УДК 004.8 DOI 10.18372/2073-4751.87.21679

Бордіян А.І.,
orcid.org/0009-0003-8305-3051,
e-mail: bordyanartem@gmail.com,

ТРАЄКТОРНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ РЕСУРСІВ ДЛЯ AI-АГЕНТІВ У СЕРВІСАХ ОНЛАЙН-НАВЧАННЯ

Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

Вступ

Сервіси онлайн-навчання дедалі частіше інтегрують AI-агентів, які поєднують міркування, вибір дій, використання інструментів та пам'яті для реалізації тьюторингових сценаріїв. Для таких сервісів визначальними є інтерактивна чутливість до затримок та стабільність якості обслуговування, що зазвичай формалізується через квантілі затримки кроків (p95/p99), обмеження частоти помилок і стійкість під піковим навантаженням. Паралельно оператори платформи мають мінімізувати споживання дорогих ресурсів, насамперед GPU-обчислень і GPU-пам'яті (KV-кеш), а також контролювати CPU, мережеві та сховищні витрати, зумовлені оркестрацією, пошуком у пам'яті та інструментальними викликами.

Структурною властивістю сучасних агентних систем є композиційність, а саме обслуговування висновків винесено в окремий шар, оркестрація та політики реалізуються незалежно, пам'ять агентів зовнішня, а конвеєри спостережуваності інтегровані як компоненти системи. Це створює передумови для незалежного масштабування CPU- та GPU-орієнтованих підсистем, однак водночас породжує невідповідність між одиницею керування та одиницею споживання ресурсу. Більшість підходів керування ресурсами оперує запитом або реплікою сервісу як одиницею керування, тоді як фактичне споживання у тьюторингових агентах формується траєкторією: багатокроковим внутрішнім навантаженням, довжина й структура якого варіюють між сесіями та еволюціонують у межах однієї сесії.

У статті запропоновано траєкторно-орієнтовану модель ресурсів, яка узгоджує одиницю обліку з одиницею споживання. Модель базується на системному аналізі щодо еволюції архітектур агентів, моделювання ресурсів і навантаження у високонавантажених сервісах із токенно-структурованими фазами інференсу, та фрагментарності методів оптимізації, які не складаються в узгоджену політику для agent-based edtech. Стаття вводить агентну траєкторію як первинну одиницю обліку ресурсів і на її основі будує декомпозицію витрат GPU/CPU/пам'яті/мережі з урахуванням двофазної природи LLM-інференсу. Окремо формалізовано обмеження QoS/SLA на рівні кроку і сесії та запропоновано мінімальну схему телеметрії, яка перетворює спостережуваність із постфактум-діагностики на складову контуру керування.

Архітектурні обмеження

У сервісах онлайн-навчання агентна система охоплює щонайменше чотири функціональні контури з різним ресурсним профілем: інференс (GPU-інтенсивний), оркестрація (CPU-інтенсивна), пам'ять/дані (I/O та сховища) і спостережуваність (накладні витрати телеметрії). Така композиція дозволяє масштабувати GPU- та CPU-компоненти незалежно, але водночас означає, що жоден компонент не знає повної вартості запиту – вона розподілена між контурами, а отже й шлях запиту стає залежним від інструментальних I/O, доступів до пам'яті та накладних витрат

телеметрії понад те, що видно на рівні окремого контуру.

Невідповідність одиниць керування

У класичних високонавантажених сервісах інтенсивність запитів і агреговані метрики утилізації часто достатні для замикання контурів керування. Для LLM-обслуговування системні роботи підкреслюють щонайменше двофазний характер інференсу (prefill та decode), а також те, що планування фаз і пакетування визначають компроміс між пропускною здатністю та затримкою та поведінку хвоста розподілу. У агентних сервісах онлайн-навчання один зовнішній запит може породжувати кілька викликів інференсу, інструментальні виклики та операції пам'яті, які чергуються в часі. Відповідно, природною одиницею обліку стає не запит, а траєкторія, яка фіксує фан-аут, синхронні залежності та зростання контексту.

