Tricks to Improve BPM Performance in SAP NetWeaver Process Integration

Applies to:

SAP NetWeaver PI 7.1 and 7.0, XI 3.0. For more information, visit the User Interface Technology homepage.

Summary

This guide will help you to improve BPM performance at run time using simple but very effective techniques that we often do not take into account while designing BPM in SAP PI. The scope of this document is to provide an overview of the necessary design tasks and tools for improving performance of BPM scenarios.

Author: Sunil Singh

Company: L&T InfoTech Ltd.

Created on: 15 January 2009

Author Bio

Sunil Singh is a SAP PI/XI Consultant in L&T InfoTech Ltd. His area of expertise includes SAP PI/XI, Java.

Table of Contents

1. Introduction

Performance of cross component Business Process Management (ccBPM) of SAP Net Weaver Process Integration is neither poor nor great, it is somewhere in-between. In SAP PI, BPM is professed as resource and time consuming process. If possible everyone wants to exonerate BPM while implement their business requirement.

But there are some features offered by SAP PI for BPM that could be used to curtail above shortcomings. This article briefs about such features of BPM.

The intention of this article is to bring in few concepts of BPM inspired by design in nutshell and to discern them in detail, references are given within the topic.

2. Message Packaging:

2.1 What is it?

It is a throughput increasing mechanism basically based on message group processing inside each pipeline step, thus taking the benefits of loaded ABAP program re-execution and massive database access.

2.2 The Concept

Message Packaging is message size dependent method. It is very suitable method in scenarios where in we have many small size messages (less than 10 MB) coming at a time. Message packaging increases throughput up to 3 times under such conditions. Message packaging provides special transaction for monitoring hence providing strength to monitoring of process.

For large size messages (greater than 12 MB) Message Split method improves the performance significantly.

2.3 Care to be taken

Packaging is intended for small asynchronous messages. It is not intended for scenarios in which it is important that individual messages are processed quickly.

2.4 Reference Links

https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/704de8f8-5806-2a10-10b5-c7b0d918822f

https://www.sdn.sap.com/irj/scn/wiki?path=/display/sandbox/Message%252bPackaging%252bfor%252bIntegration%252bProcesses

https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/10b54994-f569-2a10-ad8f-cf5c68a9447c

https://weblogs.sdn.sap.com/pub/wlg/8906

3. Parallel Processing:

3.1 What is it?

Parallel processing in BPM allows the message processing in reduced time, Hence it is very suitable in time critical processes.

3.2 The Concept

Parallel Processing Allows specific Integration processes to be executed with QoS EO, instead of EOIO by processing messages in multiple queues, taking the benefit of parallel processing.

3.3 Care to be taken

It consumes more system resources. While carrying POCs it was observed that Parallel Processing can trouble other processes which are not using parallel processing.

3.4 Reference Links

https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/f0e73c7b-5301-2a10-f1ab-832f301b6c02

https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/10195

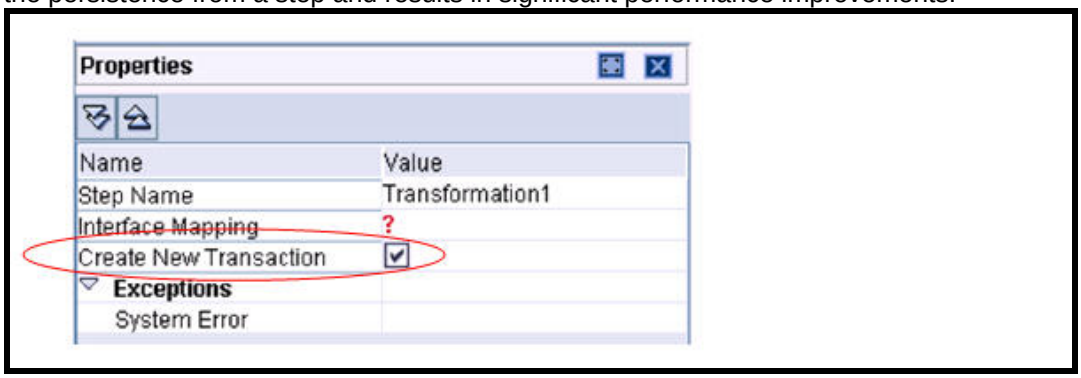
4. Enhanced Transaction Handling:

4.1 What is it?

Allows design time optimization of transaction handling, avoiding starting and ending one transaction for each integration process step. Enhanced transaction handling in BPE works as an add-on to BPE monitoring hence improving the monitoring feature of BPE.

