УДК 004.8 DOI: 10.18372/2073-4751.86.21268

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

ЕВОЛЮЦІЯ АРХІТЕКТУР AI-АГЕНТІВ ТА ПРОБЛЕМИ ОПТИМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ У СЕРВІСАХ ОНЛАЙН-НАВЧАННЯ

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

Вступ

Впровадження моделей LLM у цифрових платформах призвело до утворення нового класу систем – AI-агентів, які виконують функції інтелектуальних тьюторів, асистентів з розв’язування задач, систем діагностики помилок та персоналізованої підтримки. На відміну від традиційних сервісів обробки запитів, такі агенти функціонують у режимі багатокрокової взаємодії, підтримують стан сесії та інтегруються з інструментами й базами знань. У контексті онлайн-навчання це означає, що агент виконує послідовні кроки міркування, здійснює виклики зовнішніх сервісів (пошук, перевірка розв’язків), звертається до пам’яті (профіль студента, історія взаємодії) та адаптує стратегію підтримки з урахуванням контексту. Відповідно, навантаження визначається не окремим запитом, а траєкторією сесії, що включає змінну кількість інференс-викликів, I/O-операцій та проміжних перевірок.

У високонавантажених сервісах онлайн-навчання саме архітектурні рішення впливають на споживання обчислювальних ресурсів та стабільність SLA. Це зумовлює необхідність системного аналізу еволюції архітектур AI-агентів та методів оптимізації їх ресурсної ефективності.

Основна частина

З позицій системного проєктування AI-агент у сервісі онлайн-навчання доцільно трактувати як композицію контурів, до яких входять інференс (обробка контексту та генерація відповіді), оркестрація дій (планування кроків, вибір інструментів, допуск та пріоритизація), пам’ять і дані (профіль учня, історія взаємодії, індекси/пошук) та спостережуваність (трасування, метрики, журнали). У такій композиції «одиниця споживання» ресурсу природно відповідає траєкторії – послідовності кроків, у якій інференс, пам’ять і інструментальні виклики чергуються та утворюють шлях затримки.

Узагальнену схему взаємодії контурів агентної системи зображено на рисунку 1. У найпростішому вигляді агентну архітектуру можна представити як послідовність потоків керування та даних.

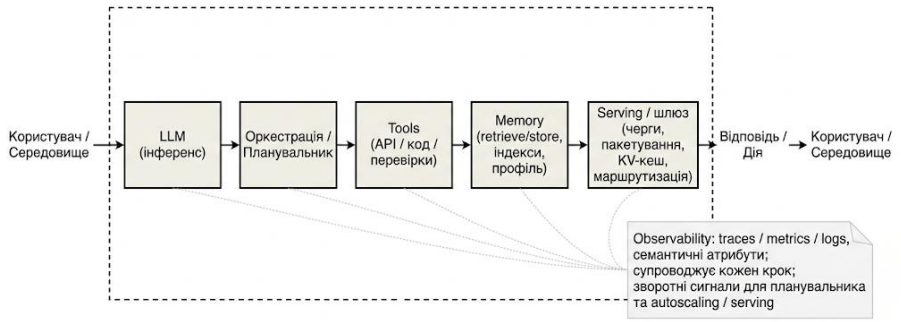

Рис. 1. Узагальнена схема архітектури AI-агента (потоки керування та даних)

На практиці така схема підкреслює дві властивості, які не завжди очевидні при описі агента як застосунку. По-перше, інференс (LLM) не є ізольованою операцією, він взаємодіє з контуром обслуговування, який визначає компроміс throughput/latency через пакетування і керування KV-кешем, а отже впливає на хвіст розподілу затримок у сесіях. По-друге, tools і memory формують негомогенний критичний шлях, оскільки вводять I/O-паузи, мережеві залежності та варіативність часу обслуговування, що прямо транслюється у tail latency та у вимоги до допуску та пріоритизації на рівні оркестрації.

