

# Approaches to Optimize Batch Processing on z/OS

Apply the latest z/OS features

Analyze bottlenecks

Evaluate critical path



Alex Louwe Kooijmans  
Elsie Ramos  
Jan van Cappelle  
Lydia Duijvestijn  
Tomohiko Kaneki  
Martin Packer





International Technical Support Organization

**Approaches to Optimize Batch Processing on z/OS**

October 2012

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (October 2012)**

This edition applies to all supported z/OS versions and releases.

This document created or updated on October 24, 2012.

**© Copyright International Business Machines Corporation 2012. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

|  |      |
|--|------|
| <b>Notices</b> .....   | v    |
| Trademarks .....   | vi   |
| <b>Preface</b> .....   | vii  |
| The team who wrote this paper .....                                  | vii  |
| Now you can become a published author, too! .....                    | viii |
| Comments welcome .....   | viii |
| Stay connected to IBM Redbooks .....                                 | ix   |
| <b>Chapter 1. Getting started</b> .....                              | 1    |
| 1.1 The initial business problem statement .....                     | 2    |
| 1.2 How to clarify the problem statement .....                       | 3    |
| 1.3 Process to formulate a good problem statement .....              | 3    |
| 1.4 How to create a good business case .....                         | 5    |
| 1.5 Analysis methodology .....                                       | 6    |
| 1.5.1 Initialization .....   | 6    |
| 1.5.2 Analysis .....   | 6    |
| 1.5.3 Implementation .....   | 10   |
| <b>Chapter 2. Analysis steps</b> .....                               | 13   |
| 2.1 Setting the technical strategy .....                             | 14   |
| 2.2 Understanding the batch landscape .....                          | 15   |
| 2.2.1 Identifying where batch runs and the available resources ..... | 15   |
| 2.2.2 Job naming conventions .....                                   | 15   |
| 2.2.3 Application level performance analysis .....                   | 17   |
| 2.2.4 Job-level understanding .....                                  | 18   |
| 2.2.5 Select jobs to tune .....                                      | 19   |
| 2.2.6 Supporting infrastructure .....                                | 19   |
| 2.3 Deriving detailed optimization actions .....                     | 20   |
| 2.3.1 Tuning long-running steps .....                                | 20   |
| 2.3.2 Splitting jobs .....   | 21   |
| 2.3.3 Removing inter-job dependencies .....                          | 23   |
| 2.4 Verification and planning .....                                  | 23   |
| <b>Chapter 3. Governance</b> .....                                   | 25   |
| 3.1 Governance defined .....   | 26   |
| 3.2 The Batch Design Authority .....                                 | 26   |
| 3.2.1 Scope .....  | 26   |
| 3.2.2 Tasks .....  | 26   |
| 3.2.3 Roles .....  | 26   |
| 3.3 Why you need governance .....                                    | 28   |
| 3.4 How to implement governance .....                                | 28   |
| 3.4.1 Appoint the Batch Design Authority .....                       | 30   |
| 3.4.2 Revise standards and guidelines .....                          | 30   |
| 3.4.3 Communicate standards and guidelines .....                     | 30   |
| 3.4.4 Set up a baseline .....  | 30   |
| 3.4.5 Monitor baseline .....   | 31   |
| 3.4.6 Start improvements .....                                       | 31   |
| 3.4.7 Control and quality assurance .....                            | 31   |

|   |    |
|---|----|
| <b>Chapter 4. Anti-patterns</b> . . . . .                                     | 33 |
| 4.1 Introduction to patterns and anti-patterns . . . . .                      | 34 |
| 4.1.1 Definition of a pattern . . . . .                                       | 34 |
| 4.1.2 Definition of an anti-pattern. . . . .                                  | 34 |
| 4.2 Performance anti-patterns. . . . .  | 34 |
| 4.2.1 Incorrect usage of a special resource . . . . .                         | 35 |
| 4.2.2 Incorrect usage of a start time . . . . .                               | 37 |
| 4.2.3 Invalid relationship . . . . .  | 39 |
| 4.2.4 Obsolete dummy jobs or dummy operations. . . . .                        | 42 |
| 4.2.5 Actual start time is not equal to end time of last predecessor. . . . . | 44 |
| 4.2.6 Workload Manager is not set up properly for batch . . . . .             | 45 |
| 4.2.7 Redundant copies of files . . . . .                                     | 48 |
| 4.2.8 Redundant image copies . . . . .  | 49 |
| 4.2.9 Batch prevented from effectively using sysplex-wide resources. . . . .  | 50 |
| <b>Chapter 5. Tools</b> . . . . .   | 53 |
| 5.1 Tivoli Workload Scheduler for z/OS . . . . .                              | 54 |
| 5.1.1 Operational efficiency . . . . .  | 54 |
| 5.1.2 Monitoring . . . . .  | 55 |
| 5.1.3 Forecasting . . . . .   | 56 |
| 5.1.4 Critical path analysis. . . . .   | 56 |
| 5.1.5 Tivoli Dynamic Workload Console . . . . .                               | 57 |
| 5.1.6 Automation . . . . .  | 59 |
| 5.1.7 Sysplex exploitation . . . . .  | 59 |
| 5.1.8 Charting . . . . .  | 60 |
| 5.1.9 Check a dependency loop . . . . .                                       | 61 |
| 5.1.10 Statistics analysis . . . . .  | 61 |
| 5.2 z/OS components and features. . . . .                                     | 62 |
| 5.2.1 Systems Management Facilities (SMF) . . . . .                           | 62 |
| 5.2.2 Step-termination exit. . . . .  | 63 |
| 5.2.3 Batch improvements in z/OS 1.13. . . . .                                | 64 |
| 5.3 Vendor tool - Source2VALUE . . . . .                                      | 64 |
| 5.3.1 Source2VALUE operation. . . . .   | 64 |
| 5.3.2 Source2VALUE output . . . . .   | 65 |
| 5.3.3 Source2VALUE for batch schedule performance optimization. . . . .       | 66 |
| <b>Related publications</b> . . . . .   | 71 |
| IBM Redbooks publications . . . . .   | 71 |
| Help from IBM . . . . .   | 71 |

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

|                           |   |              |
|---------------------------|---|--------------|
| BladeCenter®              | OMEGAMON®   | RMF™         |
| DB2®                      | Parallel Sysplex®   | System z®    |
| Global Business Services® | Rational®   | Tivoli®      |
| IBM®                      | Redbooks®   | z/OS®        |
| IMS™                      | Redpaper™   | zEnterprise® |
| MVS™                      | Redbooks (logo)  ® |              |
| NetView®                  | Resource Measurement Facility™  |              |

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

Batch performance optimization remains an important topic for many companies today, whether merging workloads, supporting growth, reducing cost, or extending the online day.

This IBM® Redpaper™ publication describes a general approach that can be used to optimize the batch window in a z/OS® environment. This paper outlines a structured methodology using anti-patterns and tools that can be followed to increase batch productivity.

## The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Alex Louwe Kooijmans** is a Senior Architect at the Financial Services Center of Excellence in IBM Systems and Technology Group. Prior to this position, he spent almost 10 years in the International Technical Support Organization, leading IBM Redbooks® projects, teaching workshops, and running technical events with a focus on using the IBM mainframe in new ways. Alex also worked as Client Technical Advisor to various banks in The Netherlands and performed various job roles in application development. His current focus is on modernizing core banking systems and the role of IBM's current mainframe technology.

**Elsie Ramos** is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. She has over 28 years of experience in IT, supporting various platforms including IBM System z® servers.

**Jan van Cappelle** is an IT Architect at IBM in the Netherlands. He has 27 years of experience in IBM mainframe environments. His areas of expertise include workload scheduling on the mainframe and multi-platform environments. He has knowledge of several workload scheduler products and has more than 20 years experience in standardizing and optimizing batch schedules.

**Lydia Duijvestijn** is a senior IT Architect and Performance Engineer in the Global Business Services® organization of IBM in the Netherlands. She has led a large number of customer engagements in the area of design for performance, performance testing, and performance troubleshooting. For 10 years she has taught the IBM Universal Method Framework classes for operational modelling (infrastructure design) and architecting for performance, in the BeNeLux region and abroad. She is a speaker at IBM internal and external conferences on topics related to IT architecture and performance. She is co-leader of the worldwide IBM Community of Practice for Performance and Capacity and led the IBM team that published the technical paper "Performance Implications of Cloud Computing". She has worked for several years with Dutch research institutes examining the quality-of-service and trustworthiness of composite IT services.

**Tomohiko Kaneki** is a Senior IT Specialist in IBM Japan. Kaneki-san has worked at IBM for over 20 years in the systems management area. His areas of expertise include IBM Tivoli® Workload Scheduler, System Automation, IBM NetView® and IBM OMEGAMON®. He is a regular speaker at the ASAP TWS users conference.

**Martin Packer** is a Consulting IT Specialist working for IBM's Worldwide Banking Center of Excellence. He has 26 years of mainframe performance experience, having consulted with

dozens of customers in that time. He has contributed to a large number of IBM Redbooks publications over the past 22 years, and has presented at many conferences. He built a number of tools for processing mainframe performance instrumentation, most notably in the areas of batch window reduction, IBM Parallel Sysplex® Performance, and IBM DB2® Tuning. He holds a Bachelor's Degree in Mathematics and Physics and a Master's Degree in Information Technology, both from University College London. Martin is a heavy user of social media, frequently blogging on mainframe performance topics and other Information Technology subjects.

Thanks to the following people for their contributions to this project:

Hans van Hal  
Wim van den Boogaard  
IBM Netherlands

Mark Taylor  
John Moore  
IBM UK

Coen Burki  
OMNEXT b.v., the Netherlands

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>





# Getting started

Before you start any project, you need to have a clear understanding of the starting points and the objectives. In a batch optimization project, this means identifying the underlying problems that must be resolved. In this chapter, we describe how to refine a vague business question or problem into a well-formulated problem statement. We also outline our methodology to analyze and tune batch processing.

## 1.1 The initial business problem statement

When we visit clients who are experiencing batch problems we often hear general or vague problem statements such as:

- ▶ “Between 10:00 a.m. and 4:00 p.m. we had bad online transaction response times.”
- ▶ “The time window to run my batch is too short.”
- ▶ “Processor utilization is too high.”

Or the client asks questions or makes a request for assistance such as:

- ▶ “When we merge company A with company B, will there still be enough time to run our batch jobs?”
- ▶ “Our batch schedule is so complex that we do not know anymore how our applications work. We are worried that when there are problems we will not be able to resolve them quickly.”
- ▶ “Can you help us reduce our batch processing costs by 20%?”
- ▶ “There is no specific department responsible for our batch schedule, every department has its own standards, and no one thinks about the overall corporate perspective and requirements; how can we standardize?”
- ▶ “We want to clean up our batch schedule but do not know where to start. Can you help?”

Often it is the client business representative that initially presents these problems or questions to an IBM representative. But neither one is a specialist who fully understands the problem; this is not unusual for many large businesses. The IT environment of today is very complex, there are many different types of machines, some with logical partitions, instead of one big computer for the entire company, and all these machines must be able to communicate and interact with each other.

Many businesses, especially large corporations, run thousands of applications, each with many jobs and job steps. Often there is a relationship between these jobs which contributes to the complexity of supporting and managing these applications. Batch processing is still a key part of the workloads running on the IBM mainframe, and runs alongside online applications, office and internet transactions. The performance of all these jobs and transactions is very important to the business. For many companies, internet and online business deals are critical to the business and batch processing continues to be a mission-critical component. Each part of the business also has its own business-critical endpoints that must be understood and addressed. For example, a bank reconciliation department has different critical endpoints from the securities department, which has to deal with stock exchanges all over the world.

All these factors add to the complexity of modern business problems. Due to this complexity, it is crucial that client problems and questions are well understood. It is important that the client business expert communicates with a batch subject matter expert (SME), using language and terms that they both understand, to formulate a clear initial business problem statement.

## 1.2 How to clarify the problem statement

The client business expert and the SME should work together to clarify the initial questions or problem statement. The SME can ensure that the problem is well understood and help to locate the required people and tools to work on the problem.

Consider each of the three initial problem statements posed previously. The following paragraphs demonstrate how they can be clarified and improved.

- ▶ “Between 10:00 a.m. and 4:00 p.m. we had bad online transaction response times.”

A better statement might be:

“Between 10:00 a.m. and 4:00 p.m. online transaction response times exceeded the service level agreement by 50%. This is causing 100 user complaints a day. We have identified that batch running during the online day is causing a processor capacity shortage.”

- ▶ “The time window to run my batch is too short.”

A better statement might be:

“We need to run our batch between 8:00 p.m. and 6:00 a.m. In particular, we have to get the replenishment application completed by 3:15 a.m. Today we finished around 2:30 a.m. but when we have a problem...”

- ▶ “The processor utilization is too high.”

A better statement might be:

“Batch is driving our processor capacity requirements (or our software bill because it drives the peak of our Rolling 4 Hour Average). We need to reduce this peak by 25% to help manage our costs.”

## 1.3 Process to formulate a good problem statement

The client business expert together with the SME can formulate a clear business problem statement using the steps depicted in Figure 1-1 on page 4 and described here:

1. IBM representative discusses problem with client

An IBM representative has the initial discussion with the client to obtain further details about the problem.

2. Client business expert speaks with the IBM SME

The client business expert and the IBM SME meet to review and discuss the problem together. Because the client business expert uses and is familiar with the application, he has an understanding of the problem and is able to explain it.

3. Correct SME?

It is possible that the SME thinks he is not the correct person because the specific problem is not in his area of expertise. For example, the SME has specialized in solving scheduling problems but the problem is more database related. In this case, a database SME is needed, and should be engaged to talk with the business expert.

4. SME understands problem?

The SME writes down the problem statement and during this process might have additional questions to ask the client. The SME speaks again with the client to obtain the necessary clarifications.

5. SME formulates clear problem statement

With a good understanding of the problem, the SME can formulate a clear business problem statement. The SME uses this problem statement to define a set of follow-on steps.

6. Client and IBM agree on further steps

Agreement on action and the further steps required to solve the problem.

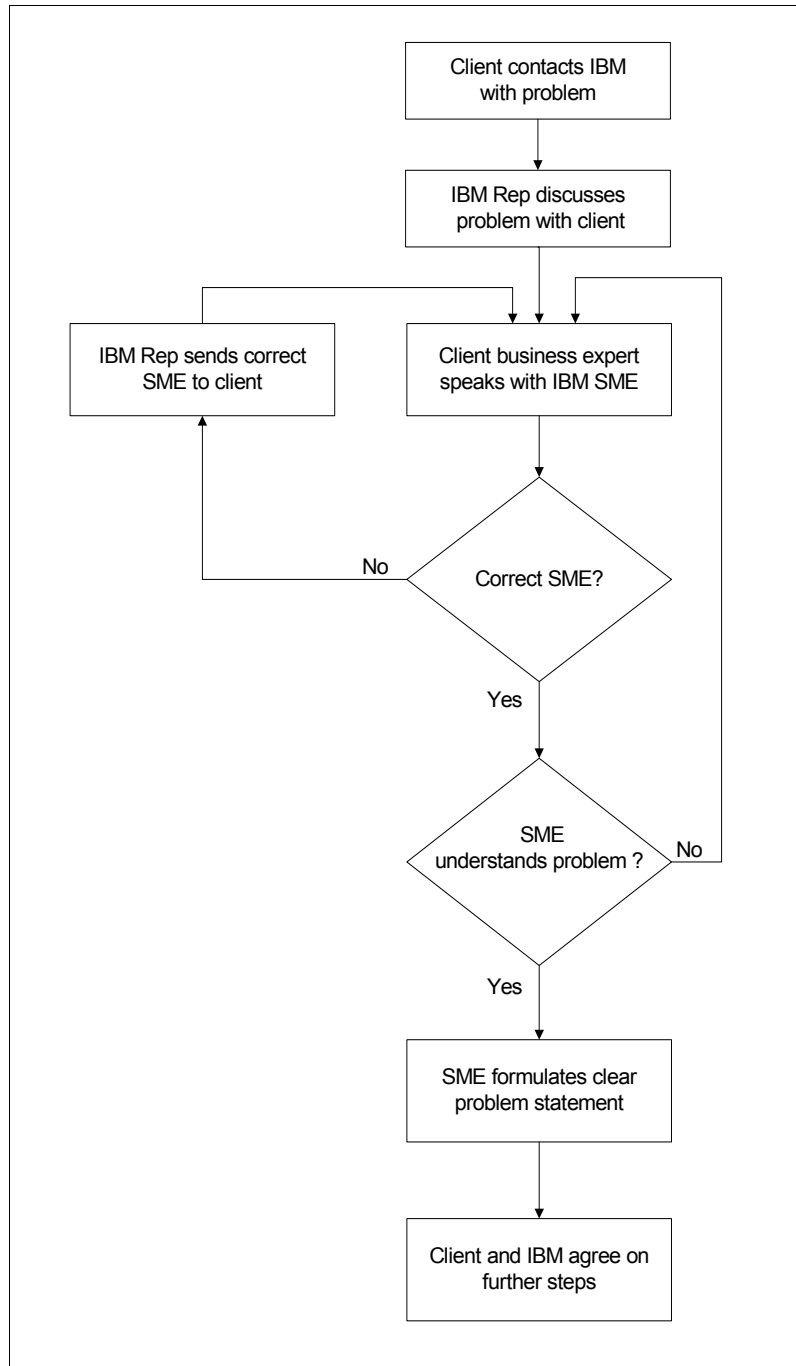


Figure 1-1 Process flow to create a clear problem statement



## 1.4 How to create a good business case

A good business case is essential for batch work because it is not always cheap or easy to implement. To resolve a problem always requires a good business case because the solution always has an associated cost. Conversely, the client benefits from the solution, so there is a balance between the cost and the benefit. In some cases, there is no quantifiable financial benefit. For example, new legislation might force you to make the change, or, if you do not make the change you are out of business.

When a problem in the batch window gets resolved, one or more of the following benefits might be realized:

- ▶ A reduction in processor usage in most cases results in lower costs, such as lower power consumption through reduced capacity requirements, and software license costs through a reduction in MSU usage.
- ▶ Eliminating batch processing bottlenecks reduces the risk of not meeting the service level agreements (SLAs).  
Failure to meet SLAs affects the business, and in certain cases may result in a penalty to be paid by the IT service provider.
- ▶ A clean schedule reduces the probability of problems and reduces possible costs as a result of not meeting SLAs.
- ▶ The simplification in the batch schedule results in lower maintenance costs, fewer skills needed by the operators, and less risk.
- ▶ Batch optimization might also result in lower disk space and network usage, thus resulting in lower costs.
- ▶ Overall, cleaning up the batch schedule might result in lower personnel costs because less work is required to maintain the schedule and less time is required to troubleshoot problems.

On the business side, any solutions implemented can have financial impacts, introduce risk, or require much time and effort. Consider the following examples:

- ▶ Removal of 30,000 obsolete jobs from a schedule of 300,000 jobs. When removing obsolete jobs, you also have to review and address any dependencies.
- ▶ Removal of changes to incorrect dependencies to 10% of the jobs in a schedule of 300,000 jobs. Making these changes might introduce significant risks.
- ▶ Installation of BatchPipes/MVS™ to create more parallelism. However, this requires you to acquire and install this program product.
- ▶ Hardware might need to be upgraded as part of the solution.

Because batch processing continues to be a fundamental, mission-critical component for most companies in every industry, the goal is to implement any required changes, but at a minimal cost. The overall costs for these changes can be reduced by using the tools described in Chapter 5, “Tools” on page 53. The tools described can help you correct and improve the batch schedule.

In summary, it is important for the benefits to exceed the costs. If not, the business case for a project to optimize your batch schedule is likely to fail.

## 1.5 Analysis methodology

Our methodology to analyze and tune batch processing is an abstract methodology that can be applied to batch processing on any platform. This platform-neutral methodology can also be applied to batch applications that span several operating environments, a characteristic of many installations today. However, in this publication we use steps specific to IBM z/OS to illustrate how to implement the methodology.

At a high level, this methodology consists of three phases: initialization, analysis, and implementation.

### 1.5.1 Initialization

Initialization is a key phase of any project, especially one as complex as a batch window reduction project. In this phase, you must define a clear scope for the project to maximize the likelihood of success. Aspects of this phase include:

- ▶ Investigate the problem  
Understand the business problem and how this translates into a technical problem.
- ▶ Governance  
Establish a formal process for making decisions and defining expectations on a continuous basis. Chapter 3, “Governance” on page 25 describes governance for our batch schedule project.
- ▶ Objectives  
Set realistic objectives and identify the benefits of the solution.
- ▶ Measurement  
Identify and define the measurement points.
- ▶ Data collection  
Collect data and information specifically needed for the batch schedule.
- ▶ Identify participants  
Define roles and agree upon the responsibilities of the participants.
- ▶ Project setup  
General project setup, such as documenting tasks, defining required activities, and so on.

