📘 **Canvas: Deep Dive – Kỹ Thuật Bất Đồng Bộ (Asynchronous I/O) trong GenAI Factory**

🎯 Mục tiêu của Async I/O

Async I/O là nền tảng giúp GenAI Factory đạt **hiệu suất cao (performance)** và **khả năng mở rộng (scalability)**.
Mục tiêu chính là **ngăn chặn I/O Blocking** – tình trạng ứng dụng bị dừng khi chờ các thao tác chậm (như gọi API qua mạng hoặc truy vấn DB).

I. ⚙️ Cơ chế Hoạt động trong LLM Wrapper (openai_llm.py)

Trong `shared_libs/atomic/llms/openai_llm.py`, Factory sử dụng client bất đồng bộ của OpenAI để định nghĩa các phương thức: `async_generate`, `async_chat`, `async_embed`.

Cấu phần	Vai trò Kỹ thuật	Cơ chế Bất đồng bộ
`self.client`	Khởi tạo client OpenAI.	Sử dụng `openai.AsyncOpenAI()` thay vì client đồng bộ. Dựa trên thư viện HTTP bất đồng bộ (`httpx`).
`await self.client.chat.completions.create(...)`	Gọi API LLM.	Khi gặp `await`, CPU trả lại quyền điều khiển cho **Event Loop**, cho phép xử lý yêu cầu khác trong khi chờ phản hồi.
Event Loop	Bộ điều phối trung tâm.	Khi API hoàn thành, Event Loop đánh thức hàm `async_generate()` để tiếp tục chạy từ điểm `await`.
Phản hồi	Xử lý kết quả trả về.	Dữ liệu được xử lý ngay khi có phản hồi mà không gây nghẽn toàn bộ ứng dụng.

💡 **Giá trị Hardening:** Async Client giúp tránh việc nghẽn toàn bộ server FastAPI khi gặp độ trễ mạng – nguyên nhân phổ biến làm chậm hệ thống GenAI.

II. 🧠 Cơ chế Hoạt động trong Agent (react_agent.py)

Agent là lớp điều phối trung tâm. Nó thực hiện nhiều vòng lặp Thought → Action → Observation và gọi các Tool hoặc LLM.
Nếu Agent hoặc Tool dùng hàm đồng bộ, toàn bộ hệ thống sẽ bị chặn.

Cấu phần	Vai trò Kỹ thuật	Cơ chế Bất đồng bộ

Cấu phần	Vai trò Kỹ thuật	Cơ chế Bất đồng bộ
`await self.llm.async_generate(...)`	Thực hiện bước **Thought**.	Agent gọi phương thức Async của LLM Wrapper. Trong khi LLM phản hồi, Agent nhường CPU cho Event Loop để xử lý tác vụ khác.
`await tool.async_run(input_dict)`	Thực hiện bước **Action**.	Tool (ví dụ: SQLTool) thực thi trong ThreadPool hoặc async DB client, đảm bảo không chặn Event Loop.
`asyncio.wait_for(self._execute_stepped_loop, timeout=timeout)`	Kiểm soát Tài nguyên.	Hardening quan trọng nhất: tự động hủy tác vụ khi vượt quá `timeout`, tránh loop vô hạn và kiểm soát chi phí inference.

💡 **Giá trị:** Bảo đảm Agent có thể chạy **song song** nhiều yêu cầu trong khi vẫn duy trì **SLA và độ ổn định** của toàn hệ thống.

III. ⚙️ Phương pháp xử lý I/O Chặn (Blocking I/O)

Một số Tool hoặc Evaluator (như thư viện CPU-bound, DB cũ) không hỗ trợ bất đồng bộ. Để giải quyết, ta dùng kỹ thuật:

```
await loop.run_in_executor(self.executor, self.evaluate, ...)
```

Ý nghĩa	Giá trị kỹ thuật
Chuyển tác vụ chặn sang Thread riêng.	Giữ cho Event Loop chính (nơi FastAPI/Agent chạy) **luôn non-blocking**.
Tăng hiệu suất toàn hệ thống.	Cho phép CPU xử lý hàng trăm yêu cầu đồng thời trong khi các thread phụ xử lý các I/O chậm.

🚀 Kết luận: Giá trị của Async I/O trong GenAI Factory

✅ **Hiệu suất cao:** CPU không bị chặn khi chờ mạng hoặc DB.
✅ **Khả năng mở rộng:** Hàng trăm request có thể chạy song song mà không tắc nghẽn.
✅ **Độ tin cậy sản xuất:** Hardening thông qua `await`, `asyncio.wait_for`, và `run_in_executor` giúp hệ thống luôn sẵn sàng 24/7.

> 🔷 **Async I/O không chỉ là kỹ thuật lập trình – mà là nền tảng của khả năng mở rộng và tính ổn định của toàn bộ GenAI Factory.**