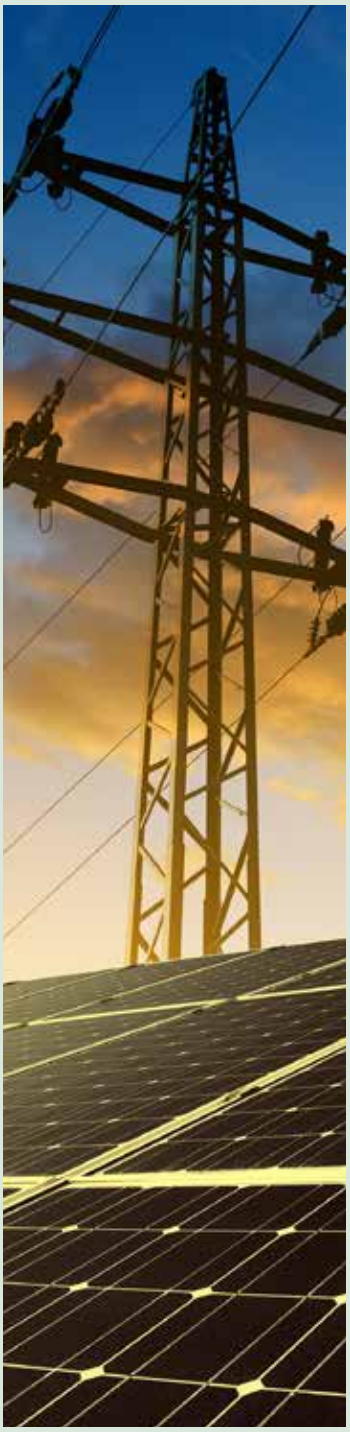
WHITE PAPER



USING BEST PRACTICES TO ACCELERATE APPLICATION PERFORMANCE IN MAINFRAMES

Abstract

An energy utility client was facing real-time infrastructure issues owing to the lag in its front-end customer application during month-end activities. The Infosys infrastructure team studied the client's environment and made recommendations to the application team based on their insights. This paper discusses the best practices Infosys used to optimize application performance. It also explains how we leveraged automation and a robust change management process to avert application outages and ensure business continuity.

Infosys®
Navigate your next

1. Introduction

An energy utility client was struggling with severe challenges in their customer care application. The application was based on mainframe with DB2 as its back-end and customer information control system (CICS) as the online transaction processing (OLTP) application.

2. Challenges

During month-end financial closing processes, the application faced heavy workloads resulting in high latency in the front-end. The root cause was the higher number of threads in the primary DB2 subsystem, leading to high CPU utilization. This resulted in high output of DB2 threads from the OLTP system whereby CPU utilization reached 99% for logical partition (LPAR) production.

Low CPU availability during these critical periods meant that regular activities, testing and processing in the LPAR were sidelined. This led to periodic crashes of business-critical applications residing on the mainframe and intermittent outages. Besides reducing overall performance, these infrastructure crises led to significant business disruption for the client.

3. Infosys approach

To uncover performance issues, Infosys set out to determine the inefficiencies in the DB2 store, the mainframe system and the business application such as high volume of adhoc data and inefficient job scheduling. Assuming that the problem was multi-faceted, we began by investigating these three levels in the mainframe environment. Capacity and performance analyses were conducted with the infrastructure as well as application teams.

For the purpose of clarity, our recommendations and measurements are classified as DB2 related and non-DB2 related.

A. DB2 recommendations and implementations

1. Queries consuming high MSUs

Infosys extracted the programs/packages that consume high MSUs using an OMEGAMON statistical analysis system (SAS) program and sent these to the application team for further analysis. This helped the application team identify programs requiring excessive CPU utilization.

Based on these findings, we implemented the DB2 plan to tune application programs and optimize the performance access path.

