InfoSphere Streams represents a paradigm shift

Operational Databases

Reporting and human analysis on historical data

Data at rest

Analysis of current data to improve business transactions

Real Time Analytic Processing (RTAP) to improve business response

Data in Motion

Stream Computing

OLTP

1968 Hierarchical

1970 Relational “System R”

1983 DB2 v1

2003 “System S”

2009 InfoSphere Streams

RTAP

OLAP

Data Warehousing
Continuous Ingestion  

A New Paradigm: In-Motion analytics for High throughput and Ultra-low latencies

Continuous Queries /Analytics on data in motion

Visual Representation
InfoSphere Streams Overview

Agile Development Environment

- Streams Processing Language (SPL)
- Rapid application development
- Eclipse IDE
- Streams LiveGraph
- Streams Debugger

Scale-out Architecture

- Clustered runtime for near-limitless capacity
- RHEL v5.3 and above
- x86 multicore hardware
- InfiniBand support
- Ethernet support

Sophisticated Analytics with Toolkits & Adapters

- Database Toolkit
- Mining Toolkit
- Financial Toolkit
- Standard Toolkit
- Internet Toolkit
- User defined toolkits
- Over 50 samples
What are People doing with Streams?

**Telephony**
- CDR processing
- Social analysis
- Churn prediction
- Geomapping

**Transportation**
- Intelligent traffic management

**Smart Grid & Energy**
- Transactive control
- Phasor Monitoring Unit

**Stock market**
- Impact of weather on securities prices
- Analyze market data at ultra-low latencies

**Law Enforcement, Defense & Cyber Security**
- Real-time multimodal surveillance
- Situational awareness
- Cyber security detection

- Detecting multi-party fraud
- Real time fraud prevention

**Fraud prevention**

**e-Science**
- Space weather prediction
- Detection of transient events
- Synchrotron atomic research

**Health & Life Sciences**
- Neonatal ICU monitoring
- Epidemic early warning system
- Remote healthcare monitoring

**Natural Systems**
- Wildfire management
- Water management

**Other**
- Manufacturing
- Text Analysis
- Who’s Talking to Whom?
- ERP for Commodities
- FPGA Acceleration
IBM Programming Model

**Input (Source):**
- NYSE
- Reuters
- Apache
- IBM

**Process:**
- Company Filter
- Usage Model
- Text Extract
- Keywords Filter
- Season Adjust
- Daily Adjust
- History
- Compare
- RSS Feed

**Output (Sink):**
- Temp Action
- Store History

**Streams Processing Language**

**Platform optimized compilation**
InfoSphere Streams provides

- a programming model for defining data flow graphs consisting of **data sources** (inputs), **operators**, and **sinks** (outputs)
- controls for fusing operators into processing elements (**PEs**)
- infrastructure to support the composition of scalable **stream processing applications** from these components
- deployment and operation of these applications across distributed **x86 processing nodes**, when scaled-up processing is required
Streams application graph:

A directed, possibly cyclic, graph

A collection of sources, operators, and sinks (the graph’s vertices)

Connected by streams (the graph’s edges)

Each complete application is a potentially deployable job
Streams application graph:

A directed, possibly cyclic, graph

A collection of sources, operators, and sinks

Connected by streams

Each complete application is a potentially deployable job

Jobs are deployed to a Streams runtime environment, known as a Streams Instance (or simply, an instance)
Streams application graph:

- A directed, possibly cyclic, graph

- A collection of sources, operators, and sinks

- Connected by streams

Each complete application is a potentially deployable job

Jobs are deployed to a Streams runtime environment, known as a Streams Instance (or simply, an instance)

An instance can include a single processing node (hardware)
Streams application graph:
   A directed, possibly cyclic, graph
   A collection of sources, operators, and sinks
   Connected by streams
Each complete application is a potentially deployable job
Jobs are deployed to a Streams runtime environment, known as a Streams Instance (or simply, an instance)
An instance can include a single processing node (hardware)
Or multiple processing nodes
Instance

Runtime instantiation of InfoSphere Streams executing across one or more hosts

Collection of components and services

Processing Element (PE)

Fundamental execution unit that is run by the Streams instance

Can encapsulate a single operator or many “fused” operators

Job

A deployed Streams application executing in an instance

Consists of one or more PEs
Optimizing scheduler assigns jobs to nodes, and continually manages resource allocation

Commodity hardware – laptop, blades or high performance clusters
Optimizing scheduler assigns PEs to nodes, and continually manages resource allocation.

Commodity hardware – laptop, blades or high performance clusters.

Dynamically add nodes and jobs.

New jobs work with existing jobs.
Streams Runtime includes High Availability

PEs on busy nodes, can be moved manually by the Streams administrator

PEs on failing nodes can be moved automatically, with communications re-routed
Streams Runtime Node Pools facilitate High Availability

- HA application design pattern
  - Source job exports stream, enriched with tuple ID
  - Jobs 1 & 2 process in parallel, and export final streams
  - Sink job imports streams, discards duplicates, alerts on missing tuples
Streams Processing Language

Designed for stream computing

- Define a streaming-data flow graph
- Rich set of data types to define tuple attributes

Declarative

- Operator invocations name the input and output streams
- Referring to streams by name is enough to connect the graph

Procedural support

- Full-featured C++/Java-like language
- Custom logic in operator invocations
- Expressions in attribute assignments and parameter definitions

Extensible

- User-defined data types
- Custom functions written in SPL or a native language (C++ or Java)
- Custom operator written in SPL
- User-defined operators written in C++ or Java
An **operator** is the fundamental building block of the Streams Processing Language. It processes data (tuples) from (zero or more) streams and may produce (zero or more) new streams. It represents a class of manipulations. It defines input ports and output ports.

A **stream** is an infinite sequence of tuples. It connects to an operator on a **port**.

A **tuple** is a structured list of attribute names and their types.

An operator **invocation** is a specific use of an operator with specific assigned input and output streams with locally specified parameters, logic, etc.

A **composite** operator is a collection of operators. An encapsulation of a subgraph of

- Primitive operators (non-composite)
- Composite operators (nested)

Similar to a macro in a procedural language.
Every graph is encoded as a composite
A composite is a graph of one or more operators
A composite may have input and output ports
Source code construct only
Nothing to do with operator fusion (PEs)

Each stream declaration in the composite
Invokes a primitive operator or
another composite operator

An application is a main composite
No input or output ports
Data flows in and out but not on streams within a graph
Streams may be exported to and imported from other applications running in the same instance
Communication among Streams jobs

Operators can define streams as exportable or importable

A newly deployed job will be dynamically linked to existing imported or exported streams
Streams Data Processing in Telco Environment

Decoding, filtering, aggregation, correlation, summation, transformation, formatting, ...

Source data format: ASN.1, XML, ASCII, binary
Standardized or proprietary, via edge adapters

Output into dashboards, databases, files
Statistics, monitoring, archiving

DB

Streams Telco
Real time Processing

CDRs
Event Data
Logs
Configuration Data
Tap into Message Transfer

Database lookup, data enrichment
Benefits of Streams in Telco Projects

Operating on data in motion, no need for dumping mass data into database

Short processing latencies (milliseconds)

High data throughput (up to 10,000 records per sec per core)

Excellent scaling (almost linearly with number of CPU cores)

Real time processing of data from different sources

Extensible wrt. deployment of additional use cases and edge adapters

Short application development cycles