Thesis: Data Stream Processing on Enterprise Integration Platforms

author: Damian Chojna

objective: Use common enterprise integration middlewares to accomodate data stream processing.

Thesis proposal

Introduction

SenseTile requirements

SenseTile system is a distributed system, organized in three layers:

- Sensors: sensors deployed in ad hoc custom board (SenseTile Sensor Board), ad hoc networks and wireless sensor networks
- Processor Units: small full blown computational units (netbooks, miniPCs, powerful PDAs)
- Server Cluster: powerful cluster, SenseTile main data storage and processing facility.

Data streams are flowing from sensors to Processor Units; in Processor Units the data streams can be merged, processed, transformed, locally stored, used for locally deployed applications, sent to the Server Cluster, sent to near by Processor Units. In the Server Cluster all the functionalities available in processor Units are replicated, with an higher level of complexities.

Data streams come in various formats and transport protocols, to accommodate a wide range of data stream sources and sinks; SenseTile system will be able to support ad hoc applications, feeding them data streams with the requested format and transport protocol. SenseTile system aims to minimize the effort to link together previously separated application sharing data and processes.

Enterprise Application Integration frameworks are suited for the integration of different applications, while they lack the data stream handling capabilities that Data Stream Management Systems have.

A prototype library of data stream oriented components, based on an existing Enterprise Application Integration framework (Apache Camel is the main candidate) will be built, based on the requirements of the SenseTile system; its effectiveness will be assessed compared to the features available in Data Stream Management Systems.

Data Strams

Data Stream storage and processing is an ever growing research field, especially in monitoring applications. The data stream model (DAHP, DBMS Active Human Passive) can differ from the conventional stored relation model (HADP, Human Active DBMS Passive) in several ways:

- Passive vs active:
  - HADP: passive repository, users issues transactions and queries.
  - DAHP: active repository, data coming continuously from external sources.
- Data state:

    - HADP: focus on current state of data.
    - DAHP: history intelligent management, typically some history data is collected and after a defined time frame discarded or reprocessed.
- Triggers:
    - HADP: triggers are not mandatory.
    - DAHP: trigger oriented.
- Queries:
    - HADP: data and queries are exact
    - DAHP: data streams data can be lost, stale or intentionally omitted, queries can be inexact even if data is exact (for instance for performance reasons).
- Real-time:
    - HADP: no real-time requirements.
    - DAHP: real-time requirements.

==== Enterprise Application Integration frameworks =====

Enterprise Application Integration (EAI) frameworks enable integration of systems and applications across the enterprise.

EAI frameworks typically relies on an Enterprise Service Bus (ESB) providing an event based and message based engine; ESB main strength is the possibility to build data routes with sources, destinations, and processing nodes. Enterprise Application Integration (EII) deals with data integration, that is, combining data coming from different sources and, eventually, providing a unified view.

Description

An EAI framework is used to accommodate functionalities that characterize Data Stream management Systems in a distributed environment.

The main test bed application is the SenseTile system: the SenseTile data stream requirements are to be supported with the data stream management and process features that will be implemented in the EAI framework.

Some of the features singled out:

- Accommodate eterogeneous data streams sources and sinks.
    - i.e. data read with custom made API (source) and stored in an XML data base (sink), data read from a local RDB storage (source) and sent to a WebService? (sink).
    - This feature is almost entirely provided by the EAI.
- Data Stream manipulation:
    - Attach meta data to data stream.
    - Merge data streams with various data rates and various reciprocal delays on data into a resultant data stream with a defined data rate and aligned data.
    - Change data stream nature: from constant rate to event based.
- History management:
    - Guarantee a definite time frame for the availability of data (using memory for short time frames, and local storage for larger ones).
    - Guarantee data availability with degrading quality: that is, guarantee a short time frame for high quality data, and progressively longer data time frames for progressively lower quality data.
- Search and select data from data streams based on meta data.
- Dynamic configuration: reconfigure part of the system according to data collected. It is often the case that a data stream will be processed in various ways, according to previously collected data.

The implementation will use the EAI framework with a special focus on components reuse: a collection of simpler components will be built and connected each other to realize the features detailed previously.

Items

Mandatory

1. Familiarization with Data Stream Management challenges, overview of Data Stream Management Systems.
2. Familiarization with EAI challenges and with one EAI platform.
3. Features implementation into an EAI framework (Apache Camel). Some of the features to be implemented have been discussed previously. Implementation steps:
    1. Define one or more scenarios related to the feature to be implemented and to the test bench application selected.
    2. Implement the feature and the related scenarios.
    3. Refactor code to ease reuse: extract simpler, reusable, one purpose components.
4. Compare the approches (EAI vs dedicated Data Stream Management Systems).

Discretionary

1. Features implementation into a distributed Data Stream Management System (Borealis is the main candidate).
2. Compare quantitatively the approches (EAI vs dedicated Data Stream Management Systems):

References

- Enterprise Integration Patterns
- Apache Camel open source integration framework based on known Enterprise Integration Patterns
- Aurora Project Data Stream management system
- Borealis Project Distributed Data Stream management system
- Apache Mina application framework which provides an abstract, event-driven, asynchronous API over various transports (TCP/IP, UDP/IP) via Java NIO