An introduction to InfoSphere Streams

A platform for analyzing big data in motion

Sherif Sakr (ssakr@cse.unsw.edu.au)
Senior Research Scientist
National ICT Australia

Skill Level: Intermediate

Date: 07 May 2013

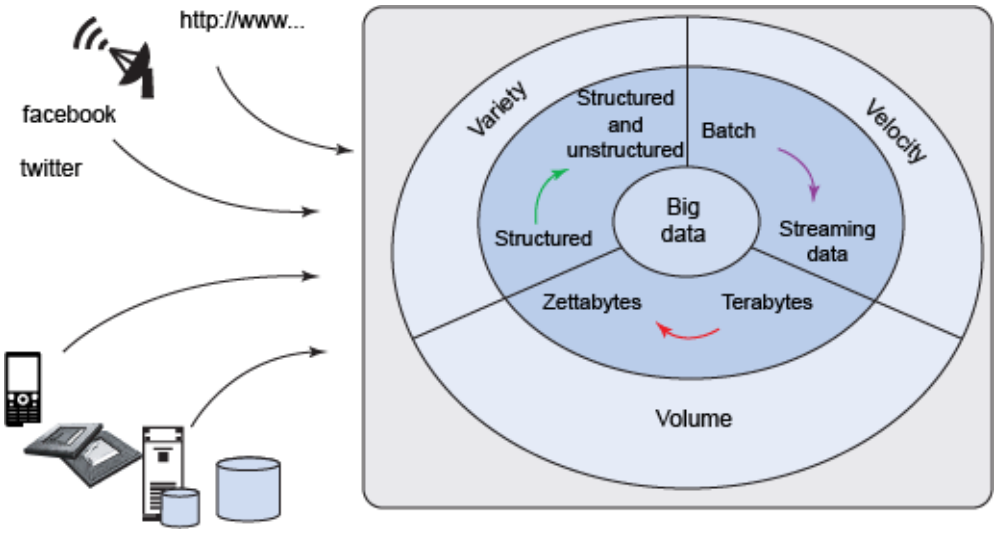
Learn about InfoSphere® Streams, part of the IBM big data platform. InfoSphere Streams addresses a crucial emerging need for platforms and architectures that can process vast amounts of generated streaming data in real time. Find out what the product is designed to do, when it can be useful, how it works, and how it can complement InfoSphere BigInsights™ to perform highly complex analytics.

Information from multiple sources is growing at a staggering rate. The number of Internet users reached 2.27 billion in 2012. Every day, Twitter generates more than 12 TB of tweets, Facebook generates more than 25 TB of log data, and the New York Stock Exchange captures 1 TB of trade information. About 30 billion radio-frequency identification (RFID) tags are created every day. Add to this mix the data generated by the hundreds of millions of GPS devices sold every year, and the more than 30 million networked sensors currently in use (and growing at a rate faster than 30 percent per year). These data volumes are expected to double every two years over the next decade.

A company can generate up to petabytes of information in the course of a year: web pages, blogs, clickstreams, search indices, social media forums, instant messages, text messages, email, documents, consumer demographics, sensor data from active and passive systems, and more. By many estimates, as much as 80 percent of this data is semi-structured or unstructured. Companies are always seeking to become more nimble in their operations and more innovative with their data analysis and decision-making processes. And they are realizing that time lost in these processes can lead to missed business opportunities. The core of the big data challenge is for companies to gain the ability to analyze and understand Internet-scale information just as easily as they can now analyze and understand smaller volumes of structured information.

Figure 1 illustrates the big data challenge of extracting insight from an immense *volume. variety,* and *velocity* of data — in context — beyond what's been possible up until now.

Figure 1. The big data challenge



In Figure 1, volume refers to the scale of data, from terabytes to zettabytes. Variety refers to the complexity of data in many different structures, ranging from relational to logs to raw text. Velocity reflects streaming data and large-volume data movements.

IBM is helping companies respond to the big data challenge, giving them tools for integrating and managing the volume and velocity of data, applying analytics in native form, visualizing available data for ad-hoc analysis, and more. This article introduces you to InfoSphere Streams, technology that enables you to analyze many data types simultaneously and perform complex calculations in real time. You'll learn how InfoSphere Streams works, what it's useful for, and how you can use it in conjunction with another IBM product for big data analytics — IBM InfoSphere BigInsights — to perform highly complex analytics.

