К использованию LLM в системах жёсткого реального времени

Старолетов Сергей Михайлович, кандидат физико-математических наук, доцент кафедры прикладной математики АлтГТУ, электронная почта: serg_soft@mail.ru

Широкое внедрение искусственного интеллекта для оценки текущего состояния переменных системы в промышленные робототехнические устройства и авионику сдерживается отсутствием формальных гарантий времени выполнения алгоритмов, ведь нормативные документы для доверенного программного обеспечения устройств (DO-178C, ISO 26262) требуют доказанного ограничения worst-case execution time. Предлагаемая работа – шаг к тому, чтобы показать, что современные облегченные языковые модели могут быть адаптированы к парадигме жёсткого реального времени.

В серии работ автор исследует свободную микроядерную ОС POK [1], реализующую принцип разделов по стандарту ARINC 653, как платформу для экспериментов, поскольку она обеспечивает концепт строгой временной и пространственной изоляции компонентов – необходимое условие для будущей сертификации. В качестве LLM модели взята квантованная версия TinyStories (15 миллионов параметров), а в качестве кодовой базы LLM – код проекта llama2.c [3], способный работать на слабом железе IoT вроде одноплатных компьютеров.

Цель работы – создать демо, моделирующего работу «умного советчика» по параметрам окружающей среды. Каждые заданное количество миллисекунд система должна:

- Считать показания четырёх датчиков или их эмуляторов (температура, влажность, давление, скорость ветра).
- Преобразовать их в лингвистические оценки (вроде «жарко», «ветрено», «сухо»).
- Сформулировать запрос на естественном языке и передать его LLM внутри системы.
- Сгенерировать текстовую рекомендацию не позднее установленного дедлайна, даже если модель не успела выдать самый лучший ответ.

Модель TinyStories (dim=288, hidden_dim=768, n_layers=6, n_heads=6, vocab_size=32000), была квантована до int8 (для уменьшения памяти и поддержки вычислений в целых числах), преобразована программой в байты данных внутри .h файла и встроенна получившимся статическим массивом в код elf-файла раздела (размер ~15 МБ). Прямой проход инференции состоит из последовательности трансформерных блоков. Время обработки одного слоя постоянно. Это даёт естественный квант для контроля времени – после завершения каждого слоя вызывается pok_time_get(), и, если текущее время превысило дедлайн, генерация завершается. Такой подход кооперативной проверки на границах слоёв гарантирует, что поток добровольно уступит управление не позже чем через время обработки одного слоя LLM.

Для сборки и успешной работы С-кода проекта llama2.c в среде POK необходимо было сделать несколько решений портирования. Оригинальный код после загрузки квантованных весов сразу де-квантизует всю таблицу эмбеддингов токенов, требуя для TinyStories около 36 дополнительных МБ (32000 * 288 * 4 байта). Под управлением POK такой единовременный запрос к менеджеру кучи приводил к отказу. Поэтому решено было полностью отказаться от хранения де-квантизованной таблицы и вычислять эмбеддинг каждого токена «на лету» при каждом прямом проходе. В целом, можно вообще отказаться от динамического выделения памяти (это в коде делается в основном до начала работы при инициализации LLM), заменив

все это на статические массивы и арифметику указателей. Далее, хотя и в POK есть libm для работы с математикой и часть libc, но было недоступны некоторые функции вроде qsort, bsearch, sprintf и так далее. Они были добавлены или код был переделан.

В итоге, в решении свободно доступном на GitHub [4], внутри одного раздела POK организованы три потока, синхронизированные с помощью семафоров: (1) sensor_job – периодически опрашивает (для демо эмулирует с помощью случайной генерации и учета предыдущих значений по методы скользящего среднего) датчики температуры, скорости ветра, давления и влажности; (2) estimator_job – преобразует числовые показатели в строковые категории; (3) llm_job – формирует промпт, запускает инференцию (вывод) и выводит ответ. Результат можно посмотреть на Рис. 1.

