# Performance Optimisation Report

**PERFORMANCE OPTIMIZATION REPORT**

**FOR**

**COMBINED SYSTEM DEVELOPMENT SERVICES**

**FOR**

**LICENSING SELF-CERTIFICATION PORTAL**

**OF**

**BUILDINGS DEPARTMENT**

**Version 0.1**

**Jan 2025**

? The Government of the Hong Kong Special Administrative Region

The contents of this document remain the property of and may not be reproduced in whole or in part without the express permission of the Government of the HKSAR.

| Amendment History |  |  |  |  |
| --- | --- | --- | --- | --- |
| Change Number | Revision Description | Pages Affected | Revision/ Version Number | Date |
| 1 | 1st draft | All | 0.1 | 16/01/2025 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**TABLE OF CONTENTS**

[**1. Introduction 5**](#introduction)

[1.1 Goal of Performance Optimization 6](#goal-of-performance-optimization)

[1.1.1 Server Loading 6](#server-loading)

[1.1.2 Bandwidth Usage 6](#bandwidth-usage)

[1.1.3 Better user experience 6](#better-user-experience)

[1.2 Performance Optimisation Actions 7](#performance-optimisation-actions)

[1.3 Storage Allocation 8](#storage-allocation)

[1.4 Required Response Time 9](#required-response-time)

[1.4.1 Online Transaction 9](#online-transaction)

[1.4.2 Online Reports 10](#online-reports)

[**2. Critical online transition timing 11**](#critical-online-transition-timing)

[**3. Critical Batch Cycle Timing 12**](#critical-batch-cycle-timing)

[**4. Optimization changes 13**](#optimization-changes)

[4.1 Optimization Actions 14](#optimization-actions)

[4.1.1 Create Stored Procedures (If Applicable) 14](#create-stored-procedures-if-applicable)

[4.1.2 Create Indexes 14](#create-indexes)

## 1. Introduction

This Performance Optimisation Report outlines strategies and considerations for enhancing the performance of the Licensing Self-Certification Portal system for the Buildings Department. The optimization efforts are primarily focused on improving the responsiveness of online transactions and ensuring efficient resource utilization.

This report is informed by an analysis of the system's database, named "bd". As of 2025/3/4 ??10:10:39, the database statistics are as follows:

* **Database Size:** 88.10 MB
* **Collections:** 12
* **Total Documents:** 1,278,983
* **Total Data Size:** 371.24 MB

The database comprises 12 collections, with sysfilerefs and adrblkfilerefs being the largest in terms of document count and data size. Optimizing queries and operations on these collections will be crucial for overall system performance.

The performance optimization of the system could be classified into optimization of Online Transaction.

## 1.1 Goal of Performance Optimization

The main goal of performance optimization is to improve the response time of the system to users. In order to achieve better response time, the program implementation should take the following areas into consideration.

### 1.1.1 Server Loading

The Capacity of Server Loading is a fixed variable of a system. An increase in server loading would increase the response time of the system. Server loading is affected by

1. the number of system users; and
2. the efficiency of the programming code.

When the number of system users increases, the server loading increases and hence there is less resource for running programming code for each system user. The response of the program would be slower as there is less server resource for the program to utilize.

On the other hand, if the programming code could use fewer server resources, there is less server loading and hence the server could respond to users quicker and serve more users simultaneously.

Therefore, response time could be reduced if the programming code could be optimized to use fewer server resources.

### 1.1.2 Bandwidth Usage

Capacity of Bandwidth Usage is a fixed variable of a system. An increase in bandwidth usage would decrease the response time of the system. Bandwidth loading is affected by

1. the number of system users; and
2. the size of transmitting resource.

When the number of system users increases, the bandwidth usage increases and hence there is less bandwidth resource for transmitting data to each system user. The response time would be slower as there is less bandwidth for each user to utilize.

On the other hand, if the size of transmitting data is smaller, there is less bandwidth usage and hence the fixed network bandwidth could serve more users simultaneously.

Therefore, response time could be reduced if the size of the transmitting resource could be optimized to use less server resources.

### 1.1.3 Better User Experience

This application will be used by the public, in order words, it represents the department. If it performs well, the department can take credit from it, the image of the department may be affected if this application performs bad or causes other problems, for example very slow or no response.

