor를 통해서 데이터를 호출 할 수 있다. “Cursor.execute”를 통해서 데이터의 쿼리를 만들어 낼 수 있다. 만들어 낸 데이터의 내용을 적용 하기 위해서는 “Connection.commit()”을 사용하여 출력 하기 위한 데이터를 저장 할 수 있다.

조회를 통해서 대용량 데이터 가져와야 한다. 조회된 데이터는 쿼리를 이용해서 해당 데이터를 만들어 낼 수 있다. Fetch() 통해서 만들고자 하는 데이터를 리스트 형식을 만들 수 있다. Fetchone(). fechmany(), fetchall() 3개의 메소드가 있는데 우리는 fechmany() 메소드를 사용하여 대용량 데이터의 리스트를 생성 할 수 있는 자료를 가져오려고 한다.[2]

대용량데이터의 해제는 close() 메소드를 사용하여 진행이 된다. Python에 경우 with 키워드를 통해서 자동으로 해제가 가능하고 자동으로 해제된 close()는 반환이 되어서 다시 데이터를 가져올 수 있도록 초기화된다.

2. 대용량 데이터 리스트 저장

대용량 데이터베이스 리스트를 만들기 위해서 SQLAlchemy, psycopg2, ipython-sql, sql_magic 패키지를 이용하여 리스트를 생성 하려고 한다.

ipython-sql 패키지로 DB로 접속을 하고 table에 해당 데이터를 리스트화 시키려고 한다.

variable_name과 같은 named style를 사용하여 table 값을 localnamespace에 생성한 변수 이름을 SQL query에 넣어서 동적으로 값을 바꾸어 가면서 query를 적용하려고 한다.[3]

variable_name를 SQL의 정수(Integer) 변수 값으로 대체하고 문자형(Character) SQL 데이터는 변수 값으로 대체 할 수 있도록 형변환을 지정해 둔다. localname space에 문자형의 bind parameter 값에 해당하는 문자형은 큰따옴표(“”)로 감싸준다.

query를 통해서 데이터 리스트를 생성하고 리스트의 형태를 만들기 위해서 row정보를 형태에 맞게 설정하고 리스트 구조를 설정하게 된다.

3. 대용량 데이터 JSON 형식 출력

대용량 데이터를 정형화된 데이터로 만들어 출력하려고 한다. 정형화된 데이터를 통해서 데이터의 구조가 정해지고 데이터를 사용 할 수 있는 구조가 완성이 된다. 정형화된 데이터구조를 JSON형식으로 만들어 제공하려고 한다. 대용량 데이터가 가지고 있는 데이터 베이스의 내용을 리스트형태로 가지고 있게 되었다. 리스트형태의 대용량 데이터를 JSON 형식의 구조로 만들고 제공 하려고 한다.

Python에서는 JSON형태의 구조로 만들기 위해서는 json.dumps 메소드를 이용한다.

json.dumps()함수는 주어진 오브젝트를 JSON형식 문자열로 직렬화 한다. json.dumps()함에서 키워드 매개변수 Indent에 양의 정수를 주어 주어진 들여 쓰기 레벨로 오브젝트가 정형화 되게 된다. Ident가 0으로 설정되면 새로운 줄로 삽입하게 지정된다. 구분 문자를 지정 할수 있고 부분한 문자 인수는 Sepatators를 지정하게 되고 index를 지정한 인수는 인수 indent가 된다. 키로 정렬을 지정하게 되면 인수는 sort_keys로 지정되어 JSON형태의 문자열로 출력이 된다.[4]

III. Go 언어 대용량 데이터 리스트 처리

본절에서는 GO언의 대용량 데이터 리스트를 처리하는 방식에 대해서 설명하려고 한다.

1. GO언어 대용량 데이터 연결

데이터베이스에 필요한 연산들을 추상 인터페이스로 만들어서 사용하게 된다 . 사용하는 데이터베이스와 비즈니스 로직을 구분 할 수 있는 방법으로 GO언어에서 PostgreSQL은 연동하기 위해서 Data Access Object를 만들어서 연결을 하도록 하겠다.

데이터베이스의 특정 정보 ID를 해당하는 작업을 달라는 것을 Get이라고 하고 특정 ID의 작업을 넘겨준 내용을 변경하는 것을 Put이라고 한다. 새로운 작업을 추가 하는 것은 Post 라고 지정하여 대용량 데이터를 연결 할 수 있도록 설정을 한다.