Для навчальних сценаріїв характерними є довгі сесії з накопиченням контексту, неоднорідність активностей (пояснення, перевірка, тестування, підказки, розбір помилки) та високий фан-аут внутрішніх операцій, де один «крок тьютора» може містити звернення до матеріалів курсу, перевірку проміжних розв'язків та повторні виклики LLM для уточнення. У зв'язку з цим традиційні підходи, що оперують запитом як атомом планування, не відображають реальної вартості обслуговування, а QoS/SLA в інтерактивних системах визначається, передусім, хвостом розподілу затримок (p95/p99) і стабільністю під піковим навантаженням.

Сучасні AI-агенти сформувалися на перетині двох напрямів, а саме розвитку LLM як «ядра агентності» та індустріалізації інфраструктури serving/оркестрації. Парадигма ReAct [1] продемонструвала можливість інтеграції міркування та дії у циклі «thought–action», де модель не лише продукує текст, а й ініціює зовнішні дії. У ресурсному вимірі це означає імпульсний профіль GPU (короткі токенні фрагменти) у поєднанні з потенційно тривалими I/O-паузами під час інструментальних викликів, що підвищує значущість планувальника.

Робота Toolformer [2] демонструє можливість інтеграції зовнішніх API-викликів безпосередньо в процес генерації. У результаті архітектура агентів стає більш залежною від інструментальних викликів, що збільшує варіативність часу обслуговування запитів. У сервісах онлайн-навчання це напряму транслюється у ризики затримки хвоста, оскільки затримка одного інструмента може блокувати важливий шлях педагогічної взаємодії.

Архітектури генеративних агентів із довготривалою пам'яттю [3] показали, що пам'ять перетворюється на активний компонент критичного шляху, де індексація, пошук, ранжування й витяг релевантного контексту виконуються регулярно та створюють мережеві й обчислювальні накладні витрати. Для

навчальних сервісів, де персоналізація є базовою вимогою, частота звернень до пам'яті зростає пропорційно активній аудиторії, а невдале рішення щодо розміщення пам'яті (локально/віддалено, синхронно/асинхронно) призводить до системної деградації QoS.

Багатоагентні системи, зокрема AutoGen [4], змістили акцент із «екземпляра агента» на «топологію взаємодії». Рольові агенти (планувальник, перевіряльник, пояснювач) дозволяють гетерогенний розподіл ресурсів, в основі якого лежить те, що частину логіки виконувати на CPU або меншому бюджеті контексту, а ресурсоємні пояснення – на GPU та більшому контексті. Водночас багатоагентність додає накладні витрати на координацію й дублювання контексту, що при наївній реалізації може мати квадратичну компоненту за кількістю повідомлень.

Окрему лінію еволюції становлять агенти з «безперервним удосконаленням» у контурі виконання (наприклад, Voyager [41]). Такі архітектури підсилюють роль виконання коду та інструментів, зміщуючи ресурсний баланс, де частина інтелекту реалізується як позамодельні обчислення. У навчальному контексті аналогічно працюють перевірки рішень, симуляції, автоматичні тести, що різко підвищує вимоги до ізоляції, квотування та безпечного мультиорендного виконання інструментів.

Паралельно еволюціонував інфраструктурний шар обслуговування (serving). vLLM [5] продемонстрував, що ефективність LLM-обслуговування визначається керуванням KV-кешем та механізмами пакетування. Це важливо для довгих діалогів у навчанні, бо без політики контролю контексту навіть високоефективний serving деградує через вичерпання відеопам'яті. Ray Serve [6] оформив програмований шар розгортання з DAG-композицією та незалежним масштабуванням компонентів, що концептуально узгоджується з потребою «масштабувати контури», а не «масштабувати все разом».

Узагальнення етапів еволюції подано в таблиці 1.