## 1.2 Performance Optimisation Actions

To serve more users simultaneously with better response time, the system and the programs should be optimized to use the server loading and bandwidth resource more efficiently. The followings are the possible measures that could be taken generally:

* Optimize the programming and query logic to reduce server loading burden.
* Optimize the page size and image size to reduce bandwidth burden.
* Improve resource retrieval speed by indexing and hashing.
* Cache frequently used resources.
* Pre-generate resource that required heavy instant server loading.
* Reduce waiting time of third-party services.
* Archive expired records to keep the size of active datastore minimal.

Based on the database analysis, potential areas for optimization include:

* **Indexing:** Given the large size of sysfilerefs and adrblkfilerefs collections, ensure appropriate indexes are in place for frequently queried fields such as sysFileRefId, adrBlkFileRefId, createdDt, and lastModifiedDt. Analyzing query patterns will help identify the most effective indexes.
* **Query Optimization:** Review queries targeting large collections to ensure they are efficient and utilize indexes effectively. Avoid full collection scans where possible.
* **Data Archiving:** Consider implementing a data archiving strategy for older records in large collections like sysfilerefs and adrblkfilerefs to reduce the active dataset size and improve query performance over time.
* **Code Efficiency:** Review backend code for potential performance bottlenecks, especially in areas that interact with the database.

## 1.3 Storage Allocation

The storage of the database data will be stored as following:

i. System data ? Store in the ?Database? server in the Integrated system.

ii. Textual data ? Store in the ?Database? server in the Integrated system.

All the required system and textual data will be logically stored within the database system.

The database "bd" currently occupies 88.10 MB of disk space with a total data size of 371.24 MB across all collections. The collections sysfilerefs (204.70 MB) and adrblkfilerefs (154.89 MB) account for the majority of the data storage. Future storage planning should consider the growth rate of these collections, particularly if file references are expected to increase significantly.

## 1.4 Required Response Time

The required response time is defined according to 2 pre-defined categories. They are Online Transaction and Online Report. All of the programs and reports that require to interact and provide immediate response to users are categorized. The categorized programs and reports should respond to the user within required response time. The required response time should be defined according to the complexity of the programs. The required response time and the complexity will be discussed in the coming section.

For those procedures or reports that could not able to optimize to have immediate response, some part of the program that take a lot of processing time will be swapped to batch job.

95% of all online functions should meet the committed response time.

The committed response time is affected by the network status of the site. The environment of testing site should meet an agreed network health level. The following criteria define the agreed network health level.

* Maximum number of Concurrent users is 100.
* Minimal bandwidth is 2Mb/s per testing machine.
* Maximum network round-trip latency (ping) to the Integrated system is 200ms.
* Remote testing site will have **50% mark-up** time to the committed response time.

### 1.4.1 Online Transaction

The programs in the following category are classified as online transaction:

* User Account Program
* Form and Record Management Program

Online transactions can be classified into the following groups:

[RY Note: Needs user input/ refer to Load Test data given by user]

| Transaction Complexity | Number of Concurrent Users |  |  |  |
| --- | --- | --- | --- | --- |
|  | 40 | 80 | 100 |  |
| Simple | < 1 sec | < 2 sec | < 3 sec |  |
| Medium | < 2 sec | < 3 sec | < 4 sec |  |
| Complex | < 2.5 sec | < 3.5 sec | < 5 sec |  |

Online transactions can also be classified as follows:

| Online transactions | Description |
| --- | --- |
| Online Update Transactions | Used to update records in LSCP, for example, create supervision plan |
| Online Enquiry Transactions | Used to retrieve records from LSCP, for example, filter site monitoring records |
| Full-text Search | Used to search for records with given key words, for example, search assigned TCP |

### 1.4.2 Online Reports

The programs in the following category are classified as online reports:

* XXXXXXXXXXX

As the report is a standalone internal system for BD, the measurement would only estimate when the concurrent users are not more than 20.

| On-line Reports | Committed Response Time |
| --- | --- |
| Report 1 | < TBD sec |
| Report 2 | < TBD sec |
| ... | < TBD sec |

# 2. Critical Online Transition Timing

The following matrix is the list of programs with the complexity and transaction type being marked.

Offline module helps users save information in local devices when Internet connectivity is not available. Once Internet connectivity is back, it will submit the data and store it in the database. The major difference between offline modules with other modules is that the offline module is asynchronous. It will not submit immediately but will wait until the Internet is available. For the asynchronous handling logic, we included it in the functional test. On the other hand, the data sending logic is no different with other API, thus we would not have a separate section for offline module in critical online transition timing.