**Note:** These phases are standard for any project, not specific to a batch project. Because an understanding of Project Management is assumed, this publication does not define these terms except as they specifically relate to batch.

### 1.5.2 Analysis

The analysis phase defines a set of coherent and focused actions that will be implemented in the final phase, described in 1.5.3, “Implementation” on page 10.

The analysis phase consists of three stages:

1. Setting the strategy
2. Understanding the batch landscape
3. Deriving detailed optimization actions

These stages are depicted in Figure 1-2. The stage to set the strategy and the stage to understand the batch landscape occur almost simultaneously. The stage to derive the optimization actions cannot start until the other two stages have completed, although it might cause some adjustments to them (as shown by the dashed lines).

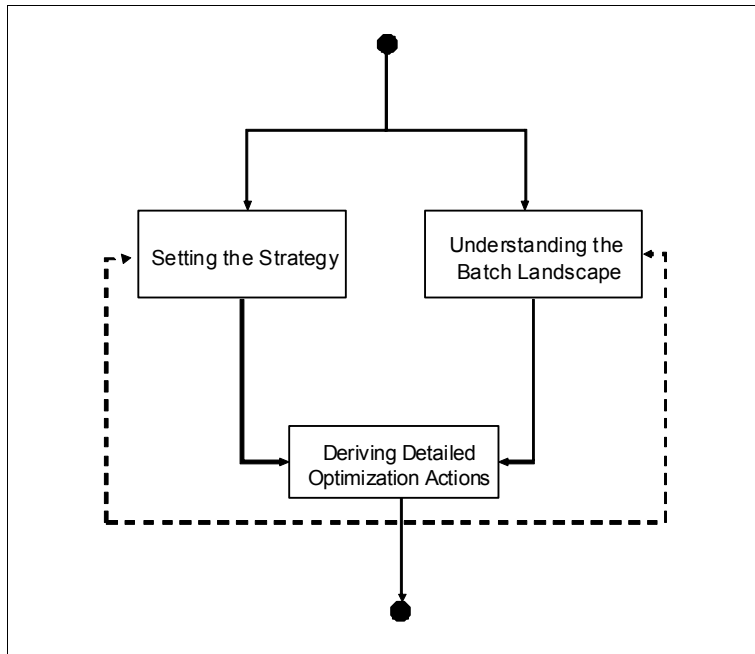


Figure 1-2 Overall analysis flow

## Setting the strategy

Setting the strategy is an important step in the overall process because it ensures that the rest of the project meets the business goals. In this stage, you can identify system conditions and any technical constraints. Do not generate optimization actions that conflict with each other.

The strategy is composed of seven major themes:

1. Ensuring the system is properly configured
2. Implementing Data In Memory (DIM)
3. Optimizing I/O
4. Increasing parallelism
5. Reducing the impact of failures
6. Increasing operational effectiveness
7. Improving application efficiency

*Batch Modernization on z/OS*, SG24-7779 contains z/OS-specific examples of each of these themes. The following are a few platform-neutral examples of these themes:

- ▶ Ensuring the system is properly configured: Verify your configuration and ensure that there are enough processor cycles.
- ▶ Implementing Data In Memory (DIM): Buffering data.
- ▶ Optimizing I/O: Plan and install disks that are fast enough and use them optimally.
- ▶ Increasing parallelism: Run more things at a time than previously.

- ▶ Reducing the impact of failures: Fix unreliable programs and other errors to reduce failures.
- ▶ Increasing operational effectiveness: Automate operational tasks to run more effectively.
- ▶ Improving application efficiency: Tune the application program code to increase efficiency.

These approaches are at a high level, but in a batch optimization project they can serve as a useful overall guide. Individual optimization actions can fit into this approach. In an actual optimization project, you create the strategy by combining these seven strategies or themes appropriately.

### ***Appropriate strategy - using DIM as an example***

We use Data In Memory (DIM) as an example of an appropriate batch strategy. This is only a brief general discussion of DIM.

DIM is an appropriate strategy if all of the following criteria are true:

- ▶ There are spare processor cycles to handle the faster consumption caused by the reduction of I/O delays.
- ▶ There is spare memory that can be used to keep the data in memory.
- ▶ There is I/O time that can be reduced.
- ▶ There are suitable buffering techniques for batch work where elapsed time is important.

These criteria are only met for a subset of the batch workload, certain systems, jobs, files, and databases. The purpose of much of the effort described in “Deriving detailed optimization actions” on page 9 is to find and identify these cases. This stage might also determine that there are no suitable cases.

With some DIM techniques, memory requirements decrease rather than increase because the task occupies memory for less time.

**Note:** In this example, the business problem is assumed to be elapsed time. If the problem is to reduce processor cycles, DIM is unlikely to be an appropriate strategy.

### **Understanding the batch landscape**

Most batch environments are extremely complex, typically with tens or hundreds of thousands of jobs, applications, files, and database objects. Complexity is the biggest obstacle to managing batch. Several tools are available that can help manage this complexity; they are described in Chapter 5, “Tools” on page 53.

Due to this complexity, it can be difficult to gain a good understanding of a batch landscape. This second stage is necessary to determine what is significant and identify what needs to be optimized.

The organization needs to fully understand their batch processing, so a review of the batch is necessary. Some examples of items that must be reviewed and verified are as follows:

- ▶ Naming conventions must be documented and fully enforced.
- ▶ Job functions must be well understood.
- ▶ Application ownership and boundaries must be clear.

Figure 1-3 on page 9 shows a case where the boundary of an application is unclear. In this example, Job PAX270D is part of Application AX, which itself is a part of Application A. But job PAX270D is also a boundary job for Application B.

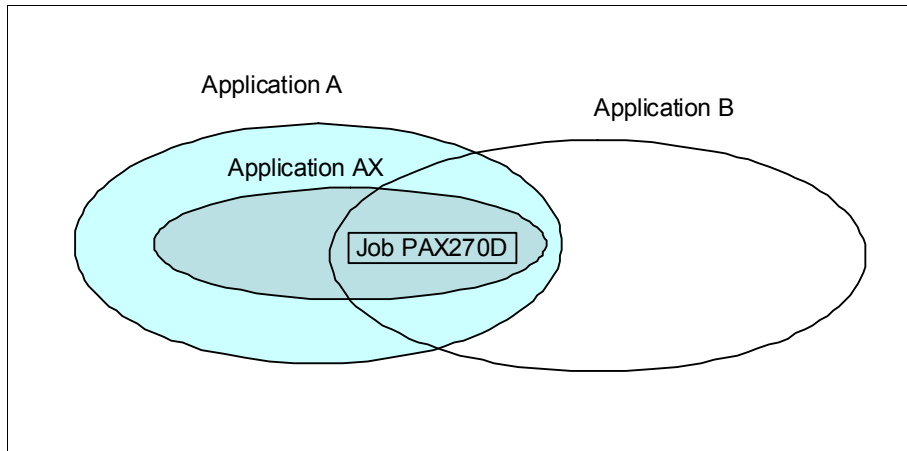


Figure 1-3 Boundary Job

- ▶ Users of every database object must be identified.
- ▶ Critical path and end points must be identified.
- ▶ Batch reliability must be assessed and opportunities to maximize it must be planned.
- ▶ Job elapsed time and resource consumption must be documented and fully understood because it can vary with business volumes.

Some of the key elements of the batch landscape that must be identified and considered are:

- ▶ Systems where the batch runs
- ▶ Database resources used by the batch
- ▶ Batch jobs that are grouped into applications
- ▶ Files
- ▶ Tape subsystems and libraries
- ▶ Bulk printers, envelope stuffers, and so on
- ▶ Transmission networks
- ▶ Batch flow endpoints and deadlines

It is important during the early stages of a batch project to identify these aspects, their interrelationships and behavior.

### Deriving detailed optimization actions

In the first part of the analysis phase we established an overall approach; in the second we developed a clear understanding of the batch environment and identified the key jobs and components that required optimization. This information can now be used in the third part of the analysis phase, which is to derive detailed optimization actions.

The implementation of the identified optimization actions is what yields the improvements for batch processing. These actions should implement the approach derived in “Setting the strategy” on page 7, although some actions might not be related to the strategy and some actions might even slightly contradict it.

To determine the job-level optimization actions to implement, examine the jobs in detail; follow a similar approach for the other components. The following tasks are some examples of optimization actions you can take:

- ▶ Examine the times of the job steps and then review file access for the major steps.
- ▶ Review and tune the database system buffer pools.
- ▶ Examine and increase the server processor speed.

### ***Patterns and Anti-Patterns***

When examining the batch processing components it is useful to look for patterns and anti-patterns, which are identified as follows:

|                     |  |
|---------------------|--|
| <b>Pattern</b>      | A named solution to a recurring problem in a context.      |
| <b>Anti-Pattern</b> | Similar to a pattern but leading to negative consequences. |

A documented pattern or anti-pattern specification can be used to communicate knowledge from an expert to a novice:

- ▶ For a pattern, the specification shows what makes the problem difficult and how it should be solved.
- ▶ For an anti-pattern, the specification shows what should be avoided and if a problem is found, how it should be fixed.

Both the patterns and anti-patterns contain enough information to decide whether they are applicable and provide enough details to be used successfully.

A pattern describes an outline for a solution to a problem but does not advocate a particular detailed implementation of that solution. As an example, in “Appropriate strategy - using DIM as an example” on page 8 the list of criteria forms a pattern, but we do not say how to specifically implement DIM.

Chapter 4, “Anti-patterns” on page 33 describes some common performance anti-patterns and possible solutions.

### **Verification and planning**

In the final stage of analysis we validate the candidate solutions, both strategic and tactical, and plan their implementation. Some of the tasks include:

- ▶ Preparing a checklist of optimization actions that can be implemented.
- ▶ An overall estimate of the likely benefits.
- ▶ An assessment of the financial cost, including personnel time.
- ▶ An evaluation of any alternative solutions.
- ▶ Resolution of any conflicts between candidate solutions.
- ▶ A decision on who should implement the solutions.
- ▶ A plan of the implementation timescales and dates.
- ▶ The measurement and monitoring of the effects from any changes implemented.

A workshop approach is beneficial to performing these tasks.

## **1.5.3 Implementation**

The changes identified in the analysis phase are enacted, tracked, and monitored in the final phase, implementation.

Batch optimization implementation is very similar, at a high level, to other implementation projects, so we do not review the generic items here.

However, there are some considerations specific to batch projects, including the following:

- ▶ The size of a batch project can be different from other implementation projects. The scale of a batch optimization can be very large because many changes to many components, such as program modules and databases, might be required. Furthermore, the potential

for a large quantity of changes means a batch project can have an impact on the entire company.

- ▶ Typically there is a need for user acceptance testing, but some required changes can be difficult to test. Testing whether a change continues to allow a low-level component, such as a batch program, to function correctly is usually straightforward. However, testing the impact of a change on the overall batch flow is much more difficult.
- ▶ The implementation of batch optimization can require a diverse set of skills, for example database administrators, workload schedulers, or application developers. But precisely which skills are needed is not known until the detailed tasks are identified.







## Analysis steps

In this chapter, we describe in more detail the methodology and steps outlined in “Analysis methodology” on page 6. This discussion is specifically about applications and software that run on z/OS. In Chapter 5, “Tools” on page 53 we describe some tools you can use to perform these steps.

## 2.1 Setting the technical strategy

As described in “Setting the strategy” on page 7, you create a technical strategy from a palette of seven candidate strategies and combine them appropriately.

Here we review some z/OS-specific examples for each strategy:

1. Ensure that the system is properly configured.  
Make sure that the LPAR where the batch runs has enough memory and that the Workload Manager (WLM) setup is appropriate for batch.
2. Implement Data In Memory (DIM).  
Implement sufficient buffering of DB2 indexspaces and tablespaces and use the Virtual Storage Access Method (VSAM) Local Shared Resources (LSR) buffering adequately.
3. Optimize I/O.  
Install enough disk I/O bandwidth and use sequential data striping appropriately.
4. Increase parallelism.  
Break multistep jobs and use BatchPipes/MVS (Pipes) to allow steps to run alongside each other. In fact, many jobs can be broken up without needing Pipes.
5. Reduce the impact of failures.  
Use an ICETOOL VERIFY step to scan a notoriously unreliable data set and only run the main processing step if ICETOOL declares it to be clean. However, a more strategic solution is to fix the causes of data corruption.
6. Increase operational effectiveness.  
Use Tivoli Workload Scheduler (TWS) functions to automate the batch, or use tape automation.
7. Improve application efficiency.  
Use Application Performance Analyzer for z/OS (APA) to tune the application program code.

There are too many z/OS-based product functions that implement these strategies to list them all here.

For example, entire books have been written about VSAM tuning, notably *VSAM Demystified*, SG24-6105 and several chapters of *System/390 MVS Parallel Sysplex Batch Performance*, SG24-2557. Most installations do not have the time or the skills to tune VSAM properly, so they have to rely on some general guidelines instead of comprehensive analysis, such as the following:

- ▶ Some of the seven strategies can have computer resource implications. DIM can have implications for memory, processor cycles, and I/O, as discussed in “Appropriate strategy - using DIM as an example” on page 8.  
Assessing system resource usage is a necessary early step. In the DIM case, for example, there is the perspective of finding free resources, memory, and processor cycles, and perhaps alleviating constraints for I/O.  
These two factors, processor and memory utilization, demonstrate that resources are available for exploiting parallelism and Data In Memory (DIM).
- ▶ Some of these strategies require analyzing operations logs and instrumentation for specific jobs to analyze the incidence, nature, and causes of failures. There is also the need to understand the batch application operational effectiveness.

- ▶ Application efficiency is difficult to assess. Take the example of a job that runs as a single step for 1 hour, consuming 45 minutes of processor cycles. Is this job inefficient? The answer is yes if you can find a technique to cut the processor cycles (for example to 5 minutes), otherwise the answer is less clear.

The existence of processor-consuming job steps can be determined, although it is best done as part of understanding the batch landscape.

## 2.2 Understanding the batch landscape

As previously mentioned, it is important to identify key processes and issues in the batch environment.

We have explained the possible complexity of batch environments. Fortunately, an understanding can be gained from the machine-readable instrumentation, so complexity is less of an issue. We discuss this instrumentation in Chapter 5, “Tools” on page 53.

### 2.2.1 Identifying where batch runs and the available resources

Installations usually know where batch is allowed to run. This can be confirmed using job runtime information or with Tivoli Workload Scheduler (TWS). For the z/OS system (LPARs) it is readily identifiable.

Some resources (such as DB2 subsystems, job classes, tape data sets, and disk data sets) are also readily identifiable with standard instrumentation. Other resources are more difficult to locate, for example, printers, transmission lines, and tape libraries. Typically, IT support would have knowledge about these.

With the advent of technologies such as the IBM DB2 Analytics Accelerator (IDAA) and the IBM zEnterprise® BladeCenter® Extension (zBX) also, resources outside the z/OS environment might have to be included in what is needed for batch to run.

**Note:** Although it is usually easy to determine which DB2 subsystem a batch job uses, it is more difficult to determine which subsystem tables and indexes it uses.

### 2.2.2 Job naming conventions

With many installations running tens of thousands of jobs per night, the identification of the production batch and categorizing it by application is a large but worthwhile task.

Installations typically maintain a naming convention that is defined and established in a planned fashion. Using a naming convention can help determine which applications are important.

Table 2-1 presents an example of a job naming convention.

Table 2-1 Example of a job naming convention

| Character position | Number of characters | Meaning   |
|--------------------|----------------------|---|
| 1                  | 1                    | P for Production<br>D for Development<br>Q for Quality Assurance                                |
| 2                  | 3                    | A three-character application name  |
| 5                  | 3                    | Individual job identifier, typically a number   |
| 8                  | 1                    | D for Daily<br>W for Weekly<br>M for Monthly<br>Q for Quarterly<br>Y for Yearly<br>A for Ad Hoc |

PLEG250D is an example of a job name in this naming convention. It is a production batch job, running daily, and it is part of the LEG (General Ledger) application.

You can assess how well an installation's naming convention is being implemented by performing the following tasks:

- ▶ Check how closely the naming convention is followed by obtaining and reviewing a list of job names. You can readily determine what proportion of jobs follow the pattern.
- ▶ Check whether the applications identified by the naming convention are meaningful to the installation.

This step is more difficult and requires you to discuss the list of applications with the appropriate experts. For example, in the review of the three-character names, discuss how the three-character codes relate to the application names, and how those batch applications group together to form the business applications.

Understanding these conventions and assessing the degree of compliance with them is important. This can be achieved with the review of standard performance data.

**Note:** In addition to job naming conventions, installations also define naming conventions for other components, such as data sets, steps, and programs.

For example, job PLEG250D might create data sets with PLEG250D as one of its qualifiers. The main processing step might be called STEP010, which runs program LEG250, and LEG250 might use plan PLEG250. Also, every job might name its steps STEP010, STEP020, STEP030, and so on.

However, the naming convention is not always followed. Some of the reasons are:

- ▶ Over time there is a tendency for an installation to deviate from them because of budget constraints, or just a lack of governance.
- ▶ In case of a merger or acquisition, batch applications within the same domain might be merged, but without necessarily making the names uniform. So, for example, a general ledger application in a large bank might still have a mixture of naming conventions because of a previous merger.
- ▶ Batch application boundaries can be fuzzy, such as those of a batch job that forms a bridge between two applications.

## 2.2.3 Application level performance analysis

If you have a good job naming convention it is easy to do application-level performance analysis because the queries that extract job-level information can be simple. Compare this to the case where you have to explicitly refer to each individual job by name. In the latter case such queries become long and unwieldy.

Two application-level queries are illustrated here:

- ▶ Display of the application run times as a Gantt chart
- ▶ Summary of a DB2 application elapsed time profile

### Gantt chart

When using a job naming convention you can readily produce a Gantt chart like the one shown in Figure 2-1. This annotated example reveals a number of interesting features about this batch application: a high number of tape mounts, jobs that are clones of each other, and apparent gaps in the schedule. All of these issues should be investigated.

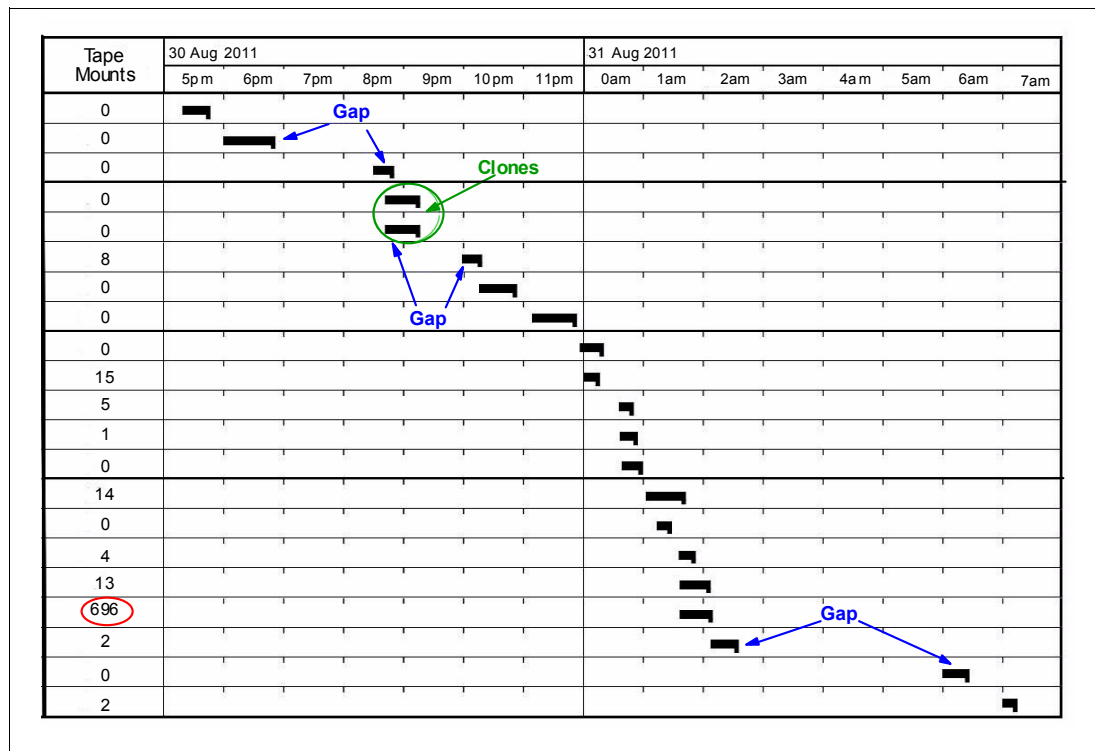


Figure 2-1 Gantt chart example

### DB2 application time profile

You can summarize a DB2 application time profile. An example is shown in Figure 2-2 on page 18 for a group of jobs deemed key and long running. In this example, the time is dominated by DB I/O, which is DB2 Database Synchronous I/O, and a small number of other elapsed time components. Such a profile guides you in your approach to tuning DB2. In this example, DB2 Buffering should help, as should tuning SQL to reduce Synchronous Database I/O and processor cycle consumption.