Таблиця 1. Етапи еволюції архітектур AI-агентів

Етап	**Ознаки**	**Ресурсні наслідки**	**Типові проблеми**
Монолітний	Єдиний процес	Неефективне GPU-резервування	Деградація SLA
Композиційний	Розділення контурів	Гнучке масштабування CPU та GPU	Складність координації
Інструментальний	Виклики API	Варіативність latency, I/O паузи	Tail latency, простої GPU
Пам'ять-центричний	Пошук/контекст у критичному шляху	Мережеві та сховищні витрати	Неконтрольоване зростання контексту
Багатоагентний	Рольова взаємодія	Гетерогенність ресурсів	Накладні витрати координації
Інфраструктурний	Оптимізований serving та оркестрація	Вища пропускна здатність	Залежність від політик контексту

Перейдемо до моделей обчислювальних ресурсів та інференс LLM. Інференс LLM має двофазну природу, а саме prefill (обробка вхідного контексту) та decode (породження токенів). Системні огляди обслуговування

висновків (inference serving) [13] підкреслюють, що ці фази мають різні властивості щодо пропускної здатності та латентності (throughput/latency) і по-різному реагують на пакетування. Sarathi-Serve [15] формалізує конфлікт throughput–latency як наслідок змішування фаз і пропонує механізми, що зменшують деградацію хвоста (p99) за високих навантажень.

WindServe [16] та semi-PD [22] розвивають фазово-дезагреговане обслуговування, де планування потоків виконання дозволяє краще завантажувати GPU при змішаних довжинах контексту. Seesaw [14] вводить перерозподіл (resharding) як підхід до масштабування з урахуванням пропускної здатності та розміщення, а ORCA [17] демонструє архітектурні рішення для розподіленого обслуговування генеративних трансформерів.

Квантування та стискання (SmoothQuant [19], GPTQ [20], AWQ [21]) зменшують вимоги до пам'яті та змінюють обчислювальний режим інференсу. Емпіричні роботи з енергетики та латентності обслуговування [23] показують, що ефект квантування проявляється нерівномірно за фазами та залежить від апаратури, що ускладнює «універсальні» політики масштабування.

Для агентних систем у навчанні ключовим є те, що ресурсна складність визначається не лише однією інференс-операцією, а поведінкою сесії, оскільки контекст росте, інструментальні паузи змінюють структуру черг, а повторні виклики моделі можуть множити токенну вартість одного «кроку тьютора». Тому модель ресурсу повинна включати токенні змінні (довжина контексту, очікувана довжина відповіді), динаміку KV-кешу та правила контролю контексту. Траєкторію сесії агента можна позначити як послідовність

$$T = \{s1, s2, \dots, sn\} \qquad (1)$$

, де кожен крок si відповідає інференс-виклику, інструментальній операції або зверненню до пам'яті.

У системному зрізі методи оптимізації ресурсів доцільно групувати за рівнями впливу:

1. Рівень моделі. Квантування та компресія (SmoothQuant [19], GPTQ [20], AWQ [21]) зменшують пам'ять та латентність, однак не формують гарантій SLA і можуть опосередковано змінювати навчальну динаміку (більше уточнень, довші діалоги).
2. Рівень inference serving. Керування KV-кешем та пакетування (vLLM [5]) підвищують пропускну здатність, фазове планування (Sarathi-Serve [15], WindServe [16]) стабілізує tail latency, а розподілені системи (ORCA [17], Seesaw [14]) підтримують масштабування великих моделей. Проте ці методи не охоплюють інструментальні паузи та пам'ять агента.
3. Рівень агентної логіки. Планування траєкторій (ReAct [1]) та інструментальність (Toolformer [2]) визначають фан-аут і структуру внутрішніх робіт, в той час, як агенти з пам'яттю [3] формують додаткові критичні компоненти (пошук/витяг). Відсутність явного «ресурсного контракту» на цьому рівні призводить до непередбачуваності профілю навантаження.
4. Рівень оркестрації. Програмовані шари (Ray Serve [6]) дозволяють відокремити масштабування компонентів, однак політики масштабування мають узгоджуватися з токенно- орієнтованими профілями.
5. Рівень прискорення декодування. Спекулятивне декодування [18] змінює структуру обчислень і може зменшувати вартість на токен, але водночас ускладнює прогнозування та керування мультиорендністю.

Зіставлення рівнів оптимізації наведено в таблиці 2.