InfoSphere BigInsights: An overview

MapReduce

The MapReduce framework, introduced by Google, enables the programming of commodity computer clusters to perform large-scale data processing in a single pass. A MapReduce cluster can scale to thousands of nodes in a fault-tolerant manner and process petabytes of data in a highly parallel and cost-effective manner. One of the framework's main advantages is its reliance on a simple and powerful programming model. In addition, it isolates the application developer from all the complex details of running a distributed program, such as issues relating to data distribution, scheduling, and fault tolerance.

An understanding of InfoSphere BigInsights will give you a fuller appreciation of the purpose and value of InfoSphere Streams. (If you're already familiar with BigInsights, feel free to jump to the next section.)

BigInsights is an analytics platform that enables companies to turn complex Internet-scale information sets into insights. It consists of a packaged Apache Hadoop distribution, with a greatly simplified installation process, and associated tools for application development, data movement, and cluster management. Hadoop, an open source implementation of the MapReduce framework, has gained significant momentum in industry and academia thanks to its simplicity and scalability. In addition to Hadoop, other open source technologies in BigInsights (all of which, with the exception of Jaql, are Apache Software Foundation projects) are:

- **Pig** — A platform that provides a high-level language for expressing programs that analyze large datasets. Pig is equipped with a compiler that translates Pig programs into sequences of MapReduce jobs that the Hadoop framework executes.
- **Hive** — A data-warehousing solution built on top of the Hadoop environment. It brings familiar relational-database concepts, such as tables, columns, and partitions, and a subset of SQL (HiveQL) to the unstructured world of Hadoop. Hive queries are compiled into MapReduce jobs executed using Hadoop.
- **Jaql** — An IBM-developed query language designed for JavaScript Object Notation (JSON) and provides a SQL-like interface. Jaql handles nesting gracefully, is heavily function-oriented, and is highly flexible. It's useful for loosely structured data and is the interface for the HBase column store and for text analytics.
- **HBase** — A column-oriented NoSQL data-storage environment designed to support large, sparsely populated tables in Hadoop.
- **Flume** — A distributed, reliable, available service for efficiently moving large amounts of data as it is produced. Flume is well-suited to gathering logs from multiple systems and inserting them into the Hadoop Distributed File System (HDFS) as they are generated.
- **Lucene** — A search-engine library that provides high-performance and full-featured text search.
- **Avro** — A data-serialization technology that uses JSON for defining data types and protocols, and serializes data in a compact binary format.
- **ZooKeeper** — A centralized service for maintaining configuration information and naming, providing distributed synchronization and group services.
- **Oozie** — A workflow scheduler system for managing and orchestrating the execution of Apache Hadoop jobs.

In addition, the BigInsights distribution includes the following IBM-specific technologies:

- **BigSheets** — A browser-based spreadsheet-like discovery and exploration interface that enables business users to gather and analyze data easily and harness the power of Hadoop. It provides built-in readers that can work with data in several common formats, including JSON, comma-separated value (CSV), and tab-separated value (TSV).

- **Text analytics** — A pre-built library of text annotators for common business entities. It provides rich language and tooling for building custom location annotators.
- **Adaptive MapReduce** — An IBM Research solution for speeding up the execution of small MapReduce jobs by changing how MapReduce tasks are handled.

The InfoSphere platform

InfoSphere is a comprehensive information-integration platform that includes data warehousing and analytics, information integration, master data management, life-cycle management, and data security and privacy. The platform improves application development processes so organizations can speed time to value, reduce integration costs, and increase information quality.

In general, BigInsights is not designed to replace a traditional relational database management system (DBMS) or traditional data warehouses. In particular, it is not optimized for interactive queries over tabular data structures, online analytical processing (OLAP), or online transaction processing (OLTP) applications. However, as a component of the IBM big data platform, BigInsights provides potential integration points with the other components of the platform including the data warehouse, data integration, and governance engines; and third-party data analytics tools. As you'll see later in this article, it can also integrate with InfoSphere Streams.

