HelixLLM

HelixLLM

من حاسوبك Anthropicو OpenAI ثنائي واحد، ستة أوضاع — استدلال متوافق مع المحمول إلى مجموعة متعددة المضيفين.

الملخص

HelixLLM هو نظام LLM موزّع على مستوى المؤسسات مبنيّ على Go: ثنائي واحد بنظام أوضاع يتدرّج من التطوير أحادي المضيف إلى الإنتاج متعدد المضيفين. يقدّم واجهات برمجة تطبيقات متوافقة بالكامل مع OpenAI وAnthropic عبر HTTP/3، مع استدلال محلي باستخدام llama.cpp، وسلسلة احتياطية متعددة المزودين مُقيّمة، وخط أنابيب RAG، ونظام وكيل ReAct.

وصف مختصر

HelixLLM هو نظام LLM موزّع ثنائي مبنيّ على Go. يعرض واجهات برمجة تطبيقات متوافقة مع OpenAI وAnthropic عبر HTTP/3، ويشغّل استدلالاً محلياً باستخدام llama.cpp، ويكتشف تلقائياً ويقيّم مزودي الخدمات السحابية المجانية ضمن سلسلة احتياطية، ويضيف خط أنابيب RAG معرفية بالإضافة إلى وكيل ReAct القادر على استدعاء الأدوات — قابل للنشر في ستة أوضاع مختلفة.

وصف مفصّل

HelixLLM هو نظام LLM موزّع على مستوى المؤسسات، مبنيّ باستخدام Go وGin، وحيلته الأساسية تكمن في أن قطعة برمجية واحدة تخدم جميع المستويات. يُترجم إلى ثنائي واحد يحدد نظام الأوضاع عند النشر ماهية هذا الثنائي: شغّله كـ `full` للحصول على مثيل شامل على حاسوب محمول، موزّعة على `control` و`agents` و`knowledge` و`brain` و`gateway` أو وزّع المسؤوليات عبر أوضاع عدة مضيفين — نفس الكود يعاد ترتيبه دون إعادة كتابته، من جهاز المطور إلى مجموعة الإنتاج.

OpenAI يتقن النظام لغتين بطلاقتهما: واجهات برمجة تطبيقات متوافقة بالكامل مع الحالية من كلا النظامين دون تعديل، وكلها تُقدّم عبر SDK بحيث تعمل عملاء Anthropic، وعمل الاستدلال المحلي. TLS 1.3 ودعم HTTP/2 مع احتياطي تلقائي لـ HTTP/3 (QUIC) بحيث يسرّع نفس البناء على أجهزة ROCm، وMetal وCUDA مع دعم llama.cpp عبر Nvidia وApple وAMD على حد سواء. أما الميزة البارزة فهي سلسلة الاحتياطية متعددة المزودين، التي تحول عدم موثوقية الاستدلال السحابي المجاني إلى مورد مُدار وقادر على الشفاء الذاتي: يكتشف HelixLLM تلقائياً النماذج المجانية من أكثر من 7 مزودي خدمات سحابية (Chutes،

OpenRouter، HuggingFace، Nvidia، Cerebras، SambaNova، Together)، ويقيمها عبر LLMsVerifier كل 5 دقائق، ويوجّه الطلبات عبر السلسلة المرتبة مع المحلي كملاذ أخير، بحيث llama.cpp مع ضمان استخدام — xxاحتياطي تلقائي للأخطاء 429/5 لا يفشل الطلب أبداً لعدم توفر مزود.

منصة تطبيقات كاملة: يتضمن خط أنابيب معرفية HelixLLM بالإضافة إلى الاستدلال الخام، يُعد مع استدعاء الأدوات ReAct ونظام وكيل (vector استيعاب، تجزئة، تضمين، بحث) RAG، ضمن نفس الثنائي. يؤتي نظام الأوضاع ثماره أيضاً على مستوى RAG وجلسات المحادثة، وتكامل المباشرة داخل العملية دون Go تتواصل جميع الطبقات عبر استدعاءات full الاتصال — ففي وضع SSEو gRPC أي عبء شبكي، بينما ينسّق الثنائي المتطابق عند توزيعه على عدة مضيفين عبر الذي يتطابق مع تنسيقات SSE وبث Brotli/gzip، ويكتمل النظام بمفاوضات المحتوى Kafka.و مع تحديد المعدل، ومقاييس JWTو API-key بايتاً ببايت، ومصادقة Anthropicو OpenAI للبنية التحتية Go ومجموعة كبيرة من وحدات OpenTelemetry، وتتبع Prometheus، الإنتاجية.