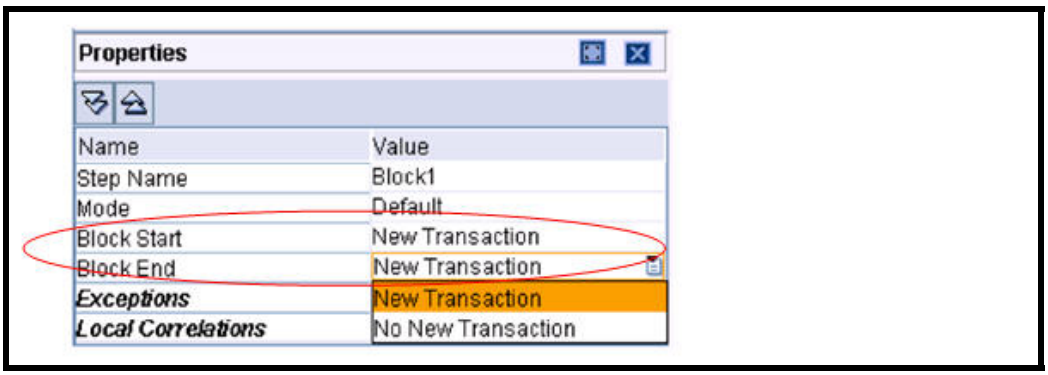
4.2 The Concept

With SP10, a new transactional behavior of ccBPM integration processes has been introduced. Before SP10**,** after each step in an integration process, the resulting data was persisted in the database, what adds overhead to the process and therefore affects performance. Now, with SP10, the persistence of these steps is configurable. As you can see in picture 1 showing the properties of an integration process step, you can now define if a new transaction needs to be created for a particular step. Not setting this checkmark removes the persistence from a step and results in significant performance improvements.



Picture 1: Simple check box config for send, transformation and receiver determination steps

For individual process steps, this behavior is available for the ‘send’, ‘transformation’ and ‘receiver determination’ step. However, as you can see in picture 2, one can also configure this for an entire ‘block’ and define the start of end of the block as the beginning of a new transaction.



Picture 2: Configuration of a block start and end

If a process instance at run time fails within a ‘block’, it is possible to restart the process from the beginning of the block if a transaction was created at the beginning of the block.
While you have to balance the use of longer running transactions with the need of memory, this new capability will improve ccBPM performance significantly.

4.3 Care to be taken

It is advisable to use it when extensive monitoring is required in process containing BPM as it creates extra burden on integration engine.

4.4 Reference Links

https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/10228

https://weblogs.sdn.sap.com/pub/wlg/4875

5. Suppress BPM Validation

5.1 What is it?

It is not a feature that BPM offers but can be seen as smart use of XSLT mapping so as to improve the performance.

5.2 The Concept

Messages, which pass BPM container, are validated for the Message Type referred in the abstract Interface.

The validation can be suppressed by serialization before and a deserialization after the Business Process. This can be done by a simple XSLT mapping, which transfers the content of a XML message to an unparsed string using the CDATA tag.

5.3 Care to be taken

Conditions given using X-path will never be fulfilled as with above technique the structure of payload gets changed.

5.4 Reference Link:

https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/3574

6. Reorganized Runtime Tables

6.1 What is it?

We have to reorganize tables related to workflow so as to improve the speed.

6.2 The Concept

ccBPMs in SAP PI are using the Workflow engine. As the system is running, index records are frequently created for the workflow runtime tables and then deleted again. This has a negative effect on the performance if we access the system using these indexes.

Depending on the data volume of the processes, indexes of the database runtime tables should be reorganized regularly (for example, on a monthly basis).

