ERP/1: Істотність

Техноробочий проект

на створення та впровадження
локальної LLM-інфраструктури
для аналізу та оцінки істотності інформації

Згідно з Постановою КМУ № 205 від 21.02.2025

Формалізація вимог за стандартом ISO/IEC/IEEE 29148:2018

Зміст

Зміст

1. Загальні відомості про засіб інформатизації

1.1. Найменування та підстави для розробки

Найменування засобу інформатизації: Модуль «Істотність» (локальна ШІ-інфраструктура) у складі інформаційної системи управління підприємством ERP/1.

Підстави для виконання робіт:

- Закон України «Про захист персональних даних»;
- Закон України «Про захист інформації в інформаційно-комунікаційних системах»;
- Порядок використання засобів інформатизації (Постанова КМУ № 205 від 21.02.2025);
- Програма модернізації обчислювальної IT-інфраструктури ЄСІКС.

1.2. Призначення та цілі створення засобу

Модуль призначений для розгортання локальних інтелектуальних сервісів аналізу та семантичного пошуку з використанням великих мовних моделей (LLM) та векторних баз даних (RAG) безпосередньо on-premises на базі споживчих GPU NVIDIA RTX 4070 Ti.

Основні цілі створення:

- Забезпечення повного циклу локального інференсу LLM (Llama-3-8B-Instruct) із квантуванням ваг.
- Організація семантичного RAG-пошуку через RocksDB та FAISS.
- Автоматизація знеособлення вхідних документів (NER-маскування).
- Побудова оркестрованого контуру управління 600+ периферійними вузлами через K8s та Rancher за методологією GitOps.
- Реалізація відмовостійкості через CPU Fallback та Peer-to-Peer failover.

1.3. План-графік виконання етапів робіт

Розробка та впровадження модуля «Істотність» розділяється на такі етапи:

1. **Етап 1: Технічне проектування** — погодження UX/UI макетів, вибір цільових моделей ШІ, розробка схем даних. (1 місяць).
2. **Етап 2: Реалізація сховища даних та RAG** — налаштування RocksDB та індексів FAISS, розробка API-інтеграції з Erlang/OTP. (1.5 місяці).
3. **Етап 3: Розробка NER та знеособлення** — реалізація локального мікросервісу розпізнавання сутностей та маскування. (1 місяць).
4. **Етап 4: Налаштування оркестрації K8s та GitOps** — побудова кластера K8s, інтеграція GPU плагінів, оновлень. (1.5 місяці).
5. **Етап 5: Державна експертиза КСЗІ** — аудит безпеки, перевірка контролів NIST SP 800-53 та отримання атестату. (2 місяці).

2. Відомості про робочий процес та умови експлуатації

2.1. Опис бізнес-процесів (BPMN / FSM)

Всі процеси керування та роботи ШІ-сервісів формалізуються у вигляді кінцевих автоматів (FSM) та виконуються рушієм процесів BPE:

2.1.1. Процес локального інференсу та RAG (ai_inference.erl)

Цей процес відповідає за обробку запиту користувача до великої мовної моделі з додаванням RAG-контексту:

1. **StartInference:** Користувач надсилає текстовий запит (промпт) або документ.
2. **AnonymizePrompt:** Фоновий процес NER-фільтрації знеособлює чутливі дані, замінюючи їх на токени.
3. **GenerateEmbeddings:** Створення векторного представлення (ембедінгу) запиту через модель `e5-large`.
4. **VectorSearch:** Пошук відповідних фрагментів документів у FAISS за косинусною близькістю.
5. **BuildRAGPrompt:** Формування підсумкового промпту для LLM, де знеособлений запит поєднується зі знайденим контекстом.
6. **ExecuteInference:** Надсилання промпту до локального сервера `llama.cpp` на GPU RTX 4070 Ti.
7. **DeanonymizeResponse:** Зворотне відновлення персональних даних у згенерованій відповіді на базі локальної RocksDB таблиці відповідностей.
8. **InferenceCompleted:** Завершення процесу, повернення відповіді користувачу.