Table 1: A sample of the report that was sent to the application team

Parameter	Programs consuming high MSUs	
Observations	There are packages that consume high MSUs as shown below:	
	Programs	MSUs (high to low)
	KIDCAB0	63.47
	KILIFG4	63.44
	KIDSA41	45.44
	KIBRFF1	37.42
	KIDCAB2	32.35
Recommendations	• DB2 database team should generate a daily report on packages that consume high MSUs • Application teams should analyze the packages that consume high MSUs and identify possibilities to reduce this	

External Document © 2018 Infosys Limited

2. Monitoring the DB2 thread levels for criticality

The escalation in the number of DB2 threads to hundred or even thousand was an early warning of an impending crisis. However, monitoring the thread count was a manual and time-consuming task.

To address this challenge, the infrastructure team created an automated mechanism using the OMEGAMON-for-DB2 tool that monitored thread count and captured the number of DB2 threads in 5-10 minute intervals. REXX/JCL programs were used to design the automated solution.

When the thread count exceeds 100, the automated solution segregates thread details according to type and sends this to the application team. The application team immediately analyzes this data and takes necessary action before the issue escalates to a critical state. This solution has helped the team receive early warning alerts, enabling them to take quick and preventive action.

Further, the approach eliminates the need for manual intervention as the trigger is sent through the Tivoli Workstation (TWS). The thread monitoring job handles the following tasks:

- **Generate DB2 threads/plans that are being executed in real-time in DBE1 along with the count of threads**
- **Measure CPU use in DBE1DBM1 and DBE1MSTR address spaces**
- **Email the thread count, plan names and CPU details in a consolidated mail to the application owners.**

Table 2: Sample of threshold levels for the thread monitoring job

Threshold Levels		
Critical	**Major**	**Minor**
Add critical threshold level (percentage or number)	Add major threshold level (percentage or number)	Add minor threshold level (percentage or number)
• Number of threads > 400 • DBE1MSTR and DBE1DBM1 > 95% • SYSE CPU > 95%	• Number of threads > 200 • DBE1MSTR and DBE1DBM1 > 65% • SYSE CPU > 80%	• Number of threads > 100 • DBE1MSTR and DBE1DBM1> 50% • SYSE CPU > 65%

External Document © 2018 Infosys Limited

3. Handling intensive and costly queries

Application and end-users used the advance query tool (AQT) to submit queries. However, these queries were often large and costly, requiring a run-time ranging from several minutes to a few hours. In cases where a significant amount of processing or a critical problem is expected such as escalating thread numbers or high CPU cost, we prescribed the following guidelines:

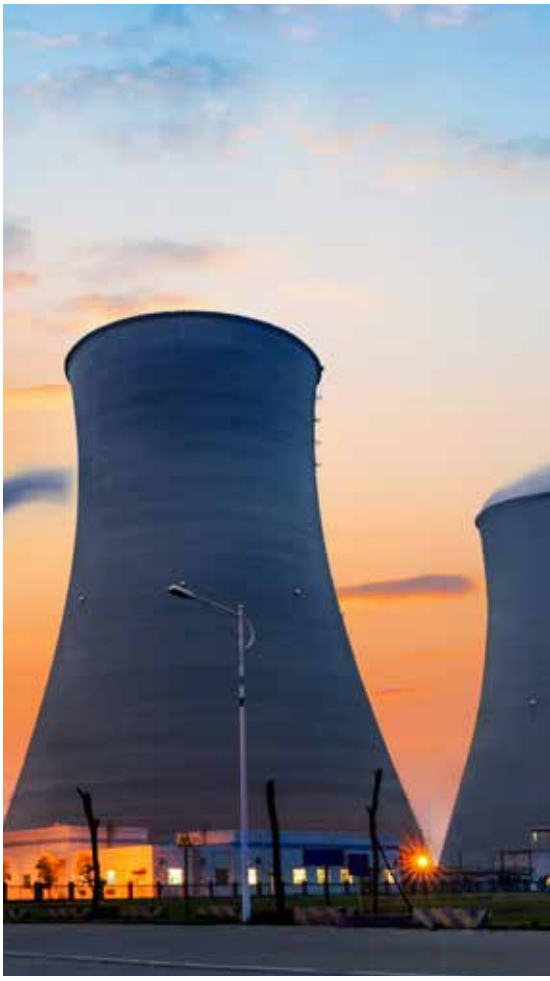
Parameter	AQT (adhoc) queries
Observations	AQT queries are run during business hours or while the batches are running
Recommendations	• Do not run AQT queries during critical periods like month-end • In unavoidable situations, run adhoc AQT queries under tight control • Use a scheduler to schedule static AQT queries during off-peak business hours • Use materialized query tables (MQT) for static queries for faster query response and to ensure query execution is not done on the base table