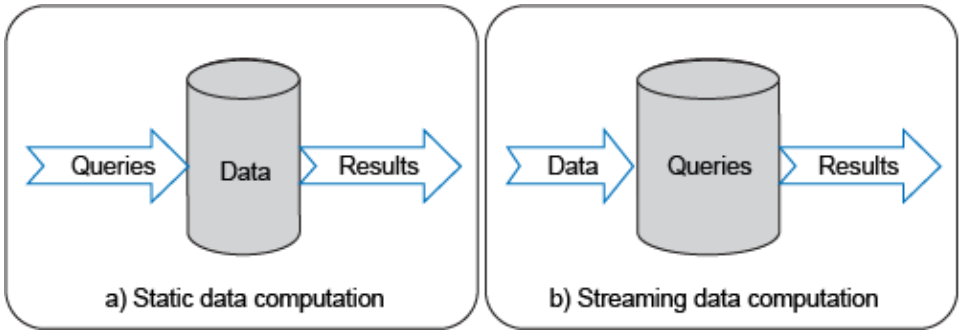
Stream computing: A new paradigm

Stream computing is a new paradigm necessitated by new data-generating scenarios, such as the ubiquity of mobile devices, location services, and sensor pervasiveness. A crucial need has emerged for scalable computing platforms and parallel architectures that can process vast amounts of generated streaming data.

The BigInsights technologies are not adequate for supporting real-time stream-processing tasks because they are mainly oriented toward supporting batch processing of static data. In the processing of static data, a query such as *List all users who have been connecting to the network* will result in a single result set. With real-time processing of streaming data, you can execute a *continuous query* such as *List all users who have been connecting to the network in the last 10 minutes*. This query will return continuously updated results. In the world of static data, the user finds the proverbial needle in a haystack; in the world of streaming data, the user finds the needle as the hay is blowing by.

Figure 2 illustrates the difference between the computations performed on static data and those performed on streaming data.

Figure 2. Static data computation versus streaming data computation



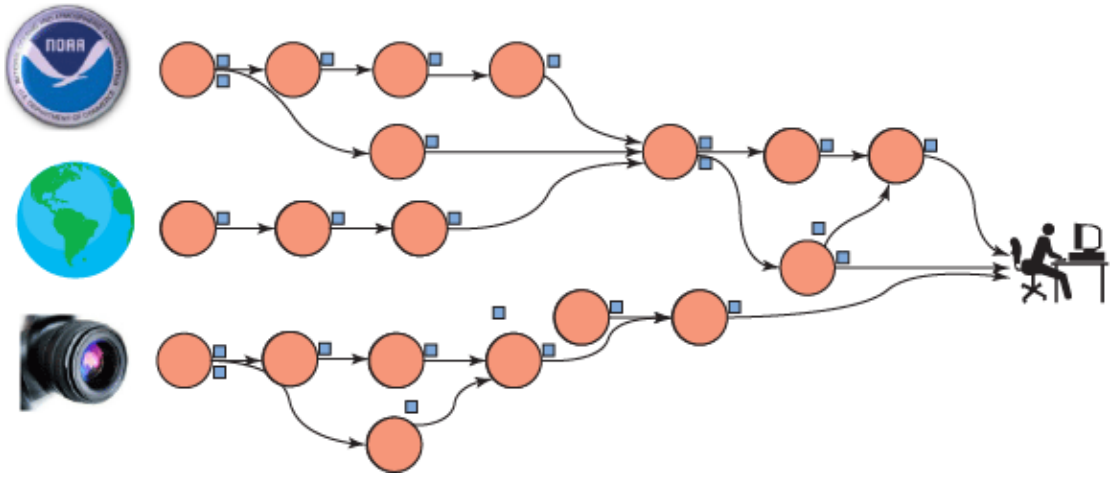
In static data computation (the left-hand side of Figure 2), questions are asked of static data. In streaming data computation (the right-hand side), data is continuously evaluated by static questions.

The InfoSphere Streams platform supports real-time processing of streaming data, enables the results of continuous queries to be updated over time, and can detect insights within data streams that are still in motion.

InfoSphere Streams overview

InfoSphere Streams is designed to uncover meaningful patterns from information in motion (data flows) during a window of minutes to hours. The platform provides business value by supporting low-latency insight and better outcomes for time-sensitive applications, such as fraud detection or network management. InfoSphere Streams also can fuse streams, enabling you to derive new insights from multiple streams, as illustrated in Figure 3.

Figure 3. Fused stream processing



The main design goals of InfoSphere Streams are to:

- Respond quickly to events and changing business conditions and requirements.
- Support continuous analysis of data at rates that are orders of magnitude greater than existing systems.

- Adapt rapidly to changing data forms and types.
- Manage high availability, heterogeneity, and distribution for the new stream paradigm.
- Provide security and information confidentiality for shared information.