Таблиця 2. Таксономія методів оптимізації ресурсів

Рівень	**Ідея**	**Метрики**	**Обмеження**
Модель	Квантування та стиснення	Пам'ять, час інференсу	Вплив на якість, апаратна залежність
Inference serving	Пакетування, KV-кеш, фази	Пропускна здатність, p95/p99	Не охоплює інструменти та пам'ять агента
Оркестрація	DAG, компоненти, автомасштабування	Черги, latency, CPU та GPU	Запізнення сигналів, проксі-метрики
Агент і пам'ять	Контроль контексту, траєкторії	Токени, кроки, fan-out	Складність формалізації, гетерогенність
Observability	Семантичні сигнали	Траси, метрики, логи	Накладні витрати, неповні конвенції

Далі варто обговорити перехід від RPS до токенної вартості. У мультиорендних LLM-платформах справедливість обслуговування є не «додатковою» вимогою, а механізмом запобігання системній деградації QoS. Робота [24] показує, що справедливість має визначатися у термінах токенної вартості (вхідні та вихідні токени), а не у термінах кількості запитів. Дослідження [25] підкреслює, що різноманітність застосунків і неоднорідність запитів роблять RPS-квоти некоректними, оскільки один «запит» може містити суттєво різний токенний та часовий бюджет.

Для онлайн-навчання ця проблема посилюється траєкторністю, оскільки одна педагогічна дія може викликати кілька LLM-кроків (пояснення, перевірка, уточнення), звернення до пам'яті та інструментів. Отже, одиницею планування має бути сесія/траєкторія, а механізми контролю доступу (admission control) і черг повинні враховувати не лише кількість запитів, а й очікувану «важкість» траєкторії, до якої входить довжина контексту, ймовірність інструментальних викликів і очікуваний фан-аут.

Також варто розглянути автомасштабування та прогнозування навантаження. Огляди та таксономії автомасштабування [8, 26, 27] узагальнюють класичні підходи як поєднання тригерів, стратегій та об'єктів масштабування. Проте для агентних сервісів реактивні сигнали (CPU%, середній latency) часто запізнюються, оскільки вони фіксують уже реалізовану деградацію хвоста затримок.

Прогностичні системи та огляди (PASS [28], application-oriented prediction [29], систематичний огляд cloud sizing [30]) демонструють, що прогнозування може зменшувати надлишок і дефіцит ресурсів. Подієво-орієнтоване масштабування у Kubernetes через KEDA [8, 9] дозволяє масштабувати за сигналами черг/подій, що є релевантним для інструментальних викликів агентів (зовнішні черги, запити до підсистем перевірки).

Втім, центральна проблема для AI-агентів полягає в тому, що прогнозувати потрібно не лише інтенсивність, а й «важкість» запитів, а саме токенні профілі, довжину контексту, ймовірність інструментальних траєкторій. Структурні підходи для мікросервісів [31] враховують граф залежностей, але для агентів граф залежить від політики планування і може бути динамічним. Це підводить до вимоги інтегрованої моделі, де навантаження описується як композиція фаз інференсу, інструментальних викликів і операцій пам'яті.

Подивимось на спостережуваність як на основу керованої оптимізації. OpenTelemetry консолідує практики телеметрії у вигляді трас, метрик і

журналів, формуючи основу для порівнянних вимірювань у розподілених системах [10, 12]. Для задач оптимізації ресурсів важливо, що спостережуваність повинна бути не лише діагностикою, а частиною керованого контуру, оскільки профілювання має постачати сигнали для планувальника й автомасштабування.

Метрики semantic conventions [32] та конвенції для генеративних систем [33, 34] створюють передумови типізованої телеметрії, до якої входять подія інференсу, параметри токенізації, подія інструментального виклику, подія звернення до пам'яті та атрибути контексту. Вибірковість трасування (sampling) [11] є критичною через накладні витрати, оскільки повне трасування кожного кроку агента у піках може саме стати джерелом деградації.