Based on these recommendations, MQTs were built into the application for large, complex and frequently used queries. This saved processing effort and time as the query was pre-built and data throughput was consistent. Since implementing this solution, the MQT has been frequently accessed during critical processing periods. Another challenge was that long-running queries in the AQT placed heavy demand on the processors. We suggested identifying such long-running queries from the DB2 accounting report that is shared with application and business users. Users were then encouraged to edit these queries over time to prevent long-running queries from entering the system.

External Document © 2018 Infosys Limited

How to identify long-running AQT queries from the DB2 accounting report

We can determine the threads that require higher CPU usage and run time using the DB2 accounting report. The DB2 SMF records (100,101,102) are routed to SQP.SMF.SYS*.** on a daily basis. The SMF dataset is a high-level qualifier (HLQ) for all the DB2 subsystems.



Then, the following job is run on the accounting report:

```
//ACCOUNT  JOB (TYYYYY,XXXX),
//       CLASS=6,MSGCLASS=P,MSGLEVEL=(1,1),REGION=0M,
//       NOTIFY=DB00SXA,
// TIME=(50)
/*JOBPARM S=SYSC
//*************************************************************
//AUTH    EXEC PGM=ACF2NCB
//*************************************************************
//*************************************************************
//*************************************************************
//STEP1    EXEC PGM=FPECMAIN
//STEPLIB  DD DSN=SQP.OMEGXE52.SMPE.TKANMOD,DISP=SHR
//INPUTDD  DD DSN=SQP.SMF.SYSE.G2660V00,DISP=SHR
//SYSOUT   DD SYSOUT=*
//DPMLOG   DD SYSOUT=*
//JOBSUMDD DD SYSOUT=*
//ACRPTDD  DD SYSOUT=*
//SYSIN    DD *
GLOBAL
FROM(06/01/15,00:01) TO(06/01/15,23:59)
 INCLUDE(SUBSYSTEMID(DBE1))
ACCOUNTING
REPORT
LAYOUT(SHORT)
EXEC
```

Through this report, we uncovered two AQT queries from users as described below:

```
1)          JOSEPT                      4     4   0.00   1.00        1.00     40.206210    2.50    0.00
 aqtv9_x6                     4   0.00   0.00   0.00    41.109528     5.944577   38.25      0
                         0   0.00   0.00   0.00     5.944803      872.7K 27474.50

   ------------------------------------------------------------------------------------------------------------
   |PROGRAM NAME      TYPE    #OCCURS  #ALLOCS  SQLSTMT  CL7 ELAP.TIME   CL7 CPU TIME  CL8 SUSP.TIME  CL8 SUSP|
   |SYSSH200          PACKAGE      4       4    2.50     40.206066      5.944481     28.571494     21.1K|
   ------------------------------------------------------------------------------------------------------------

   ------------------------------------------------------------------------------
   |REQUESTER       METH  #DDFS   TRANS  #ROLLBK  #COMMIT  SQLRECV  ROWSENT  CONVI|
   |::159.108.159.44  DRDA    4     N/A      0       4     2.50    51.00   0.00|
   ------------------------------------------------------------------------------
2) PA                          27     27   0.00   0.89        1.00     0.016038    0.56    0.00
 aqtv9_x6                     27   0.00   0.00   0.00     0.454371      0.004417    1.00      0
                         0  12.93   0.00   0.00     0.004805        93.19    3.63

   ------------------------------------------------------------------------------------------------------------
   |PROGRAM NAME      TYPE    #OCCURS  #ALLOCS  SQLSTMT  CL7 ELAP.TIME   CL7 CPU TIME  CL8 SUSP.TIME  CL8 SUSP|
   |SYSSH100          PACKAGE      27      27   15.63      0.015456      0.004375      0.005057      0.93|
   ------------------------------------------------------------------------------------------------------------

   ------------------------------------------------------------------------------
   |REQUESTER       METH  #DDFS   TRANS  #ROLLBK  #COMMIT  SQLRECV  ROWSENT  CONVI|
   |::159.108.153.39  DRDA    27     N/A      0      27    15.63  2722.52   0.00|
   ------------------------------------------------------------------------------
```