Неоднорідність навантаження в тьюторингових агентах структурується за трьома осями. Перша – довжина і склад контексту: довгий діалог із доданими навчальними матеріалами істотно збільшує prefill-час порівняно з коротким запитом, оскільки фаза prefill лінійно залежить від довжини вхідного контексту [14]. Друга – траєкторія виконання: сесія з перевіркою коду може породжувати кілька інструментальних викликів у межах одного педагогічного кроку, тоді як пояснювальна сесія може взагалі не звертатися до зовнішніх інструментів. Третя – токенний профіль відповіді: генерація покрокового розв'язку задачі дає суттєво більший L_out, ніж підтверджувальна репліка. Ігнорування цих осей веде до типових відмов. а саме реактивне автомасштабування бачить лише запізнілі агрегати, справедливість на основі лічильника запитів некоректна при токенно-орієнтованій вартості та оптимізація середнього випадку не стабілізує хвіст затримки при довгих контекстах та інструментальних паузах.

Вимоги QoS/SLA в онлайн-навчанні

Онлайн-навчання є інтерактивним сервісом, отже QoS та SLA доцільно задавати на рівні кроку (однієї відповіді агента) та на рівні сесії. Нехай L – це затримка педагогічного кроку, а E – подія помилки (тайм-аут, відмова інструмента, внутрішня помилка або відхилення допуском). Репрезентативний SLA задається квантільними обмеженнями $p95(L) \le Lmax$, $p99(L) \le L99max$ та обмеженням частоти помилок $Pr(E) \le$. У агентних системах ці обмеження мають виконуватися не лише для зовнішніх запитів, а для траєкторних подій, оскільки користувацьке сприйняття визначається часом до наступної педагогічної репліки й стабільністю протягом довгої сесії.

Пов'язані дослідження

Дослідження архітектур LLM-агентів формалізують патерни «міркування + дія» з інструментами та підкреслюють роль пам'яті у довгих взаємодіях, що мотивує сесійне моделювання та профілювання [1–4, 8, 12, 13]. Паралельно системи inference serving демонструють, що ефективність GPU визначається керуванням KV-кешем, пакетуванням і фазово-обізнаним плануванням, а отже практична межа «пропускна здатність – затримка» задається саме системними механізмами [5]. Зокрема, архітектурне розірвання фаз prefill і decode [14] та узгоджене планування батчів і підкроків при змішаному навантаженні [15] безпосередньо підтверджують двофазну природу інференсу, формалізовану в (9).

Розподілені системи обслуговування показують кластерні обмеження та взаємодію розміщення з продуктивністю [6]. Роботи зі справедливості у мультиорендному LLM-serving підкреслюють, що квоти за кількістю запитів є некоректними, коли домінує токенна вартість, і що потрібен токенно-обізнаний облік [16, 17]. Зокрема, підхід Virtual Token Counter [16] вводить функцію вартості обслуговування на основі кількості вхідних та вихідних токенів, а аналіз великомасштабної мультиорендної платформи [17] емпірично підтверджує, що варіації довжини токенів між застосунками унеможливлюють справедливість на основі лічильника запитів.

Огляди автомасштабування систематизують реактивні та прогностичні контури для cloud-native навантажень, однак типово припускають, що навантаження адекватно описується агрегованими метриками, а відображення залежності запит – споживання ресурсу є відносно стабільним [7, 9, 10]. Ці припущення порушуються для агентних траєкторій з інструментами, пам'яттю та довгими контекстами, що потребує семантично

значущих дескрипторів навантаження. Стандарти спостережуваності консолідують практики збирання трас, метрик і журналів та розвивають семантичні конвенції. Для GenAI та агентів формуються напрями, які підтримують тезу, що телеметрія є передумовою керування, а не лише діагностики [11, 19].

Запропонована робота відрізняється тим, що вводить траєкторію як первинний об'єкт ресурсного обліку саме для сервісів онлайн-навчання, де багатокрокові сесії, тьюторингові SLA та мультиорендність одночасно визначають і споживання ресурсів, і цілі керування.

Траєкторно-орієнтована модель ресурсів

Розглянемо сервіс онлайн-навчання як розподілену систему з компонентами, до яких відносяться оркестрація, inference serving, пам'ять/дані, інструменти (зовнішні API та виконання коду) та pipeline спостережуваності. Нехай агентна траєкторія (trajectory) T є впорядкованою послідовністю кроків, що реалізують фрагмент навчальної взаємодії в межах сесії:

$$T = \langle s_1, s_2, \dots, s_n \rangle \quad (1),$$

де кожен крок s_i відповідає одній педагогічній відповіді та включає внутрішнє навантаження, до якого відноситься інференс, інструментальні виклики, операції пам'яті та емісію телеметрії.

