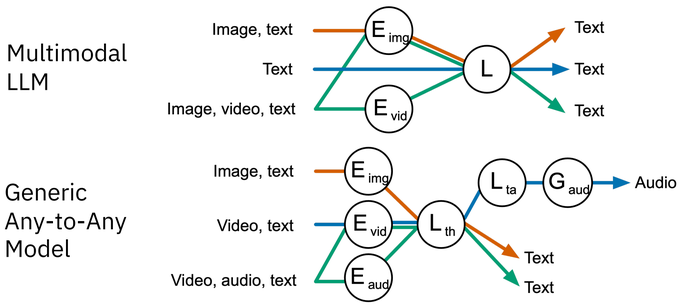
Cornserve: A Distributed Serving System for *Any-to-Any* Multimodal Models

Jae-Won Chung[*,1] **Jeff J. Ma**[*,1] Jisang Ahn[1] Yizhuo Liang[2] Akshay Jajoo[3] Myungjin Lee[3] Mosharaf Chowdhury[1]

[*] *Equal contribution* [1] University of Michigan [2] University of Southern California [3] Cisco Research

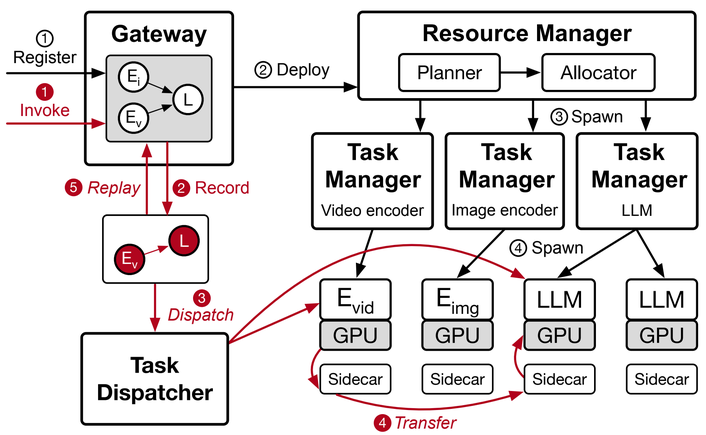
https://cornserve.ai

Any-to-Any Multimodal Models



Model	Input Modality	Output Modality
Qwen 2.5 / 3 / 3.5 Omni	T I V A	T A
Qwen 2.5 / 3 VL, Qwen 3.5, InternVL 3	T I V	T
DeepSeek Janus	T I	T I
LTX-2	T I	V A
Qwen-Image, GLM Image	T	I

System Architecture



Deployment flow

Gateway analyzes app
→ Resource Manager
→ Task Managers & Executors

Request flow

Gateway task record pass
→ Task Dispatcher
→ Task Executors & Sidecars
→ Gateway task replay pass

Tasks and Apps

- **Unit Task**: An atomic unit of computation handled by a single Task Manager
- **Composite Task**: Multiple Unit Tasks put together, record & replay happens
- **App**: Multiple Composite Tasks put together with arbitrary async Python logic

Multimodal LLM Task = Encoder Unit Task(s) + LLM Unit Task

```python
from cornserve.task import Task, Stream
from cornserve_tasklib import EncoderTask, LLMUnitTask, Modality

class MLLMTask(Task[ChatRequest, Stream[ChatCompletionChunk]]):
    model_id: str
    modalities: list[Modality] = []
    encoder_fission: bool = True

    def post_init(self) -> None:
        if self.encoder_fission:
            self.encs = {m: EncoderTask({self.model_id}, modality=m)
                         for m in self.modalities}
        self.llm = LLMUnitTask(self.model_id, receive_embeddings=self.encoder_fission)

    def invoke(self, req: ChatRequest) -> Stream[ChatCompletionChunk]:
        if self.encoder_fission:
            encoder_embeddings = [
                self.encs[mm.modality].invoke(EncoderInput(req.model, [mm.url])).embeddings[0]
                for mm in extract_multimodal_items(req)
            ]
            req.cornserve_embeddings = encoder_embeddings
        return self.llm.invoke(req)
```

Disaggregated encoders instantiated

Symbolic `DataForward` descriptor

Omni Task = MLLM Task + Talker Unit Task + Vocoder Unit Task

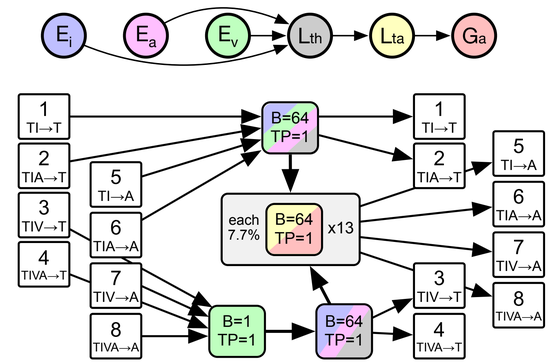
```python
from cornserve.task import Task, Stream
from cornserve_tasklib import (MLLMTask, MLLMEmbeddingTask,
                               OmniTalkerEmbeddingTask, AudioGeneratorTask)

class OmniTask(Task[OmniInput, Stream[ChatCompletionChunk]]):
    model_id: str
    modalities: list[Modality] = []
    encoder_fission: bool = True

    def post_init(self) -> None:
        mid, mods, ef = self.model_id, self.modalities, self.encoder_fission
        self.thinker_text = MLLMTask(mid, mods, ef)
        self.thinker_emb  = MLLMEmbeddingTask(mid, mods, ef)
        self.talker_emb   = OmniTalkerEmbeddingTask(mid)
        self.vocoder      = AudioGeneratorTask(mid)

    def invoke(self, req: OmniInput) -> Stream[ChatCompletionChunk]:
        if not req.return_audio:
            return self.thinker_text.invoke(req)

        emb = self.thinker_emb.invoke(req)

        talker_in = OmniTalkerInput(**req.dict(), thinker_hidden=emb.embeddings)
        emb_a = self.talker_emb.invoke(talker_in).embeddings

        return self.vocoder.invoke(AudioGeneratorInput(emb_a))
```

If no audio output, just invoke multimodal LLM

Cornfigurator: Automated Deployment Planner

- Which components should be **disaggregated**?
- How many **replicas** does each component need?
- How should requests be **routed** across replicas?

Deployment plan for Qwen 3 Omni on 16xA100 GPUs



The planner specializes a path with a disaggregated video encoder for heavy video-input requests and shares the talker + vocoder across all paths.

Toward a Unified Corn Stack

*Cornserve and Cornfigurator sit at the **narrow waist** of the stack, providing **generic** planning and orchestration compatible with many first-class application libraries and runtimes/executors.*