A SOA Middleware for High-Performance Communication

M.Swientek[1,2,3], B.Humm[1], U.Bleimann[1] and P.S.Dowland[2]

[1]University of Applied Sciences Darmstadt, Germany
[2]Centre for Security, Communications and Network Research,
University of Plymouth, United Kingdom
[3]Capgemini sd&m AG, Offenbach, Germany
e-mail: martin@swientek.org

Abstract

Systems for bulk data processing are often implemented as batch processing systems. While this type of processing in general delivers high throughput, it cannot provide near-time processing of data. Message-based solutions such an ESB are able to provide near-time processing but cannot provide high throughput. This paper presents a new approach to the problem of delivering near-time processing while providing very high throughput by adjusting the data granularity at runtime. It describes how existing SOA middleware can be extended to implement this approach.

Keywords

Middleware, Bulk data processing, Near-time processing, Performance, SOA

1. Introduction

Business software systems like customer-billing systems or financial transaction systems are required to process large volumes of data in a fixed period of time. For example, a billing system for a large telecommunication provider has to process more than 1 million bills per day. It consists of several sub components that process the different billing sub processes like mediation, rating, billing and presentment (see Figure 1).

Figure 1: Billing sub processes

The mediation components receive usage events from delivery systems, like switches and transform them into a format the billing system is able to process. For example, transforming the event records to the internal record format of the rating and billing engine or adding internal keys that are later needed in the process. The rating engine assigns the events to the specific customer account, called guiding, and determines the price of the event, depending on the applicable tariff. It also splits events if more than one tariff is applicable or the customer qualifies for a discount. The billing engine calculates the total amount of the bill by adding the rated events, recurring

and one-time charges and discounts. The output is processed by the presentment components, which format the bill, print it, or present it to the customer in self-service systems, for example on a website.

The performance requirements for such a billing system are high. It has to process more than 1 million records per hour and the whole batch run needs to be finished in a limited timeframe to comply with service level agreements with the print service provider. Since delayed invoicing causes direct loss of cash, it has to be ensured that the bill arrives at the customer on time.

The traditional operation paradigm of such a system for bulk data processing is batch processing (see Figure 2).

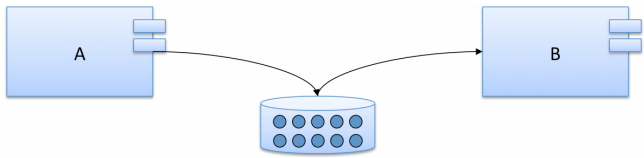

Figure 2: Batch processing

Batch processing exhibits the following properties (Swientek et al., 2008):

- **Bulk processing of data**
  A Batch processing system processes several gigabytes of data in a single run. Multiple systems are running in parallel controlled by a job scheduler to speed up processing.

- **No user interaction**
  There is no user interaction needed for the processing of data. It is impossible due to the amount of data being processed.

- **File- or database-based interfaces**
  Input data is read from the file system or a database. Output data is also written to files on the file system or a database. Files are transferred to the consuming systems through FTP by specific jobs.

- **Operation within a limited timeframe**
  A batch processing system often has to deliver its results in a limited timeframe due to service level agreements (SLA) with consuming systems.

- **Offline handling of errors**
  Erroneous records are stored to a specific persistent memory (file or database) during operation and are processed afterwards.

While such a batch processing system is able to process bulk data and thus delivering a high throughput, it is not able to deliver near-time processing. That is, the latency of a batch processing system is high.

Near-time processing reduces the latency of the system, that is, the time that is spent between the occurrence and the processing of an event. In case of a billing system, it is the time between the user making a call and the complete processing of this call including mediation, rating, billing and presentment. From the customer point of view, an event should be viewable in the customer self-care website shortly after the call has been made. This requirement cannot be implemented using batch processing.

To decrease the latency of the system a message-based approach is needed (see Figure 3), for example by utilising an Enterprise Service Bus (ESB). While this approach provides near-time processing of data, it is not able to deliver the same throughput as batch processing.