У навчальних сервісах спостережуваність має додатковий вимір, оскільки потрібна кореляція системних сигналів (latency, GPU memory) з «семантикою навчальної взаємодії» (тип активності, довжина сесії, крок плану). Без цього неможливо коректно порівнювати альтернативи оптимізації та будувати відтворювані висновки.

Також для освітніх сервісів системні метрики недостатні без педагогічних критеріїв. TutorBench [35] та MathTutorBench [36] оформлюють оцінювання тьютора як багатовимірну рубрику, що відокремлює предметну правильність від дидактичної адекватності. Таксономія оцінювання педагогічних здібностей [37] формує єдиний «словник» критеріїв для порівняння моделей і протоколів.

AgentBench [38] переводить оцінювання в площину «дії в середовищі», що релевантно для агентів, які користуються інструментами (перевірки, бази задач). Підходи LLM-судді (GEval [39]) масштабують оцінювання, але мають ризики упередженості й змішування «якість тексту» з «педагогічною коректністю». Платформи преференційного оцінювання (Chatbot Arena [40]) дають сигнал людської переваги, однак їхні висновки потребують узгодження з вимогами QoS/SLA і ресурсними бюджетами.

З точки зору методології, ключовою вимогою є інтеграція трьох вимірів, а саме педагогічна якість, QoS/SLA (особливо p95/p99 «час до наступної репліки») та ресурсна доцільність (токени, GPU-час, витрати інструментів). Така інтеграція природно веде до парето-порівняння конфігурацій та до необхідності семантичної спостережуваності, яка дозволяє вимірювати витрати на рівні кроку або сесії.

Аналіз наведених підходів дозволяє виявити спільну закономірність. Більшість методів оптимізації сильні на власному рівні (модель, serving, autoscaling), але вони не утворюють інтегрованого методу для AI-агентів у високонавантажених сервісах онлайн-навчання. Причиною є невідповідність між одиницею керування (запит, контейнер, сервіс) та одиницею споживання (траєкторія або сесія з токенно-інструментальним профілем).

На основі цього доцільно зробити узагальнення з позицій що саме оптимізується. Наявні підходи, як правило, оптимізують модельні параметри або формат обчислення (квантування та стиснення), локальну ефективність inference serving (KV-кеш, пакетування, фазове планування), або реактивні чи прогностичні контури автомасштабування за агрегованими метриками компонентів. Однак у агентних освітніх сервісах «вартість» і «ризик порушення SLA» визначаються не окремою інференс-операцією і не середнім завантаженням контейнера, а траєкторією сесії як композицією кроків, де змінюється довжина контексту, повторюються виклики моделі, з'являються інструментальні паузи та звернення до пам'яті.

Відсутність траєкторії як первинної одиниці ресурсного обліку призводить до системного ефекту, де політики керування (допуск, пріоритизація, квоти, autoscaling)

працюють з проксі-метриками і «бачать» деградацію вже постфактум, тоді як ключові показники порушення QoS та SLA закладені у структурі та еволюції траєкторії під час сесії. Отже, саме узгодження траєкторної деталізації з політиками serving, оркестрації та масштабування і з семантичними сигналами observability є тим місцем, де поточні дослідження залишаються неповними.

Це узгоджується з висновками щодо токенно-орієнтованої справедливості та фазово-орієнтованих властивостей інференсу, які додатково підкреслюють структурну неоднорідність навантаження.

Справедливість обслуговування у токенних термінах [24, 25] та фазово-орієнтовані моделі інференсу [15, 16] підкреслюють, що навіть усередині одного запиту існує структурна неоднорідність. Для агентних систем ця неоднорідність посилюється інструментальними викликами [2] та використанням пам'яті [3], що робить реактивне масштабування за агрегованими метриками недостатнім. Це дозволяє сформулювати наукову прогалину дослідження. Відсутній узгоджений метод, який зв'язує траєкторну модель навантаження, політики обслуговування і оркестрації, систему семантичної observability та вимоги QoS/SLA і педагогічної ефективності.