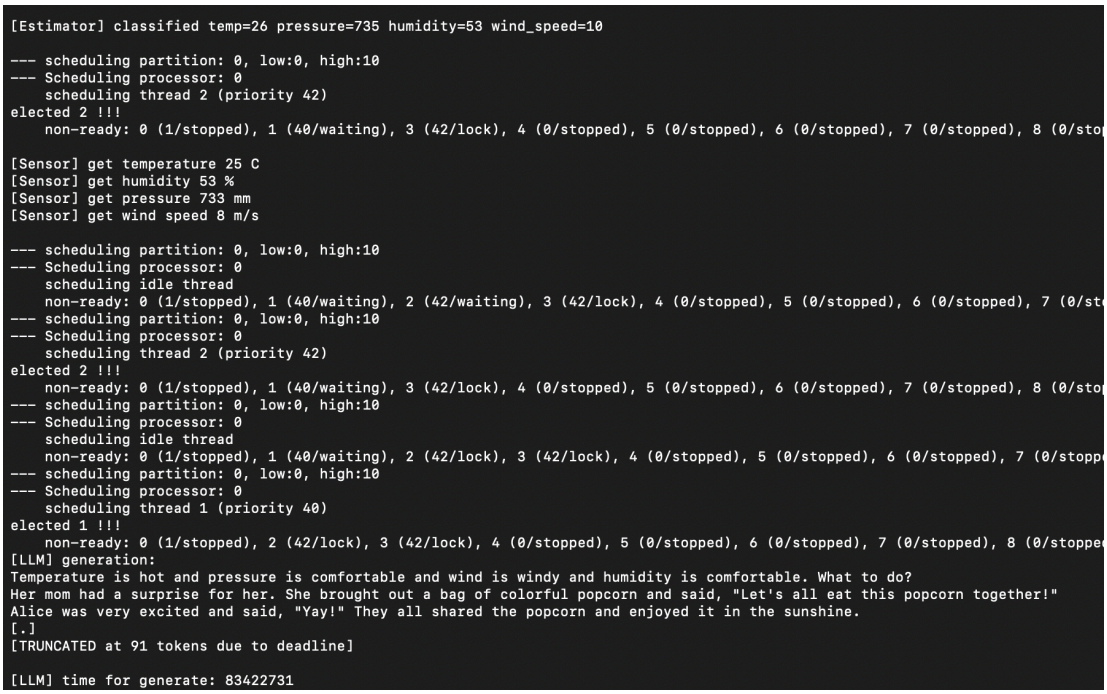

Рис.1 – Работа POK LLM демо [4] (модель Stories 15M_Q8) на qemu-system-i386

В качестве продолжения возможна реализация полнофункционального демо с двумя разделами (партициями) – одним для LLM, а другим для датчиков и запуск его на устройстве. Модель уже была проверена на микрокомпьютере семейства Raspberry Pi и показала достаточно слабую скорость генерации – 2.18 токена/секунду, но это и нормально для чистого C-кода без оптимизаций. Но и получается, что LLM модель должна работать в секундных дедлайнах, выполняя долгосрочные прогнозы.

Также, простое завершение генерации по времени может оставить ответ синтаксически незавершённым. Можно реализовать более элегантный двухступенчатый механизм. За заданное в процентах до истечения дедлайна время может активироваться мягкий дедлайн. Модель при этом продолжает генерировать токены, но ей будет разрешено выдать не более заданного небольшого количества дополнительных токенов. Если в течение этого окна встречается знак окончания предложения – генерация немедленно останавливается и ответ получается грамматически законченным, иначе продолжается до истечения дедлайна и ответ получается как есть.

В итоге, можно констатировать, что задача может быть решена со свойственными таким моделям ограничениями и нужно искать случаи ее полезного использования.

Литература

1. POK kernel. Safe and secure for safety-critical systems. URL: https://github.com/pok-kernel/pok
2. Tiny Stories. URL: https://huggingface.co/datasets/roneneldan/TinyStories , https://arxiv.org/abs/2305.07759
3. llama2.c. URL: https://github.com/karpathy/llama2.c
4. LLM-POK Демо. URL: https://github.com/SergeyStaroletov/LLM-POK