# 3. Critical Batch Cycle Timing

The below are the list of batch programs. The cycle timing of the batch is identified in the tables below. They are different from modules in the last section whereas they will run a thousand times everyday but the modules in this section probably run once or twice daily. Moreover, most of the modules described in this section are scheduled jobs running in backend, normally, the fluctuation of performance in these items would not affect the end user. Except, the generate report module, has a higher impact, thus it is the only item required to optimize.

| **Program ID** | **Program Name** | **Online Web** | **Mobile** | **Update Batch** | **Require Optimization?** | **Cycle Timing** |
| --- | --- | --- | --- | --- | --- | --- |
| **BATCH PROGRAM** |  |  |  |  |  |  |
| BATCH-01 | Daily Data Archiving |  |  | X | Yes | Daily |
| BATCH-02 | Weekly Report Generation |  |  | X | Yes | Weekly |
| BATCH-03 | System Maintenance |  |  | X | No | Monthly |
| ... | ... |  |  |  |  |  |

# 4. Optimization changes

The optimization will **only** focus the performance on programs and reports.

## 4.1 Optimization Actions

### 4.1.1 Create Stored Procedures (If Applicable)

While the current database is MongoDB and does not directly use stored procedures in the traditional relational database sense, the principle of pre-compiled and optimized database operations is still relevant. For frequently executed complex queries or aggregations, consider using:

* **Optimized Aggregation Pipelines:** MongoDB's aggregation framework allows for complex data transformations and analysis within the database. Optimizing these pipelines can significantly improve performance for reporting and data processing tasks.
* **Server-Side JavaScript Functions (with caution):** For highly specific or complex logic, server-side JavaScript functions can be used within MongoDB. However, use this feature judiciously as it can introduce performance overhead if not carefully implemented.

### 4.1.2 Create Indexes

Indexes are crucial for efficient query performance in MongoDB. Based on the database schema analysis and common query patterns, the following indexing strategies are recommended:

| **Collection Name** | **LSCP Entity** | **Key Nature** | **Index Field(s)** | **Index Type** | **Rationale** |
| --- | --- | --- | --- | --- | --- |
| tasks | Task | Query | status, application, submissionCase, taskType, user | Single Field, Compound | Optimize task retrieval and filtering based on status, application, etc. |
| eminutes | eMinute | Query | eminuteId, efolio, submissionCase, from, to, status | Single Field, Compound | Optimize eMinute retrieval and filtering. |
| applications | Application | Query, Search | ApplicationNo, NameOfSchoolEN, assignedBS, assignedGR, createdAt | Single Field, Text, Compound | Optimize application retrieval, search, and filtering by assigned personnel. |
| notifications | Notification | Query | notificationType, task, user, createdAt | Single Field, Compound | Optimize notification retrieval and filtering. |
| bsblocks | BSBlock | Query | blockId, bdgis | Single Field, Compound | Optimize block retrieval. |
| cases | Case | Query, Search | Category, CaseOfficer, ReceivedDate, application, assignedBS, assignedGR, team | Single Field, Compound | Optimize case retrieval, filtering, and reporting. |
| oauthtokens | OAuthToken | Query | accessToken, refreshToken, user | Single Field, Unique | Ensure fast token lookup and uniqueness. |
| sysfilerefs | SysFileRef | Query | sysFileRefId, createdDt, lastModifiedDt, frefPref, frefSeq, frefYr | Single Field, Compound | Optimize file reference retrieval and filtering, especially by date and ref. |
| attachments | Attachment | Query | application, submissionCase, sysFileRefId, type, subType | Single Field, Compound | Optimize attachment retrieval and filtering by application, case, etc. |
| users | User | Query, Search | osdpLoginId, email, role, department, team, name | Single Field, Text, Compound | Optimize user retrieval, search, and filtering by role, department, etc. |
| adrblkfilerefs | ADRBlkFileRef | Query | adrBlkFileRefId, adrBlkId, sysFileRefId, createdDt, lastModifiedDt | Single Field, Compound | Optimize ADR block file reference retrieval and filtering. |

It is recommended to:

* **Analyze Query Patterns:** Conduct further analysis of application query patterns to refine and add to the suggested indexes.
* **Monitor Index Usage:** Regularly monitor index usage to ensure indexes are being effectively utilized and to identify any missing or redundant indexes.
* **Consider Compound Indexes:** For queries that frequently filter on multiple fields, create compound indexes to improve performance. The order of fields in compound indexes matters; place the most selective fields first.
* **Text Indexes for Search:** For fields requiring text search functionality (e.g., NameOfSchoolEN in applications, name and email in users), create text indexes.

<< End of Document>>