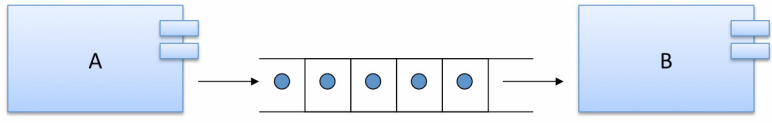

Figure 3: Message based processing

This paper describes a new approach to the problem of delivering near-time processing while providing very high throughput. It is organised as follows: The next section defines the performance attributes throughput and latency in more detail and explains why they are contrary to each other in this case. Section 3 defines the term data granularity and explains how throughput and latency depend on it. Section 4 describes how this approach can be implemented using Sopera ASF which provides an open-source SOA platform. The paper concludes with a summary of the described approach and an outlook to further research.

2. Throughput vs. latency

Throughput and latency are performance metrics of a system. We use the following definitions of throughput and latency in this paper:

- **Throughput**
  The number of events the systems is able to process in fixed timeframe.

- **Latency**
  The period of time between the occurrence of an event and its processing.

In the case of bulk data processing, throughput and latency are contrary to each other (as illustrated in Figure 4). A high throughput, as provided by batch processing, leads to a high latency, which impedes near-time processing. On the other hand, low latency, as provided by a message-based system, cannot provide the throughput needed for bulk data processing.

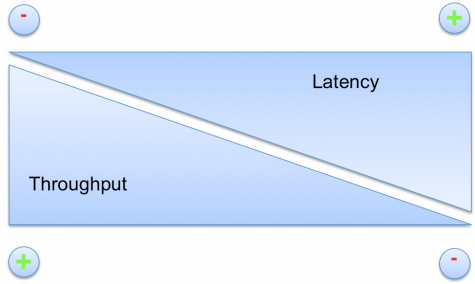

Figure 4: Throughput vs. latency

In order to achieve near-time processing with very high throughput, we propose a combination of both processing types (see Figure 5).

Figure 5: Combining batch processing with message-based processing

This solution should provide the best possible latency with the lowest throughput that is still acceptable to meet the performance requirements.

3. Data granularity

Throughput and latency of the system depend on the granularity of data that is being processed. Data granularity relates to the amount of data that is processed in a unit of work, for example in a single batch run or an event. Haesen et al. distinguishes between two types of data granularity (Haesen et al., 2008):

- **Input data granularity**
  Data that is sent to a component

- **Output data granularity**
  Data that is returned by a component

Additionally, data granularity can relate to different orientations:

- **Horizontal data granularity**
  Refers to the amount of data or fields that is contained in a single record

- **Vertical data granularity**
  Refers to the total number of records

The remainder of this paper focuses on vertical data granularity. No distinction is being made regarding input and output data granularity.

Batch processing uses a high granularity of data, which leads to high throughput and high latency. Message-based processing uses low granularity of data, which leads to low latency but also low throughput. The optimum data granularity would allow having the lowest possible latency with the lowest acceptable throughput and thus providing near-time processing of bulk data (see Figure 6).

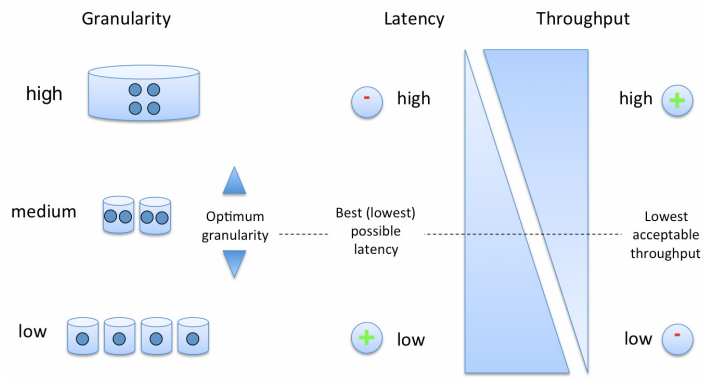

Figure 6: Throughput and latency depend on data granularity

3.1. Variable adjustment of granularity

The granularity of the data processed in one message will be adjusted at runtime. A middleware is needed that provides services to constantly measure the throughput and latency of the system and to control the granularity of the data (see Figure 7). If the throughput drops below the acceptable minimum, the granularity of the data needs to be higher. On the other hand, the granularity can be lowered, if the throughput of the system is above the minimum.