Кожен крок s_i параметризуємо токенно-структурованими величинами інференсу та фан-аутом оркестрації:

$$s_i = (L_i^{in}, L_i^{out}, k_i^{tool}, k_i^{mem}, \Delta KV_i, \pi_i) \quad (2),$$

де L_i^{in}– довжина вхідного контексту в токенах, L_i^{out}– довжина відповіді, k_i^{tool}– кількість інструментальних викликів, k_i^{mem}– кількість операцій пам'яті (retrieve/store/index), ΔKV_i – приріст KV-кешу, а π_i – стан політики, що впливає на фан-аут і вибір компонентів.

Декомпозиція вартості

Визначимо вартість кроку як вектор споживання гетерогенних ресурсів:

$$C(s_i) = (C_i^{gpu}, C_i^{cpu}, C_i^{mem}, C_i^{net}, C_i^{obs}) \quad (3),$$

а вартість траєкторії – як суму по кроках:

$$C(T) = \sum_{i=1}^{n} C(s_i) \quad (4)$$

Формула (4) вірна для адитивних ресурсів, а саме GPU-секунд, CPU-секунд, токенів, мережевого трафіку та обсягу телеметрії. Для ємнісних ресурсів адитивна апроксимація не може бути використаною. Тому загальну ресурсну характеристику траєкторії подаємо як два компоненти:

$$C(T) = (C^{add}(T), C^{peak}(T)) \quad (5), \text{ де}$$

$$C^{add}(T) = \sum_{i=1}^{n} C^{add}(s_i) \quad (6)$$

є накопичувальними витратами, а

$$C^{peak}(T) = R(t) \quad (7)$$

визначає пікове використання ємнісного ресурсу, зокрема GPU-пам'яті або KV-cache. Детальне моделювання динаміки KV-cache з урахуванням eviction і TTL є предметом подальшої роботи; у поточній моделі добуток $\alpha_{kv}\Delta KV_i$ використовується як консервативна верхня оцінка тиску на пам'ять.

GPU-складова вартості визначається двофазною структурою інференсу. Нехай t_i^{pre} та t_i^{dec} – часи обслуговування фаз prefill та decode на кроці i. Тоді:

$$C_i^{gpu} = \alpha_{gpu}(t_i^{pre} + t_i^{dec}) \quad (8),$$

де α_{gpu} переводить GPU-час у нормовані одиниці вартості. На практиці α_{gpu} визначається апаратною конфігурацією і тарифом оренди (для хмарного розгортання це вартість GPU-години, поділена на максимальну пропускну здатність); для локального кластера замість вартості підставляється нормована частка ємності. Аналогічно α_{kv} відображає тиск на HBM-пам'ять прискорювача, а α_{cpu} – вартість ядро-секунди CPU. Конкретні значення залежать від платформи і мають бути виміряні або отримані з трас; у моделі вони є параметрами, а не константами.

Зовнішні операції пам'яті v_i^{mem} залишаються адитивною складовою; KV-cache характеризується піковим обмеженням $K_i \le K^{max}$, де K_i накопичується між кроками траєкторії.

Планування та пакетування трансформує залежність токенів від часу, тому t_i^{pre} і t_i^{dec} задаємо як функції токенної структури та параметрів політики обслуговування:

$$t_i^{pre} = f_{pre}\left(L_i^{in}, \Delta KV_i, \theta\right),$$
$$t_i^{dec} = f_{dec}(L_i^{out}, \Delta KV_i, \theta) \quad (9),$$

де θ – параметри політики (batching, фазове планування, пріоритети).

CPU-складова охоплює оркестрацію, підготовку контексту, ранжування й постобробку:

$$C_i^{cpu} = \alpha_{cpu}(t_i^{orch} + t_i^{prep} + t_i^{post}) \quad (10)$$

Складова пам'яті/сховищ включає тиск на KV-кеш прискорювача та зовнішні операції пам'яті:

$$C_i^{mem} = \alpha_{kv}\Delta KV_i + \alpha_{store}\, v_i^{mem} \quad (11),$$

де v_i^{mem} – абстрактна міра навантаження на сховище чи індекс, індукованого k_i^{mem} операціями. У персоналізованих довгих сесіях пам'ять стає активним компонентом критичного шляху, де системи кешування стану розмови [18] показують, що повторне обчислення KV-стану при зверненні до історії діалогу створює саме той тиск на ΔKV_i, який формалізовано у (11).