인터페이스를 구현하는 구현체를 지정하고 지속적으로 작업 목록을 저장하고 있으면 데이터베이스에서 할 일의 목록을 미리 저장해두어야 한다. 서버가 죽으면 작업 목록도 모두 사라지지만 데이터베이스를 지원하고 싶을 때 접근하는 DataAccess 인터페이스를 지정해두면 불필요한 연결 지연 없이 동작 할 수 있다. 이러한 DataAccess를 MemoryDataAccess라고 한다. MemoryDataAccess를 DataAccess의 인터페이스를 구현하고 컴파일러를 지정하면 대용량 데이터베이스에 연결의 구성을 지정할 수 있다.

2. 대용량 데이터 리스트 저장

대용량 데이터를 GO언어에서 리스트형식으로 하기 위해서는 배열과 슬라이스를 이용하여 리스트를 저장할 수 있다. 배열과 슬라이스 모두 연속된 메모리 공간을 순차적으로 이용하는 자료구인데 배열이 직접 사용되는 경우도 있지만 주로 슬라이스를 이용하여 간접적으로 배열을 이용하여 사용된다.

배열은 크기가 자료형에 고정이 되어 있다. 하지만 슬라이스는 “var bigdata [] string” 으로 사용이 되고 빈 슬라이스에 nil 값이 들어간다.

슬라이스의 대용량 데이터를 len(nums) +1 < cap(nums), 공식으로 리스트를 지정할 것이다. 즉 용량이 초과하지 않는 경우에 시작 위치에서 길이만큼 오른쪽으로 이동한 위치에서 새로운 값을 집어넣고 길이가 증가한 슬라이스를 반환하도록 지정한다. 길이가 증한 슬라이스를 nums에 다시 할당해야 하므로 두 번 반복해서 사용하도록 한다. 만약 용량이 초과하게 될 경우는 더 큰 배열의 새로운 배열을 만들고 슬라이스로 지정 하여 거기에 맞는 데이터 리스트를 반환하도록 할당한다.

3. 대용량 데이터 JSON 형식 출력

대용용량 데이터의 리스트 형태의 데이터를 JSON의 구조형태로 출력하여 지원하려고 한다. GO언어에서는 json.marshal함수를 호출하여 바이트 슬라이스와 구조체

내부에 있는 필드들을 대문자로 시작하여 바꾸어 진행하게 된다. 대문자로 시작하는 필드들은 같은 패키지뿐만 아니라 다른 패키지에서도 지정 할 수 있도록 설계를 하고 진행을 한다. GO언어에서는 JSON구조체를 직렬화를 하고 json.Unmarshal를 이용하여 구조체의 안에 값을 채워 줄 수 있다. 이때에는 포인터를 이용하여 수정된 것이 반영되므로 데이터형식의 대신 변환형 데이터로 지정해야 한다.

JSON 직렬화를 지정하기 위해서는 Status는 숫자로 직렬화가 되고 문자열은 역직렬화로 지정이 된다.

UnmarshalJSON함수를 사용하여 역따옴표로 한번 더 둘러싼 것은 넘어오는 데이터가 따옴표까지 포함된 문자열이기 때문에 JSON에서는 따옴표가 없이 수를 표현하고 따옴표를 넣는 형식은 문자열로 간주하여 직렬화를 진행한다.[8]

JSON의 직렬화 데이터를 구조화 하기위해서는 “json.string” 태그를 이용하여 문자열로 전달하여 직렬화 구조화 되고 사용 할 수 있게 된다.

Ⅳ. Go 언어의 동시성

GO언어의 동시성은 가벼운 쓰레드와 같은 것으로 현재 수행 흐름과 별개로 흐름을 만들어 사용하도록 한다. 생성하는 법은 함수를 호출하는 방식과 비슷하다. f(x,z,y)이렇게 함수를 호출하면 수행되는 흐름이 함수를 따라가서 수행이 종료되고 돌아온다. 따라서 현재 함수는 f가 종료 될 때까지 다른 수행을 하지 않는다.

GO언의의 동시성은 함수 앞에 go를 붙여서 이와 같이 함수를 호출한다. f(x,z,y) 호출과 현재의 함수의 흐름은 메모리를 공유하는 논리적으로 별개의 흐름으로 진행되게 된다. 동시에 각각의 본인의 기능을 별개로 진행하는 것이다. 동시성이 있는 두 루틴은 서로 의존관계가 없다. 동시성은 병렬성과는 다르시만 동시성이 있어야 병렬성이 생기게 된다. 서로 어느 것이 먼저 되어야 하는 의존 관계가 있는 것은 함께 진행 될 수 없지만, 선 순서에 맞게 함수를 진행하게 된다.[6]

동시성은 메모리도 서로 공유하게 된다. 파일시스템을 이용하는 것이 아니더라도 고루틴이 변수의 포인터를 받아서 해당변수에 원하는 값을 넣어 줄 수 있다. 이렇게 사용 하는 GO언어의 동시성은 함수가 가지고 있는 고유의 기능을 순차적으로 동시에 사용하게 된다.