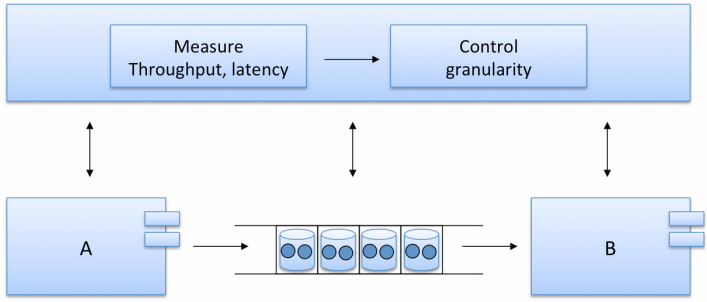

Figure 7: Variable adjustment of granularity

4. Implementation

This section describes how to implement the variable adjustment of data granularity by extending the Sopera ASF platform.

4.1. Sopera Advanced Service Factory

The Sopera Advanced Service Factory (Sopera ASF) provides an open source SOA (Service Oriented Architecture) platform, which has been developed and successfully deployed at Deutsche Post AG. The core of the platform is the Sopera ESB. The Sopera ESB is implemented as a distributed service bus. An Enterprise Service Bus (ESB) is an integration platform that combines messaging, web services, data transformation and intelligent routing (Schulte, 2002).

The main components of the Sopera ESB are the Sopera Library (SSB Library) and the Sopera Service Management (SSM). The Sopera Library represents the service container of the Sopera ESB and provides access for all participants, mediation of the SOA functionality and message exchange. Sopera Management provides functionality for monitoring the operations of the SOA platform including performance, error handling and reporting and provides methods to control the behaviour of the service participants.

Additional infrastructure services are provided as plug-ins. Sopera ASF includes the following plug-ins:

- Service registries/repositories
- Security services
- Messaging/Transport services
- Orchestration/Workflow server

Sopera ASF supports different Message Queuing Server such as Apache ActiveMQ, JORAM and IBM WebSphere MQ. In addition to the ESB, Sopera ASF also provides an extensive tool suite based on the Eclipse IDE including editors to define services, policies and process flows.

We will use the Sopera ASF platform to implement the adjustment of data granularity to reduce the latency of bulk data processing as introduced in section 3. The platform has been chosen because of its best of breed approach using open source components. All source code is freely available. Additionally, the reliability of the platform has been proven in a huge deployment at Deutsche Post.

The next section describes the design of the components that comprise the proposed solution.

4.2. Component architecture

Figure 8 shows the components, which are involved in the adjustment of data granularity at runtime.

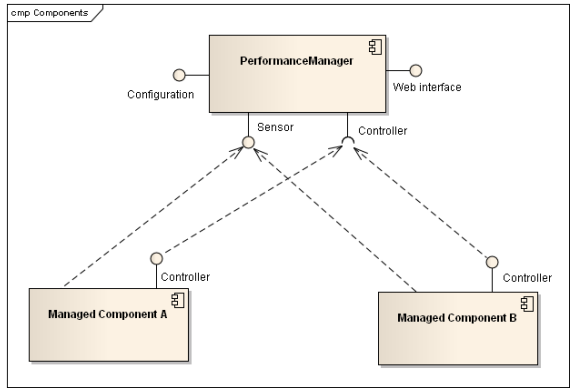

Figure 8: Components

The main component is the Performance Manager. It constantly measures the throughput and latency of managed components and controls their data granularity. Every managed component constantly sends a notification to the Performance Manager containing its current throughput and latency. The Performance Manager buffers the incoming notification messages and computes the current throughput and latency of the complete business process. If the computed latency exceeds the pre-defined limit, the Performance Manager adjusts the data granularity of the managed components.

The communication between the Performance Manager and the managed components will be implemented using Java Management Extensions (JMX).