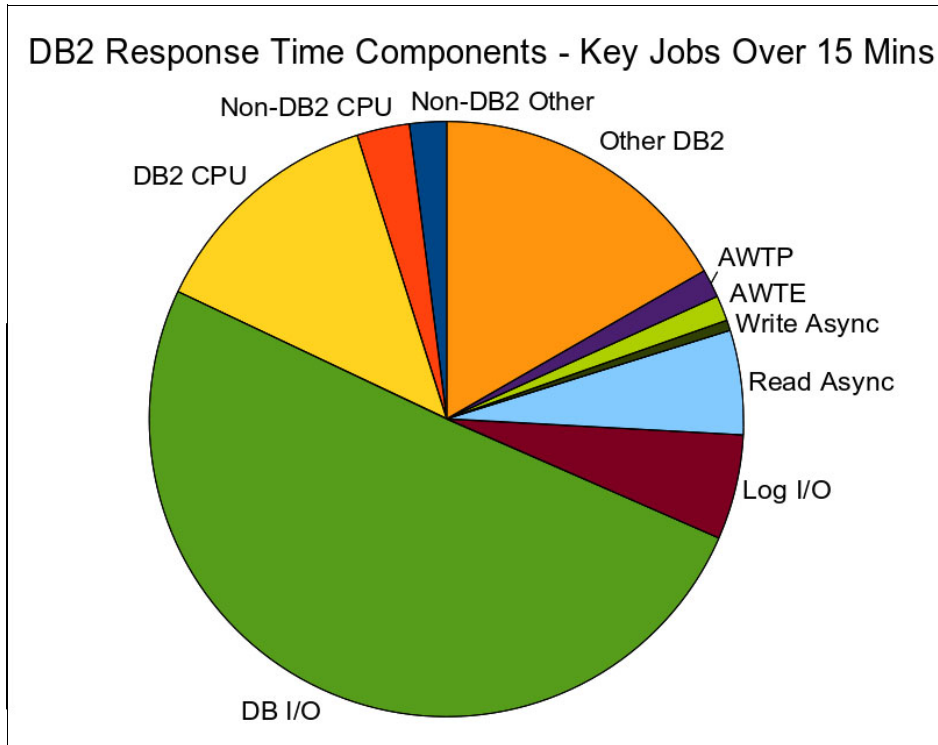


Figure 2-2 DB2 application time profile

## 2.2.4 Job-level understanding

After you identify the jobs in the important applications, you need to determine which jobs are performance sensitive and how these jobs will perform with varying business volumes.

**Note:** IBM Tivoli Workload Scheduler (TWS) has a view of job runtimes and you can control how TWS calculates job run time, but it cannot display how run times vary. For this you need actual run times from instrumentation. This is described in 5.2.1, “Systems Management Facilities (SMF)” on page 62.

It is useful to characterize the types of jobs, for example, DB2, IBM IMS™ Batch Message Processing (BMPs), or tape-intensive jobs. Another aspect of this characterization is to identify how many steps a job has and how many of these steps have significant run times.

Here is an example of categorizing different types of jobs:

- ▶ A job with a single step.
- ▶ A job with one significant processing step plus a setup and a termination step.
- ▶ A job with a small number of processing steps. Sometimes these processing steps can have setup or checking steps between them.
- ▶ A job with a large number of steps doing a variety of things.

Jobs with many steps can cause a lot of problems. They generally are more difficult to manage, take longer to recover from a failure, and perform poorly. In general, it is not easy to break up these jobs into multiple jobs with fewer steps, but it is usually worth the effort. The larger the number of steps the more worthwhile it is to break up the job.

## 2.2.5 Select jobs to tune

Key applications typically have thousands of jobs, so it is impractical to tune all of them. Therefore, you should select the ones that would be most beneficial to tune.

You can use the following criteria to help you select the jobs to tune:

- ▶ Processor cycles

If your challenge is to reduce processor cycles, choose the highest processor consuming jobs. Perhaps restrict yourself to jobs that run at the peak of the rolling four hour average of processor consumption.

- ▶ Critical path

The longest running critical path jobs should be in the list. Sometimes jobs near one of the critical paths are also worth tuning, so consider those as well.

- ▶ Similar jobs

Jobs that are similar to many other jobs might be worth tuning. Also, if you can easily apply the lessons learned from tuning them to other jobs, consider adding those to the list as well.

Sometimes jobs which do not initially meet these criteria might emerge from tuning those that do. For example, a job that writes a big data set read by a selected job might emerge as also being worthwhile to tune. Therefore, the list of jobs to tune can change as you go through this process.

## 2.2.6 Supporting infrastructure

Some batch applications are essentially stand-alone, and do not depend on other components, but many other applications rely on infrastructure components such as database managers and tape libraries.

You should examine these components and assess whether they are tuned for batch. Such components might include:

- ▶ DB2 subsystems
- ▶ IMS subsystems
- ▶ Tape drives and libraries
- ▶ Workload Manager (WLM) policies
- ▶ Transmission mechanisms
- ▶ Job classes and initiators

### Tuning for batch performance

Most components that support batch also support other workloads and these two requirements can be in conflict. Fortunately, there are usually time periods when only one predominates, but this should be reviewed to determine whether there is a conflict.

Assuming a period of batch-intensive work, you can tune the components specifically for batch. Some examples of batch tuning are:

- ▶ Changing the DB2 buffer pools to favor sequential processing. Batch often has a more sequential data access pattern than other workloads.
- ▶ Increasing the buffer pool size to take advantage of the lower demand for memory in the batch window. However, this might not always apply because many installations keep other applications available throughout the batch window.

- ▶ Switching WLM policies to favor batch. To accomplish this, some installations have different WLM policies for overnight and daytime processing.
- ▶ Changing Logical Partition (LPAR) weights to favor the LPAR where batch runs. This might be a manual procedure, or Intelligent Resource Director (IRD) might automatically shift weights between LPARs.
- ▶ Ensuring DB2 or other logging bandwidth is adequate for the batch peak. In this way it will not conflict with other user logging requirements, which might actually be higher than that of the batch.
- ▶ Provisioning more processor capacity to reduce queuing, or provisioning more memory to prevent paging.

Be aware that these techniques are different from those described in the next section because they apply to many jobs, not just individual ones. These techniques are often easier to implement, or minimally can take less time for each affected job.

**Note:** We discussed the need to produce a list of tuning actions as though you were a consultant or analyst, but we have not discussed the specific implementation of these tuning changes. There is a brief overview of this topic in 1.5.3, “Implementation” on page 10.

## 2.3 Deriving detailed optimization actions

After the technical strategy is set and there is an understanding of the batch landscape, you can derive detailed optimization actions.

Make a list of jobs to tune, then tune each job individually. To tune a job, perform three kinds of analysis on it:

- ▶ Find the longest running steps and tune these individually.
- ▶ Examine the inter-step dependencies and assess whether they can be eliminated, splitting the job where possible.
- ▶ Examine the inter-job dependencies and strive to eliminate these, adjusting the batch schedule as appropriate.

It does not matter in which order you do this analysis, but tuning long-running steps is generally of lower risk than either of the other two.

**Important:** When it comes to eliminating dependencies between jobs and job steps, thorough application knowledge should generally be at hand.

### 2.3.1 Tuning long-running steps

If your goal is to reduce processor cycle consumption, the tuning of long-running steps is the only approach needed; splitting jobs or removing inter-job dependencies will not help.

**Note:** We have seen cases where the aim is to speed up the batch so its overall processor use could be slowed down. Under these circumstances the other two approaches have value.



Long-running steps are readily identifiable. As a general guideline, any step that takes more than 10% of the job's run time is worth considering.

Some examples of how you might tune a step's run time are:

- ▶ Tune the application code. This might also reduce processor consumption.
- ▶ Tune data set access. This reduces the run time I/O component.
- ▶ Split the step into clones. Each clone runs against a subset of the data.
- ▶ Tune DB2 aspects. For example, a locking problem can contribute to the readily measurable component of run time.

Many steps consist of more than one phase. A well known example of this is a sorting step. A sorting step has three phases:

- ▶ Input phase
- ▶ Work phase
- ▶ Output phase

Usually the work phase runs for a short time. Therefore, one of the most common tuning actions is to eliminate this phase, which can be done by eliminating the intermediate merges. The length of the input and output phases is determined by how fast the sort can read and write the data and how much of the data is written out.

The phase analysis approach can be applied to any step. However, it might not produce additional step understanding.

As with a DFSORT sort operation, the data set open and close times within a step indicate the start and stop times of the step's phases. However, determining the step phases and timings is part of the analysis because you probably do not know the phases before you do the analysis.

For more complex DB2 steps, you can sometimes divide the step into phases using DB2 Accounting Trace (SMF Type 101) record timestamps. There are two cases where these can be used:

- ▶ When a DB2 step uses CP Query Parallelism you can set up DB2 to write SMF 101 records for each parallel leg. With this setup you can determine whether the legs take different amounts of time as well as the timings for parallel groups.
- ▶ If a step leaves DB2 entirely and returns to DB2 again you get multiple SMF 101 records.

## 2.3.2 Splitting jobs

In many cases, jobs can be split at the step level, as opposed to within a step, as was discussed in 2.3.1, "Tuning long-running steps".

It is important to determine why one step runs after another. Some of the reasons are:

- ▶ One step writes a data set that another step needs to process.
- ▶ One step accesses DB2 and then another step needs to be scheduled later.
- ▶ Additional function was added to a pre-existing job by simply adding steps.
- ▶ The original design did not take into consideration whether steps were independent or dependent on each other.

Most of these reasons are similar to the reasons one job might be scheduled after another, as discussed in 2.3.3, "Removing inter-job dependencies" on page 23.

The Life Of A Data Set (LOADS) is a technique for examining SMF (System Management Facility) data. It is discussed in *Batch Modernization on z/OS*, SG24-7779. LOADS allows you to consider ways to speed up data set access. This technique also enables you to determine where the dependencies caused by data set access exist.

Another technique is to examine the job's Job Control Language (JCL). Examine the JCL when you are reviewing a very small number of jobs and look for reader and writer DD statements. This is similar to how you use LOADS with SMF. (LOADS is easier to use when you need to examine many jobs.)

Some patterns that suggest that a dependency exists are:

- ▶ A writer followed by a reader
- ▶ A writer followed by a writer
- ▶ A reader followed by a reader
- ▶ A reader followed by a writer

A reader followed by a reader does not necessarily indicate a dependency because either reader can execute first, or they can run in parallel if the job is split up. A data set on tape is a frequently observed example of a dependency because concurrent reads of a physical tape data set are not possible.

**Note:** Both a data set name and a volume serial number (VOLSER) are required to uniquely identify a data set, especially for tape. The member name in a partitioned data set and the generation number for a generation data group are important elements that can be checked to determine whether a dependency exists.

For example, tape data set OTNAY.AWAY.ULLFAY.ITEMWAY.AMENAY, with volser APETAY, written by TEP1SAY and read by TEP2SAY are required elements to uniquely characterize the dependency.

Many installations catalog internal tape data sets, but this does not eliminate the need to view the volser as part of the unique identifier.

When splitting jobs with a number of steps into smaller jobs, if a dependency exists between two steps, evaluate whether the dependency can be removed, although usually this is not possible. These are two examples of how a dependency might be removed:

- ▶ For a sequential data set, it might be possible to use BatchPipes/MVS to remove the dependency.
- ▶ For a Virtual Sequential Access Method (VSAM) data set, one step might open the data set for update and a subsequent step opens the data set for read. Viewing VSAM statistics, in the SMF Type 64 records, you can see whether the step updated, deleted, or inserted records.

If there are never any updates, deletes, or inserts in the step that opens the data set for update, there might not be any dependency.

For the reader followed by writer case, examine the time lag between the reader closing the data set and the writer opening it. If the time lag is sufficiently long, the dependency might not need to be coded in the scheduler. For example, if the reader and writer steps are part of a daily job and the lag is 23 hours between the reader and the writer, there is probably no need to code a dependency between the jobs.

There are other cases where a dependency might not be real. For example, some installations have many jobs where the first step updates a control data set and a last step

updates it again. In this case it might be possible to split the job up because this data set does not represent a real data set.

Life Of A Data Set (LOADS) can be simplified to ignore the possibility that a data set is opened more than twice, although for most data sets this is not the case. However, this technique for dependency validation can be easily extended. For example, a data set with one writer followed by two readers creates a pair of dependencies. In this case both readers are dependent on the writer, but not on each other.

### 2.3.3 Removing inter-job dependencies

Inter-step dependencies are implicit in the running order of steps within a job. Inter-job dependencies are explicitly coded in the scheduler.

Once you have identified the inter-job dependencies from the scheduler, use the same process to examine them that you used for inter-step dependencies. Determine whether the dependency is real or not for data set access patterns, or other resource usage patterns.

The concept of lag is also important. If the first close and a subsequent open of the data set are sufficiently far apart, this dependency might not be coded in the scheduler. Nonetheless the dependency is real and it should be taken into account when tuning. Consider adding these missing dependencies to the schedule. However, consider this carefully because it is easy to define too many of this kind of dependency, which would add unnecessary work to the scheduler.

## 2.4 Verification and planning

Next we validate the candidate solutions, both strategic and tactical, and plan their implementation.

Some of these tasks include:

- ▶ Checking whether the optimization actions can be implemented.
- ▶ Estimating, probably very loosely, the likely benefit.
- ▶ Assessing the cost, whether personnel time or financial.
- ▶ Evaluating the alternative solutions.
- ▶ Resolving the conflicts between candidate solutions.
- ▶ Deciding who should implement the solutions.
- ▶ Planning the implementation timescales and dates.
- ▶ Considering how to measure and monitor the effects of changes.

A workshop approach works well for performing these tasks because the analysis naturally breaks into stages and then common understanding can be reached at the end of each stage.

In each workshop, the team that analyzed the batch should present their findings for further discussion. The list of tasks includes several for which debate and negotiation are required. For example, evaluating alternative solutions is a task that requires a number of perspectives.

Our experience shows that the involvement of a number of teams is important for the success of a workshop. These teams include:

- ▶ Performance and Capacity
- ▶ Operations
- ▶ Application Development
- ▶ Systems Programming

- ▶ Database Administration

Each team brings their own perspective and expertise. Often batch projects are sponsored by Operations and sometimes they do not include Application Development. In our experience, this is not ideal. With a wide range of participants the workshops are better, even if there is a perception that the teams speak different languages.



# Governance

In this chapter, we discuss what governance is, why it is needed, and how to implement it, specifically for the batch schedule. We also review the scope and role of the Batch Design Authority (BDA) in governance.

## 3.1 Governance defined

*Governance* is the formal process and the policies for controlling, directing, and making batch decisions. Governance empowers you with consistent management, cohesive policies, guidance, and decision rights for your batch actions and solutions. You can use these processes to verify your batch performance. The Batch Design Authority (BDA) serves as a guide in this process.

In this publication, we describe governance specifically for the batch schedule.

## 3.2 The Batch Design Authority

A key element of batch governance is to have an organization in place with a focus on batch processing. This organization is the Batch Design Authority (BDA). In this section, we explain the BDA in more detail, specifically:

- ▶ The scope of the BDA
- ▶ The tasks of the BDA
- ▶ The roles of the BDA

### 3.2.1 Scope

The scope of the Batch Design Authority is to:

- ▶ Design the end-to-end batch schedule of the enterprise
- ▶ Document and enforce the standards and guidelines to run the batch schedule
- ▶ Assure the quality of the batch schedule during design, build, operation, and change phases
- ▶ Make architectural decisions about the batch schedule, especially for new features
- ▶ Make proposals for improvement

### 3.2.2 Tasks

The BDA performs tasks over three time frames, with the scope increasing over time.

- ▶ **Short Term:** Addresses the current batch problems and how they can be solved quickly. Manages risk in the current environment and maintains up-to-date standards and guidelines.
- ▶ **Medium Term:** Creates and maintains the future architecture, including new machines and new applications. Involved with the creation of new solutions.
- ▶ **Long Term:** Aligns the business strategy and the batch strategy and implements this strategy effectively. Develops strategic and innovative initiatives.

### 3.2.3 Roles

The BDA has a role with Enterprise Architecture and IT Management in addition to interacting with Development and Operations. These roles are part of the overall governance model and are depicted in Figure 3-1 on page 27.

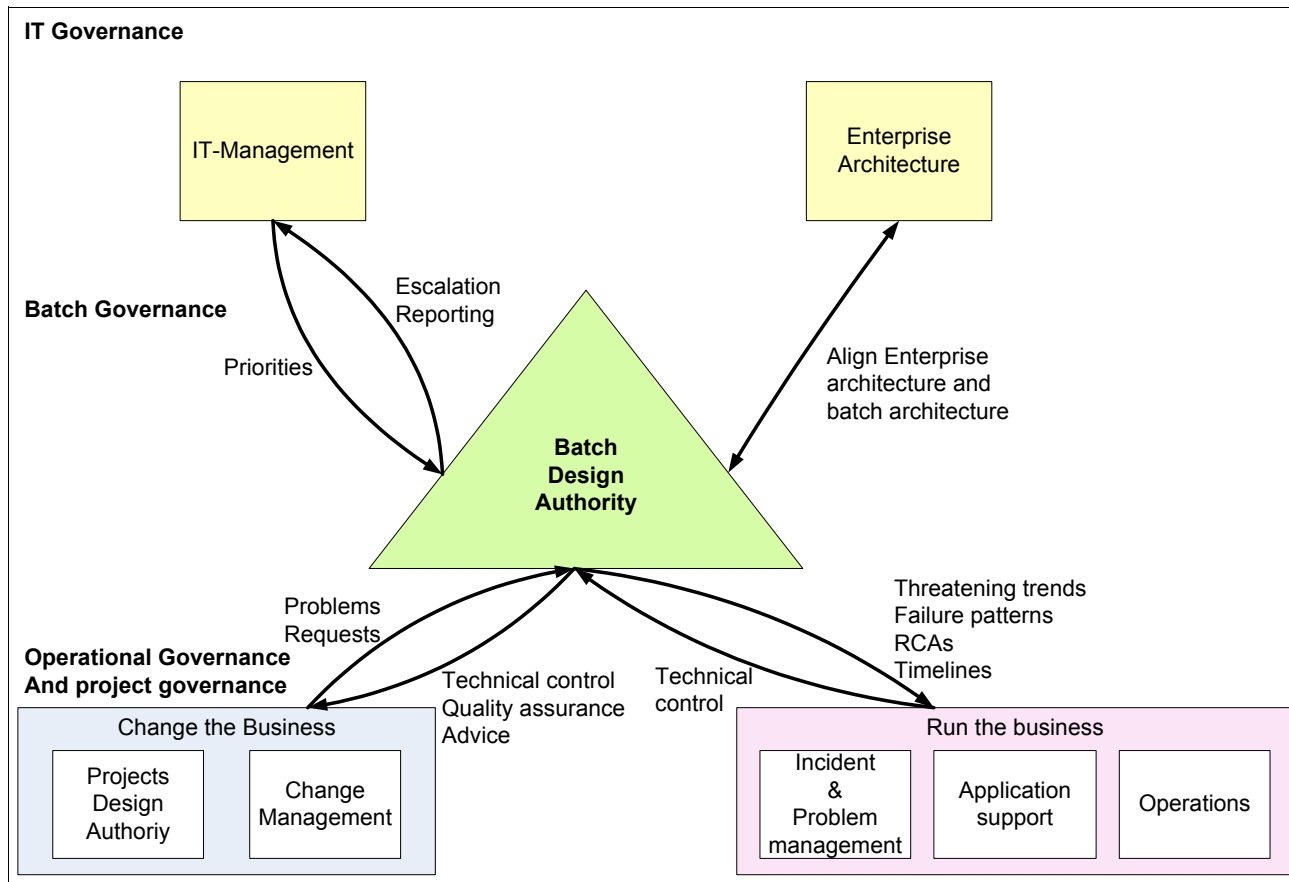


Figure 3-1 Batch Design Authority and roles of governance model

The BDA reports and advises IT management on a number of topics, including:

- ▶ Batch service level reporting
- ▶ Conflicting viewpoints

Operations and development can have conflicting viewpoints, so the BDA aligns the enterprise architecture with the batch architecture. It is important to align these viewpoints and decide what viewpoint should get priority.