Мережеву складову в основному формують інструментальні виклики та віддалені доступи:

$$C_i^{net} = \alpha_{net}(v_i^{tool} + v_i^{remote}) \quad (12),$$

де v_i^{tool} агрегує трафік або взаємодію з інструментами, а v_i^{remote} – трафік, пов'язаний з віддаленою пам'яттю.

Накладні витрати спостережуваності включаємо явно:

$$C_i^{obs} = \alpha_{obs}\, v_i^{tel} \quad (13),$$

де v_i^{tel} – обсяг телеметрії та витрати на її експорт та обробку. Без агентно-семантичних сигналів оптимізація спирається на непрямі проксі-метрики та деградує до евристик без гарантій.

Формалізація обмежень SLA/QoS

Нехай L_i – випадкова величина end-to-end затримки кроку s_i, що вимірюється від надходження введення користувача до доставки наступної педагогічної репліки. Затримка індукується критичним шляхом через компоненти:

$$L_i = l(t_i^{pre}, t_i^{dec}, t_i^{orch}, t_i^{tool}, t_i^{mem}, t_i^{obs}) \quad (14),$$

де $l(\ldots)$ відображає синхронні залежності. Інструментальні виклики можуть породжувати довгі паузи, а доступи до пам'яті – суттєві мережеві накладні витрати, отже t_i^{tool} і t_i^{mem} не можна ігнорувати.

SLA на рівні кроку задаємо квантільними обмеженнями:

$$Q_{0.95}(L_i) \le L_{0.95}^{max},$$
$$Q_{0.99}(L_i) \le L_{0.99}^{max} \quad (15),$$

де $Q_p(\ldots)$ – p-квантіль. Обмеження за помилками:

$$Pr(E_i = 1) \le \varepsilon \quad (16),$$

де E_i – індикатор відмови (тайм-аут, збої інструмента з абортом, відхилення допуском, внутрішня помилка).

Оскільки навчальні сесії багатокрокові, корисно формалізувати обмеження й на рівні траєкторії. Нехай L(T) – метрика сесійної чутливості, наприклад середній час до наступної репліки або максимум у вікні:

$$L(T) = g(L_1, \ldots, L_n) \quad (17)$$

і задаємо:

$$Q_{0.95}(L(T)) \le L^{max} \quad (18)$$

SLA на рівні зовнішнього запиту є похідним, а первинними є SLA на рівні етапів і сесійної взаємодії.

Сигнали спостережуваності та схема телеметрії

Траєкторно-орієнтоване керування потребує телеметрії, семантично узгодженої з параметрами траєкторії. Системні метрики (утилізація CPU/GPU) недостатні, бо не розрізняють важкість запитів, траєкторний фан-аут і токенну структуру. Аналогічно, справедливість і допуск мають спиратися на токенну та траєкторну вартість, а не на лічильники запитів.

Задамо мінімальну схему телеметрії, яка підтримує реконструкцію траєкторії та атрибуцію витрат. Кожен крок емісує проміжок трасування для інференсу,

інструментів, пам'яті й оркестрації з атрибутами, що дозволяють агрегувати події в ознаки траєкторії. Таблиця 1 наводить компактну схему, придатну для стандартних pipeline observability за умови наявності семантичних атрибутів.

Схема є мінімальною і не нав'язує конкретних засобів інструментування, окрім вимоги щодо стандартизації сигналів та їх семантичної узгодженості між компонентами. Ключова умова полягає у стабільному відображенні спостережуваних сигналів у параметри (2).

Таблиця 1. Схема телеметрії для траєкторного обліку й керування

Сигнал	**Область**	**Ключові атрибути (приклади)**
Span інференсу	Крок	L_{in}, L_{out}, фаза (prefill/decode), проксі повторного використання KV
Span інструмента	Крок	тип інструмента, k_{tool}, тривалість, прапорець збою, клас віддаленої залежності
Span пам'яті	Крок	тип операції (retrieve/store/index), k_{mem}, проксі розміру даних
Span оркестрації	Крок	стан політики π, роль, рішення маршрутизації, ідентифікатор кроку в T
Лічильники/метрики	Сесія	токени на сесію, кроки на сесію, частота інструментів, квантілі tail latency
Журнали	Сесія	причини помилок, рішення допуску, події обрізання/узагальнення контексту

