Аналіз Бекенду та Бази Даних (Лаба 7)

Частина 1. Аналіз запитів до MongoDB

Мета: Перевірити ефективність роботи бази даних під час виконання основних сценаріїв (пошук фільму, перегляд деталей).

Інструменти:

- Вбудований профайлер MongoDB (db.setProfilingLevel(2)).
- Команда explain("executionStats") для аналізу плану виконання запитів.

Результати аналізу:

1. **Швидкість виконання:** Аналіз системного профілю (system.profile) показав, що абсолютна більшість запитів виконується за час **<1 мс** (відображається як 0 ms у логах). Це свідчить про відсутність затримок на рівні БД при поточному обсязі даних.
2. **Використання індексів:**
   - Для перевірки було проаналізовано запит пошуку фільму за movieId: db.movies.find({ movieId: "438631" }).
   - Результат explain() показав використання стадії IXSCAN (сканування індексу movieId_1).
   - **Ефективність:** totalKeysExamined: 1, totalDocsExamined: 1. Співвідношення 1:1 є ідеальним показником — база даних не переглядає жодного зайвого документа.
   - Повне сканування колекції (COLLSCAN) відсутнє для критичних запитів.

   Висновок по БД: База даних спроєктована оптимально. Індекси, створені автоматично через Mongoose (unique: true в схемах), забезпечують миттєвий доступ до даних.

Частина 2. Репорт по навантаженню Бекенду (Artillery + Clinic Doctor)

1. **Загальний результат тесту (Health Check)**

Висновок про здоров'я системи: Система працює. Вона здатна обробляти близько 86 запитів на секунду. Проте, 33.8% віртуальних користувачів (863 з 2550) не змогли завершити свій сценарій через помилки. У пікові моменти сервер не встигає згенерувати/повернути токен або відбуваються таймаути.

2. **Аналіз часу відповіді (Latency)**

Висновок про Latency: Хоча середній час відповіді чудовий, 1% запитів, які займають понад 100 мс, вказує на те, що система стикається з блокуваннями CPU (наприклад, bcrypt під час логіну) або затримками БД (запит без індексу).

Висновок по Artillery:

1. **Пропускна здатність (RPS):** Система стабільно витримує ~86 RPS.
2. **Функціональна проблема:** Існує висока частка (33.8%) збоїв на етапі автентифікації (401 Unauthorized).
3. **"Вузькі місця" (Bottlenecks):**
    - Низький показник Median (3 мс) вказує на ефективне використання кешування (Redis).
    - Високий показник p99 (106.7 мс) вказує на проблему з CPU-інтенсивними (хешування паролів) або DB-інтенсивними (повільні запити/відсутні індекси) операціями.

3. **Загальний стан (Alerts)**

Detected a potential I/O issue

- **Діагноз:** Clinic виявив, що значна частина часу була витрачена на операції вводу/виводу (I/O). Це може бути час очікування відповіді від бази даних (MongoDB), зовнішніх API або файлової системи.
- **Висновок:** Сервер не завантажений обчисленнями JavaScript, а заблокований очікуванням зовнішніх ресурсів.

2. **Графік CPU Usage % (Центральний процесор)**

- **Значення:** Графік показує рідкісні, але високі піки використання CPU (до 300%+), з довгими періодами низького використання (близько 0%).
- **Аналіз:**

    - **Високі піки (до 300%+):** Оскільки Node.js є однопотоковим, піки вище 100% (до 400% на 4-ядерному процесорі) свідчать про те, що застосунок використовує пул потоків (libuv thread pool) для інтенсивних операцій:
        - **Bcrypt:** Хешування паролів.
        - **I/O:** Операції з диском.
    - **Низька база (Near 0%):** Між піками CPU не використовується. Це підтверджує, що більшість часу Node.js просто чекає на завершення операцій I/O (відповіді від MongoDB/зовнішніх API).
- **Висновок:** Вузьке місце знаходиться у виконанні важких операцій у пулі потоків (Bcrypt/Hashing) або у затримках мережі/БД.

3. **Графік Memory Usage MB (Оперативна пам'ять)**

- **Значення:** Використання пам'яті (Heap Used та RSS) залишається відносно стабільним на рівні 20-40 МБ.
- **Аналіз:** Пам'ять не зростає, немає різких падінь.
- **Висновок:** У застосунку немає витоків пам'яті (memory leaks) або проблем із вивантаженням сміття (Garbage Collection).

4. **Графік Event Loop Delay ms (Затримка циклу подій)**

- **Значення:** Переважно низькі затримки (0-5 мс), з декількома помітними піками до 35 мс.
- **Аналіз:** Якщо затримка перевищує 50 мс, це свідчить про блокування циклу подій. 35 мс — це допустимо, але піки корелюють із піками CPU. Вони, ймовірно, викликані тим, що інтенсивні хеш-операції (Bcrypt) повністю займають пул потоків, через що цикл подій затримується.
- **Висновок:** Цикл подій не є серйозним вузьким місцем.