- ▶ Risks and issues
- ▶ Plan improvements

IT management reviews the improvements and then decides what plans are executed.

The BDA delivers the batch standards and guidelines to development, describes them, and answers any questions. The BDA also advises development on how to build the batch schedule and performs quality control over the delivered components.

The BDA is involved with the running of the business. Incident and problem management, operations, and application support report on threatening trends, failure patterns, root cause analysis, and timelines of critical endpoints and individual jobs. The BDA assures the technical control for running the business.

The BDA is also involved with the operational and project governance. Change management and the Project Design Authority manage the changes to the business. The BDA assures the technical control and quality assurance of all changes to the business.

## 3.3 Why you need governance

Governance is needed to ensure that the problem which precipitated the batch optimization project (the issue you identified in 1.1, “The initial business problem statement” on page 2), never occurs again. Implementing governance reduces the likelihood of such problems even occurring in the first place.

Some of the advantages of governance are:

- ▶ Hardware resource utilization.  
When you remove jobs, copies, and relations that are not needed, then the processors, storage, and network usage decreases.
- ▶ Decrease of application maintenance.  
When you remove jobs, copies, and application relationships that are not needed, then the costs for application maintenance decreases.
- ▶ Decrease of application development.  
When the batch schedule is efficiently arranged, the cost for application development decreases.
- ▶ Batch schedule is conveniently arranged.  
You can easily manage your schedule and understand the relationships so that you can find your applications and determine their status.
- ▶ The Batch Design Authority makes architectural decisions in advance.  
When a new feature is introduced or a project needs a new feature, the BDA first decides whether they want to use it and then decides how to use it. The decisions are documented in the standards and guidelines. Every project uses the same solution; when the solution is documented, other projects can reuse it.
- ▶ Advance planning.  
Due to timely planning, you know in advance that the proposed solution fits within the time window.
- ▶ Fewer errors in the batch.  
Because the batch schedule consists of only the jobs needed and they have the correct relationships, you have fewer problems with the batch.
- ▶ Easier to troubleshoot.  
Because your batch is conveniently arranged, it is easier for you to do troubleshooting of any problems or issues.
- ▶ Achieving SLAs.  
It is easier to meet and maintain your service level agreements.

## 3.4 How to implement governance

Perform the following steps to implement governance:

1. Appoint the Batch Design Authority.
2. Revise the standards and guidelines.
3. Communicate the standards and guidelines.
4. Set up a new baseline of critical endpoints and elapsed times



5. Monitor baseline.
6. Start making improvements.
7. Control and perform quality assurance.

These steps are depicted in Figure 3-2.

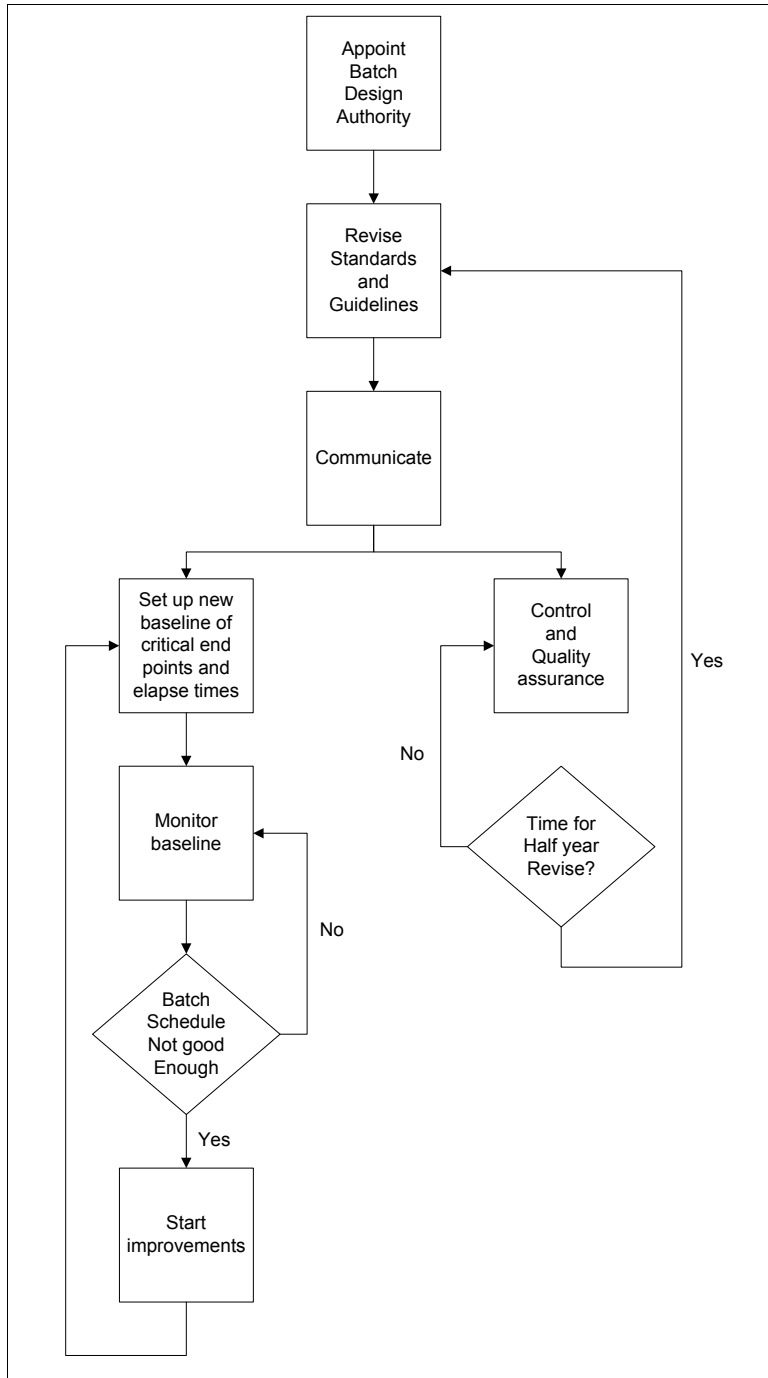


Figure 3-2 Process flow to create governance

### 3.4.1 Appoint the Batch Design Authority

The first step is to appoint your Batch Design Authority (BDA). The people in the BDA should be knowledgeable in the following areas:

- ▶ Batch scheduling
- ▶ Job scripting language
- ▶ Use of databases in the batch
- ▶ Business-critical endpoints
- ▶ Interaction of the various servers and machines of the business

### 3.4.2 Revise standards and guidelines

Every company has standards and guidelines, but often they are out of date and do not reflect state-of-the-art concepts. It is good practice to review and revise the standards and guidelines periodically, but it is especially useful at the beginning of a batch optimization project. Discuss the existing standards and guidelines with all the involved parties. Retain and update the standards and guidelines that are still applicable, then add any new standards or guidelines that you want to implement.

If you use a tool to generate your batch, schedule and job control components make sure that it generates components that adhere to the new standards.

If you deploy the batch components to a test, acceptance, or production environment, and this tool checks the standards, update the tool to enforce the new standards.

When using any other tools to check your batch components, update these tools as well to enforce the new standards.

### 3.4.3 Communicate standards and guidelines

After you update the standards and guidelines, communicate them to all the involved parties, including:

- ▶ IT management
- ▶ Enterprise architecture
- ▶ Development
- ▶ Project management
- ▶ Incident management
- ▶ Problem management
- ▶ Change management
- ▶ Application support
- ▶ Operations

After they have been communicated, the updates to the standards and guidelines for the batch components can start to monitor and improve your batch.

### 3.4.4 Set up a baseline

When your batch is under control, set up a new baseline of critical endpoints and elapsed times to:

- ▶ Record the end times of your critical endpoints.
- ▶ Record the elapsed times of your jobs.

### 3.4.5 Monitor baseline

After it is set up, monitor the baseline to accomplish the following:

- ▶ After you start to monitor, you can see whether the timelines of your critical endpoints are being met.
- ▶ You can see whether the elapsed times of jobs are growing.
- ▶ Perform trend analysis.
- ▶ If the trend analyses indicates that your elapsed time is growing or if your critical endpoint is being reached later, take measurements.

### 3.4.6 Start improvements

After you have implemented your new standards and guidelines, you can start to improve your batch. However, batch components might not comply with the new standards and guidelines because:

- ▶ The standard is new and the batch component is old.
- ▶ A component went into production without a proper check against the standards.
- ▶ The program using the batch component was enhanced but the batch component was forgotten.
- ▶ The components are not needed anymore but no one removed them from production.

You can now start projects to improve the batch by following these steps:

1. Define anti-patterns you do not want to see and define the patterns that you do want to see.
2. Change a component that has an anti-pattern in a component with a pattern.

After you have done one or more improvement projects, you can create a new base line.

### 3.4.7 Control and quality assurance

After you have created your new standards and guidelines and started improving your batch, you want to remain in control. This entails:

- ▶ Regularly reviewing and updating the standards and guidelines
- ▶ Performing quality assurance for test, acceptance, and production
- ▶ Reviewing innovative development and implementing new features





# Anti-patterns

In this chapter, we introduce some typical performance patterns and anti-patterns, describe how to analyze them, and suggest possible solutions.

## 4.1 Introduction to patterns and anti-patterns

Software design patterns and anti-patterns were first discussed in *Design Patterns*<sup>1</sup>. The authors derived their approach from Christopher Alexander's *Pattern Language*, which was developed for towns, buildings, and construction, and which dates back to the publication of *A Pattern Language*<sup>2</sup> in 1977.

### 4.1.1 Definition of a pattern

A pattern (DO) is a named solution to a recurring problem within a context. A documented pattern specification communicates expertise from an expert to a novice and educates the novice about what makes the problem difficult and how to solve it. A pattern should provide the reader with sufficient information to decide whether the pattern is applicable to the problem, and with enough understanding to use it successfully. A pattern describes how to solve a problem, but does not advocate the implementation of a particular solution.

### 4.1.2 Definition of an anti-pattern

An anti-pattern (DON'T) is similar to a pattern, but leads to negative consequences. An anti-pattern describes what must be avoided and, when a problem is located, how to fix it.

## 4.2 Performance anti-patterns

Performance anti-patterns were introduced in *Performance Solutions*<sup>3</sup>. Batch elapsed time is one of the non-functional requirements for which performance anti-patterns can be determined.

To identify the performance anti-patterns that affect batch elapsed time, we subdivide the batch elapsed time into:

- ▶ Wait time  
Wait time in this case is indicated on the Gantt chart by *white spaces* that represent the batch critical timeline. You must minimize or preferably remove this type of wait time in consultation with the business owner.
- ▶ Time spent on work that is not needed  
Whether work is needed is a decision of the business owner. You must minimize or preferably remove work that is not needed in consultation with the business owner.
- ▶ Time spent on work that is needed  
Whether work is needed is a decision of the business owner. You must optimize work that is needed. There are various techniques to do so, such as parallelization, source code optimization, and system performance tuning. Meaningful time at the resource level is subdivided into:
  - Wait time
  - Processing time
  - I/O time

<sup>1</sup> Design Patterns - Elements of reusable software. ISBN: 0201633612. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison-Wesley, 1995.

<sup>2</sup> A Pattern Language. ISBN: 0195019199, Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King and Shlomo Angel. OxfordUniversity Press, New York, 1977.

<sup>3</sup> Performance Solutions - A practical guide to creating responsive, scalable software. ISBN: 0201722291, Connie U. Smith, Lloyd G. Williams. Addison-Wesley, 2002.

With these classifications in mind, we define the various candidate performance anti-patterns and describe them in the following sections.

The following anti-patterns are discussed in the chapter:

- ▶ 4.2.1, “Incorrect usage of a special resource” on page 35
- ▶ 4.2.2, “Incorrect usage of a start time” on page 37
- ▶ 4.2.3, “Invalid relationship” on page 39
- ▶ 4.2.4, “Obsolete dummy jobs or dummy operations” on page 42
- ▶ 4.2.5, “Actual start time is not equal to end time of last predecessor” on page 44
- ▶ 4.2.6, “Workload Manager is not set up properly for batch” on page 45
- ▶ 4.2.7, “Redundant copies of files” on page 48
- ▶ 4.2.8, “Redundant image copies” on page 49
- ▶ 4.2.9, “Batch prevented from effectively using sysplex-wide resources” on page 50

### 4.2.1 Incorrect usage of a special resource

It is important to use special resources correctly. You can specify a special resource in Tivoli Workload Scheduler (TWS) to prevent multiple jobs from running at the same time. This can be used when there is no dependency between the jobs and it does not matter what job runs first provided that the jobs do not run concurrently. It is a good idea to give the special resource a name that reflects the reason the jobs cannot run at the same time. For example, when both jobs use the same database, you can give the special resource the name of the database.

A special resource can be either *shared* or *exclusive*. For example, for reading you can use *shared*; for updates that do not allow any other jobs reading the same database, you can use *exclusive*.

#### Problem

Incorrect use of a special resource by a batch job can increase the batch elapsed time. If other jobs also need the special resource, but are not able to access it because it has been allocated exclusively to the first job, this can result in unnecessary delays in the batch schedule.

The risk of using a special resource is that it can cause bottlenecks in the batch schedule; therefore, it should be avoided as much as possible. Use a special resource only when two jobs do not have a dependency, but both require the special resource, either *shared* or *exclusively*. An example of this is when two jobs are updating the same table in a database, and there is no locking mechanism in place.

#### Method to identify

A special resource can be identified in TWS and it can have a *shared* or *exclusive* usage. If the special resource is defined in the batch schedule, it can also be found in the job control language (JCL), scripting language, program source, or data manipulation language. If the special resource is only mentioned in the batch schedule, it is being used incorrectly. When the special resource is found in both the batch schedule and the JCL, scripting language, program source, or data manipulation language, its usage can be compared. If the usage is *exclusive* in the schedule, then the usage in the JCL, scripting language, program source, or data manipulation language must be *update*. If the usage is *shared* in the schedule, then the usage in the JCL, scripting language, program source, or data manipulation language must be *read*.

Both the combination of *exclusive* in the schedule with *update* in other sources and the combination of *shared* in the schedule with *read* in other sources would mean that the

definitions have been made consciously and therefore probably are valid. Other combinations indicate an incorrect use of the special resource. When a special resource is defined in a batch schedule, there must be at least one job that has exclusive usage and one job that has a shared usage. If there is no job with exclusive usage, then the special resource is obsolete. Table 4-1 displays the allowed combinations.

Table 4-1 Identification of special resources

| Other sources | Exclusive in TWS | Shared in TWS |
|---------------|------------------|---------------|
| Update        | OK               | Not OK        |
| Read          | Not OK           | OK            |

### Sources needed to identify

The following sources are used to find and analyze this anti-pattern:

- ▶ The batch schedule
- ▶ Job control language or scripting language
- ▶ Program sources
- ▶ Data Manipulation Language (DML)

### Solution

After discussing with the business owner, correct the usage of the special resource and remove the obsolete special resource from the batch schedule. For any remaining special resources, attempt to make them obsolete by defining an appropriate locking mechanism in the JCL, scripting language, program source, or data manipulation language. Another way to make the special resource obsolete is to introduce a dependency.

### Performance benefits

The performance benefits attained with the correct usage of a special resource are:

- ▶ Elapsed time - For an individual occurrence of this anti-pattern, the potential elapsed time improvement equals the total wait time for the special resource.
- ▶ Resource utilization - For an individual occurrence of this anti-pattern, the potential elapsed time improvement equals the total processing time to allocate the special resource.

### Example

An example of the SQL code that can be used to select data is in Example 4-1.

Example 4-1 SQL code to select data

---

```

00177      EXEC SQL
00178      DECLARE MKB-CURSOR CURSOR WITH HOLD FOR
00179              SELECT CLIENTNUMBER,
00180                      ACCOUNTNUMBER,
00181                      TOT_CR_TURNOVR_L4Q,
00182                      DATE
00182              FROM VSAMPLEA
00183              ORDER BY ACCOUNTNUMBER
00184      END-EXEC.

```

---

Example 4-2 on page 37 is an example of the corresponding special resource that is used in TWS. In this example, the special resource for Operation SC6V 005 is exclusive and it is not OK.



Example 4-2 Special resource for operation SC6V 005

```
----- SPECIAL RESOURCES ----- Row 1 to 1 of 1
Command ==>                               Scroll ==> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete

Operation                : SC6V 005

Row  Special          Qty  Sir Keep On Avail on
cmd  Resource
'''  PROD.SAMPLEA.TODAY_____ X  N      Y
***** Bottom of data *****
```

### 4.2.2 Incorrect usage of a start time

It is important to use the start time correctly. A start time is the specification of the time that a job can start in the batch schedule. Start times of jobs can indicate the beginning of the schedule. A start time of a job can also be based on a business reason, for example, after office hours or after shopping hours.

#### Problem

When a start time is not specified, jobs can start immediately after the plan for the next 24 hours is finished. Not specifying a start time can cause problems when there are delays in the previous plan. A job start time specification that is too early can cause problems for the business. Likewise, a job start time specification that is too late can cause extra elapsed time for the batch.

#### Method to identify

The start time of jobs can be found in the batch schedule. You can compare these start times with the table of defined start times per application for the business.

#### Sources needed to identify

Use the following sources to find and analyze this anti-pattern:

- ▶ The batch schedule
- ▶ Table of business-defined start times per application

#### Solution

Update and correct the start time in the schedule based on the start time indicated in the table of defined start times per application.

#### Benefits

The performance benefits of the correct usage of start times are:

- ▶ No problems in case the previous plan was delayed - No jobs are submitted before the plan of the previous day is finished.
- ▶ No problems for the business - Because the job does not start too early, it does not impact the business and it can continue to work without problems.
- ▶ Elapsed time - Because jobs do not start too late, the elapsed time of the batch is reduced.

## Examples

Example 4-3 shows that for application SAMPLEB for the operation SC6V 020 SAMPLEB, time dependent is set to Yes.

*Example 4-3 Job options panel where you specify time dependent yes or no*

---

```
----- JOB, WTO, AND PRINT OPTIONS -----
Command ==>
Enter/Change data below:
Application   : SAMPLEB           A job desc with CMX
Operation    : SC6V 020 SAMPLEB
JOB CLASS    :      _             Job class of corresponding job
ERROR TRACKING :      Y           Y means error automatically tracked
HIGHEST RETURN CODE :      _       Highest return code not in error
EXTERNAL MONITOR :      N         Job monitored by external product (Y/N)
CENTRALIZED SCRIPT :      N       Centralized script Y/N (for FTW only)
COND RECOVERY JOB :      N        Recovery job for a cond predecessor (Y/N)
CRITICAL     :      N            Critical job: N P W
POLICY       :      _ CLASS     : SAMP _ WLM Policy and Service Class
Job release options:
SUBMIT       :      Y           Automatically submitted
HOLD/RELEASE :      Y           Automatically held and released
TIME DEPENDENT :      Y         Run the job at specified time
SUPPRESS IF LATE :      N       Suppress the job if not on time
DEADLINE WTO :      N           Deadline WTO, Y or N
WS fail options:
RESTARTABLE :      _           Operation is restartable
REROUTEABLE :      _           Operation is eligible for reroute
Print options:
FORM NUMBER  :      _           SYSOUT CLASS      :      _
```

---

Example 4-4 shows that the actual start time of the application and the operation details are entered. It also displays the correct start time agreed upon with the business.