Similarly, we can determine the threads that require high CPU use from the accounting report using the ELAP time.

External Document © 2018 Infosys Limited

4. Using history tables and conducting routine maintenance activities on DB2 objects

The infrastructure team recommended using a history table and archiving the oldest rows from the DB2 application tables that are commonly used in the CICS online DB2 programs.

Parameter	Use of history tables
Observations	Some of the application tables are huge and contain millions of rows as shown below: Table name — Total number of rows KIRMOUSG — 166,215,805 KICHRG — 108,045,567 KICSL — 100,409,002 KIFEVDTL — 73,953,608 KICADL — 66,849,379
Recommendations	• Use history tables to store data that may not be used for periodic reporting • This will ensure that the base tables have limited number of rows, resulting in better I/O and improved performance This depends on how the data can be moved according to the application's requirement

5. Making changes to indexes in huge tables

Large financial tables were studied to identify and implement new indexes and rebound the relevant packages.

This improved queries where date ranges were queried against the large financial tables.

Some of the expensive queries were studied first and the underlying table spaces were investigated for additional indexes.

6. Monitoring deadlocks in DB2

SQL code -904 in DB2 indicates a deadlock. Previously, this was monitored retrospectively and the application team was informed only when a deadlock occurred in the production system.

Infosys devised a way of automating these -904 reports for the application team and this is in development.

Note: While the mainframe system recommendations followed are not listed here, they helped address all the critical issues faced by the client.

B. Mainframe system (non-DB2) recommendations and Implementations

1. Reclassification of AQT/user queries

Some of the AQT-based queries submitted by users were long running and non-business-critical, leading to heavy processor load during business hours. Thus, there was a need to appropriately classify such queries so that vital application jobs and functional business transactions could be processed quickly and on priority.To address this challenge, the mainframe infrastructure team along with business owners reprioritized critical AQT queries to run during critical business hours and days. The remaining AQT queries were assigned to a lower service class in the workload manager (WLM) for z/OS.

Parameter	WLM service class definition to reclassify AQT queries
Observation	AQT queries receive a higher service class priority when they execute
Recommendations	Assign a lower service class to AQT queries either permanently or during the critical period to avoid any impact on batch operations and web/IVR transactions

While REORG and RUNSTATS jobs were already running in the system, all maintenance jobs were reviewed and corrected. It was important to carefully study the critical financial tables that were widely used during crises to ensure that no data is missed during maintenance. This also helped accelerate processing.

Parameter	Routine maintenance activities
Recommendations	• Identify the most critical tables related to financial closing and monthly reporting • Ensure that the monthly REORG activities are diligently followed and that all critical tables are included in the activity • Remove unused temporary tables from critical table spaces • Remove obsolete database objects from the subsystem

2. Limiting test logical partitions (LPARs) during critical periods

As test LPARs contributed significantly to CPC utilization, we decided to limit their use to critical times while also limiting development and testing.