6.3 Reference

For more information refer SAP Note 72873

7. Processing Messages Locally Within the Block and Using Local Correlation

7.1 What is it?

In this we decide the scope of Message whether the message should be available throughout the process or only within the block.

7.2 The concept

Less the number of messages in global container more is the throughput. So it becomes added benefit if scope of the message/correlation is known while design phase**.**

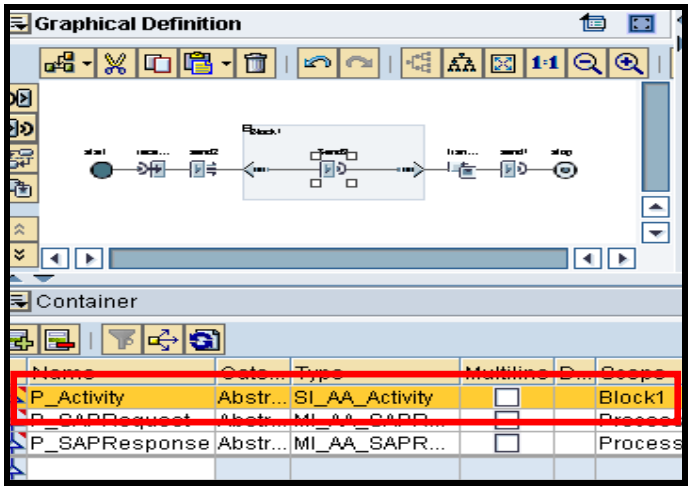


Fig. Assigning the Local Visibility to a Container Variable

7.3 Care to be taken

Scope of the message must be clearly known while deciding the scope of message/correlation otherwise it may lead to ambiguity at run time.

8. Other Points that Need to Be Considered While Designing

8.1 Avoiding the use of Transformation Steps

Transformation step consumes considerable resource and times while its execution. It is always good approach to mention the interface mapping in Interface determination in Integration Directory

8.2 Multi-mappings

While ccBPM allows you to easily collect and split messages, you can achieve the same using a multi-mapping and mapping based message split. Check the documentation for multi-mappings.

8.3 Sequential mappings

Sometimes we use ccBPM processes with the only purpose of receiving a message, performing two or three successive transformation steps, and then sending this message off. For such a process we can simply use one interface mapping with sequential mapping steps

8.4 Message collection

For message collection scenarios from single sender, try to collect messages on the sender system and send it out as a single message to your integration server.

8.5 Sync to Async Bridge

Previously, implementation of the Sync to Async Bridge (or vice versa) required the use of ccBPM. With SP10, the same can be achieved using modules “RequestResponseBean” and “ResponseOneWayBean”.

8.6 Single message loops

Do not use loops and process single messages at a time. Instead, collect your messages first and combine into one large message with several business documents inside.

8.7 Large messages

Do not use ccBPM for processing of large messages, especially in collect and split patterns. Start paying close attention to resource consumption when using messages above a size of 20 MB; remember this is highly dependent on your available hardware resources. - And while XSLT mappings should be avoided for large messages in general, this applies especially for ccBPM

Conclusion

If ccBPM is designed properly then it is very prevailing while if we use BPM just because we have to then we have to pay the price, either in extra hardware or in poor performance.

Related Content

https://weblogs.sdn.sap.com/pub/wlg/2927

http://help.sap.com/saphelp_nw04s/helpdata/en/21/6faf35c2d74295a3cb97f6f3ccf43c/frameset.htm

https://weblogs.sdn.sap.com/pub/wlg/3115

https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/704de8f8-5806-2a10-10b5-c7b0d918822f

https://www.sdn.sap.com/irj/scn/wiki?path=/display/sandbox/Message%252bPackaging%252bfor%252bIntegration%252bProcesses

https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/10b54994-f569-2a10-ad8f-cf5c68a9447c

https://weblogs.sdn.sap.com/pub/wlg/8906

https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/f0e73c7b-5301-2a10-f1ab-832f301b6c02

https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/10195

For more information, visit the User Interface Technology homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.