서로 다른 단계를 동시에 이루어질 수 있기 때문에 성능상의 장점이 있을 수 있다. 컴퓨터 시스템에서는 특히 서로 다른 종류의 하드웨어들이 어떤 일을 해야 할 때 파이프라인이 큰 효과를 가져 올 수 있다. 소프트웨어에서는 들어오는 데이터와 나가는 데이터에 집중하여 문제를 풀고 자 할 때 장점이 있고 버퍼를 활용하면 경우에 따라 성능상의 장점도 얻을 수 있다. 이렇게 사용한 메모리의 버퍼를 통해서 같은 함수뿐만 아니라 형태만 같다면 서로 다른 함수들도 이어 붙일 수 있다. 파이프라인의 앞의 과장에서는 결과물이 빨리 나오는데 뒤의 과정은 시간이 오래 걸려서 여러 곳에 결과물을 나오는 과정을 채널 하나를 여럿이서 사용 할 수 있도록 동시성의 패턴을 연결 할 수 있다. 이렇게 사용한 채널은 채널이 닫히면 채널에서 자료를 받아가는 동시성이 몇 개인지 상관없이 모든 동시성의 채널을 닫고 데이터 함수 처리의 결과를 반환한다.

동시성의 파이프라인을 통해서 처리된 데이터를 통과시키고 함수화 시키면 분산처리가 된다. 설정해 놓은 함수는 파이프라인 패턴에서 처리를 하고 분산처리 함수로 돌려주는 방식이다. 이렇게 다양한 방식으로 파이프라인을 구성하여 분산처리 하는 방식을 통해서 GO언의 동시성 처리를 진행하게 된다.[7]

대용량 처리의 리스트를 만들 때 리스트 row에 진행되는 각 자료형을 함수로 지정하여 함수처리를 동시성으로 사용하려고 한다.

Ⅴ. 실험 및 결과

Python를 이용한 대용량 리스트 처리 기능과 GO언어의 동시성을 이용한 대용량 리스트 처리기능의 처리 속도를 테스트하여 비교하려고 한다.

실험을 진행하기 위해서는 Python을 실행하기 위한 서버와 GO언어를 실행하기 위한 서버를 구성하였다.

서버는 클라우드서버를 활용하였고 PostgreSQL과 연동이 되기 때문에 인터넷 회선을 100Mbps기준으로 구성을 하고 PostgreSQL 데이터베이스 안에 테이블 정보인 URSR_INFO의 23,698건으로 실험을 진행하려고 한다.

Python에서 query를 통해서 리스트형태로 만들기위해서 호출이 되는 속도롤 비교하려고 한다.

서버의 사양은 4Core CPU에 Memory는 4GB로 지정

하였다.

OS는 CentOS7.0으로 최신 업데이트를 지정하였다.

방화벽은 별로 지정하지 않고 PostgreSQL 호출을 하기 위한 기본 Port만 오픈하였다. 서버의 가동률을 95%까지 설정을 하였고 같은 조건으로 실행하기 위한 준비를 하였다.

표 1. Python과 GO언의 PostgreSQL query 리스트 호출 시간
Table 1. PostgreSQL query list call time in Python and GO language

	결과 ROW건수	조회 시간
Python	23,698	1.124sec
GO언어	23,698	1.348sec

표 1에서와 같이 PostgreSQL ROW 데이터인 23,698건의 query 리스트를 만들기 위한 호출 속도를 비교 하였다. Python 조회시간이 1.123sec가 나왔고 GO언의조회시간은 1.328sec로 GO언어가 조금 더 느리게 query 리스트를 생성하기 위한 데이터 조회의 속도차가 나고 있다는 것을 시간으로 차이가 난다.

그렇다면 JOSN의 구조를 가진 대용량 데이터 리스트를 생성하여 출력하려고 한다. PostgreSQL 테이블 필드에 맞는 데이터 형식으로 함수를 만들고 Python은 생성하면서 함수를 실행하는 방식이고 GO언어는 생성하면서 동시성을 사용해서 함수를 연결 해두었다.

각 자료형 함수 마다 순번을 설정했고 실행되는 함수마다 데이터가 처리되는 각 단계를 로그로 기록하여 처리가 되는 형태를 파악하고 출력되는 시간을 기록 할 수 있도록 진행하였다.