Постановка задачі оптимізації ресурсів

На основі введеної моделі сформулюємо задачу оптимізації. Нехай Θ – множина керованих параметрів політики обслуговування, а саме пакетування, пріоритизація, допуск запитів, маршрутизація, бюджет контексту та інструментальних викликів та параметри автомасштабування. Задача полягає у мінімізації очікуваної адитивної вартості траєкторій за умови дотримання QoS/SLA та ємнісних обмежень:

$$ET \sim D\ [C^{add}(T;\ \theta)] \quad (19)$$

за умов:

$$Q_{0.95}(L_i;\ \theta) \le L_{0.95}^{max} \quad (20)$$

$$Q_{0.99}(L_i;\ \theta) \le L_{0.99}^{max} \quad (21)$$

$$Pr(E_i = 1;\ \theta)\ \le\ \varepsilon \quad (22)$$

$$C_{gpu}^{peak}(T;\ \theta)\ \le\ M_{gpu}^{max} \quad (23)$$

$$K_i(\theta)\ \le\ K^{max} \quad (24)$$

У нашому випадку ця робота задає структуру сформульованої задачі. Вона визначає простір станів (траєкторія), функцію вартості (декомпозиція C(T)) та обмеження (QoS/SLA і ємнісні). Конкретні політики планування, допуску та автомасштабування на основі цієї постановки є предметом наступної роботи.

Обговорення

Траєкторно-орієнтоване моделювання переформульовує оптимізацію ресурсів для agent-based тьюторингу як багатокомпонентну задачу керування, де первинним об'єктом контролю є траєкторія, а не запит. Це має практичні наслідки для планування, допуску, автомасштабування та справедливості.

Декомпозиція (3) найпомітніше проявляється там, де планування і допуск перетинаються. Якщо планувальник

оптимізує лише локальну GPU-ефективність, він не бачить, що простій GPU під час I/O-очікування інструмента – це не проблема завантаження, а проблема порядку виконання кроків: жодна додаткова GPU-потужність цього не виправить, потрібне узгодження з контуром оркестрації. Та сама прогалина проявляється і в допуску запитів. Сесія з верифікацією коду та кількома інструментальними викликами має суттєво вищий ресурсний профіль, ніж коротка пояснювальна сесія; конкретне співвідношення залежить від моделі, довжини контексту та політики кешування, але платформа, що квотує за кількістю запитів, бачить обидві як «один запит» і не може розрізнити їх до моменту фактичного навантаження.

Реактивне автомасштабування реагує на вже наявну деградацію p95; траєкторні передвісники дозволяють виявити зростання навантаження до накопичення черги. Наприклад, зростання середньої довжини контексту в поточних сесіях скорочує доступний GPU-бюджет для нових сесій ще до того, як черга починає накопичуватись. Водночас зростає ціна observability (13) та складність трансляції семантичних сигналів у масштабувальні дії.

Головним обмеженням моделі є те, що функції f_pre, f_dec і коефіцієнти α не є універсальними константами – вони залежать від моделі, апаратної конфігурації, inference-serving платформи та профілю навантаження. Тому модель не задає готових числових порогів, а визначає структуру вимірювання й оптимізаційної постановки, яка має бути параметризована за даними конкретного розгортання. Додатковим обмеженням є адитивна апроксимація (4), яка некоректна для конкурентного споживання GPU-пам'яті в мультиорендному середовищі – цей ефект потребує нелінійного моделювання у подальших роботах.

У контексті edtech модель підтримує причинно-наслідкове узгодження системних цілей і вимог педагогічної взаємодії. Хоча педагогічна якість у цій статті не оптимізується явно, методологія наголошує на ортогональних вимірах оцінювання (педагогічна якість, QoS/SLA, ресурсна доцільність) та на парето-узгоджених порівняннях. Траєкторна вартість забезпечує ресурсний вимір на коректній гранулярності (крок/сесія) і створює основу для спільного аналізу компромісів після інтеграції педагогічних метрик.

Висновки

У статті показано, що традиційна одиниця керування ресурсами – запит або репліка сервісу – не відповідає реальній одиниці споживання в тьюторингових агентах, якою є траєкторія. На основі цього спостереження побудовано модель, що декомпонує вартість траєкторії за GPU, CPU, пам'яттю, мережею і спостережуваністю та задає квантильні обмеження QoS/SLA на рівні кроку і сесії. На основі декомпозиції сформульовано оптимізаційну постановку: мінімізація адитивних витрат траєкторій за умови дотримання квантильних SLA-обмежень і ємнісних обмежень інфраструктури. Постановка не залежить від конкретної реалізації inference-сервера чи оркестратора і може бути параметризована під конкретне розгортання.