Proper scheduling during non-critical hours can improve CPU utilization. Thus, when critical problems emerge, test LPARs are removed or change moratoriums are announced in the test system, thereby allowing production systems to use the CPU effectively. In other words, resources from the test system are diverted to the production system. This was executed without business disruption by properly scheduling all the activities in the test system, thereby resolving performance issues.

Parameter	Use of test LPARs during critical periods
Observation	Test LPARs increase mainframe CPC to 100% during business hours
Recommendations	• Shut down the test LPAR for testing purposes during critical business hours like monthly close • Allocate the resources of the test LPAR to the production LPAR

External Document © 2018 Infosys Limited

3. Implementing changes to batch cycle

Usually, the application batch cycle runs at the start of business hours and requires core processing. However, CICS-DB2 programs enter the system at the same time through web and IVR channels causing heavy load on the CPU.

We optimized/rescheduled the batch cycle to run during the night, thereby eliminating load on the system when user queries and other online transactions demanded system usage.

By rescheduling time-consuming processing jobs at night or during weekends, we reduced the number of CICS and DB2 threads running simultaneously in the system. Additionally, the application team rescheduled and modified some of these jobs without impacting business applications, thereby decreasing the peak load on the CPU.

4. Validating program temporary fixes (PTFs)

Infosys reviewed and analyzed some of the PTFs for any issues since the last recommended service upgrade (RSU), particularly 'hold actions' in the DB2. The vendor was also contacted to determine any specific and important PTFs that may have been missed since the upgrade.

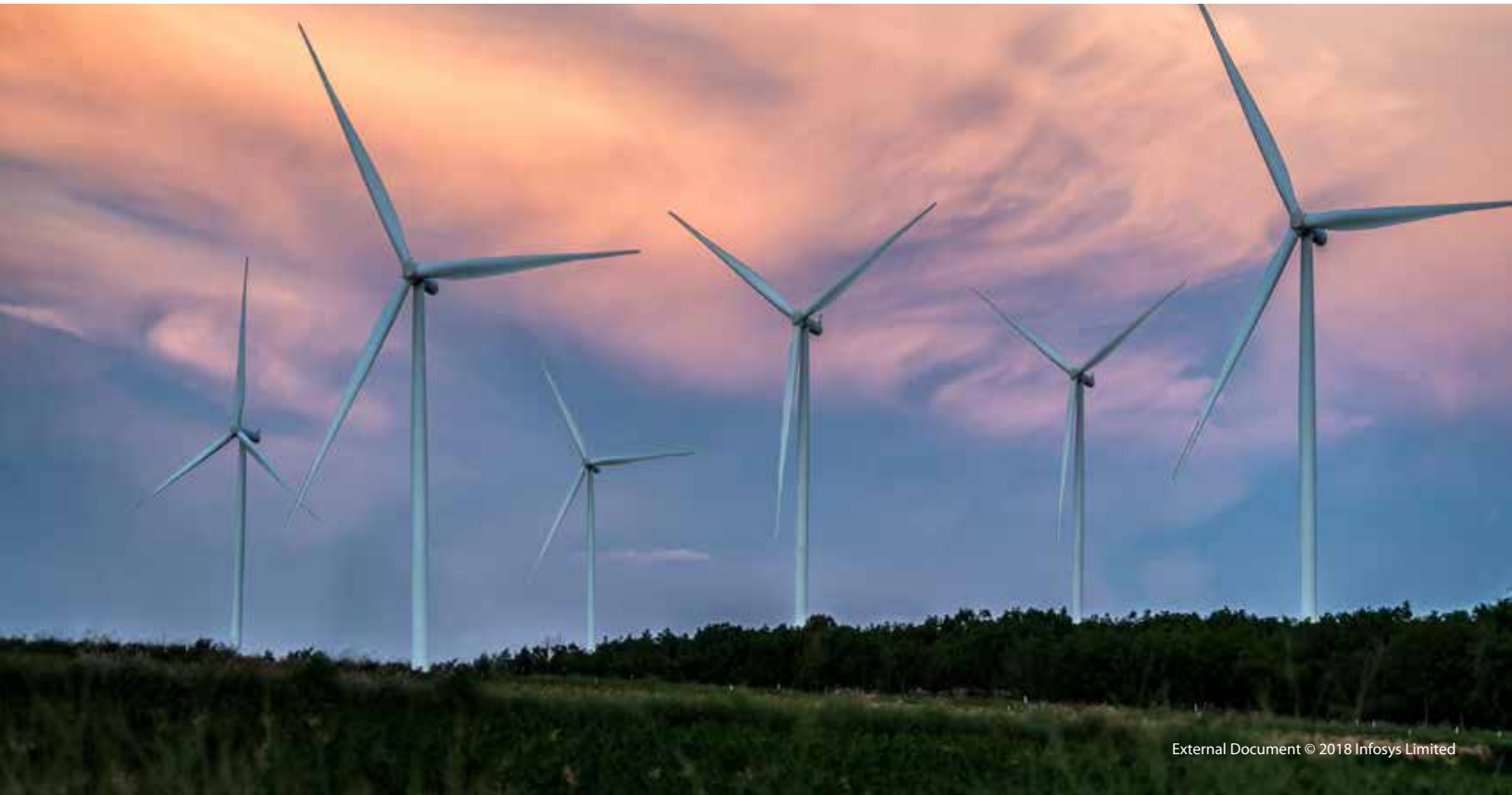
Parameter	Status of PTF actions after DB2 RSU1406 implementation
Observation	RSU1406 was implemented during March-April 2015
Recommendations	Analyze and validate all PTF actions after completing RSU1406 implementation Note: RSU – Recommended service upgrade