Висновки

Аналіз існуючих систем продемонстрував, що сучасні архітектури AI-агентів є багаторівневими композиціями, у яких інференс, оркестрація, пам'ять і спостережуваність формують взаємозалежні контури. Існуючі методи оптимізації ефективні локально (квантування, обслуговування, фазове планування, автомасштабування), проте не формують узгодженого методу керування ресурсами на рівні агентної траєкторії.

Наукова прогалина полягає у відсутності інтегрованого методу, що поєднує:

- траєкторну модель навантаження (токени, кроки, fan-out, інструменти, пам'ять);
- політики inference serving (KV-кеш, пакетування, фази);
- автомасштабування з прогнозуванням важкості запитів;
- семантичну спостережуваність як джерело керованих сигналів;
- критерії педагогічної ефективності у зв'язку з ресурсною доцільністю.

Подальші дослідження мають бути спрямовані на формалізацію такого методу та його експериментальну верифікацію в умовах реальних сервісів онлайн-навчання.

Література

1. *Yao, Shunyu, et al.* 'ReAct: Synergizing Reasoning and Acting in Language Models'. arXiv [Cs.CL], 2023, arxiv.org/abs/2210.03629. arXiv.

2. *Schick, Timo, et al.* 'Toolformer: Language Models Can Teach Themselves to Use Tools'. arXiv [Cs.CL], 2023, arxiv.org/abs/2302.04761. arXiv.

3. *Park, Joon Sung, et al.* 'Generative Agents: Interactive Simulacra of Human Behavior'. arXiv [Cs.HC], 2023, arxiv.org/abs/2304.03442. arXiv.

4. *Wu, Qingyun, et al.* 'AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation'. arXiv [Cs.AI], 2023, arxiv.org/abs/2308.08155. arXiv.

5. *Kwon, Woosuk, et al.* 'Efficient Memory Management for Large Language Model Serving with PagedAttention'. Proceedings of the 29th Symposium on Operating Systems Principles, Association for Computing Machinery, 2023, pp. 611–626, https://doi.org/10.1145/3600006.3613165. SOSP '23.

6. Ray Project. *(2026). Ray Serve:* Scalable and programmable serving (documentation).

7. *Schroeder, Christian, et al.* 'Comparison of Autoscaling Frameworks for Containerised Machine-Learning-Applications in a Local and Cloud

Environment'. arXiv [Cs.DC], 2024, arxiv.org/abs/2311.18659. arXiv.

8. *Tina Lekshmi Kanth.* 'Predictive autoscaling in Kubernetes microservices with KEDA and time series forecasting'. Journal of Information Systems Engineering and Management, 2026, pp. 151–163, e-ISSN:2468-4376.

9. KEDA. Kubernetes event-driven autoscaling / URL: https://keda.sh/

10. OpenTelemetry. Project and roadmap update from KubeCon / URL: https://opentelemetry.io/blog/2022/kubecon-na-project-update/

11. OpenTelemetry. Sampling milestones (Tracing specification update / URL:https://opentelemetry.io/blog/2025/sampling-milestones/

12. OpenTelemetry. Observability primer (concepts) / URL: https://opentelemetry.io/docs/concepts/observability-primer/

13. *Li, Baolin, et al.* 'LLM Inference Serving: Survey of Recent Advances and Opportunities'. arXiv [Cs.DC], 2024, arxiv.org/abs/2407.12391. arXiv.

14. *Su, Qidong, et al.* 'Seesaw: High-Throughput LLM Inference via Model Re-Sharding'. arXiv [Cs.DC], 2025, arxiv.org/abs/2503.06433. arXiv.

15. *Agrawal, Amey, et al.* 'Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve'. Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, 2024. OSDI'24.

16. *Feng, Jingqi, et al.* 'WindServe: Efficient Phase-Disaggregated LLM Serving with Stream-Based Dynamic Scheduling'. Proceedings of the 52nd Annual International Symposium on Computer Architecture, Association for Computing Machinery, 2025, pp. 1283–1295, https://doi.org/10.1145/3695053.3730999. ISCA '25.