المحتوى

لماذا قمنا ببنائه

تحتاج الفرق إلى استدلال قابل للنقل ومتوافق مع المعايير ومرن — دون الحاجة إلى إعادة كتابة بحيث يمكن لنفس الملف التنفيذي HelixLLM العملاء أو الارتهان بمزود واحد أو جهاز واحد. تم بناء أن يعمل محلياً للتطوير ويتوسع ليشمل مجموعة إنتاجية متعددة المضيفين، ويتحدث لهجات OpenAI وAnthropic التي يستخدمها العملاء بالفعل.

لماذا يُعد ثورة في المجال

إنه يدمج مجموعة كاملة من بنية الاستدلال — البوابة، والاستدلال المحلي، والعودة إلى السحابة، والوكلاء — في ملف تنفيذي واحد يتحكم فيه مفتاح وضع التشغيل، بحيث يصبح قرار نشر RAG، وو البنية المعمارية قراراً يتم اتخاذه في وقت التشغيل بدلاً من أن يكون مشروع إعادة منصة. كما يحول ما كان يُعد نقطة ضعف إلى ميزة: تصبح موثوقية مزودي السحابة أولوية رئيسية تُقاس باستمرار، وتُدار من خلال سلسلة عودة ذاتية الإصلاح تُقيّم المزودين كل بضع دقائق وتتحول دائماً إلى الاستدلال المحلي المضمون. الإمكانية التي يتيحها هي نقطة نهاية واحدة يمكنك الاعتماد عليها حقاً — متوافقة مع المعايير، قابلة للنقل من الحاسوب المحمول إلى المجموعة، وغير قادرة على التوقف عن العمل لأن أحد المزودين الرئيسيين قد فرض قيوداً على المعدل أو فشل.

ما الجديد في هذا الحل

- ملف تنفيذي واحد بنظام مكون من ستة أوضاع يعمل إما بشكل متكامل أو كأدوار موزعة — عند الفصل gRPC/SSE/Kafka، وfull المباشرة داخل العملية في وضع Go مكالمات بحيث تتغير طوبولوجيا النشر دون تغيير في الكود أو فرض ضرائب شبكية غير مطلوبة.
- سلسلة عودة متعددة المزودين مكتشفة تلقائياً ومقيّمة عبر أكثر من 7 مزودين مجانيين، تُصنّف باستمرار بواسطة LLMsVerifier 429/5 مع تجاوز تلقائي لأخطاء xx وضمان استخدام llama.cpp كملاذ أخير — تحويل قدرة الطبقة المجانية إلى قدرة موثوقة.
- مع عودة تلقائية HTTP/3 تُقدّم عبر Anthropic وOpenAI واجهات متوافقة مع كل من HTTP/2، إلى بحيث يتصل العملاء من أي من النظامين البيئيين دون تعديل.

- من قاعدة كود واحدة — نفس الإصدار ROCmو Metalو CUDA استدلال محلي يشمل AMDو Appleو Nvidia يعمل بتسريع على أجهزة.

أكبر التحديات التقنية وكيف تغلبنا عليها

- تفرض معظم الأنظمة حداً صارماً بين **التوسع من مضيف واحد إلى عدة مضيفين دون إعادة كتابة.** التطوير المحلي" و"الإنتاج الموزع"، وعبر هذا الحد يتطلب إعادة هيكلة. لقد ألغينا هذا الحد بنظام" أوضاع على ملف تنفيذي واحد: تتواصل نفس الطبقات عبر مكالمات مباشرة داخل العملية في عبر الأوضاع الموزعة، بحيث يصبح gRPC/SSE/Kafka وتتحول بسلاسة إلى `full` وضع التوسع مجرد تغيير في التكوين بدلاً من نقل.
- الاستدلال في الطبقة المجانية **مزودو السحابة المجانيون غير الموثوقين والمحدودي المعدل.** سريع حتى يفرض قيوداً على المعدل أو يختفي أثناء الطلب. جعلناه موثوقاً باكتشاف النماذج وتتبع رؤوس قيود المعدل بشكل استباقي، LLMsVerifier المتاحة تلقائياً، وتقييمها باستخدام لتوجيه الطلبات بعيداً عن المزودين على وشك التقييد، والعودة تلقائياً عبر السلسلة المصنفة إلى المحلي — بحيث لا تصل تقلبات المجموعة أبداً إلى المتصل llama.cpp.
- إعادة كتابة العملاء لاعتماد خلفية استدلال جديدة أمر غير **توافق العملاء عبر نظامين بيئيين.** وصولاً إلى — Anthropicو OpenAI لكل من API مقبول. لقد قمنا بتنفيذ أشكال المختلفة الخاصة بكل منهما — بحيث تشير مجموعات تطوير البرمجيات SSE تنسيقات بث وتعمل ببساطة HelixLLM من أي من المعسكرين إلى.