InfoSphere Streams provides a programming model and IDE for defining data sources, and software analytic modules called *operators* fused into processing execution units. It also provides infrastructure to support the composition of scalable stream-processing applications from these components. The main platform components are:

- **Runtime environment** — This includes platform services and a scheduler for deploying and monitoring Streams applications across a single host or set of integrated hosts.
- **Programming model** — You can write Streams applications using the Streams Processing Language (SPL), a declarative language. You use the language to state what you want, and the runtime environment accepts the responsibility for determining how best to service the request. In this model, a Streams application is represented as a graph that consists of operators and the streams that connect them.
- **Monitoring tools and administrative interfaces** — Streams applications process data at speeds much higher than those that the normal collection of operating system monitoring utilities can efficiently handle. InfoSphere Streams provides the tools that can deal with this environment.

Streams Processing Language

SPL, the programming language for InfoSphere Streams, is a distributed data-flow composition language. It is an extensible and full-featured language like C++ or Java™ that supports user-defined data types. You can write custom functions in SPL or a native language (C++ or Java). You can write user-defined operators in C++ or Java.

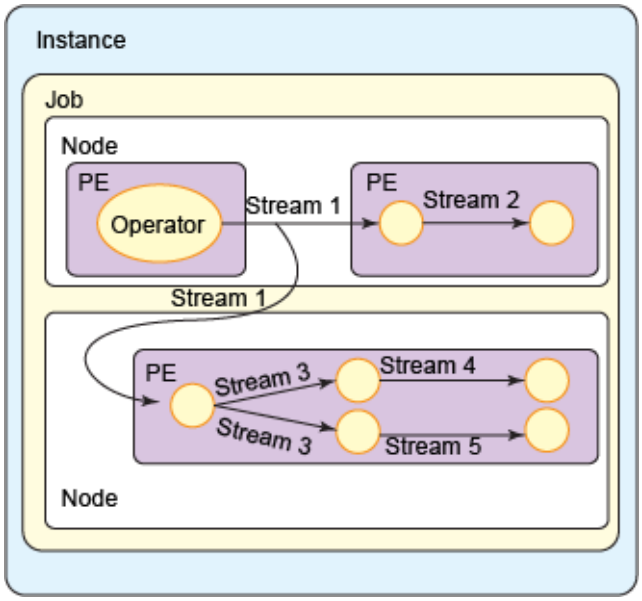
An InfoSphere Streams continuous application describes a directed graph composed of individual operators that interconnect and operate on multiple data streams. Data streams can come from outside the system or be produced internally as part of an application. The basic building blocks of SPL programs:

- **Stream** — An infinite sequence of structured tuples. It can be consumed by operators on a tuple-by-tuple basis or through the definition of a window.
- **Tuple** — A structured list of attributes and their types. Each tuple on a stream has the form dictated by its stream type.
- **Stream type** — Specifies the name and data type of each attribute in the tuple.
- **Window** — A finite, sequential group of tuples. It can be based on count, time, attribute value, or punctuation marks.
- **Operator** — The fundamental building block of SPL, its operators process data from streams and can produce new streams.

- **Processing element (PE)** — The fundamental execution unit. A PE can encapsulate a single operator or many fused operators.
- **Job** — A deployed Streams application for execution. It consists of one or more PEs. In addition to a set of PEs, the SPL compiler also generates an Application Description Language (ADL) file that describes the structure of the application. The ADL file includes details about each PE, such as which binary file to load and execute, scheduling restrictions, stream formats, and an internal operator data-flow graph.

Figure 4 illustrates the InfoSphere Streams runtime view of SPL programs:

Figure 4. InfoSphere runtime execution



An operator represents a reusable stream transformer that transforms some input streams into output streams. In SPL programs, operator invocation implements the specific use of an operator, with specific assigned input and output streams and with locally specified parameters and logic. Each operator invocation names the input and output streams. Various built-in InfoSphere Streams operators offer a range of powerful functions:

- **`Source`** — Reads the input data in the form of streams.
- **`Sink`** — Writes the data of the output streams to external storage or systems.
- **`Functor`** — Filters, transforms, and performs functions on the data of the input stream.
- **`Sort`** — Sorts streams data on defined keys.
- **`Split`** — Splits the input streams data into multiple output streams.
- **`Join`** — Joins the input streams data on defined keys.
- **`Aggregate`** — Aggregates streams data on defined keys.
- **`Barrier`** — Combines and coordinates streams data.
- **`Delay`** — Delays a stream data flow.

- **`Punctor`** — Identifies groups of data that should be processed together.

