# 4. Morpheus Advanced Pipelines: 비선형, 라우팅 및 대규모 모델 평가

## 프로젝트 개요

Morpheus의 고급 기능을 종합적으로 활용하여 **비선형 데이터 흐름**, **조건부 라우팅**, **커스텀 스테이지 그래프 구축**, 그리고 **대규모 언어 모델(LLM) 평가** 솔루션까지 구현했습니다. 이는 복잡한 엔터프라이즈급 AI 파이프라인을 설계하고 운영하는 능력을 입증합니다.

## ⚙️ 사용 기술 및 역할

* **주요 기술:** Morpheus Pipeline (비선형), RouterStage, MRC Builder, DeepSpeed (Assessment 추론), SinglePortStage.
* **핵심 역할:**
  1. **비선형 파이프라인 구축:** Non-linear-passthrough.ipynb에서 Pipeline 클래스를 사용하여 병렬 처리 및 분기/통합이 가능한 복잡한 데이터 그래프를 설계했습니다.
  2. **동적 라우팅 제어:** Router.ipynb를 통해 메시지 페이로드(데이터)의 조건에 따라 메시지를 서로 다른 다운스트림 포트로 분기하는 **RouterStage** 논리를 구현했습니다.
  3. **커스텀 스테이지 내 그래프:** Stage-graphs.ipynb를 통해 단일 커스텀 스테이지 내부에 MRC 노드들을 연결하여 복잡한 로직을 모듈화하는 방법을 습득했습니다.
  4. **다단계 PCAP 분석:** 복잡한 PCAP 데이터 분석을 여러 커스텀 스테이지로 분리하여 선형 파이프라인으로 구성했습니다. (Exercise-multi-stage-pcap.ipynb)

## 🛠️ 주요 구현 내용 (Assessment\_sol.ipynb 기반)

1. **대규모 모델 평가 솔루션:** Assessment\_sol.ipynb에서는 **minGPT**와 **Vision Transformers** 같은 대규모 모델을 **DeepSpeed** 프레임워크로 포팅하고 최적화하여, **활성화 체크포인팅(Activation Checkpointing)** 및 **ZeRo 리던던시 옵티마이저**를 포함한 복잡한 딥러닝 트레이닝 파이프라인을 구축했습니다.
2. **파이프라인 비선형 분기:** RouterStage의 key\_fn을 사용하여, 메시지를 '성공' 처리 브랜치와 '실패' 처리 브랜치로 나누는 동적 라우팅을 구현하여 파이프라인의 유연성을 극대화했습니다.
3. **유닛 테스트:** 모든 커스텀 스테이지는 독립적으로 메시지를 처리할 수 있도록 설계되었으며, 이는 복잡한 파이프라인의 디버깅 및 유지보수를 용이하게 합니다.

## 📌 주요 코드 로직 (비선형 라우팅 추상화)

from morpheus.stages.general.router\_stage import RouterStage  
from morpheus.messages import MessageMeta  
  
# 라우팅 키 함수: 메시지 상태를 기반으로 분기 결정  
def router\_key\_fn(message: MessageMeta) -> str:  
 # 이 로직은 캡스톤 프로젝트에서 NVSMI와 PCAP을 분기하는 데 사용됨  
 # message.get\_metadata('route') 등을 사용하여 태그된 값으로 분기합니다.  
 if message.get\_metadata('route') == 'nvsmi\_data':  
 return "nvsmi\_route"  
 else:  
 return "pcap\_route"  
  
# 비선형 파이프라인 구축  
pipeline = Pipeline(config)  
router = pipeline.add\_stage(RouterStage(config, keys=["nvsmi\_route", "pcap\_route"], key\_fn=router\_key\_fn))  
  
# Router의 출력 포트 연결  
pipeline.add\_edge(router.output\_ports[0], nvsmi\_preprocessing\_stage)  
pipeline.add\_edge(router.output\_ports[1], pcap\_preprocessing\_stage)