Запропонована мінімальна схема телеметрії забезпечує реконструкцію траєкторій і трактує спостережуваність як передумову керування. Головним обмеженням моделі лишається відсутність емпіричної перевірки α-коефіцієнтів і нелінійних ефектів конкуренції за GPU-пам'ять, описаних у п. 4.2 – без цього модель задає структуру обліку, але не дає готових числових порогів для конкретного розгортання.

Література

1. Yao S., Zhao J., Yu D., Du N., Shafran I., Narasimhan K., Cao Y. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv preprint. 2022.

arXiv:2210.03629. DOI: 10.48550/arXiv.2210.03629.

2. Schick T., Dwivedi-Yu J., Dessì R., Raileanu R., Lomeli M., Zettlemoyer L., Cancedda N., Scialom T. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv preprint. 2023. arXiv:2302.04761. DOI: 10.48550/arXiv.2302.04761.

3. Park J. S., O'Brien J. C., Cai C. J., Morris M. R., Liang P., Bernstein M. S. Generative Agents: Interactive Simulacra of Human Behavior. arXiv preprint. 2023. arXiv:2304.03442.DOI: 10.48550/arXiv.2304.03442.

4. Wu Q., Bansal G., Zhang J., Wu Y., Li B., Zhu E., Jiang L., Zhang X., Zhang S., Liu J., Awadallah A. H., White R. W., Burger D., Wang C. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv preprint. 2023. arXiv:2308.08155. DOI: 10.48550/arXiv.2308.08155.

5. Kwon W., Li Z., Zhuang S., Sheng Y., Zheng L., Yu C. H., Gonzalez J. E., Zhang H., Stoica I. Efficient Memory Management for Large Language Model Serving with PagedAttention. Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '23). New York, NY: ACM, 2023. P. 611–626. DOI: 10.1145/3600006.3613165.

6. Moritz P., Nishihara R., Wang S., Tumanov A., Liaw R., Liang E., Elibol M., Yang Z., Paul W., Jordan M. I., Stoica I. Ray: A Distributed Framework for Emerging AI Applications. Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18). Carlsbad, CA: USENIX Association, 2018. P. 561–577.

7. Schroeder C., et al. Comparison of Autoscaling Frameworks for Containerised Machine-Learning-Applications in a Local and Cloud Environment. arXiv preprint. 2023. arXiv:2311.18659. DOI: 10.48550/arXiv.2311.18659.

8. Wang G., Xie Y., Jiang Y., Mandlekar A., Xiao C., Zhu Y., Fan L., Anandkumar A. Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv preprint. 2023. arXiv:2305.16291. DOI: 10.48550/arXiv.2305.16291.

9. Ilyushkin A., Ali-Eldin A., Ghit B., Iosup A., Epema D. A Workload Forecasting Model for Private Clouds. IEEE Transactions on Cloud Computing. 2017. Vol. 7, No. 4. P. 949–963.

10. KEDA Project. Kubernetes Event-Driven Autoscaling (KEDA): Documentation. URL: https://keda.sh/docs/

11. OpenTelemetry Project. Observability Primer: Documentation. URL: https://opentelemetry.io/docs/concepts/observability-primer/

12. Luo J., Zhang W., Yuan Y., Zhao Y., Yang J., Gu Y., Wu B., Chen B., Qiao Z., Long Q., et al. Large Language Model Agent: A Survey on Methodology, Applications and Challenges. arXiv preprint. 2025. arXiv:2503.21460. DOI: 10.48550/arXiv.2503.21460.

13. Xu W., Huang C., Gao S., Shang S. LLM-Based Agents for Tool Learning: A Survey. Data Science and Engineering. 2025. Vol. 10. P. 533–563. DOI: 10.1007/s41019-025-00296-9.

14. Zhong Y., Liu S., Chen J., Hu J., Zhu Y., Liu X., Jin X., Zhang H. DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving. Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24). USENIX Association, 2024.

15. Agrawal A., Kedia N., Panwar A., Mohan J., Kwatra N., Gulavani B. S., Tumanov A., Ramjee R. Taming Throughput–Latency Tradeoff in LLM Inference with Sarathi-Serve. Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24).USENIX Association,2024. P. 117–134.