*Example 4-4 Time specification panel*

---

```
----- TIME SPECIFICATIONS -----
Command ==>

Enter/Change data below:

Application time specifications:
Input arrival time   :    15.00
Deadline day/time    : 00 16.00

Operation            : SC6V 020

Operation input arrival:
DAY                  :      00           The day the input arrives for operation,
                                         relative to application start day
                                         (0 means the start day).
TIME                  :      15.00      Arrival time of the input
                                         in the format HH.MM

Operation deadline:
DAY                  :      _           Deadline day for operation completion,
                                         relative to application start day
```



Table 4-2 Analyze need for job relationships

| Successor        | QSAM input | QSAM output   | VSAM read  | VSAM update   | DB2 table read | DB2 table update | IMS read   | IMS update    |
|------------------|------------|---------------|------------|---------------|----------------|------------------|------------|---------------|
| Predecessor      |            |               |            |               |                |                  |            |               |
| QSAM input       | Not needed | OK, but check |            |               |                |                  |            |               |
| QSAM output      | OK         | OK            |            |               |                |                  |            |               |
| VSAM read        |            |               | Not needed | OK, but check |                |                  |            |               |
| VSAM update      |            |               | OK         | OK            |                |                  |            |               |
| DB2 table read   |            |               |            |               | Not needed     | OK, but check    |            |               |
| DB2 table update |            |               |            |               | OK             | OK               |            |               |
| IMS read         |            |               |            |               |                |                  | Not needed | OK, but check |
| IMS update       |            |               |            |               |                |                  | OK         | OK            |

### Method to identify a missing relationship

In the case of a missing relationship, there is no relationship in the workload scheduling tool. However, according to the JCL, scripting language, program source, or Data Manipulation Language, both jobs have a common resource. This is the opposite of the superfluous relationship.

### Sources needed to identify

The following are the sources you can use to find and analyze this anti-pattern:

- ▶ The batch schedule
- ▶ JCL or scripting language
- ▶ Program sources
- ▶ Data Manipulation Language
- ▶ Business knowledge

**Note:** For QSAM and VSAM data sets, SMF data can also be used as source.

### Solution

After discussing with the business owner, remove invalid relationships and create missing relationships.

### Benefits

For an individual occurrence of this anti-pattern, the potential elapsed time improvement is:

$$t1 - t0$$

t0 is the end time of the last valid predecessor of the successor in the relationship and t1 is the start time of the successor in the relationship.

## Examples

Example 4-5 shows how a predecessor is recognized in TWS. It shows that job SAMPLEC runs after job SAMPLE8.

### Example 4-5 Predecessor relation in TWS

---

```
----- PREDECESSORS ----- Row 1 to 1 of 1
Command ==>                               Scroll ==> PAGE

Enter the COND command to view the list of conditions, enter/change
data in the rows, and/or enter any of the following row commands:

I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Description of external dependency

Application      : SAMPLEC                A job desc in ADGROUP
Operation        : SC6V 015   SAMPLEC
No. of conditions: 0

Row Oper        Transport time  Application id      Jobname
cmd  ws  no.      HH.MM          (for ext pred only)
''' CPU1 015          _____ SAMPLE8_____ SAMPLE8_
***** Bottom of data *****
```

---

Example 4-6 shows that procedure SAMPLEC is used by job SAMPLC has an input data set from job SAMPLE8. The naming standards used for TWS, JCL, source code, and databases are important for recognizing the link between the different components.

### Example 4-6 JCL example

---

```
//*****
//SAMPLEC   PROC
//*****
//STEP02   EXEC PGM=SAMPLE
//STEPLIB DD DSN=PROD.PGMLOA01.SAMPLE,
//          DISP=SHR
//SAMPLEI1 DD DSN=PROD.SAMPLE01.SAMLE08,
//          DISP=SHR
//DATE     DD DSN=PROD.SHOPPING.DATE,
//          DISP=SHR
//SAMPLE01 DD DSN=PROD.SAMPLE01.SAMPLEC(+1),
//          DISP=(,CATLG,DELETE),
//          DATACLAS=DCALNL,
//          DCB=(SYS1.MODEL,DSORG=PS,RECFM=FB,LRECL=657,BLKSIZE=0)
//SAMPLE02 DD DSN=PROD.SAMPLE02.SAMPLEC(+1),
//          DISP=(,CATLG,DELETE),
//          DATACLAS=DCALNS,
//          DCB=(RECFM=VBA,LRECL=181,BLKSIZE=0)
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=P
```

---

Example 4-7 on page 42 shows how you can recognize a database in an IMS program specification block (PSB). The DBDNAME is the name of the database used and PROCOPT is the processing options for the database. In this example, database SAMP01 is used for a read operation.

*Example 4-7 IMS PSB showing database and process option*

```
*****
*
*      DB/DC DATA DICTIONARY PSB_OUT FUNCTION          07/16/95
*
*      PSB DICTIONARY KEY = SAMPLED                    000
*
*****
      PCB  TYPE=DB,                                     X
          DBDNAME=SAMP01,                               X
          PROCOPT=G,                                   X
          KEYLEN=206
```

Example 4-8 illustrates how you can identify what DB2 table is used by examining the SQL statements.

*Example 4-8 SQL statements*

```
00177 EXEC SQL
00178 DECLARE MKB-CURSOR CURSOR WITH HOLD FOR
00179         SELECT CLIENTNUMBER,
00180                ACCOUNTNUMBER,
00181                TOT_CR_TURNOVR_L4Q,
00182                DATE
00182         FROM VSAMPLEA
00183         ORDER BY ACCOUNTNUMBER
00184 END-EXEC.
```

## 4.2.4 Obsolete dummy jobs or dummy operations

A dummy job does not contain any meaningful processing. A job that has become obsolete is replaced by the dummy job. If there is not sufficient knowledge about the consequences of removing an obsolete job, especially with regard to predecessor and successor relationships, the job is replaced by a dummy job.

### Problem

Although the dummy step does not result in significant resource utilization, it might cause processing delays due to the relationships to and from this job.

### Method to identify

A dummy job can be recognized in several ways:

- ▶ In TWS, the operation has a dummy workstation. It is still in the schedule but it is not actually running.
- ▶ In the JCL, the program that used to be executed is replaced by an IEFBR14, an assembler program that does a branch to register 14, which is a dummy step.

### Sources needed to identify

The following are sources that you can use to find and analyze this anti-pattern:

- ▶ TWS
- ▶ JCL
- ▶ Business knowledge

## Solution

After discussion with the business owner, remove the obsolete dummy step, including its relationships, and create the correct relationships in the remaining batch schedule.

## Benefits

For an individual occurrence of this anti-pattern, the potential elapsed time improvement depends on the situation.

- ▶ If the relationship between one or more of the predecessors and one or more of the successors of the dummy job is valid, the potential elapsed time improvement equals the time needed to process the dummy job, which could be insignificant.
- ▶ If the relationship between one or more of the predecessors and one or more of the successors of the dummy job is invalid, the potential elapsed time equals the minimum of the potential elapsed time improvements of all invalid relationships.

## Examples

Example 4-9 displays the JCL of a job that executes the dummy program IEFBR14.

*Example 4-9 Job that executes dummy program IEFBR14*

---

```
//SAMPLEC JOB CLO00000,MSGCLASS=2
//*OPCEXIT
//*ROUTING
//STEP01 EXEC PGM=IEFBR14
```

---

Example 4-10 illustrates how a dummy workstation is defined in TWS.

*Example 4-10 Definition of a dummy workstation in TWS*

---

```
----- BROWSING A WORK STATION DESCRIPTION -----
Command ==>

Enter the command R for resources A for availability 0 for end-to-end options
or D for Destinations above

Work station      : DUMM
Description       : DUMMY

Work station type : General
Reporting attribute : Non reporting
Printout routing  : SYSPRINT
Server usage      : Planning
Destination       :

Splittable        : No      Job setup          : No
Started task STC  : No      WTO              : No
AUTOMATION        : No      Fault-tolerant agent : No
WAIT              : No      z-centric agent    : No
VIRTUAL           : No      Dynamic            : No
Remote engine type :
Transport time    : 00.00
Duration          :
Last updated by   : JANVC   on 11/09/08 at 18.51
```

---

Example 4-11 illustrates how the dummy workstation is used in job SAMPLED.

*Example 4-11 The use of a dummy workstation in TWS*

```
----- OPERATIONS ----- Row 1 to 1 of 1
Command ==> Scroll ==> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details, J - Edit JCL
Enter the PRED command above to include predecessors and number of conditions
in this list, or, enter the GRAPH command to view the list graphically.

Application          : SAMPLED          A job desc in ADGROUP

Row Oper      Duration Job name Operation text
cmd ws   no.  HH.MM.SS
'''' DUMM 015  00.01.00  SAMPLED_ _____
***** Bottom of data *****
```

## 4.2.5 Actual start time is not equal to end time of last predecessor

There are three types of dependencies in a batch schedule that a job waits for:

- ▶ A start time - Indicates the time the job is allowed to start, for example, after the close of business.
- ▶ A data set - Job B can start after job A delivers a data set that job B must process.
- ▶ Functional - Job B can start after job A processes a certain functionality. For instance, you process the payments and then process the account statements. Both job A and job B use the same databases.

One of the goals of the batch schedule is to have a continuous flow of jobs with no unused time in between. When there is a difference between the end time of the last predecessor and the start time of a job, then there is unused processing time between those jobs.

### Problem

Unused processing time between jobs can indicate that there is not optimal use of the batch window.

### Method to identify

Perform the following steps to identify the unused wait times:

1. Identify the predecessors of the job that you are investigating.
2. Compare the end times of the predecessors with the start time of the job.
3. If the latest end time is the same as the start time, then there is no problem.
4. If there is a difference, analyze what the job is waiting for. Some possible reasons are:
  - The job is waiting for a special resource.
  - The job is waiting for another resource, for instance, an initiator or a tape device.



## Sources needed to identify

You can use the following sources to find and analyze this anti-pattern:

- ▶ TWS to identify the predecessors of a job.
- ▶ TWS tracklog or SMF to identify end times and start times of jobs.
- ▶ SMF or joblog for further analysis.

## Solution

When you have found the reason for the delay, you can take the appropriate action. For instance, you might need to use a DASD data set instead of a tape data set, or if you are waiting for an initiator, you might need to increase the number of initiators.

## Benefits

One benefit is that you gain the time that was between the end time of the last predecessor and the start time of the job.

**Note:** This anti-pattern might also cover some other possible anti-patterns.

## Example

As depicted in Figure 4-1, there is a white space between Job 2 and Job 4. If the white space is removed then your total timeline becomes shorter.

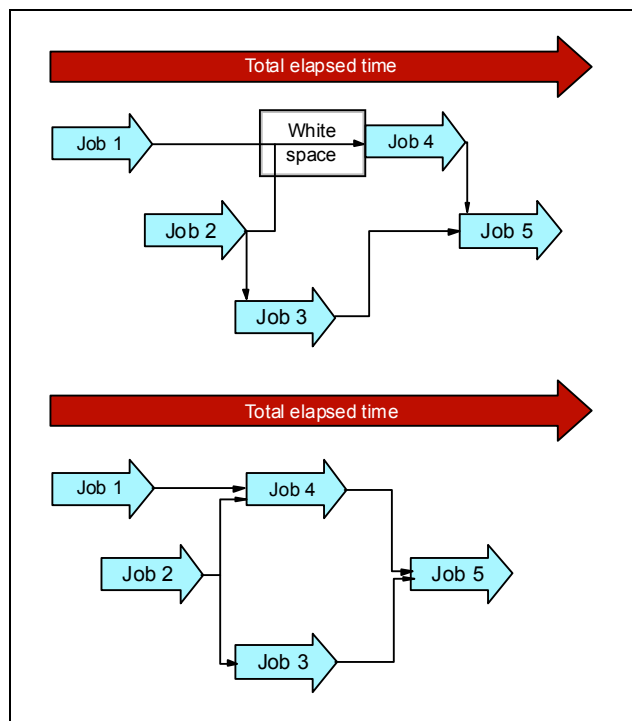


Figure 4-1 Timeline with white space

## 4.2.6 Workload Manager is not set up properly for batch

The anti-pattern regarding the Workload Manager (WLM) setup and how it relates to production batch is broadly defined. It also includes considerations for Tivoli Workload Scheduler (TWS) and the Job Entry Subsystem (JES) interactions with WLM. Resource

requirements, such as memory and processor capacity, should also be considered in the WLM setup.

## Problem

Most installations give a lot of thought and planning to how to set up WLM for the online day. However, the same amount of care is not always taken for the WLM setup for the batch window. The proper setup of WLM is even more critical when resources, such as processor cycles, are scarce.

Traditionally, processor capacity requirements were driven by the online day, and WLM was used primarily to allocate cycles to the most important workloads. Increasingly, capacity planning is now driven by batch as much as by the online day. With the growth of concurrent batch and online services, prioritizing work in the batch window also requires the protection of online services.

WLM primarily manages processor cycles. But the degree of parallelism, partially managed by limiting the number of batch initiators, can influence the amount of memory used in the batch window.

Often the interaction between WLM and TWS is not understood well. This situation can lead to unrealistic expectations. Similarly, the interaction between WLM and JES2 is subtle.

## Method to identify

Because it is broadly defined, an incorrect WLM setup can manifest itself in a number of different ways.

- ▶ Queuing for the processor is detected at the service class level, using IBM Resource Measurement Facility™ (IBM RMF™) Workload Activity Report data.  
Detecting or inferring queuing is not sufficient, the correct service classes must experience queuing for this to be a problem.
- ▶ Queuing for the processor is detected at the job level.  
Detecting or inferring queuing is not sufficient, the correct jobs must be queuing for this to be a problem.
- ▶ Important jobs are waiting for the initiators.

Incorrect expectations are another problem. For example, an installation might erroneously assume that:

- ▶ WLM directs batch work to specific systems.  
WLM can manage pools of server address spaces, such as initiators. It can start and stop initiators as necessary. It can balance a service class goal attainment against the system throughput and other service class goal attainment.  
It is the responsibility of the JES initiators to select work, on whatever system they are on. WLM does not direct work to specific systems and does not actively balance processor cycle consumption.
- ▶ WLM optimizes processing for critical path jobs identified with TWS.  
Although TWS can promote work to a different service class, there is no guarantee that:
  - It will run any faster.  
If there is no contention for processor cycles, or the job is not processor-bound, it probably will not run any faster.
  - The jobs are critical path jobs.  
Generally, TWS is working from estimates and cannot predict the future outcome.

## Sources needed to identify

For the examples in “Method to identify” on page 46, the data sources that can be used are:

- ▶ Queuing for the processor detected at the service class level, using the IBM RMF Workload Activity Report data.

Although this can be detected using the Workload Activity Report, we prefer to use System Management Facility (SMF) Type 72 records. Processing the SMF records allows you to graph such delays through the window, and enables you to identify when the problem occurs.

- ▶ Queuing for the processor is detected at the job level.

You can use job-level and step-level SMF data (Type 30 subtypes 4 and 5) or DB2 Accounting Trace (Type 101) records.

Although you do not directly see processor queuing delays, you can infer their likely presence, for example, in the following cases:

- In the SMF type 30 record the difference between processor cycles and elapsed time represents several things including I/O time. If the I/O time does not account for most of the difference, it might suggest that the processor queuing affects the job or step.
- In the SMF type 101 record the Unaccounted For Time often represents processor queuing delays, although there are other minor reasons for this time.

- ▶ Important jobs are waiting for initiators.

Use the SMF type 30 job end records to measure initiator delays for the most important jobs.

## Solution

The solution is to implement an effective batch-oriented WLM policy that favors critical batch over non-critical batch. Ideally, you would achieve this with a single policy that covers both daytime and overnight. However, it might prove impossible to have a single policy. If online work runs while your batch window WLM policy is in effect, ensure that the policy prioritizes the online work appropriately.

An effective batch policy has:

- ▶ A hierarchy of batch service classes, separated by WLM importance and perhaps velocity.
- ▶ JES job classes assign appropriate batch service classes.
- ▶ Appropriate service class period aging.

You can analyze processor and memory capacity requirements to determine whether the batch window drives these requirements more than the online day, then provision memory and processor resources accordingly.

Ensure that you understand how JES initiator-driven workload balancing works and how effective the interaction between TWS and WLM is in your environment. If, for example, near-critical jobs need help from WLM but are not getting promoted to a different service class, ensure that they are in this class anyway.

## Performance benefits

The performance benefits for performance-critical batch jobs include:

- ▶ Enough processor cycles to run as fast as possible
- ▶ No delays waiting for an initiator

## Example

One example is DB2 job delays.

### ***DB2 job delayed waiting for processor***

For example, a single-step DB2 job runs for 60 minutes and DB2 Accounting Trace shows that 20 minutes are spent consuming processor cycles, 20 minutes are spent waiting for DB2 I/O to complete, and the remaining 20 minutes are in Unaccounted For time.

Analysis of the system, using RMF, shows that the job runs during a time of high contention for processor cycles, the job does not run in the highest batch service class, and there is no paging.

Although the job is not on the critical path, its performance is important. Any significant extension to its elapsed time would put it on the critical path, especially if jobs on the current critical path were shortened.

Analysis of the job's current service class shows a significant number of Delay For CPU samples. This suggests that the service class is prone to queuing for processor cycles.

This job is suffering from processor queuing and it must be moved to a service class with better performance characteristics. This can be accomplished by using the TWS Critical flag or by manually changing the job class so that the job changes service class.

If this situation is pervasive, it might indicate the need for a processor upgrade or the need to tune down processor-intensive work of lower criticality.

## **4.2.7 Redundant copies of files**

Often different departments use copies of files to access data because the files are easy to use: they just rename the files and copy them to DASD that is managed by each department. For example, the customer administration department creates a file of the customer list, the sales department copies the file to their DASD, and then the delivery department copies the same file to their DASD.

### **Problem**

Copies of files are redundant and can cause problems. At first the files are the same, but if the original file is updated, the copied file is no longer the same. Redundant copies of files also occupy additional DASD space. If you copy them with DFSMSDss the files are locked and have exclusive enqueue, which means other batch jobs must wait for completion of the previous copy job. These copy jobs also consume resources such as CPU time and I/O time.

### **Method to identify**

There are different ways to identify redundant files. For example, there are copy statements in batch jobs. The JCL or scripting language has copy statements such as PGM=IEBCOPY, IEBGENER, IDCAMS, or ADRDSSU. If you copy files with your program, the program source has a copy statement. The original file name of a redundant copy can also be found in SMF. SMF type 14, 15, and 64 records shows the job names that opened and closed the original data set.

### **Sources needed to identify**

The following sources can be used to find and analyze this anti-pattern:

- ▶ JCL or scripting language
- ▶ Program sources
- ▶ SMF record dump

## Solution

Search for any redundant copies of files, then remove the copy statement in the JCL, scripting language, or program source. In addition, change the read statement for the copied files to the original files in the JCL, scripting language, or program source.

## Performance benefits

Removing the copy statement produces the following performance benefits:

- ▶ CPU time - The CPU time consumed by the redundant copy job is removed.
- ▶ I/O time - The I/O time consumed by the redundant copy job is removed.
- ▶ Elapsed time - The batch job does not wait for the completion of the other job.

## Examples

Figure 4-2 illustrates how different departments create redundant copies of the same file. In this example, Job A creates a customer file, then both the sales department and the delivery department make their own copies of the same file.

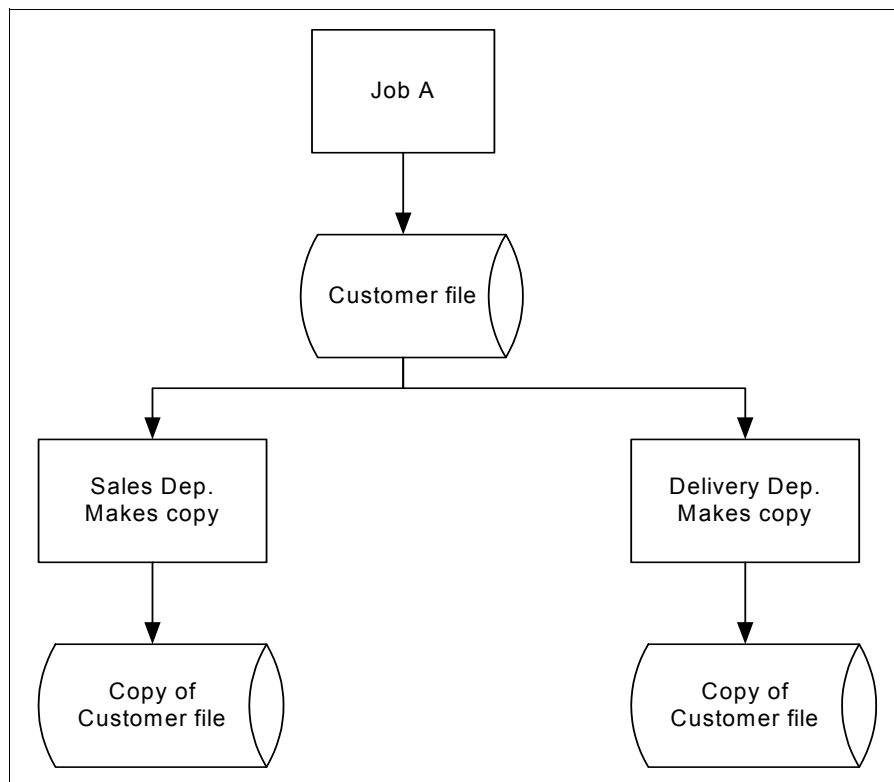


Figure 4-2 Different departments make their own copy of the same file

## 4.2.8 Redundant image copies

Batch jobs are used to make image copies of a database. These image copies are then used in conjunction with the database log file to recover the database if it is broken or corrupted. Image copies are copied frequently. For example, image copies for an important database are copied every hour.