4.2.1. Performance Manager

The Performance Manager is an infrastructure service and will be implemented as a Web application, which runs inside a standard Servlet container (see Figure 9).

The Performance Manager provides the following interfaces.

- **Sensor interface**
  The Sensor interface receives JMX (Java Management Extension) notification messages from managed components containing their current throughput and latency.

- **Performance Manger Client interface**
  The Performance Manager client interface exposes the Performance Manager Client application and is used to set the pre-defined limits for the latency and throughput of the business process.

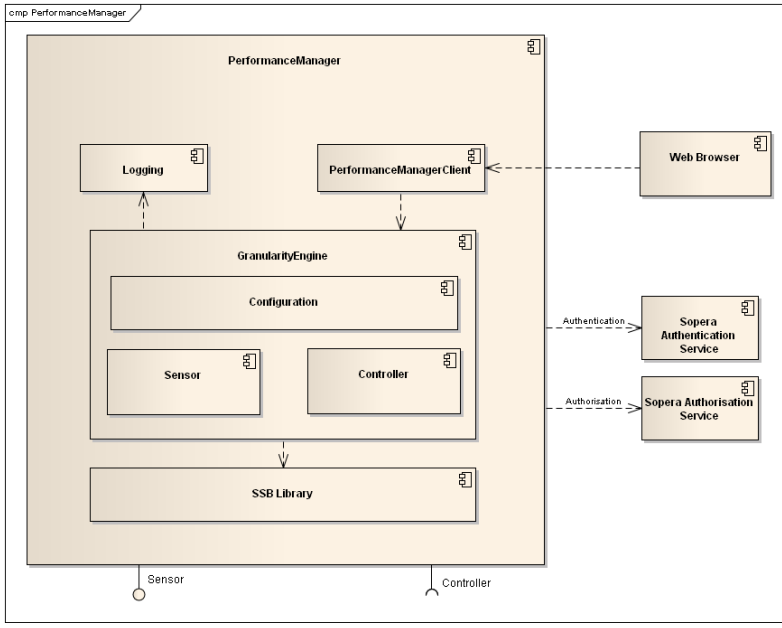

Figure 9: Component Performance Manager

The Performance Manager is comprised of the following sub components.

- **Granularity Engine**
  The Granularity Engine is the core of the Performance Manager service. It consists of the components Sensor, Controller and Configuration. The Sensor component receives JMX notification messages from managed components. The Granularity Engine computes the throughput and latency of the complete business process using the notifications and compares the computed values with the pre-defined limits stored in the Configuration component. If the computed values exceed the pre-defined limits, the Controller sends a message to the corresponding managed components to adjust the data granularity.

- **SSB Library**
  The SSB Library provides the integration of the Performance Manager in the Sopera ASF platform. It is used to receive the notification messages of the managed components.

- **Logging**
  The Logging component logs all measured and computed values of the managed components and all adjustments of the data granularity performed by the Granularity Engine. The logs can be viewed using the Performance Manager Client application.

- **Performance Manager Client**
  The Performance Manager Client application provides a user interface to set the pre-defined limits for throughput and latency. It also offers functionality to manually control the data granularity and to view the logs written by the Logging component.

- **Authentication and Authorisation**
  The Performance Manager uses the Authentication and Authorisation services provided by the Sopera ASF platform.

4.2.2. Managed Component

The SSB Library already contains an SSM module which provides the management functionality for a service participant. The SSM module contains several MBeans (Management Beans), which monitor the message traffic that passes through an instance of the SSB Library, including the average number of requests per minute, the total number of request for a fixed time and the percentage of failed requests. The data is available at different levels of aggregation. The ParticipantMonitor provides data about the service participant. The ServiceMonitor and OperationMonitor provide data about a service and an operation respectively (Sopera Operations and Administration Guide).