Results

Using an approach that leveraged best-practices, we helped the client overcome critical application performance challenges. The solution delivered results within 3 weeks. Further, it reduced the number of critical P1 and P2 tickets by 50% within 3 months. Enabling effective capacity and performance management was done in gradual phases across databases as well as systems, without changing the application or adding processors and additional CPU to the infrastructure. While there is a long-term plan to increase CPU capacity, we were able to consistently eliminate a majority of the issues in the mainframe environment by following best-practices and a proactive approach. Many of these approaches can be tailored to the local mainframe, however, some of them are applicable to nearly all mainframe environments. The Infosys team refrained from suggesting shifting the workloads to a different environment as this would have involved higher cost and business disruption. By adopting a realistic and proactive approach, we helped the client implement a cost-effective solution that solved critical problems, allowing them to resume business as usual.

Conclusion

To help the client handle excessive CPU utilization and business outages during month-end processing activities, Infosys conducted a careful analysis of the customer application, databases and mainframe. We leveraged proactive monitoring tools and automation scripts that trigger alerts for the database and systems, thereby eliminating performance bottlenecks before they occur. By leveraging best-practices for application performance management, Infosys helped the client save costs on processor upgrade/additional capacity. The solution improved application performance, increased application throughput to DB2 and reduced critical tickets by 50%.



External Document © 2018 Infosys Limited

About the Author

Venkatesh Pandian Rajagopalan is a Lead Consultant with 14 years of experience in application development and maintenance, database administration and mainframe performance, and capacity planning. Having worked as a major incident manager overseeing VMware servers, Linux, AIX, and database and mainframe tracks, he has wide experience in infrastructure projects. In addition to a bachelor's degree in computer science, Venkatesh is an IBM-certified IT specialist, IBM-certified database administrator and ITIL v3 foundation-certified.

Email: Venkatesh_r20@infosys.com

References

1. DB2 10 for z/OS Performance Topics- SG24-7942-00
2. Monitoring Performance from the IBM Tivoli OMEGAMON Enhanced 3270 User Interface- SH12-7056-00
3. System Programmer's Guide to: Workload Manager - SG246472
4. IBM Tivoli Workload Scheduler: User's Guide and Reference - SC32-1274-15

For more information, contact askus@infosys.com

Infosys®
Navigate your next

© 2018 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.