## Problem

The image copies of a database are redundant. After the image copy was taken, only one image copy can be recovered with the use of log files. The image copy jobs consume many resources, such as CPU time and I/O time, and redundant image copies files also occupy DASD space. An image copy job locks the database while it is running, which means that any transaction must wait for completion of the image copy job.

## Method to identify

Image copy statements are in batch jobs. JCL or scripting language has image copy statement such as DSNUPROC.

## Sources needed to identify

The following sources are used to find and analyze this anti-pattern:

- ▶ JCL or scripting language
- ▶ The batch job schedule

## Solution

After searching for a redundant image copy of a database, remove the image copy statement in the JCL or scripting language.

## Performance benefits

Removing the image copy statement produces the following performance benefits:

- ▶ CPU time - CPU time consumed by the redundant image copy is removed.
- ▶ I/O time - I/O time consumed by the redundant image copy is removed.
- ▶ Elapsed time - Any other transaction batch job no longer has to wait for the completion of the image copy job.

## Examples

An example is the image copy jobs that are scheduled in TWS. The application description of the jobs has a run cycle with the *every* option. The *every* option schedules the application repeatedly, from the input arrival time until the repeat end time. The image copy job must be removed in the application description.

## 4.2.9 Batch prevented from effectively using sysplex-wide resources

This anti-pattern is broadly defined, covering various scenarios where work is prevented from utilizing all of the resources. Typically, these scenarios involve the static partitioning of the workload or the constraint of the entire workload to run on only a subset of the parallel sysplex members.

There is no general rule; each situation must be evaluated on an individual basis.

## Problem

To schedule work anywhere within the Parallel Sysplex (plex), there should not be any member-specific dependencies. For example, the work should run in a job class that has initiators on all systems.

The advantages of removing system affinities include:

- ▶ The ability to use resources wherever they become available, most notably processor cycles.

- ▶ The ability to keep the batch running even when one member fails.
- ▶ The ability to avoid the exclusive resources of a failed member, for example DB2 retained locks.

These advantages can be significant, particularly when resources would otherwise be constrained. Examples of these include:

- ▶ When all the memory on the members is needed to drive hard with Data In Memory (DIM).
- ▶ When processor capacity would otherwise be insufficient to support increased parallelism.

Disadvantages include:

- ▶ The potential for additional costs, for example, the propagation of DB2 locks between members.
- ▶ Additional scheduling complexity for work that must run together. For example, two jobs using a BatchPipes/MVS pipe could run on different members in a sysplex. If they ran on two members, using BatchPipePlex, they might run slower than if the two jobs ran on the same member.

The balance of batch work is another consideration. The idea that batch work must be balanced across the members of a sysplex is an attractive one. However, technically work must be routed in a manner that meets the installation's batch goals, regardless of whether it results in equal processor utilization.

### **Method to identify**

Generally, installations are aware of whether they have rigidly partitioned their batch and this can be confirmed by examining the job end data. The impact on system usage can be assessed using system-level data.

An installation should determine whether there are any technical inhibitors to a more flexible batch environment, and attempt to resolve them. For example, the installation can examine whether the DB2 Global Locking cost would be excessive if rigid partitioning was abandoned.

### **Sources needed to identify**

The following sources can be used to find and analyze this anti-pattern:

- ▶ At the job level, the system it runs on is in the SMF header of Type 30 job end records.
- ▶ If the job uses DB2 and DB2 subsystem affinity, both the system and the DB2 subsystem names can be found in the DB2 Accounting Trace (Type 101) record.
- ▶ System conditions caused by rigidity in scheduling can be identified using RMF (Types 70 to 79) SMF records.
- ▶ DB2 subsystem conditions, such as excessive lock propagation as a result of not partitioning, can be analyzed using both Type 101 and DB2 Statistics Trace (Types 100 and 102) SMF records.

### **Solution**

To solve this anti-pattern, implement these two changes:

- ▶ Resolve, or mitigate, any technical inhibitors to flexible running. For example, manage DB2 locking contention down.
- ▶ Alter the setup to allow more flexible scheduling. For example, change the Tivoli Workload Scheduler (TWS) setup so that jobs are scheduled flexibly across the sysplex.

## **Performance benefits**

The performance effects depend on the situation and range from negative to very positive. A positive effect is, that after unnecessary inhibitors have been removed, the spare processor cycles can be accessed more easily. Conversely, an increase in locking effects might be detrimental.

## **Examples**

Inflexible partitioning and balancing are examples of this situation.

### ***Inflexible partitioning***

As an example, an installation divides its work into 16 identical cloned streams of jobs. The data is mainly logically partitioned into 16 and some of the data cannot be effectively partitioned.

Each of the two members of the plex runs eight streams. Every few weeks, operations decides the streams to run on each member based on the balance or imbalance of run times between the 16 streams. This decision remains in effect until the next rebalancing is done.

This scheme is meant to minimize inter-DB2 locking costs and conflicts. But, it has a number of disadvantages, including:

- ▶ Workload cannot be balanced across the two members and an instantaneous shortage in processor capacity on one member cannot be relieved by directing jobs to run on the other member.
- ▶ If one member fails, moving the batch over to the surviving member involves a lot of work.
- ▶ The data segregation implemented by this scheme is not perfect:
  - There is common data accessed by all the streams, so cross-member sharing costs cannot be entirely eliminated.
  - When other work shares the data with the batch streams and it cannot be divided, it results in the cross-members sharing costs.
- ▶ The effort to decide what streams should run on each member is substantial.

Over time, any inhibitors to flexible work scheduling across the plex must be removed.

### ***Balancing***

An installation notices that batch tends to run on the fastest member in a plex, leaving slower members much less busy, and this is a concern. The installation must be concerned if this is adversely affecting performance, whether of batch or other workloads. This is a common occurrence because the faster or higher capacity member often acquires more work. But, usually this is not harmful.

You need to understand whether this imbalance is delaying work. If it is delaying work, adjustments to the WLM policy might be required.





## Tools

In this chapter, we discuss at a high level some of the tools that can be used to improve and optimize batch processing. The following tools are covered:

- ▶ Tivoli Workload Scheduler for z/OS
- ▶ z/OS tools and components
- ▶ Source2VALUE, a vendor tool

## 5.1 Tivoli Workload Scheduler for z/OS

Tivoli Workload Scheduler for z/OS (TWS) provides mainframe batch and real-time workload scheduling. It is a valuable tool for batch analysis and optimization activities.

Among the many capabilities and benefits of TWS are:

- ▶ Critical path analysis
- ▶ Integration with Workload Manager (WLM), making it easier to manage and monitor jobs
- ▶ Management of real-time and calendar-triggered workloads

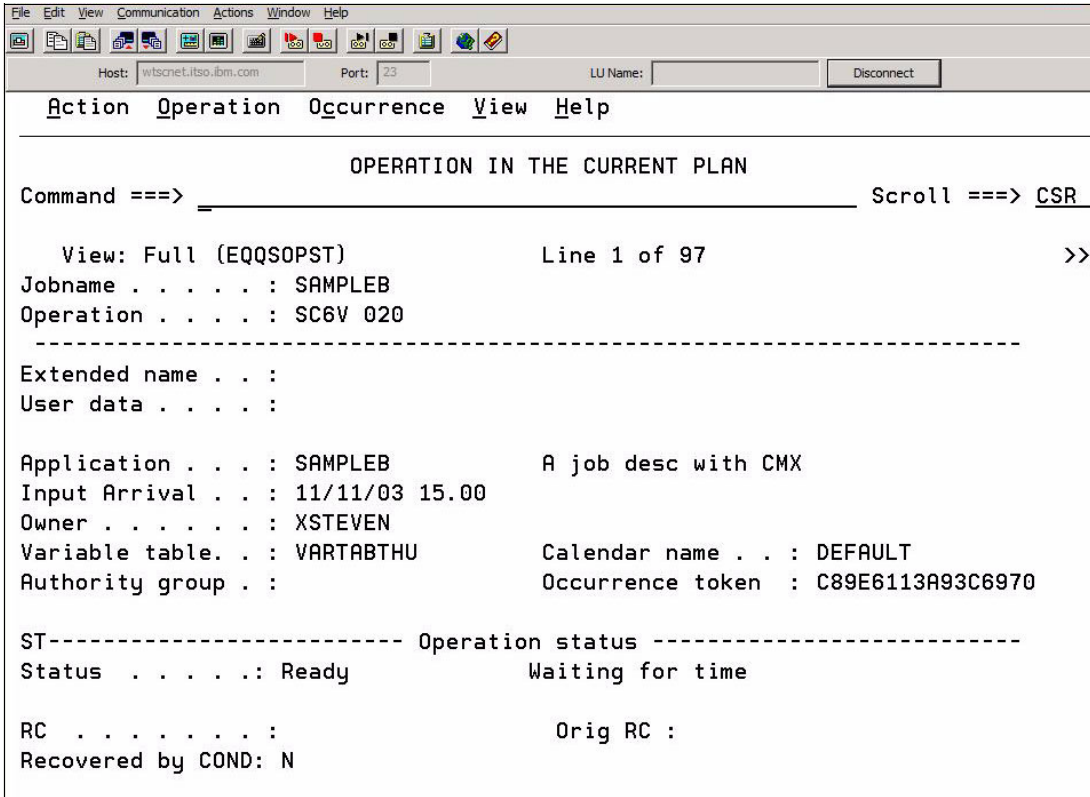
### 5.1.1 Operational efficiency

You can use TWS to review the operation processing of your current production batch to determine whether it is operating efficiently. It can provide information about your jobs and identify any dependencies.

#### List plan status and list dependencies

The Query Current Plan (QCP) and Modify Current Plan (MCP) ISPF panels provide information about the status of your current production. You can request detailed or summary information about individual applications, operations, or workstations, such as z/OS images. With QCP, you can also obtain summary information concerning all operations.

The QCP panel looks at the current plan and is continuously updated as the operations are processed. Figure 5-1 is an example of the *Operation in the Current Plan* view for the SAMPLEB job.



```
File Edit View Communication Actions Window Help
Host: wtsnet.itso.ibm.com Port: 23 LU Name: Disconnect
Action Operation Occurrence View Help
OPERATION IN THE CURRENT PLAN
Command ==> _____ Scroll ==> CSR
View: Full (EQQSOPST) Line 1 of 97 >>
Jobname . . . . . : SAMPLEB
Operation . . . . . : SC6V 020
-----
Extended name . . . :
User data . . . . . :
Application . . . . . : SAMPLEB A job desc with CMX
Input Arrival . . . : 11/11/03 15.00
Owner . . . . . : XSTEVEN
Variable table . . . : VARTABTHU Calendar name . . : DEFAULT
Authority group . . . : Occurrence token : C89E6113A93C6970
ST----- Operation status -----
Status . . . . . : Ready Waiting for time
RC . . . . . : Orig RC :
Recovered by COND: N
```

Figure 5-1 Operation status

The QCP panel can be used to determine why an operation has not started. The QCP and MCP panels can also be used to provide status information and to display a list of operations that ended in error.

With this information, you can decide whether intervention is required to speed up the processing of specific applications. You can display the applications that are most critical and those that have missed, or that are near to missing, the defined deadline. You should check and review this information before making modifications to the current plan.

You can also display a list of all the dependencies for an operation. This function is particularly beneficial to quickly identify the outstanding predecessors that have not completed and to determine the impact of an operation that has ended in error.

### Parallelism

Tivoli Workload Scheduler for z/OS has parallel servers on the workstation. TWS can control the parallelism of batch jobs. The maximum number of parallel servers is 99. TWS can submit more than 99 jobs at the same time if the parallel servers are not being used. The virtual workstation has a value of up to 65535 parallel servers at each destination.

**Note:** Increased parallelism is not guaranteed by creating more parallel servers. The batch workload must be capable of running in parallel to a greater degree.

## 5.1.2 Monitoring

You can use TWS for monitoring, for example, to monitor job errors and delays.

### Monitor job errors

TWS can monitor job return codes as well as job end codes. A return code can be one of two types:

**Highest return code** The highest return code of all the performed steps.

**Last return code** The return code of the last step.

You can specify the highest return code not in error with the *initial statements* parameter for all jobs or with each job definition in the application descriptions.

### Monitor job delays

TWS can monitor whether a job in the current plan is late. A job is considered late if it reaches its latest start time and does not have a status of started. The latest start time is calculated by its deadline minus each jobs duration time.

TWS can specify the normal highest return code with the *initial statements* parameter or with the application description for each job.

### Long duration jobs

TWS can monitor whether a job in the current plan is active for an unexpectedly long time. This means that the job must be active longer than its estimated duration.

### 5.1.3 Forecasting

TWS plans your production workload schedule. It produces both high-level and detailed plans. These plans both drive the production workload and provide you with the status of the production workload on your system at any specified time. You can produce trial plans to forecast future workloads.

#### Long-term planning (LTP)

The *long-term plan* is a high-level schedule of your anticipated production workload. It lists, by day, the instances of job streams to be run during the period of the plan. Each instance of a job stream is called an *occurrence*. The long-term plan shows when occurrences are scheduled to run. It also shows the dependencies that exist between the job streams. You can view these dependencies graphically on your workstation, as a network, to check that work has been defined correctly. The LTP can help you to forecast and plan for heavy processing days. The long-term planning function can produce histograms to display planned resource use for individual workstations during the plan period.

You can also use the long-term plan as the basis to document your service level agreements (SLAs). It lets you correlate an SLA directly to your production workload schedules so that your customers can see when and how their work is to be processed.

The long-term plan provides a window to the future, and you can specify how far into the future that should be, from a span of one day to four years. You can also produce long-term plan simulation reports for *any* future date. Tivoli Workload Scheduler for z/OS can automatically extend the long-term plan at regular intervals. You can print the long-term plan as a report, or you can view, alter, and extend it online using the available dialogs.

Normally, most customers use the long-term plan as a monthly plan. This means that they extend the long-term plan to the following month on the last day of the month.

#### Current plan

The *current plan* is at the center of Tivoli Workload Scheduler for z/OS processing. It drives the production workload automatically and provides a way to check its status. The current plan is produced by a run of the batch jobs and it extracts from the long-term plan the occurrences that fall within the specified period of time by viewing the job details. The current plan selects a window from the long-term plan and makes the jobs ready to run. They are started based on the identified restrictions, for example, dependencies, resources availability, or time dependency.

The current plan is a rolling plan that can cover several days. A common method is to cover one to two days with regular extensions for each shift. The production workload processing activities are listed by minute.

You can either print the current plan as a report, or view, alter, and extend it online, by using the dialogs.

### 5.1.4 Critical path analysis

TWS can provide analysis for the critical jobs for the business.

#### Critical path

Tivoli Workload Scheduler for z/OS provides a *critical path* feature to automatically manage SLAs for milestone jobs. Milestone jobs are jobs that are critical to the business and must not miss their deadlines.

## Critical job

The user can flag *critical jobs* and Tivoli Workload Scheduler for z/OS can automatically calculate the critical path to those jobs. It will promote jobs on the critical paths that are late and might affect the critical jobs deadline. The promotion algorithm uses the WLM integration.

If an operation is critical for your business and must complete by the set deadline, you can specify that the operation be considered as the target of a critical path. Then during daily plan processing, a critical path that includes the internal and external predecessors of the target operation is calculated, and a specific dependency network is created. While the plan is running, the scheduler monitors the critical paths that are consuming their slack time and, these become more critical than the paths calculated at plan generation.

## 5.1.5 Tivoli Dynamic Workload Console

The Tivoli Dynamic Workload Console provides graphical view tools to manage your workload and to generate and run customized SQL reports. With IBM Tivoli Dynamic Workload Console (TDWC), you can be proactive and monitor your workloads and IT infrastructure resources.

### Plan view

You can use the *Plan view* for monitoring, to get a high-level representation of a plan of any type, and to display a filtered set of job streams and their mutual dependencies.

### Job stream view

You can use the *Job stream view* for monitoring, to get a graphical representation of a single job stream in plan. This view provides a direct way to work with a job stream and its dependencies.

### Impact view

The *Impact view* is an expandable graphical representation of job streams and jobs in plan. It provides a multilevel analysis of how job and job stream completion affects the plan.

By default, the Impact View displays all the internal predecessors and successors of the selected object and the first level of external predecessors and successors with Tivoli Dynamic Workload Console and with the Job Schedule Console (JSC). Jobs that belong to the same job stream but that are not predecessors or successors of the selected job are not displayed, unless you choose a deeper dependency level, or switch to the job stream view.

Figure 5-2 on page 58 shows the Tivoli Dynamic Workload Console Impact View for the SAMPLEB job.

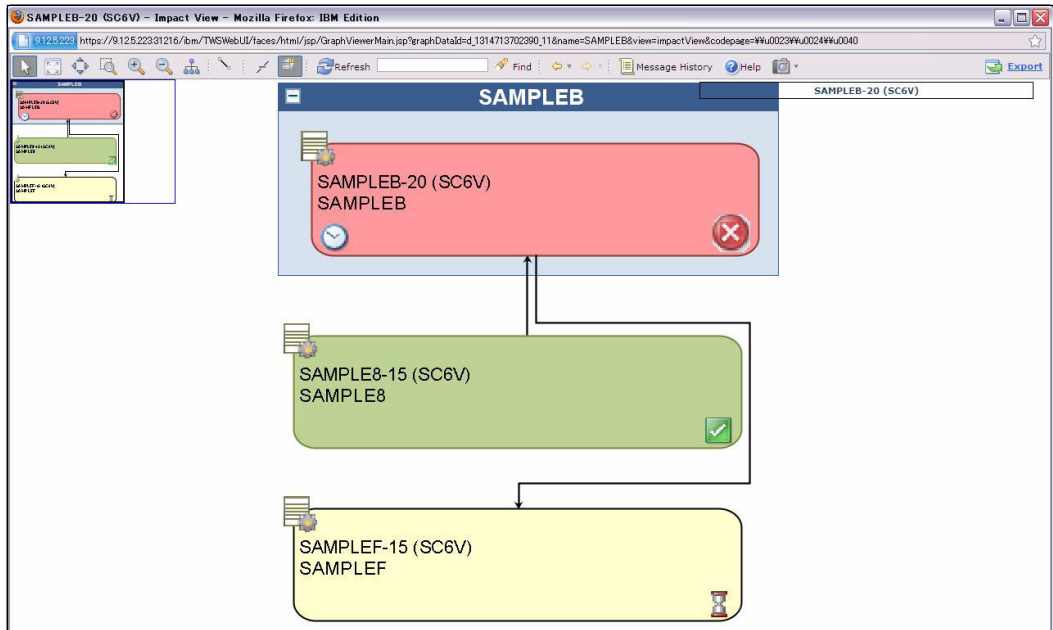


Figure 5-2 TDWC Impact View

Figure 5-3 shows the Impact View of the Job Schedule Console for the SAMPLEB job.