المحتوى

مجموعة الأدوات التقنية

- **Go + Gin** — اختيرت لأن بيئة التشغيل ذات الملف الثنائي الواحد والموجهة نحو التوازي هي ما يجعل نظام الأوضاع بأكمله ممكنًا: بناء واحد يمكن أن يكون خادمًا محليًا أو دورًا ضمن مجموعة خوادم. تحمل النظام بأكمله بالإضافة إلى طبقة HTTP الخاصة بالبوابة.
- **HTTP/3 (QUIC) + TLS 1.3، مع الرجوع التلقائي إلى HTTP/2** — اختيرت لنقل البيانات الحديث منخفض الكمون والمرن في الاتصالات، وتُعرض كواجهة الخادم مع التفاوض بهدوء إلى QUIC التلقائي بحيث تعود العملاء غير القادرين على استخدام HTTP/2.
- **llama.cpp (CUDA/Metal/ROCm)** — اختيرت للاستدلال المحلي المحمول من قاعدة برمجية واحدة؛ كما تعمل AMDو Appleو Nvidia الذي يتسارع عبر منصات كخيار احتياطي مضمون يمنع سلسلة التراجع من الوصول إلى طريق مسدود.
- **LLMsVerifier** — اختيرت لتحويل "أي مزود جيد الآن" إلى رقم؛ حيث تقوم بتقييم وترتيب سلسلة التراجع السحابية كل خمس دقائق بحيث تتبع التوجيه الجودة الحية بدلاً من الافتراضات القديمة.
- **موفرو الخدمات السحابية (Chutes، OpenRouter، HuggingFace، Nvidia، Cerebras، SambaNova، Together)** — اختيروا لاستغلال السعة المجانية عبر عدة مصادر؛ حيث يتم اكتشافهم وترتيبهم تلقائيًا في سلسلة تراجع واحدة بحيث لا يشكل أي مزود نقطة فشل واحدة.
- **gRPC + SSE + Kafka** — اختيروا كوسائط نقل بين الأوضاع للنشر الموزع: gRPC للمكالمات بين الخدمات، و SSEللبث المباشر، و Kafkaللتدفق الأحداث المفككة بين الأدوار.
- **مخزن المتجهات / embeddings** — اختير لتشغيل خط المعرفة RAG من البداية إلى النهاية: استيعاب، تقطيع، تضمين، والبحث في المستندات التي تدعم إجابات النموذج.
- **Prometheus + OpenTelemetry** — اختيرا لقياس الأداء وتتبع الطلبات الموزعة عبر أي أوضاع يتم نشرها.

- اختيرت لإعادة استخدام المكونات الأساسية — **vasic-digital الفرعية من Go وحدات** للبنية التحتية الإنتاجية المجربة بدلاً من إعادة بنائها، مما يحافظ على اتساق أساس النظام مع المجموعة التقنية الأوسع.

ملاحظات الحالة والتراهة

- **الحالة: تجريبية.** نظام استدلال موزع وظيفي قيد التطوير النشط.
- **الرخصة: لم تُحدد بعد.** المستودع لا يعلن عن رخصة في بيانات التعريف (`licenseInfo` فارغ) — هذا الأمر غير مُتحقق منه ويجب حله قبل تحديد الرخصة.
- المستودع المرجعي حاليًا يشير إلى `github.com/HelixDevelopment/llm`؛ بينما يعيد التوجيه إليه. أرقام عتبات التغطية وعدد الوحدات الفرعية في ملف `HelixLLM` مسار README هي بيانات ذاتية التقرير.

درجة الأولوية: Helix-أساسية.