16. Sheng Y., Cao S., Li D., Hooper C., Lee N., Yang S., Chou C., Zhu B., Zheng L., Keutzer K., Gonzalez J. E., Stoica I. Fairness in Serving Large Language Models. Proceedings of the 18th USENIX Symposium

on Operating Systems Design and Implementation (OSDI '24). USENIX Association, 2024.

17. Khan R. I. S., Jain K., Shen H., Mallick A., Parayil A., Kulkarni A., Kofsky S., Choudhary P., St. Amant R., Wang R., Cheng Y., Butt A. R., Rühle V., Bansal C., Rajmohan S. Ensuring Fair LLM Serving Amid Diverse Applications. arXiv preprint. 2024. arXiv:2411.15997.

18. Gao B., He Z., Sharma P., Kang Q., Jevdjic D., Deng J., Yang X., Yu Z., Zuo P. Cost-Efficient Large Language Model Serving for Multi-Turn Conversations with CachedAttention. arXiv preprint. 2024. arXiv:2403.19708.

19. OpenTelemetry. GenAI Semantic Conventions. URL: https://github.com/open-telemetry/semantic-conventions-genai

Бордіян А.І.

ТРАЄКТОРНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ РЕСУРСІВ ДЛЯ AI-АГЕНТІВ У СЕРВІСАХ ОНЛАЙН-НАВЧАННЯ

AI-агенти на базі великих мовних моделей дедалі частіше впроваджуються у сервіси онлайн-навчання, де взаємодія має сесійний, багатокроковий та інструментально-розширений характер. У таких системах споживання ресурсів не описується традиційною одиницею керування. Натомість один користувацький запит породжує внутрішню траєкторію навантаження, що складається з кількох викликів інференсу, інструментальних I/O, операцій пам'яті та накладних витрат спостережуваності, при цьому GPU-профіль визначається токенною структурою та двофазною природою інференсу (prefill проти decode), а довгі контексти критично впливають на хвіст розподілу затримок. У статті запропоновано траєкторно-орієнтовану модель споживання ресурсів AI-агентами у високонавантажених освітніх платформах. Формалізовано агентну траєкторію як первинний об'єкт обліку, введено позначення, що пов'язують параметри траєкторії з витратами GPU/CPU/пам'яті/мережі, та задано обмеження QoS/SLA через квантільні характеристики затримки й обмеження частоти помилок, узгоджені з інтерактивними вимогами тьюторингу. Модель враховує архітектурне розділення шарів інференсу й оркестрації та роль інструментальних викликів і пам'яті як складових критичного шляху затримки. Окремо формалізовано відмінність між спостережуваністю як керованим контуром і як постфактум-діагностикою.

Ключові слова: *онлайн-навчання; оптимізація ресурсів; розподілені системи; обслуговування висновків; QoS/SLA; автомасштабування; спостережуваність; токенна вартість.*

Bordiian A.

TRAJECTORY-ORIENTED RESOURCE MODELING FOR AI AGENTS IN ONLINE LEARNING SERVICES

AI agents based on Large Language Models are increasingly being integrated into online learning services, where interaction is session-based, multi-step, and extended through the use of external tools. In such systems, resource consumption cannot be described by a traditional unit of control. Instead, a single user request generates an internal workload trajectory that may include several inference calls, tool-related I/O operations, memory operations, and observability overhead. At the same time, the GPU workload profile is shaped by the token structure and the two-phase nature of inference, namely prefill and decode, while long contexts have a critical impact on the tail of the latency distribution. This paper proposes a trajectory-oriented model of resource consumption by AI agents in high-load educational platforms. The agent trajectory is formalized as the primary object of resource accounting. The paper introduces notation that links trajectory parameters with GPU, CPU, memory, and network costs, and defines QoS/SLA constraints through latency quantiles and error-rate limits aligned with the interactive requirements of tutoring. The model takes into account the architectural separation between the inference and orchestration layers, as well as the role of tool calls and memory operations as components of the critical latency path. The paper also formalizes the distinction between observability as part of a control loop and observability as post-factum diagnostics.

Keywords: *online learning; resource optimization; distributed systems; inference serving; QoS/SLA; autoscaling; observability; token cost.*

Стаття подана до редакції: 04/09/2026
Стаття прийнята до опублікування:12/09/2026
Стаття опублікована: 10/11/2026

Стаття поширюється на умовах ліцензії CC BY 4.0