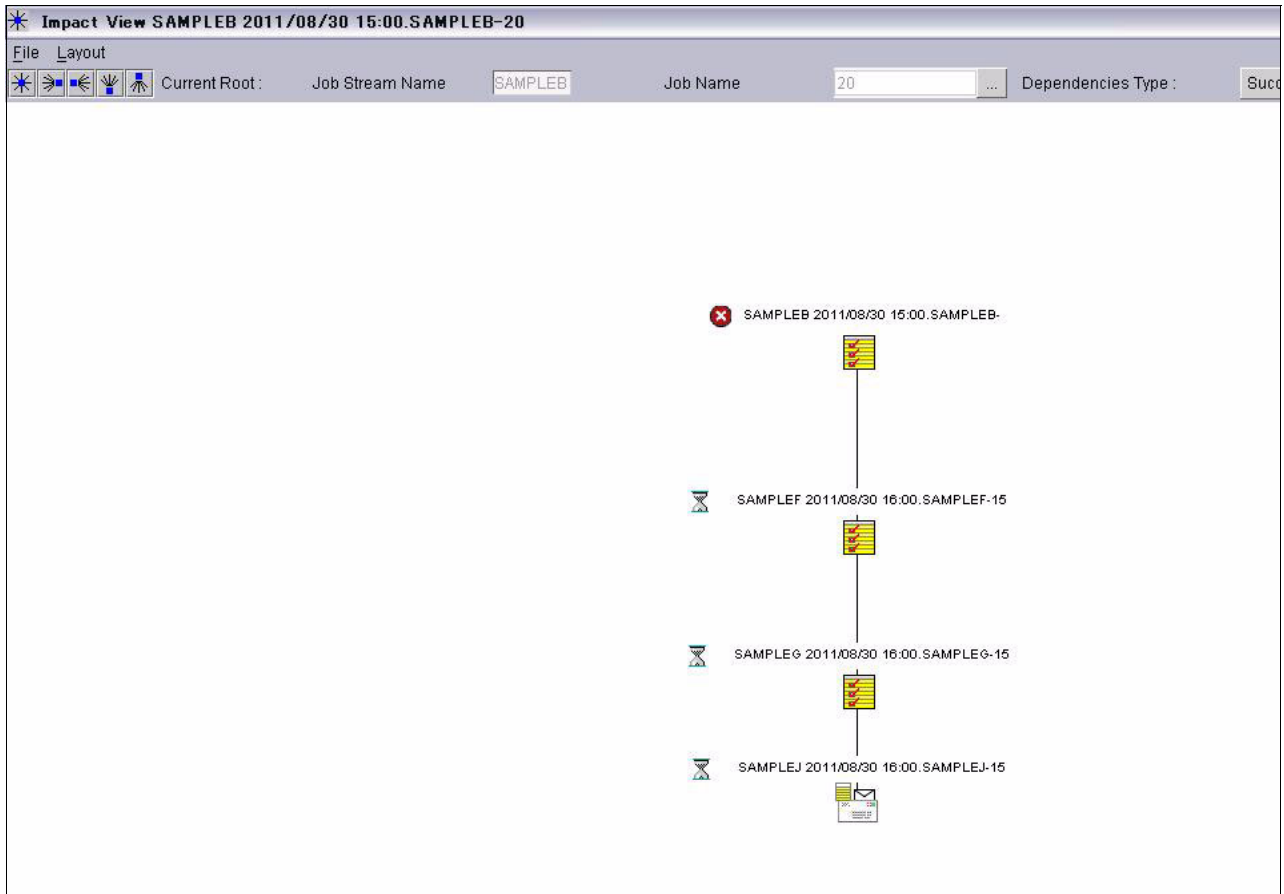


Figure 5-3 JSC impact view

## 5.1.6 Automation

With Tivoli Workload Scheduler for z/OS, you can plan and automatically schedule the production workload, meaning you can use it to increase your automation.

### Automatic job restart

When Tivoli Workload Scheduler for z/OS detects a job failure, it can automatically recover from the problem provided you have indicated what to do. This would be done through recovery statements in the JCL. When a job ends in error, Tivoli Workload Scheduler scans the JCL looking for statements of this type, and stops when it finds a statement where one of the conditions is true.

### Automatic job recovery

Tivoli Workload Scheduler for z/OS provides a Restart and Cleanup function to automatically restart any job in the plan, both failing and successful, from any step, including:

- ▶ Capability to recover the data set or catalog configuration to the state prior to the running of the job.
- ▶ Capability to restart the job at any step with optional adjustment of Generation Data Group (GDG) generation numbers to match the ones generated in one of the previous job runs.
- ▶ Automatic determination of *restartable* steps, based on referenced data sets, and the best step to restart the job from data set cleanup and data set protection.

### Application description feedback

Tivoli Workload Scheduler for z/OS automatically monitors the actual duration of operations. It can use these durations to modify the estimates in the application description database.

### Resource distribution

Resource distribution can be done using the IBM Workload Manager (WLM) service class. A Tivoli Workload Scheduler for z/OS operation integrates with IBM Workload Manager to provide the ability to dynamically route workloads, based on resource availability and capacity.

## 5.1.7 Sysplex exploitation

Tivoli Workload Scheduler for z/OS Workstation integrates with IBM Workload Manager, a z/OS component, to provide the ability to dynamically route workloads to the best available IBM System z resources, based on resource availability and capacity.

Integration with WLM is provided in either of two ways:

- ▶ Using the WLM scheduling environment (SE)

WLM SE is an attribute of Tivoli Workload Scheduler z/OS operations. WLM SE availability status is checked before the job is submitted to avoid queuing jobs that cannot be executed. Jobs are automatically resubmitted when the SE availability status changes.
- ▶ Using WLM service class

A Tivoli Workload Scheduler for z/OS operation that is on a critical path and that is running late can be automatically promoted to a better performance WLM service class (refer to “Critical path” on page 56). You can associate different WLM service classes to different operations, which provides you with the flexibility to accelerate or slow down your workload.

## Virtual workstation

The use of virtual workstations improves workload balancing and the monitoring of system availability. This feature automatically directs the submission of a workload to different destinations and removes the need to associate a workstation to a specific destination. You can define a list of destinations for the submission of a workload and the scheduler distributes the workload to automatically selected active destinations, according to a round-robin scheduling approach.

You can activate this feature by specifying the new virtual option at the workstation definition level. This option is allowed for computer workstations with the automatic reporting attribute, and is supported by all the interfaces available to define, modify, and monitor workstations.

With the use of virtual workstations, the scheduler distributes the workload across your trackers evenly, thus avoiding bottlenecks when you submit or run jobs. In fact, the scheduler splits the workload among the available destinations, so that the Job Entry System (JES) and Workload Manager (WLM) do not find overloaded input queues when they select a job action.

## 5.1.8 Charting

Charting can provide a graphical representation of the jobs and job streams.

- ▶ The Tivoli Dynamic Workload Console is a web-based graphical user interface for Tivoli Workload Scheduler for z/OS that enables you to monitor and control scheduled objects. It also simplifies report creation and customization.
- ▶ The Tivoli Job Scheduling Console is an older graphical user interface that runs on Windows, Linux, and UNIX, which provides charting. The newest Tivoli Workload Scheduler 8.6 does not yet officially support the Job Scheduling Console.

### Gantt charts

Gantt charts provide a table view for the current plan in a graphical layout, showing timelines of jobs and job streams. This view shows the planned and the actual run time of jobstreams graphically. The Job Scheduling Console is shown in Figure 5-4.

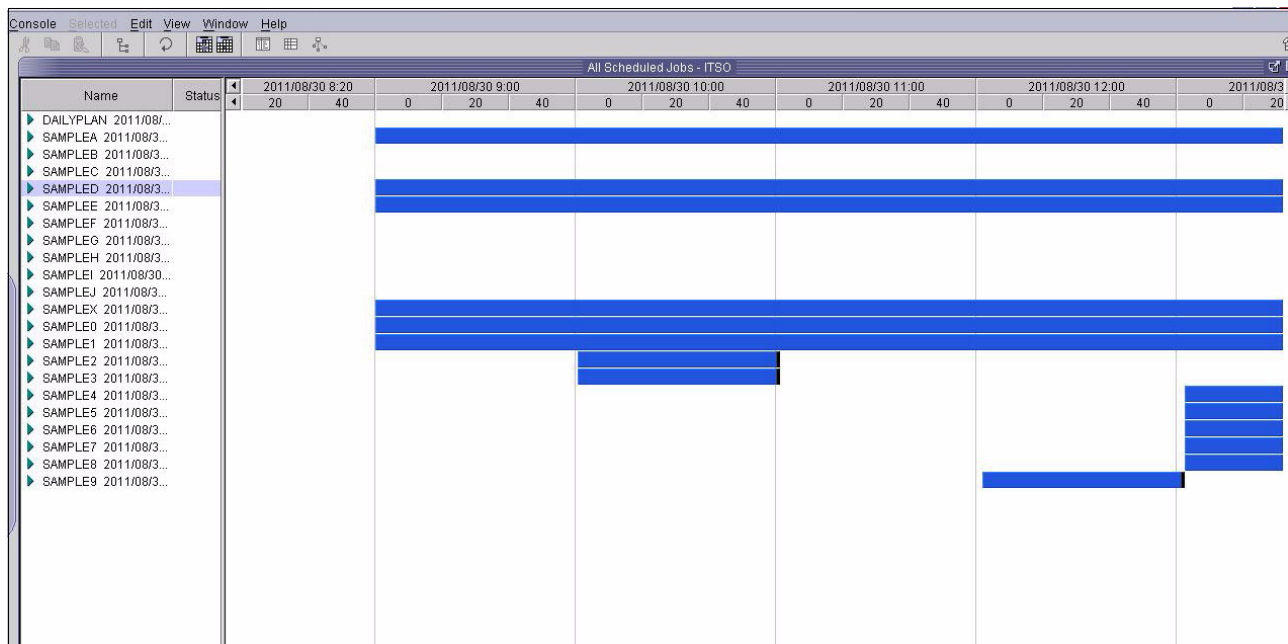


Figure 5-4 Job Scheduling Console table view



## Dependency diagrams

Dependency diagrams can be used for modeling and monitoring job streams.

- ▶ Job stream view (for modeling) is a graphical extension to the Workload Designer that shows a graphical representation of the job stream definitions in the database. It provides an intuitive way to create and maintain them.
- ▶ Job stream view (for monitoring) is a graphical representation of a single job stream in plan. It provides a direct way to work with it and its dependencies (see Figure 5-5).

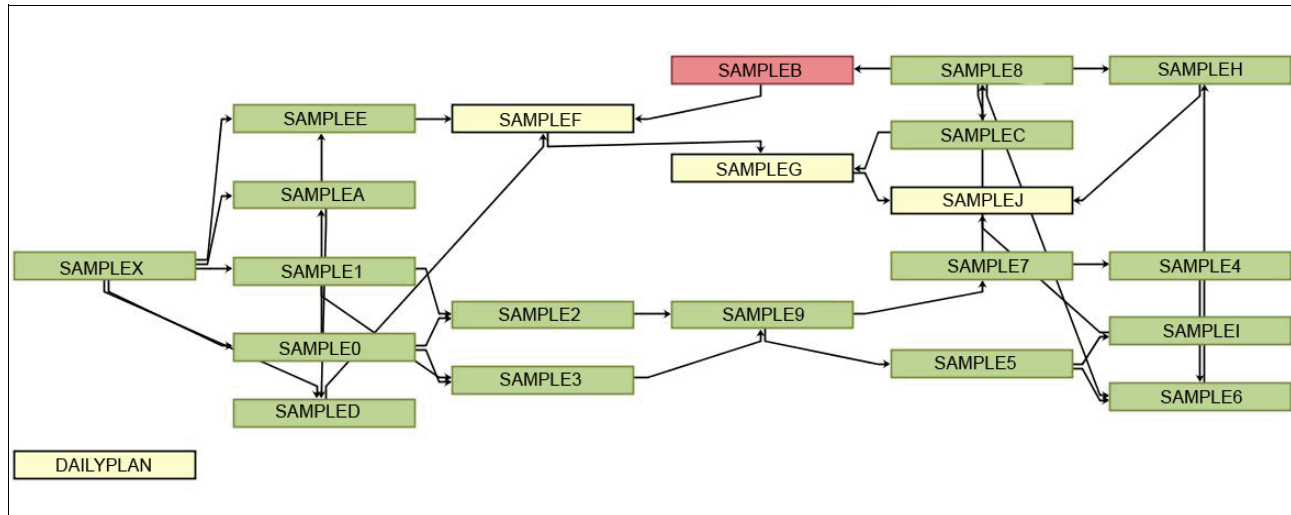


Figure 5-5 TDWC job stream view

### 5.1.9 Check a dependency loop

You can validate your plan with Tivoli Workload Scheduler for z/OS. You can use Tivoli Workload Scheduler to analyze a loop and report on the operations.

You can detect a loop using a trial plan. To help you correct a dependency loop, the daily planning program analyzes the loop and reports only those operations directly involved in the loop rather than all operations in the network. For example, if the daily planning process detects a loop in a chain of dependent jobs, a new current plan is not created.

### 5.1.10 Statistics analysis

You can obtain statistics with log analysis and provide reports with Tivoli Dynamic Workload Console.

#### Analysis track log

You can create a report of track log events. The following options are available when creating a report:

- ▶ Audit, debug, or both
- ▶ Audit created through job batch submission

When creating the report, the only required value is the type of input for the report. If you do not provide data for the remaining fields, all the records in the input files are selected. The output of the report is written to the file specified during installation.

## Reporting

The standard and user-defined reports on the Tivoli Dynamic Workload Console can:

- ▶ Display product plan details
- ▶ Track and tune workload capacity
- ▶ Control the timely workload execution according to goals and SLAs

Figure 5-6 is an example of Tivoli Dynamic Workload Console web reporting.

| Job Run History Listing |              |                 |                             |            |             |                          |                      |                            |                               |                               |                             |               |            |                  |                |
|-------------------------|--------------|-----------------|-----------------------------|------------|-------------|--------------------------|----------------------|----------------------------|-------------------------------|-------------------------------|-----------------------------|---------------|------------|------------------|----------------|
| SAMPLE0                 |              |                 |                             |            |             |                          |                      |                            |                               |                               |                             |               |            |                  |                |
| Job Name                | Status       | Job Stream Name | Scheduled Time (Job Stream) | Job Number | Workstation | Workstation (Definition) | Actual Start         | Actual Duration (hh:mm:ss) | Estimated Duration (hh:mm:ss) | Started Late (delay hh:mm:ss) | Ended Late (delay hh:mm:ss) | Long Duration | Error Code | Iteration Number | Job Identifier |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/2/11 9:00 AM GMT-4        | 5          | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:27:00                      |                             | N             |            | 1                | 7539           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/2/11 9:00 AM GMT-4        | 24         | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:25:00                      |                             | N             |            | 1                | 7546           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/2/11 9:00 AM GMT-4        | 15         | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:26:00                      |                             | N             |            | 1                | 7541           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/1/11 9:00 AM GMT-4        | 20         | SC6V        | SC6V                     | 9/2/11 8:54 AM GMT-4 | 00:00:01                   | 00:01:00                      | 16:02:00                      | 15:54:00                    | N             |            | 1                | 7372           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/1/11 9:00 AM GMT-4        | 5          | SC6V        | SC6V                     | 9/2/11 8:54 AM GMT-4 | 00:00:01                   | 00:01:00                      | 21:58:00                      | 15:54:00                    | N             |            | 1                | 7365           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/1/11 9:00 AM GMT-4        | 24         | SC6V        | SC6V                     | 9/2/11 8:54 AM GMT-4 | 00:00:01                   | 00:01:00                      | 21:56:00                      | 15:54:00                    | N             |            | 1                | 7373           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/1/11 9:00 AM GMT-4        | 10         | SC6V        | SC6V                     | 9/2/11 8:54 AM GMT-4 | 00:00:01                   | 00:01:00                      | 16:03:00                      | 15:54:00                    | N             |            | 1                | 7370           |
| SAMPLE0                 | ✓ Successful | SAMPLE0         | 9/1/11 9:00 AM GMT-4        | 15         | SC6V        | SC6V                     | 9/2/11 8:54 AM GMT-4 | 00:00:01                   | 00:01:00                      | 21:57:00                      | 15:54:00                    | N             |            | 1                | 7369           |
| SAMPLE1                 |              |                 |                             |            |             |                          |                      |                            |                               |                               |                             |               |            |                  |                |
| Job Name                | Status       | Job Stream Name | Scheduled Time (Job Stream) | Job Number | Workstation | Workstation (Definition) | Actual Start         | Actual Duration (hh:mm:ss) | Estimated Duration (hh:mm:ss) | Started Late (delay hh:mm:ss) | Ended Late (delay hh:mm:ss) | Long Duration | Error Code | Iteration Number | Job Identifier |
| SAMPLE1                 | ✓ Successful | SAMPLE1         | 9/2/11 9:00 AM GMT-4        | 15         | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:26:00                      |                             | N             |            | 1                | 7544           |
| SAMPLE1                 | ✓ Successful | SAMPLE1         | 9/2/11 9:00 AM GMT-4        | 5          | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:27:00                      |                             | N             |            | 1                | 7538           |
| SAMPLE1                 | ✓ Successful | SAMPLE1         | 9/2/11 9:00 AM GMT-4        | 25         | SC6V        | SC6V                     | 9/2/11 4:23 PM GMT-4 | 00:00:01                   | 00:01:00                      | 05:25:00                      |                             | N             |            | 1                | 7548           |

Figure 5-6 TDWC web reporting

## 5.2 z/OS components and features

This section describes some of the z/OS components and recent updates that can help you manage and improve batch performance.

### 5.2.1 Systems Management Facilities (SMF)

*Batch Modernization on z/OS*, SG24-7779 includes a review of the instrumentation that you can use to tune batch performance. This book continues to be a good resource because there have not been significant changes in this area since its publication.

This section summarizes the SMF instrumentation that is most relevant to batch. Table 5-1 provides details about some of the SMF records written by the Resource Measurement Facility (RMF) and z/OS, as well as records written by specific software such as DFSORT and DB2.

Table 5-1 SMF records

| SMF Record Type   | Written by                          | Information provided   |
|-------------------|-------------------------------------|--|
| Types 70 to 79    | Resource Measurement Facility (RMF) | <ul style="list-style-type: none"> <li>▶ System level - Including processor, logical partition, memory, disk, Parallel Sysplex information</li> <li>▶ Workload Manager (WLM) - Workload and service class resource usage and performance attainment information.</li> </ul>  |
| Type 30           | z/OS                                | Information at the address space level: <ul style="list-style-type: none"> <li>▶ Interval records - Good for long-running address spaces, such as DB2</li> <li>▶ Job and step-end records - Timing and resource consumption information for individual batch jobs</li> </ul> |
| Types 14 and 15   | z/OS                                | Performance for non-VSAM conventional data sets  |
| Types 62 and 64   | z/OS                                | Performance for VSAM data sets   |
| Type 92           | z/OS                                | Performance for Hierarchical file systems- HFS and zFS   |
| Type 16           | DFSORT                              | Detailed record for each invocation of DFSORT  |
| Type 101          | DB2                                 | Step-level <i>Accounting Trace</i> , more detailed than step-level summary, down to package level  |
| Types 100 and 102 | DB2                                 | Subsystem-level <i>Statistics Trace</i> records on a timer interval  |
| Type 102          | DB2                                 | Extremely detailed <i>Performance Trace</i> records that can be used, along with more static information, to perform SQL tuning  |

SMF records have numerous advantages, most notably that of permanence. You can continue to process SMF records for months after they were written. You can also process records from different days, for example, to check how a job's run time varies over time or varies with I/O count.

To be able to process historical SMF data, you must design your SMF retention policy carefully. Collect key metrics into a performance database, such as IBM Tivoli Decision Support. Such metrics include run times, processor consumption, and I/O counts at the job and step level. These allow you to perform trend and variability analysis.

It is impractical to retain some high volume record types indefinitely, including those related to data set access, and DB2 Accounting Trace.

## 5.2.2 Step-termination exit

When a job or step terminates, an exit routine installed at the IEFACRT exit point receives control from the system. The exit routine is called whether the step ended normally or abnormally.

Every time certain types of SMF records are about to be written the exit is called, and it has a pointer to the record in memory. The types that the exit is called for are 4, 5, 30, 32, 34, and 35.

Most installations write only Type 30 records because, for batch, Type 30 largely replaces the other types of records. There might be more than one Type 30 record at step or job termination, and the exit would be called for each one. The exit has access to the entire SMF record data, including zIIP and zAAP CPU information. Most installations have an IEFACTRT routine that provides some basic information about the step.

IBM supplies two sample exit routines for this exit point. One of them, IEEACTRT, writes a summary of the step in the job's log. The IEEACTRT sample shipped with z/OS Release 7 includes support for zAAP and zIIP time.

Tivoli Workload Scheduler for z/OS also supplies sample exit routines for the tracker, to obtain job and step termination information.

In the absence of SMF analysis, perhaps because it has not been processed yet, the output from this exit can be a useful piece of instrumentation. You can use the System Display and SPOOL Facility (SDSF) to examine the job output, either while it is running or after it has completed. However, this is a transient mechanism in comparison to SMF.

### 5.2.3 Batch improvements in z/OS 1.13

Various updates were made in z/OS 1.13 to improve batch job processing. These updates include:

- ▶ Support for instream data in PROCs and includes
- ▶ A mechanism for the control of job return codes
- ▶ The ability to spin any SPIN data set by time, size, or operator command
- ▶ A feature to requeue a job by command on a step boundary

For further details, refer to *Batch Modernization on z/OS*, SG24-7779.

