# Performance Optimisation 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.

## Distribution

| Copy No. | Holder |
| --- | --- |
| 1 | Buildings Department (BD) |
| 2 | Master Concept (Hong Kong) Limited (MC) |

## 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**](#introduction)

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

[1.1.1 Server Loading](#server-loading)

[1.1.2 Bandwidth Usage](#bandwidth-usage)

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

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

[1.3 Storage Allocation](#storage-allocation)

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

[1.4.1 Online Transaction](#online-transaction)

[1.4.2 Online Reports](#online-reports)

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

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

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

[4.1 Optimization Actions](#optimization-actions)

[4.1.1 Create stored procedures](#create-stored-procedures)

[4.1.2 Create clustered indexes](#create-clustered-indexes)

## 1. Introduction

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

## 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 logical stored in various filegroup of Microsoft SQL Server database. The following table shows the logic data storage in Microsoft SQL Server database.

| FileGroup | TableName | TableSize |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The filegroup growth size has set to meet the recommendation for below 256 MB for data files.

## 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:

| 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 |
| --- | --- |
|  |  |
|  |  |

# 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.

| **Module ID** | **Program ID** | **Program Name** | **Complexity (Simple / Medium / Complex)** | **Online Transaction Program** | **Report Program** | **Type of transactions** | **Mobile Apps** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **USER ACCOUNT PROGRAM** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# 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** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# 4. Optimization changes

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

## 4.1 Optimization Actions

### 4.1.1 Create stored procedures

All reports are created by stored procedures. Stored procedures are precompiled as opposed to the dynamic prepared statements that are compiled whenever your application code invokes a call. Once you execute a stored procedure, it remains in the cache, saving the execution time. In addition, the stored procedures are mostly using primary key for searching.

### 4.1.2 Create clustered indexes

When creating a primary key constraint, a unique clustered index on the column or columns is automatically created if a clustered index on the table does not already exist and you do not specify a unique non-clustered index. The primary key column cannot allow NULL values. In addition, when creating a unique constraint, a unique non-clustered index is created to enforce a unique constraint by default. When designing a clustered index, we have considered that the data types to be used as clustering keys. For instance, the primary keys are BIGINT data type which is the best choices as clustered index key.

| **Table ID** | **Table Name** | **LSCP Entity** | **Key Nature** | **Index Field** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

<< End of Document >>

## Appendix: Database Analysis and Potential Optimizations

This section provides a detailed analysis of the database schema, focusing on potential areas for performance optimization. The analysis is based on the provided database\_schema.md file.

### Database Overview

* **Database Name:** bd
* **Last Updated:** 2025/3/4 ??10:10:39
* **Database Size:** 88.10 MB
* **Collections:** 12
* **Total Documents:** 1278983
* **Total Data Size:** 371.24 MB

### Key Observations and Recommendations

1. **Large Collections:** The sysfilerefs and adrblkfilerefs collections are significantly larger than the others, both in terms of document count and data size. This suggests that queries against these collections could be performance bottlenecks.
   * **Recommendation:** Ensure appropriate indexes are in place for frequently queried fields in these collections. Consider data archiving strategies to reduce the size of these collections if older data is not frequently accessed.
2. **Collection Size vs. Document Count:** Some collections have a relatively small document count but a larger size (e.g., cases, applications). This could be due to large documents or inefficient data storage.
   * **Recommendation:** Analyze the structure of documents in these collections to identify potential areas for optimization. Consider using data compression techniques if applicable.
3. **Mixed Data Types:** Several fields have mixed data types (e.g., tasks.user, eminutes.from, eminutes.to, applications.AgeOfStudent, applications.ApplicantNameEN, applications.ApplicantTel, applications.DescriptionOfSchool, applications.EstimatedNoOfStudent, cases.ActualReplyDate, cases.ReceivedDate, cases.Nature). This can hinder indexing and query performance.
   * **Recommendation:** Enforce consistent data types for each field. If mixed types are unavoidable, ensure that queries handle them appropriately. Consider data cleaning and transformation to standardize data types.
4. **Null Values:** Many fields contain null values. While this is not inherently a problem, it can affect index performance and query optimization.
   * **Recommendation:** Understand the significance of null values in each field. Consider using sparse indexes if null values are prevalent.
5. **Submissions Collection:** The submissions collection is empty.
   * **Recommendation:** Investigate if this collection is still needed. If not, consider removing it to reduce database overhead. If it's needed, investigate why it's empty and ensure data is being written to it correctly.
6. **Indexing Strategy:** The provided schema analysis doesn't include information about existing indexes.
   * **Recommendation:** Conduct a thorough review of existing indexes. Identify missing indexes for frequently executed queries. Use the database's query analyzer to identify slow queries and suggest appropriate indexes.
7. **Application Collection:** The applications collection has a large number of fields. Many fields have low occurrences.
   * **Recommendation:** Review the application collection to determine if the data model can be simplified. Consider if some fields can be combined or removed.