17. *Yu, Gyeong-In, et al.* 'Orca: A Distributed Serving System for Transformer-Based Generative Models'. 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), USENIX Association, 2022, pp. 521–538, www.usenix.org/conference/osdi22/presentation/yu.

18. *Yan, Minghao, et al.* 'Decoding Speculative Decoding'. arXiv [Cs.LG], 2025, arxiv.org/abs/2402.01528. arXiv.

19. *Xiao, Guangxuan, et al.* 'SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models'. Proceedings of the 40th International Conference on Machine Learning, JMLR.org, 2023. ICML'23.

20. *Frantar, Elias, et al.* 'GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers'. arXiv [Cs.LG], 2023, arxiv.org/abs/2210.17323. arXiv.

21. *Lin, Ji, et al.* 'AWQ: Activation-Aware Weight Quantization for LLM Compression and Acceleration'. arXiv [Cs.CL], 2024, arxiv.org/abs/2306.00978. arXiv.

22. *Hong, Ke, et al.* 'Semi-PD: Towards Efficient LLM Serving via Phase-Wise Disaggregated Computation and Unified Storage'. arXiv [Cs.CL], 2025, arxiv.org/abs/2504.19867. arXiv.

23. *Delavande, Julien, et al.* 'Understanding Efficiency: Quantization, Batching, and Serving Strategies in LLM Energy Use'. arXiv [Cs.LG], 2026, arxiv.org/abs/2601.22362. arXiv.

24. *Sheng, Y., et al.* Fairness in serving large language models. In Proceedings of OSDI '24, 2024. USENIX.

25. *Khan, Redwan Ibne Seraj, et al.* 'Ensuring Fair LLM Serving Amid Diverse Applications'. arXiv [Cs.LG], 2024, arxiv.org/abs/2411.15997. arXiv.

26. *Xu, Minxian, et al.* 'Auto-Scaling Approaches for Cloud-Native Applications: A Survey and Taxonomy'. arXiv [Cs.DC], 2025, arxiv.org/abs/2507.17128. arXiv.

27. *Jeong, Byeonghui, and Young-Sik Jeong.* 'Autoscaling Techniques in Cloud-Native Computing: A Comprehensive Survey'. Computer Science Review, vol. 58,

2025, p. 100791, https://doi.org/10.1016/j.cosrev.2025.100791

28. *Guo, Yunda, et al.* 'PASS: Predictive Auto-Scaling System for Large-Scale Enterprise Web Applications'. Proceedings of the ACM Web Conference 2024, Association for Computing Machinery, 2024, pp. 2747–2758, https://doi.org/10.1145/3589334.3645330. WWW '24.

29. *Feng, Binbin, and Zhijun Ding.* 'Application-Oriented Cloud Workload Prediction: A Survey and New Perspectives'. Tsinghua Science and Technology, vol. 30, no. 1, 2025, pp. 34–54, https://doi.org/10.26599/TST.2024.9010024.

30. *Muhammad Herwindra Berlian.* A Systematic Review on Cloud Sizing Automation for PaaS Using Historical Workload Analysis. TechRxiv. November 20, 2025.

31. *ZargarAzad, M., et al.* (2023). An auto-scaling approach for microservices in cloud environments. The Journal of Supercomputing.

32. OpenTelemetry. (2025). Metrics semantic conventions (specification) /URL:https://opentelemetry.io/docs/specs/semconv/general/metrics/

33. OpenTelemetry. (2024–2025). Semantic conventions for generative AI systems. OpenTelemetry Specification / URL:https://opentelemetry.io/docs/specs/semconv/gen-ai/

34. OpenTelemetry. (2025). AI agent observability: Evolving standards and best practices. OpenTelemetry Blog / URL: https://opentelemetry.io/blog/2025/ai-agent-observability/

35. *Srinivasa, Rakshith S., et al.* 'TutorBench: A Benchmark To Assess Tutoring Capabilities Of Large Language Models'. arXiv [Cs.LG], 2025, arxiv.org/abs/2510.02663. arXiv.