2.1.2. Процес нічного оновлення моделей (ai_deployment.erl)

Забезпечує автоматизоване GitOps-оновлення ваг моделей:

1. **TriggerUpdate:** Спрацювання cron-задачі о 01:00.
2. **CheckRemoteCatalog:** Перевірка наявності нових версій моделей у центральному сховищі Scality RING.
3. **VerifyDiskSpace:** Перевірка вільного місця на NVMe накопичувачі локального сервера.
4. **PullModelWeights:** Асинхронне скачування GGUF файлу ваг із обмеженням швидкості.
5. **VerifyChecksum:** Перевірка цілісності файлу за MD5/SHA256 хешами.
6. **RolloutBlueGreen:** Оновлення поду в K8s зі зміною монтування на новий файл моделі.
7. **VerificationTest:** Виконання тестового інференсу, перевірка GPU-метрик.
8. **DeploymentCompleted:** Фіксація успішного оновлення.

2.2. Опис ролей та прав доступу (ABAC)

Доступ до управління периферійними вузлами ШІ та перегляду журналів інференсу обмежується на базі ABAC:

```
allow(User, Action, Node) ->
    UserRole = maps:get(role, User),
    UserRegion = maps:get(region, User),
    NodeRegion = maps:get(region, Node),

    IsCDTO = (UserRole == cdto_admin),
    MatchesRegion = (UserRegion == NodeRegion),

    case {IsCDTO, Action} of
        {true, _} -> true;
        {false, view_metrics} -> MatchesRegion;
        {false, query_inference} -> true;
        _ -> false
    end.
```

2.3. Специфікація апаратного забезпечення

В якості периферійного апаратного прискорювача використовуються споживчі GPU NVIDIA GeForce RTX 4070 Ti або RTX 4070 Ti Super. Детальні технічні параметри прискорювачів наведено в таблиці:

Параметр	**RTX 4070 Ti**	**RTX 4070 Ti Super**
Об'єм VRAM	12 GB GDDR6X	16 GB GDDR6X
Шина пам'яті	192-bit	256-bit
Пропускна здатність	504 GB/s	672 GB/s
CUDA-ядра	7680	8448
Тензорні ядра	240 (4-gen)	264 (4-gen)
Енергоспоживання	285 W	285 W

2.4. Умови експлуатації та системне середовище

Робочі станції ШІ-серверів експлуатуються в приміщеннях канцелярій чи комутаційних шафах установ. Завдяки Midi-Tower корпусу та активному повітряному охолодженню RTX 4070 Ti, додаткове кондиціонування не є обов'язковим. Системне середовище включає Ubuntu LTS 24.04, драйвери NVIDIA 550+, CUDA 12.x та Docker/K8s runtime.

2.5. Специфікація мовних моделей для локальної генерації

Для забезпечення локальної генерації текстів та семантичного аналізу документів в межах апаратних обмежень споживчих GPU (з обсягом відеопам'яті 12 або 16 ГБ VRAM) використовуються квантовані великі мовні моделі (LLM) класів Llama та Phi:

1. **Llama-3-8B-Instruct** (квантування Q4_K_M або Q8_0):
    - Використовується для загальних задач текстової генерації, підготовки проектів рішень та семантичного аналізу.

- При квантуванні Q4_K_M обсяг пам'яті для моделі разом із контекстом 8K складає близько 6.5 ГБ VRAM, що дозволяє паралельно запускати кілька сесій інференсу на GPU з 12 ГБ VRAM.
- При квантуванні Q8_0 обсяг пам'яті становить близько 9 ГБ VRAM (рекомендовано для прискорювачів з 16 ГБ VRAM).