### Detailed Collection Analysis and Recommendations

#### Collection: tasks

* **Potential Indexing Candidates:** application, submissionCase, taskType, team, user, status, createdAt
* **Data Type Consistency:** user field has mixed types (string, objectId). Standardize to objectId if possible.

#### Collection: eminutes

* **Potential Indexing Candidates:** efolio, eminuteId, from, status, subject, submissionCase, sysFileRefId, to, createdAt
* **Data Type Consistency:** from and to fields have mixed types (objectId, string). Standardize to objectId if possible.

#### Collection: submissions

* **Recommendation:** Investigate why this collection is empty.

#### Collection: applications

* **Potential Indexing Candidates:** ApplicationNo, ApplicationType, BlockID, NameOfSchoolCN, NameOfSchoolEN, assignedBS, assignedGR, assignedSBS, createdAt
* **Data Type Consistency:** AgeOfStudent, ApplicantNameEN, ApplicantTel, DescriptionOfSchool, EstimatedNoOfStudent have mixed types (string, null). Standardize if possible.

#### Collection: notifications

* **Potential Indexing Candidates:** eminute, notificationType, task, user, createdAt

#### Collection: bsblocks

* **Potential Indexing Candidates:** bdgis, blockId

#### Collection: cases

* **Potential Indexing Candidates:** Category, Nature, Referrer, Region, application, assignedBS, assignedGR, team, createdAt
* **Data Type Consistency:** ActualReplyDate, ReceivedDate have mixed types (date, null). Standardize if possible.

#### Collection: oauthtokens

* **Potential Indexing Candidates:** accessToken, refreshToken, user, accessTokenExpiresAt, refreshTokenExpiresAt

#### Collection: sysfilerefs

* **Potential Indexing Candidates:** sysFileRefId, createdDt, createdName, createdPost, createdSection, lastModifiedDt, lastModifiedName, lastModifiedPost, lastModifiedSection

#### Collection: attachments

* **Potential Indexing Candidates:** application, efolio, file, submissionCase, sysFileRefId, type, receivedDate, createdAt

#### Collection: users

* **Potential Indexing Candidates:** bdgis, email, name, osdpEmail, osdpLoginId, position, role, team, userType

#### Collection: adrblkfilerefs

* **Potential Indexing Candidates:** adrBlkFileRefId, adrBlkId, sysFileRefId, createdDt, createdName, createdPost, createdSection, lastModifiedDt, lastModifiedName, lastModifiedPost, lastModifiedSection

### General Database Optimizations

* **Regular Database Maintenance:** Implement regular database maintenance tasks such as index rebuilding and statistics updates.
* **Connection Pooling:** Use connection pooling to reduce the overhead of establishing database connections.
* **Query Optimization:** Use the database's query analyzer to identify and optimize slow-running queries.
* **Data Archiving:** Implement a data archiving strategy to move older, less frequently accessed data to a separate storage location.
* **Database Monitoring:** Implement database monitoring tools to track performance metrics and identify potential issues.

This report provides a starting point for performance optimization. A more in-depth analysis, including query profiling and index analysis, is recommended to identify specific areas for improvement.