36. *Macina, Jakub, et al.* 'MathTutorBench: A Benchmark for Measuring Open-Ended Pedagogical Capabilities of LLM Tutors'. arXiv [Cs.CL], 2025, arxiv.org/abs/2502.18940. arXiv.

37. *Maurya, Kaushal Kumar, et al.* 'Unifying AI Tutor Evaluation: An Evaluation Taxonomy for Pedagogical Ability Assessment of LLM-Powered AI Tutors'. arXiv [Cs.CL], 2025, arxiv.org/abs/2412.09416. arXiv.

38. *Liu, Xiao, et al.* 'AgentBench: Evaluating LLMs as Agents'. arXiv [Cs.AI], 2025, arxiv.org/abs/2308.03688. arXiv.

39. *Liu, Yang, et al.* 'G-Eval: NLG Evaluation Using GPT-4 with Better Human Alignment'. arXiv [Cs.CL], 2023, arxiv.org/abs/2303.16634. arXiv.

40. *Chiang, Wei-Lin, et al.* 'Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference'. Proceedings of the 41st International Conference on Machine Learning, JMLR.org, 2024. ICML'24.

41. *Wang, Guanzhi, et al.* 'Voyager: An Open-Ended Embodied Agent with Large Language Models'. arXiv [Cs.AI], 2023, arxiv.org/abs/2305.16291. arXiv.

Бордіян А.І.

ЕВОЛЮЦІЯ АРХІТЕКТУР AI-АГЕНТІВ ТА ПРОБЛЕМИ ОПТИМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ У СЕРВІСАХ ОНЛАЙН-НАВЧАННЯ

У статті здійснено системний огляд еволюції архітектур AI-агентів у розподілених інтелекту- альних системах із фокусом на ресурсоємність, масштабованість та забезпечення QoS/SLA у сервісах онлайн-навчання. Проаналізовано трансформацію від монолітних реалізацій до багатоконтурних композиційних архітектур із виділенням інференсу, оркестрації, пам'яті та спостережуваності. Узагальнено класи методів оптимізації ресурсів на рівні моделі, обслуго- вування висновків, автомасштабування, керування потоками та агентної логіки. Виявлено наукову прогалину, що полягає у відсутності інтегрованого методу, який поєднує оптимізації на рівні моделі, inference serving та автомасштабування з траєкторією

агентної сесії як первинною одиницею ресурсного обліку і керування, тобто узгоджує токенно-інструментальний профіль сесії з політиками допуску, пріоритизації та масштабування в мультиорендному середовищі. Сформульовано передумови для розроблення методу та засобу оптимізації обчислювальних ресурсів AI-агентів у сервісах онлайн-навчання.
Ключові слова: *AI-агент, онлайн-навчання, LLM, інференс, масштабованість, QoS, SLA, спостережуваність, автомасштабування.*

Bordiian A.

EVOLUTION OF AI AGENT ARCHITECTURES AND THE CHALLENGES OF OPTIMIZING COMPUTATIONAL RESOURCES IN ONLINE LEARNING SERVICES.

The article provides a systematic review of the evolution of AI agent architectures in distributed intelligent systems, focusing on resource intensity, scalability, and ensuring QoS/SLA in online learning services. It analyzes the transition from monolithic implementations to multi-loop compositional architectures, distinguishing inference, orchestration, memory, and observability. The study generalizes classes of resource optimization methods at the model level, inference serving, autoscaling, flow management, and agent logic. A research gap is identified: the lack of an integrated method that combines model-level optimization, inference serving, and autoscaling with the trajectory of an agent session as the primary unit of resource accounting and control—that is, aligning the token-and-tool profile of a session with admission, prioritization, and scaling policies in a multi-tenant environment. Preconditions are formulated for developing a method and a tool for optimizing the computational resources of AI agents in online learning services.
Keywords: *AI agent, online learning, LLM, inference, scalability, QoS, SLA, observability, autoscaling.*

Стаття подана до редакції: 15/04/2026
Стаття прийнята до опублікування: 21/04/2026
Стаття опублікована: 30/05/2026