2. **Phi-3-medium-14B** (квантування Q4_K_M):
   - Використовується для складного логічного аналізу та перевірки фактів.
   - Потребує близько 10.5 ГБ VRAM для інференсу з контекстом 4K. Рекомендовано для прискорювачів з 16 ГБ VRAM.
3. **Phi-3-mini-4B** (квантування Q8_0):
   - Надлегка і швидка модель для задач експрес-класифікації та локального розпізнавання сутностей (NER).
   - Потребує близько 4.8 ГБ VRAM, забезпечуючи максимальну швидкість інференсу (понад 100 токенів/сек) на обох варіантах прискорювачів.

2.6. Архітектура локальної генерації та оптимізації VRAM

Локальний інференс та генерація текстів на периферійних вузлах реалізуються за допомогою спеціалізованого оптимізованого інференс-сервера `llama.cpp`, запущеного в ізольованому Docker-контейнері під управлінням K8s. Взаємодія з графічним прискорювачем здійснюється через драйвер NVIDIA CUDA 12.x з використанням обчислювальних ядер CUDA та Tensor Cores 4-го покоління.

2.6.1. Профіль розподілу та бюджетування VRAM (Memory Math)

Для забезпечення безперебійної роботи без виходу за межі пам'яті (Out of Memory - OOM) на споживчих картах розраховано детальний бюджет відеопам'яті:

1. **Сценарій 1: Робоча станція з NVIDIA RTX 4070 Ti (12 GB VRAM)**
   - *Цільова модель*: Llama-3-8B-Instruct (квантування Q4_K_M, вага файлу $\approx$ 4.8 GB).
   - *Статичне резервування під модель*: $\approx$ 5.2 GB VRAM (з урахуванням завантаження CUDA графів).
   - *KV-кеш (Context Window 8K)*: Об'єм пам'яті під KV-кеш для одного активного слота користувача при $N_{layers} = 32$, $N_{heads} = 8$, $D_{head} = 128$ та довжині контексту 8192 токенів розраховується як:

$$\text{Size}_{KV} = 2 \times N_{layers} \times N_{heads} \times D_{head} \times 2 \text{ bytes (FP16)} \times 8192 = 1.07 \text{ GB}$$

   - *Багатокористувацький ліміт*: Для підтримки 5 паралельних користувацьких сесій виділяється 5×1.07 GB $= 5.35$ GB VRAM.
   - *Загальний обсяг використання VRAM*: 5.2 GB(модель)$+5.35$ GB(5 сесій KV)$+1.0$ GB(системний оверхед/д 11.55 GB VRAM (запас $\approx$ 450 MB).
2. **Сценарій 2: Робоча станція з NVIDIA RTX 4070 Ti Super (16 GB VRAM)**
   - *Цільова модель*: Phi-3-medium-14B (квантування Q4_K_M, вага файлу $\approx$ 8.5 GB).
   - *Статичне резервування під модель*: $\approx$ 9.1 GB VRAM.

- *KV-кеш (Context Window 4K)*: Об'єм пам'яті під KV-кеш для одного користувача при контексті 4092 токенів становить $\approx$ 0.75 GB VRAM.
- *Багатокористувацький ліміт*: Для підтримки 8 паралельних сесій виділяється 8×0.75 GB $= 6.0$ GB VRAM.
- *Загальний обсяг використання VRAM*: 9.1 GB(модель)$+6.0$ GB(8 сесій KV)$+0.8$ GB(оверхед CUDA) $= 15.9$ GB VRAM.

2.6.2. Паралельне планування та оптимізація пропускної здатності

Для досягнення високої швидкості генерації та ефективного використання Tensor Cores застосовуються такі технологічні механізми:

- **Continuous Batching (Ітераційне пакетування)**: Інференс-сервер `llama.cpp` запускається з параметром `-cont-batching` (або `-cb`), що дозволяє обробляти запити в режимі реального часу. Нові запити додаються в конвеєр обробки на кожній ітерації декодування токенів без необхідності очікування завершення попередньої генерації.
- **PagedAttention (Сторінковий KV-кеш)**: Пам'ять KV-кешу розбивається на логічні сторінки (блоки токенів розміром 16 чи 32) та динамічно виділяється через таблицю сторінок, аналогічно віртуальній пам'яті в ОС. Це повністю запобігає фрагментації відеопам'яті та дозволяє оптимізувати простір під паралельні запити.
- **FlashAttention / SDPA**: Використання оптимізованих ядер обчислення уваги, що значно знижує кількість операцій читання/запису у глобальну пам'ять GPU та прискорює етап Prefill у 2.5–3 рази.

