# 3. Morpheus ControlMessage 구조 및 활용

## 프로젝트 개요

Morpheus 파이프라인에서 데이터와 메타데이터, 태스크 정보를 통합적으로 관리하는 핵심 메시지 타입인 \*\*ControlMessage\*\*의 구조와 사용법을 학습하고 실습했습니다. 이는 복잡한 파이프라인 로직과 SIMD(Single Instruction Multiple Data) 처리를 위해 필수적입니다.

## ⚙️ 사용 기술 및 역할

* **주요 기술:** NVIDIA Morpheus SDK, RAPIDS cuDF
* **핵심 역할:** ControlMessage 인스턴스 생성 및 MessageMeta(cuDF DataFrame 기반) 페이로드 설정, 메타데이터(metadata) 및 태스크(tasks) 추가/조회/제거 API 활용.

## 💡 ControlMessage의 4가지 구성 요소

ControlMessage는 기존 MessageMeta의 기능을 확장하여 4가지 주요 정보를 담습니다.

1. **Payload (페이로드):** 실제 데이터(MessageMeta 또는 MultiMessage 인스턴스, cuDF DataFrame 포함).
2. **Metadata (메타데이터):** 메시지 전체에 적용되는 임의의 키-값 쌍 (예: priority, source).
3. **Tasks (태스크):** 메시지 페이로드에 적용할 작업 목록 및 상세 정보 (SIMD 처리에 활용).
4. **Tensors (텐서):** 다차원 배열 데이터 (주로 ML 추론에 사용).

## 📌 주요 코드 로직 (Metadata 및 Task 활용)

### 1. ControlMessage 구성 및 페이로드 설정

import cudf  
from morpheus.messages import ControlMessage, MessageMeta  
  
# 1. 데이터 로드 (cuDF DataFrame)  
df = cudf.read\_json('data/simple\_user\_log.jsonlines', lines=True)  
mm = MessageMeta(df) # DataFrame을 MessageMeta로 감싸기  
  
# 2. ControlMessage 생성 및 MessageMeta를 Payload로 설정  
cm = ControlMessage()  
cm.payload(mm)

### 2. Metadata 및 Task API 활용

# 3. Metadata 설정 및 조회 (메시지 전체의 속성)  
cm.set\_metadata("tracing", True)  
cm.set\_metadata("priority", "high")  
  
# 조회  
print(cm.has\_metadata("tracing")) # True  
print(cm.get\_metadata()['priority']) # 'high'  
  
# 4. Task 추가 및 활용 (메시지 페이로드에 적용할 작업 정의)  
process\_task\_details = {"column": "status", "sleep\_secs": 2}  
cm.add\_task("process\_data", process\_task\_details)  
  
# 조회  
print(cm.has\_task("process\_data")) # True  
print(cm.get\_tasks()) # {'process\_data': [{'column': 'status', 'sleep\_secs': 2}]}  
  
# 5. Task 실행 및 제거 (SIMD 처리 논리)  
if cm.has\_task("process\_data"):  
 task = cm.remove\_task("process\_data")[0] # 태스크 정보 추출 및 목록에서 제거  
  
 with cm.payload().mutable\_dataframe() as df\_mutable:  
 # 실제 데이터 처리 로직 실행  
 df\_mutable[task["column"]] = df\_mutable[task["column"]].str.upper()  
  
# 결과 확인: 'status' 컬럼의 모든 값이 대문자로 변환됨  
# print(cm.payload().get\_data()['status'])