## 5.3 Vendor tool - Source2VALUE

In this section, we review one vendor tool, Source2VALUE from Omnext.

You can use Source2VALUE from Omnext to perform the batch schedule analysis. Source2VALUE is a cloud-based software analysis tool that provides insight regarding the quality, productivity, and functionality of software systems.

Source2VALUE performs an *MRI Scan* of the software source code, and other sources like workflow data, batch schedule information, and performance data. This scan provides a holistic view of the quality, size, design, and change behavior of the sources. The results of the scan are presented through a web portal that supports enhanced reporting, overviews, drill-down functionality, metrics, and productivity measurements. The Source2VALUE web portal accelerates analysis by presenting all relevant information in an attractive and hyperlinked way.

### 5.3.1 Source2VALUE operation

During processing of the source code, Source2VALUE performs the following steps:

- ▶ Parse the sources
- ▶ Resolve relations (cross reference)
- ▶ Calculate quality and size metrics

- ▶ Check standards and guidelines and anti-patterns
- ▶ Detect code duplication
- ▶ Detect dead code
- ▶ Create design models
- ▶ Analyze differences with previous version
- ▶ Store all information in the Data Warehouse

Figure 5-7 depicts the process flow of Source2VALUE processing.

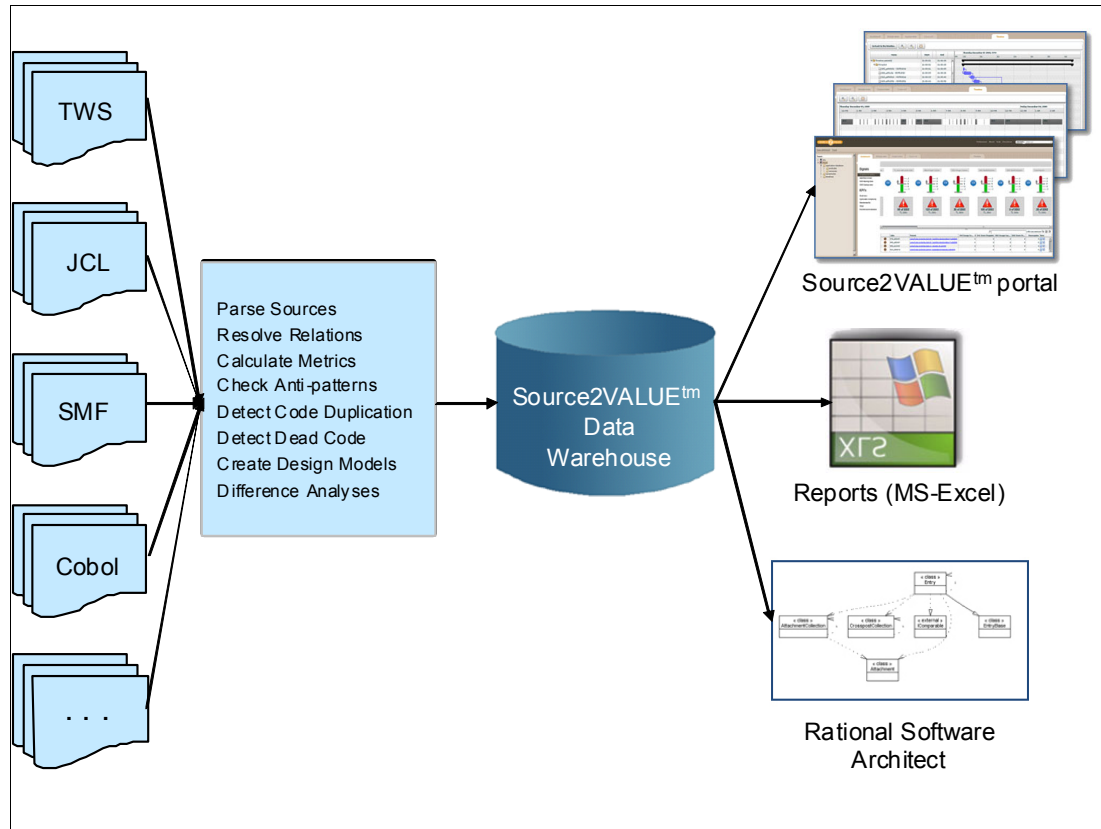


Figure 5-7 Source2VALUE processing

### 5.3.2 Source2VALUE output

The Source2VALUE Data Warehouse is the source for all reporting.

An extensive Source2VALUE Portal is created with the following functionality:

- ▶ Dashboard - The Dashboard identifies or signals all objects that do not comply with the configured Corporate Risk Policy. The Corporate Risk Policy defines the customer-specific or project-specific signals and thresholds that are checked in the source code.
- ▶ Source View - The actual source code can be examined in the Source View. Relations to other sources are hyperlinked.
- ▶ Design View - Depending on the type of source, the Design View shows Unified Modeling Language (UML) Class Diagrams, UML Sequence Diagrams, Flow Charts, or Gantt Charts.
- ▶ Code Duplication View - This view helps in identifying and analyzing *cloned* source code.

- ▶ Cross Reference View - This view shows all incoming and outgoing relations of a selected object.
- ▶ Standards and Guidelines View - This view helps to find all objects that do not comply with a selected Standard or Guideline.
- ▶ ISO-9126 Maintainability View - This view shows the maintainability aspects of the analyzed code according to ISO-9126.

All views in the portal have an export to MS-Excel option. Custom MS-Excel reports can also be created using the Source2VALUE Data Warehouse.

Design models can be exported to UML Modeling tools such as IBM Rational® Software Architect.

### 5.3.3 Source2VALUE for batch schedule performance optimization

Source2VALUE has been adapted and configured to be able to process and analyze batch schedule information. It now processes TWS Batch Loader scripts, TWS Current Plans, TWS Critical timelines, TWS Track Log data, JCL, and SMF records, and stores the information in the Source2VALUE Data Warehouse.

To locate performance optimization possibilities, you can find and analyze white spaces in critical timelines, and you can investigate anti-patterns in the batch schedule. Both types of analysis are supported by Source2VALUE.

#### **White space analysis and timeline comparison**

The white space overview of a batch critical timeline enables you to identify wait time in the batch critical timeline. Wait time is visualized by a white space in the Gantt chart of the critical timeline.

When you double-click a block in the Gantt chart that follows a white space, the tool displays the characteristics of the subsequent job and thus shows, for example, whether a start time or a dependency has been specified (see Figure 5-8 on page 67). Whether this wait time is justified can only be derived from the functional specifications of the batch critical timeline. If these specifications are not available, a discussion with the functional owner is necessary.

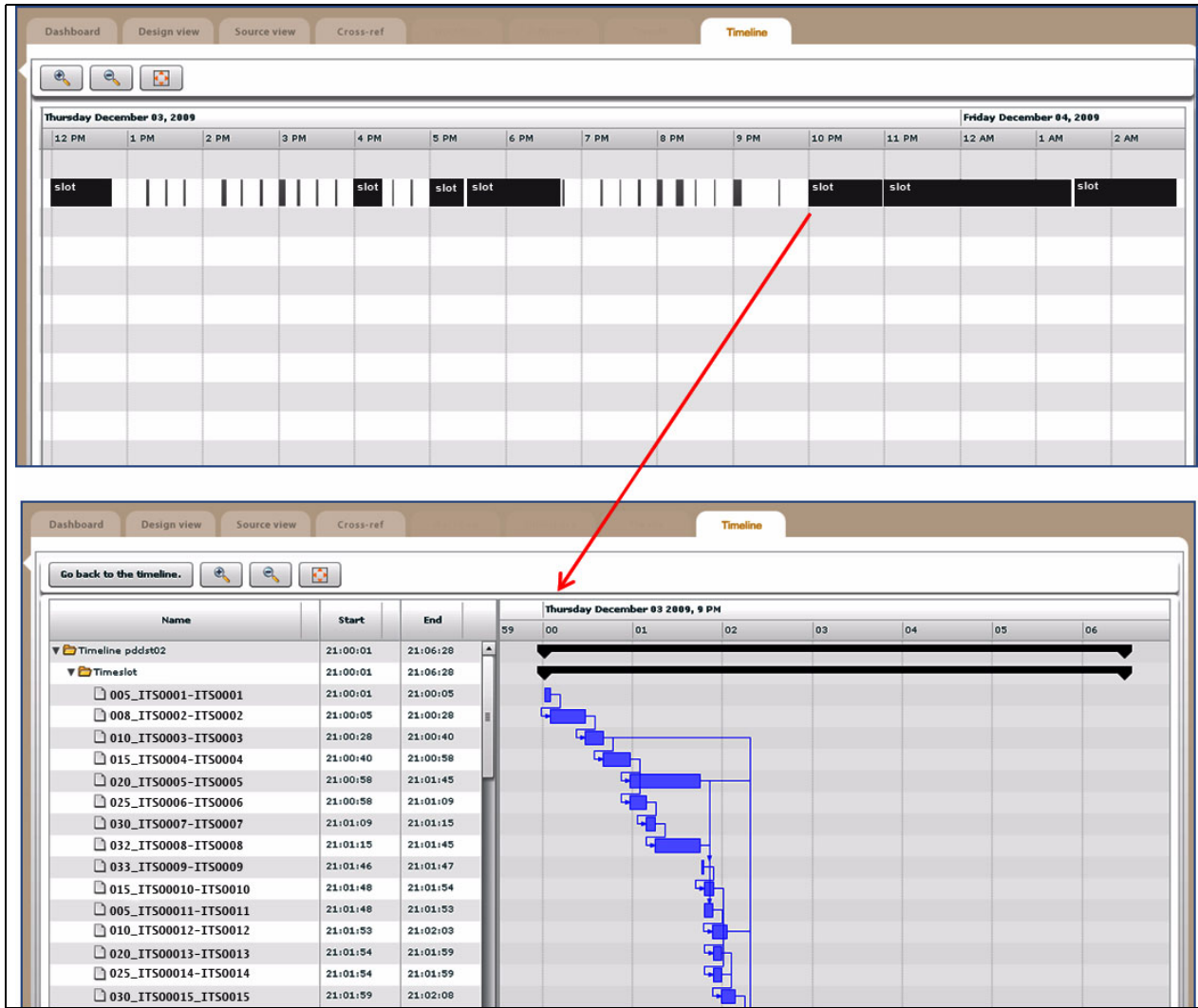


Figure 5-8 White space overview and Gantt chart with job details

In addition to analyzing wait times in critical timelines, the Gantt charts enable you to compare the course of a critical timeline during a consecutive number of batch runs with a calculated baseline.

Figure 5-9 on page 68 displays the same critical timeline for ten consecutive batch runs. The bottom Gantt chart represents the baseline of the critical timeline. The baseline is established by calculating the trimmed average start time and duration of a representative month worth of data.

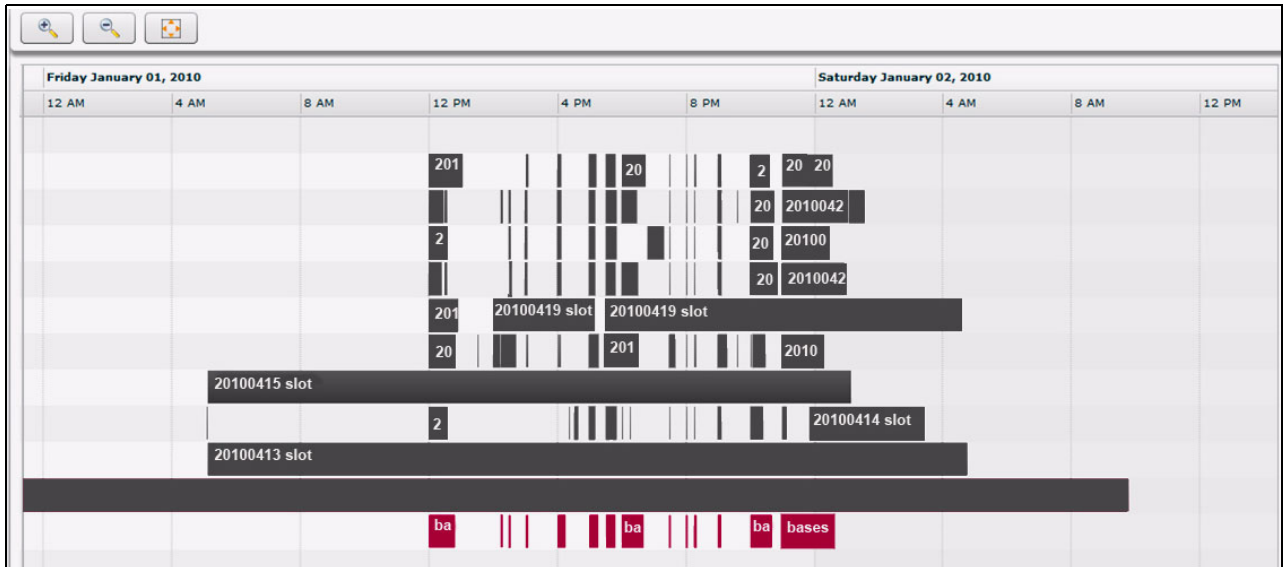


Figure 5-9 Baseline at bottom (in red) with 10 consecutive batch runs

By comparing the critical timelines during testing, you can easily detect long running processes as well as errors that occur during the testing. It is also possible to make a baseline for your production environment and actively compare your current production timelines with your baseline.

### Performance anti-pattern analysis

Another way to discover performance optimization possibilities is to perform an anti-pattern analysis on a batch critical timeline, starting from the Source2VALUE dashboard. The dashboard displays as many indicators as there are performance anti-patterns to be evaluated. For each performance anti-pattern, the dashboard documents the threshold that was set for the occurrence of the anti-pattern and the number of occurrences that are found to cross the threshold. For instance, if you want to see all jobs that started more than 900 seconds later than the baseline, the threshold is set to 900. If the anti-pattern does not have a specific threshold, the threshold is set to 1.

Figure 5-10 on page 69 shows a Source2VALUE Dashboard with performance anti-patterns displayed.



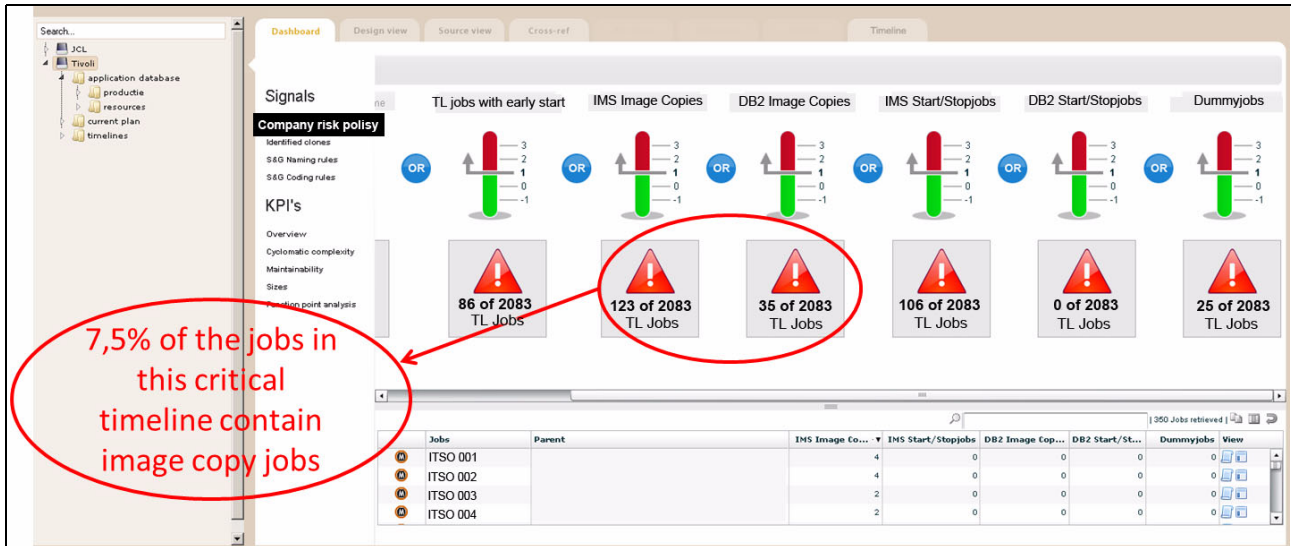


Figure 5-10 The Source2VALUE Dashboard with performance anti-patterns

The batch maintenance specialist can use these tools to analyze specific anti-patterns. When you click an anti-pattern in the overview, the result is a list of all occurrences of that anti-pattern.

Click the source view icon in the last column of the list to view detailed information about the selected occurrence, such as its job control information, relations with predecessors and successors, and system resource utilization.

Figure 5-11 is a display from the source and the cross reference views.

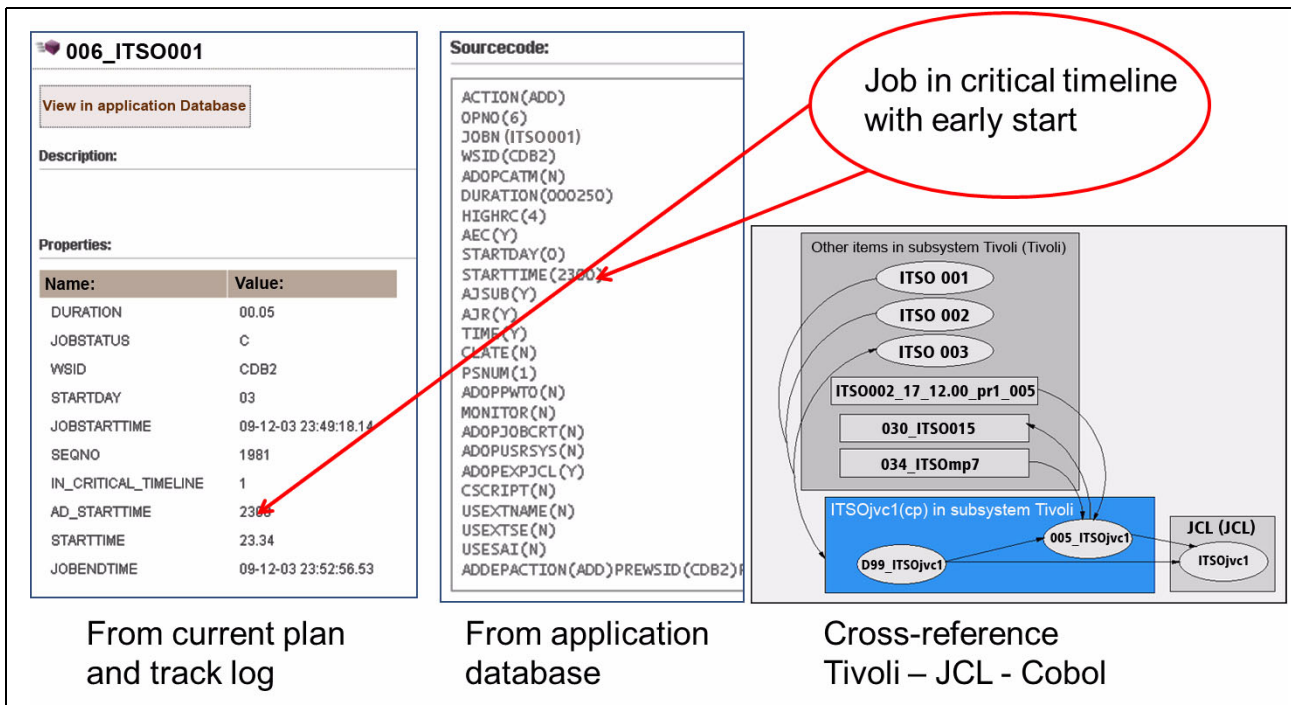


Figure 5-11 Details from the source view and cross reference view



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Batch Modernization on z/OS*, SG24-7779
- ▶ *System/390 MVS Parallel Sysplex Batch Performance*, SG24-2557
- ▶ *VSAM Demystified*, SG24-6105

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)







# Approaches to Optimize Batch Processing on z/OS



**Apply the latest z/OS features**

Batch performance optimization remains an important topic for many companies today, whether merging workloads, supporting growth, reducing cost or extending the online day.

**Analyze bottlenecks**

**Evaluate critical path**

This IBM Redpaper publication describes a general approach that can be used to optimize the batch window in a z/OS environment. This paper outlines a structured methodology using anti-patterns and tools that can be followed to increase batch productivity.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)