3. Інформаційне та програмне забезпечення

3.1. Інформаційне забезпечення (Схеми та Дані)

3.1.1. Визначення структур даних (Erlang Records)

Всі ключові сутності модуля описуються у вигляді рекорді Erlang:

```
-record(ai_node_status, {
    node_id,                    % UUID вузла (PK)
    hostname,                   % Текстове ім'я хоста
    ip_address,                 % IP-адреса вузла
    gpu_temp = 0,               % Температура GPU RTX 4070 Ti в градусах
    vram_used = 0,              % Зайнято відеопам'яті (MB)
    vram_free = 0,              % Вільно відеопам'яті (MB)
    power_limit_pct = 80,       % Power Limit у відсотках (80% за замовчуванням)
    status = offline,           % online | offline | degraded | cooling_down
    last_seen_at = 0            % Timestamp останнього пінгу
}).

-record(ai_model_config, {
    model_id,                   % UUID моделі (PK)
    name,                       % Назва моделі (напр. Llama-3-8B-Instruct)
    quant_type,                 % Тип квантування (q4_k_m | q8_0)
    file_path,                  % Шлях до GGUF файлу на NVMe
    checksum_sha256,            % Контрольна сума файлу
    context_window = 8192,      % Максимальний контекст (токенів)
    is_active = false,          % Чи завантажена модель у пам'ять GPU
    updated_at = 0
}).

-record(ai_anonymization_dictionary, {
    dict_id,                    % UUID словника (PK)
    mapping_table = #{},        % Таблиця відповідностей {Token -> OriginalValue}
    created_at = 0,
    updated_at = 0
}).

-record(ai_inference_task, {
    task_id,                    % UUID задачі (PK)
    user_id,                    % UUID користувача, що надіслав запит
    node_id,                    % UUID обчислювального вузла, який виконав запит
    prompt_hash,                % SHA256 хеш промпту
    response_hash,              % SHA256 хеш згенерованої відповіді
    tokens_generated = 0,       % Кількість згенерованих токенів
    time_taken_ms = 0,          % Час виконання запиту в мілісекундах
    status = pending,           % pending | processing | completed | failed
    created_at = 0
}).
```

```
-record(ai_rag_index_state, {
    index_id,                   % UUID індексу (PK)
    num_vectors = 0,            % Кількість векторів
    file_path,                  % Шлях до FAISS HNSW індексу
    last_sync_at = 0,           % Час останньої синхронізації з Scality RING
    status = active             % active | syncing | stale
}).
```

3.2. Програмне забезпечення (Реалізація)