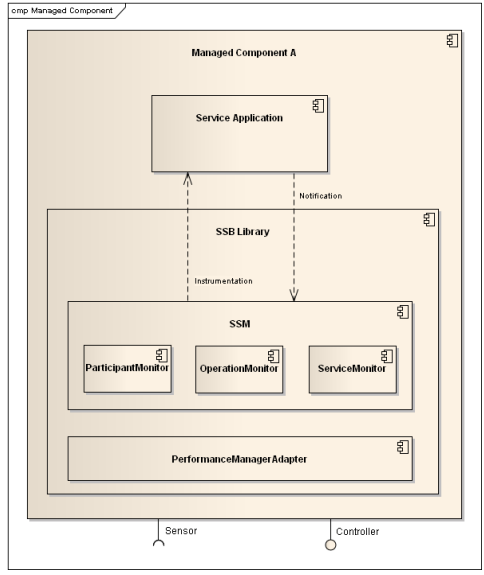

Figure 10: Managed Component

We will extend the existing SSM components to measure the throughput and latency of the service participant and add operations to control its data granularity.

The Performance Manager Adapter provides the Controller interface used by the Performance Manager to control the MBeans of the SSM components. Additionally,

the adapter publishes notification messages containing the current throughput and latency of the managed component.

4.3. Implementation considerations

The following considerations need to be taken into account when implementing the proposed solution.

4.3.1. Measuring Throughput and latency of orchestrated services

In order to compute the throughput and latency of a complete business process, the Performance Manager needs to know which services are orchestrated to compose this business process. The Performance Manager will be able to retrieve business process definitions from the workflow engine attributed with limits for the maximum latency and minimum acceptable throughput. It might be necessary to extend the utilised business process language to support these attributes including the corresponding tools.

4.3.2. Transport of large messages

The message size cannot be arbitrarily increased because very large messages cannot be transported efficiently by the messaging system. If the data granularity exceeds a certain level, it might be required that the payload of the message is transported outside of the messaging system by using FTP (File Transfer Protocol) or similar transports.

5. Conclusion

Business software systems for bulk data processing commonly utilise batch processing. These systems are more and more faced to also provide near-time processing due to changed business requirements such as customer demand. While a batch processing system is able to provide the required high throughput, it cannot meet the requirements regarding low latency necessary for near-time processing. On the other hand, message-based processing is able to deliver low latency but cannot provide the required high throughput.

Latency and throughput depend on the granularity of data that is being processed. Batch processing uses coarse-grained data and therefore exhibits a high latency. Message-based processing uses fine-grained data, i.e. messages, and therefore exhibits a low latency. The optimum data granularity would allow having the lowest possible latency with the lowest acceptable throughput and thus providing near-time processing of bulk data. We suggest that the granularity of data will be adjusted at runtime by a middleware, which continuously measures the throughput and latency of the system.

Sopera ASF is an adequate integration platform to implement the described approach. The necessary infrastructure services for monitoring the throughput and

latency of the system and for adjusting the granularity of data will be implemented as plug-ins of the Sopera ESB.

The next step is the implementation of the proposed solution along with comprehensive performance tests.

6. References

Chappel, D. (2004), *Enterprise Service Bus*, O'Reilly, ISBN 0-596-00675-6.

Haesen, R., Snoeck, M., Lemahieu, W. and Poelmans, S. (2008), "On the definition of service granularity and its architectural impact", *CAiSE '08: Proceedings of the 20th international conference on Advanced Information Systems Engineering*, Springer Verlag, Berlin, Heidelberg, Germany, pp. 375–389.

JMX, Java Management Extensions, http://java.sun.com/j2se/1.5.0/docs/guide/jmx/index.html, (Accessed 29. September 2009).

Schulte, R. W. (2002), "Predicts 2003: Enterprise Service Buses Emerge", Gartner.

Sopera ASF, http://www.sopera.de/en/home, (Accessed 10. August 2009).

Sopera Operations and Administration Guide, http://www.sopera.de/nc/en/support/bibliothek/sopera-32/opadmin32/, (Accessed 30. September 2009).

Swientek, M., Bleimann U. and Dowland P.S. (2008), "Service-Oriented Architecture: Performance Issues and Approaches", *Proceedings of the Seventh International Network Conference (INC 2008)*, Plymouth, UK, pp. 261-269.