5. **Графік Active Handles (Активні хендли)**

- **Значення:** Кількість активних хендлів (мережеві з'єднання, таймери, файлові дескриптори) залишається стабільною на рівні 10-12.
- **Висновок:** Коректно. Це стандартна поведінка для застосунку, який підтримує постійну кількість активних з'єднань (наприклад, до бази даних).


```
--------------------------------
Summary report @ 19:05:52(+0200)
--------------------------------

errors.Failed capture or match: ..... 863
http.codes.200: ..... 5061
http.codes.201: ..... 1687
http.codes.401: ..... 863
http.downloaded_bytes: ..... 17667449
http.request_rate: ..... 86/sec
http.requests: ..... 7611
http.response_time:
  min: ..... 0
  max: ..... 395
  mean: ..... 20
  median: ..... 3
  p95: ..... 71.5
  p99: ..... 106.7
http.response_time.2xx:
  min: ..... 0
  max: ..... 395
  mean: ..... 22.2
  median: ..... 3
  p95: ..... 71.5
  p99: ..... 106.7
http.response_time.4xx:
  min: ..... 0
  max: ..... 158
  mean: ..... 3.2
  median: ..... 2
  p95: ..... 8.9
  p99: ..... 24.8
http.responses: ..... 7611
vusers.completed: ..... 1687
vusers.created: ..... 2550
vusers.created_by_name.User Workflow (Login, Search, Create Post): ..... 2550
vusers.failed: ..... 863
vusers.session_length:
  min: ..... 64.7
  max: ..... 780
  mean: ..... 93.7
  median: ..... 74.4
  p95: ..... 162.4
  p99: ..... 262.5
```

Приклад запиту до БД на знаходженя фільма

```
movie-aggregator-db> db.movies.find({ movieId: "438631" }).explain("executionStats")
```

```
movie-aggregator-db> db.movies.find({ movieId: "438631"
     }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'movie-aggregator-db.movies',
    indexFilterSet: false,
    parsedQuery: { movieId: { '$eq': '438631' } },
    queryHash: '3C352FFA',
    planCacheKey: '0CC6DA86',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN', ---------------------- БД використовує індекс
        keyPattern: { movieId: 1 },
        indexName: 'movieId_1',
        isMultiKey: false,
        multiKeyPaths: { movieId: [] },
        isUnique: true,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { movieId: [ '["438631", "438631"]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 20,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
      stage: 'FETCH',
      nReturned: 1,
      executionTimeMillisEstimate: 11,
```

```
    works: 2,
    advanced: 1,
    needTime: 0,
    needYield: 0,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    docsExamined: 1,
    alreadyHasObj: 0,
    inputStage: {
      stage: 'IXSCAN',
      nReturned: 1,
      executionTimeMillisEstimate: 11,
      works: 2,
      advanced: 1,
      needTime: 0,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      keyPattern: { movieId: 1 },
      indexName: 'movieId_1',
      isMultiKey: false,
      multiKeyPaths: { movieId: [] },
      isUnique: true,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { movieId: [ '["438631", "438631"]' ] },
      keysExamined: 1,
      seeks: 1,
      dupsTested: 0,
      dupsDropped: 0
    }
  }
},
command: {
  find: 'movies',
  filter: { movieId: '438631' },
  '$db': 'movie-aggregator-db'
},
serverInfo: {
  host: 'fe0caba74d58',
```

```
    port: 27017,
    version: '6.0.26',
    gitVersion: '0c4ec4b6005f75582ce208fc800f09f561b6c2e8'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1
}
```

Приклад запиту при тестуванні БД

```
movie-aggregator-db> db.system.profile.find({ op: { $ne: 'command' } }).sort({ millis: -1 }).limit(5).pretty()
```

```
{
    op: 'query',
    ns: 'movie-aggregator-db.users',
    command: {
      find: 'users',
      filter: { email: 'user1@test.com' },
      limit: 1,
      singleBatch: true,
      lsid: { id: UUID('1a925067-aaac-4c98-9553-c2cc34a401e9') },
      '$db': 'movie-aggregator-db'
    },
    keysExamined: 0,
    docsExamined: 0,
    nBatches: 1,
    cursorExhausted: true,
    numYield: 0,
    nreturned: 0,
    queryHash: 'DFF5CD1D',
    planCacheKey: 'D871B341',
    queryFramework: 'classic',
    locks: {
      FeatureCompatibilityVersion: { acquireCount: { r: Long('1') } },
      Global: { acquireCount: { r: Long('1') } },
```

```
    Mutex: { acquireCount: { r: Long('1') } }
  },
  flowControl: {},
  responseLength: 114,
  protocol: 'op_msg',
  millis: 0, ------------------------------ Час обробки запиту 1мс<
  planSummary: 'IXSCAN { email: 1 }',
  planningTimeMicros: 109,
  execStats: {
    stage: 'LIMIT',
    nReturned: 0,
    executionTimeMillisEstimate: 0,
    works: 1,
    advanced: 0,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    limitAmount: 1,
    inputStage: {
      stage: 'FETCH',
      nReturned: 0,
      executionTimeMillisEstimate: 0,
      works: 1,
      advanced: 0,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 0,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 0,
        executionTimeMillisEstimate: 0,
        works: 1,
        advanced: 0,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
```

```
        keyPattern: { email: 1 },
        indexName: 'email_1',
        isMultiKey: false,
        multiKeyPaths: { email: [] },
        isUnique: true,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { email: [ '["user1@test.com",
   "user1@test.com"]' ] },
        keysExamined: 0,
        seeks: 1,
        dupsTested: 0,
        dupsDropped: 0
      }
    }
  },
  ts: ISODate('2025-12-13T17:10:10.169Z'),
  client: '172.18.0.3',
  allUsers: [],
  user: ''
}
```