3.2.1. ai_inference.erl (DSL процесу RAG інференсу)

```
-module(ai_inference).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'AI Inference and RAG Workflow',
        beginEvent = 'StartInference',
        endEvent = 'InferenceCompleted',
        tasks = [
            #beginEvent{name='StartInference'},
            #serviceTask{name='AnonymizePrompt', module=?MODULE},
            #serviceTask{name='GenerateEmbeddings', module=?MODULE},
            #serviceTask{name='VectorSearch', module=?MODULE},
            #serviceTask{name='BuildRAGPrompt', module=?MODULE},
            #serviceTask{name='ExecuteInference', module=?MODULE},
            #serviceTask{name='DeanonymizeResponse', module=?MODULE},
            #endEvent{name='InferenceCompleted'}
        ],
        flows = [
            #sequenceFlow{name='1', source='StartInference', target='AnonymizePrompt'},
            #sequenceFlow{name='2', source='AnonymizePrompt', target='GenerateEmbeddings'},
            #sequenceFlow{name='3', source='GenerateEmbeddings', target='VectorSearch'},
            #sequenceFlow{name='4', source='VectorSearch', target='BuildRAGPrompt'},
            #sequenceFlow{name='5', source='BuildRAGPrompt', target='ExecuteInference'},
            #sequenceFlow{name='6', source='ExecuteInference', target='DeanonymizeResponse'},
            #sequenceFlow{name='7', source='DeanonymizeResponse', target='InferenceCompleted'}
        ],
        roles = [operator, system]
    }.

action({serviceTask, 'AnonymizePrompt'}, Proc) ->
    % Виклик локального мікросервісу знеособлення NER
    {reply, Proc};
action({serviceTask, 'VectorSearch'}, Proc) ->
    % Пошук у локальному індексі FAISS
    {reply, Proc};
action({serviceTask, 'ExecuteInference'}, Proc) ->
    % Інференс на GPU RTX 4070 Ti через llama.cpp
    {reply, Proc};
action(_, Proc) -> {reply, Proc}.
```

3.2.2. ai_validator.erl (Модуль валідації ШІ вузла)

```
-module(ai_validator).
-export([validate_model_hash/2, check_gpu_temperature/1]).

validate_model_hash(FilePath, ExpectedHash) ->
    case file:read_file(FilePath) of
        {ok, Binary} ->
            Calculated = crypto:hash(sha256, Binary),
            case Calculated of
                ExpectedHash -> ok;
                _ -> {error, checksum_mismatch}
            end;
        _ -> {error, file_not_found}
    end.

check_gpu_temperature(GpuTemp) ->
    MaxAllowed = 80,
    if
        GpuTemp > MaxAllowed -> {error, overheating};
        true -> ok
    end.
```

4. Вимоги до засобу за ISO/IEC/IEEE 29148

4.1. Функціональні вимоги

Модуль «Істотність» повинен виконувати аналіз завантажених документів, генерувати семантичні ембедінги, проводити знеособлення даних, здійснювати пошук контексту у RocksDB/FAISS та забезпечувати інференс мовної моделі.

4.2. Вимоги до інтерфейсів та дизайну

- Веб-інтерфейс адміністратора вузла повинен базуватися на фреймворку N2O/NITRO.
- Панель моніторингу повинна виводити статус GPU (температура, VRAM) та кількість активних сесій.
- Дизайн інтерфейсів має повністю відповідати стилю ERP/1 (Sleek Dark Mode).

4.3. Вимоги до безпеки та захисту інформації

- Шифрування розділів NVMe накопичувачів на базі LUKS з інтеграцією апаратного модуля TPM 2.0.
- Ізоляція мережевого трафіку ШІ від загального контуру LAN за допомогою Network Policies в CNI-плагіні Cilium.
- Заборона передачі будь-яких даних за межі локальної фізичної мережі установи (Air-Gapped режим).

4.4. Вимоги до продуктивності та надійності

- Середній показник доступності (SLA) периферійного сервера не менше 99.9%.
- Час відповіді на RAG-пошук (векторизація + FAISS) не більше 250 мс.
- Максимальний час перемикання на CPU при виході з ладу GPU не більше 15 секунд.

4.5. Вимоги до логування та аудиту

- Логування кожного запиту: ідентифікатор користувача, час, хеш промпту, хеш відповіді, згенеровані токени.
- Журнал аудиту повинен бути захищений від змін та автоматично передаватися на центральний сервер логування в нічний час.

4.6. Адміністративні та юридичні вимоги

- Всі компоненти ПЗ поставляються з відкритими ліцензіями (MIT/Apache 2.0).

- Застосування споживчих GPU RTX 4070 Ti у судах та офісних приміщеннях не підпадає під заборону використання GeForce у ЦОД (Datacenters) згідно з EULA NVIDIA, оскільки вузли є робочими станціями (Workstations) або крайовими пристроями (Edge Devices).