표 2. Python과 GO언의 PostgreSQL 대용량 데이터를 JSON 구조로 출력한 시간 비교
Table 2. Time comparison of outputting large amounts of PostgreSQL data in JSON structure between Python and GO language

	결과 ROW건수	처리 함수 건수	JOSN 구조 출력 시간
Python	23,698	27건	1.629sec
GO언어	23,698	27건	1.484sec

그렇다면 JOSN의 구조를 가진 대용량 데이터 리스트를 생성하고 각 필드의 자료형를 처리하는 함수를 지정하고 처리 했을 때 출력되는 시간이 어떻게 되는지 확인 할 수 있다.

표 2에서와 같이 총 23,698건의 데이터 ROW와 PostgreSQL의 자료형을 처리하는 27건의 함수를 처리하고 JSON구조로 출력되는 시간을 작성하였다. 출력시간이 Python은 함수처리를 진행하고 JSON 구조로 출력된 시간은 1.629sec이고 GO언의 동시성을 이용하여 함수 처리 27건을 진행한 JOSN형태의 출력이 1.484sec가 나오게 되었다. 각 함수의 처리 결과를 로그에 기록된 내용으로 확인해보면 Python은 함수 처리 결과가 순차적으로 출력 되어 있고 GO언어의 함수 처리 로그의 내용은 순차적이지 않고 먼저 처리가 된 순서대로 로그의 순번이 출력 되어 있었다.

Ⅵ. 결 론

GO언어의 동시성을 사용하여 대용량 데이터의 처리를 하고 데이터리스트 형태의 JSON구조로 변경하여 기본 Python에서 사용되는 형태의 대용량 데이터 처리 방식과 JSON형태의 리스트를 출력하는데 처리 결과가 어떻게 차이가 나는지 확인을 하였다. Python에서는 PostgreSQL을 호출하는 query의 속도는 GO언어보다 빠른 결과를 확인 할 수 있었다. 하지만 JSON형태로 변형하고 PostgreSQL의 테이블 데이터 자료형을 변환하긴 위한 함수를 적용하고 처리하는 과정에서의 결과값은 GO언어 동시성을 이용하여 처리하는 속도가 조금 더 빠르게 처리되었다. 실험의 결과가 특정 기능은 특정 개발 언어의 따라서 처리 속도가 다를 수 있겠지만 GO언어 동시성을 이용하여 처리하는 기능을 적용 하며 순차적으로 사용되는 다른 특정 언어 보다 조금 더 빠른 처리 결과를 만들어 낼 수 있을 것이다. 특정 다른 개발언어에서도 분명 GO언어와 비슷한 처리 방법이 있겠지만 GO언어의 특성상 만들어 놓은 함수에 GO라는 처리를 통해서 간단하게 동시에 프로세스를 처리 할 수 있게 하는 동시성의 기능을 이용한다면 한꺼번에 처리해야 하는 대용량 데이터베이스 처리를 위한 방법으로 좋은 발전이 될거라고 생각한다.

References

[1] So .hee. Kim, “Big data Analysis using Pyhon in Agriculture Forestry and Fisheries” The International Journal of Advanced Culture Technology (IJACT),

Vol. 5, No. 1, pp. 47-50, March 2016.
[2] Hye-Wuk Jung, “A Case Study of Python programming Error in an Online Learning Environment” the journal of the Convergence on Culture Technology (JCCT), Vol. 7, No. 3, pp. 247-253, August 2021.
[3] Hee-chan Yang, “An Implementation of Python Web Crawler Using Thread” Proceedings of korea Information Processing Society Conference (KEPA), Vol. 26, No. 2, pp. 70-72, 2019.
[4] Wonjun Jang, “An Efficient Data Transfer Algorithm based on Multi-thread for Big Data Transfer System” Proceedings of korea Information Processing Society Conference (KEPA), Vol. 26, No. 2, pp. 85-88, 2019.
[5] DaeWha Seo, “Parallel Processing : Distrbuted Shared Memory Scheme for Multi-thread programming” Proceedings of korea Information Processing Society Conference (KEPA), Vol. 3, No. 4, pp. 791-802, 1996.
[6] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, Koray Kavukcuoglu. "Efficient neural audio synthesis", arXiv preprint. https://arxiv.org /pdf/1802.08435.pdf, 2018, Feb.
[7] Hyun-sik An, “Constructing Effective Code Analyzer to Measure theQuality of Blockchain Code based on Go Languag” Proceedings of korea Information Processing Society Conference (KEPA), pp. 694-696, 2019.
[8] Hyo-Jong Kim, “Crepe Search System Design using Web Crawling” Journal of digital convergence Vol. 15, No. 11, (JDC), pp. 261-269, 2017.