The place where a stream connects to an operator is called a *port.* Many operators (`Functor`, for example) have one input port and one output port, but operators can also have zero input ports (`Source`, for example), zero output ports (`Sink`, for example), or multiple input or output ports (`Split` and `Join`, for example). Listing 1 shows an SPL example for `Sink`, which has a single input port and writes the output tuples to a disk file.

Listing 1. `Sink` example

```
() as Sink = FileSink(StreamIn) {
    param
    file : "/tmp/people.dat";
    format : csv;
    flush : 20u;
}
```

In Listing 1, `file` is a mandatory parameter that provides the path of the output file. The `flush` parameter flushes the output buffer after a given number of tuples. The `format` parameter specifies the format of the output file.

A *composite operator* is a collection of operators. It represents an encapsulation of a sub-graph of *primitive* (noncomposite) operators or composite (nested) operators. It is similar to a macro in a procedural language.

An application is represented by a *main composite operator* that has no input or output ports. Data flows in and out, but not onto streams within a graph; and streams can be exported to and imported from other applications running in the same instance. The code in Listing 2 represents the skeleton of the main composite operator.

Listing 2. Structure of the main composite operator

```
composite Main {
    graph
    stream ... {
    }
    stream ... {
    }
    ...
}
```

As an example, take a simple streaming application called WordCount that counts the lines and words in a file. The program consists of the following stream graph:

- A `Source` operator invocation that reads a file and sends lines to the data stream.
- A `Functor` operator invocation that counts the lines and words for each individual line of data and sends the statistics to its output stream.
- A `Counter` operator invocation that aggregates the statistics for all lines in the file and prints them at the end.

Before you look at the main composite operator for WordCount, I'll define some helpers. I'll use a `LineStat` type for the statistics about a line. In addition, I need to build a `countWords(rstring line)` function that counts the words in a line, and a `addM(mutable LineStat x, LineStat y)` function to add two `LineStat` values and store the result. Listing 3 defines these helpers.

Listing 3. WordCount helper definitions

```
type LineStat = tuple<int32 lines, int32 words>;

    int32 countWords(rstring line) {
        return size(tokenize(line, " \t", false));
    }

    void addM(mutable LineStat x, LineStat y) {
        x.lines += y.lines;
        x.words += y.words;
    }
```

Now I'm ready to define the main composite operator, as shown in Listing 4.

Listing 4. WordCount's main composite operator

```
composite WordCount {

    graph
    stream<rstring line> Data = FileSource() {
        param file : getSubmissionTimeValue("file");
        format : line;
    }
    stream<LineStat> OneLine = Functor(Data) {

        output OneLine : lines = 1, words = countWords(line);
    }

    () as Counter = Custom(OneLine) {

        logic state : mutable LineStat sum = { lines = 0, words = 0 };
        onTuple OneLine : addM(sum, OneLine);
        onPunct OneLine : if (currentPunct() == Sys.FinalMarker)

        println(sum);

        }

}
```

Development environment

InfoSphere Streams provides an agile development environment consisting of the Eclipse IDE, the Streams Live Graph view, and a streams debugger. The platform also includes toolkits that speed and simplify the development of solutions for particular functionalities or industries:

- **Standard Toolkit** — Contains the default operators shipped with the product:
  - *Relational* operators such as `Filter`, `Sort`, `Functor`, `Join`, `Punctor`, and `Aggregate`

    - *Adapter* operators such as `FileSource`, `FileSink`, `DirectoryScan`, and `Export`
    - *Utility* operators such as `Custom Split`, `DeDuplicate`, `Throttle`, `Union`, `Delay`, `ThreadedSplit`, `Barrier`, and `DynamicFilter`
- **Internet Toolkit** — Includes operators such as `HTTP`, `FTP`, `HTTPS`, `FTPS`, and `RSS`.
- **Database Toolkit** — Supports DBMS, including DB2®, Netezza, Oracle Database, SQL Server, and MySQL.
- **Other built-in toolkits** — Financial, data mining, big data, and text toolkits.

Try it

Download a 30-day full-featured trial version of InfoSphere Streams.

In addition, you can define your own toolkits that provide reusable sets of operators and functions and let you create cross-domain and domain-specific accelerators. They can include primitive and composite operators, and they can use both native and SPL functions.

Integration and interaction between BigInsights and InfoSphere Streams

Companies that continuously generate large volumes of valuable data from their systems are grappling with the problem of analyzing the data for two important purposes: sensing and respond to current events in a timely fashion and making predictions from historical learning to guide the response. This situation raises the need for the seamless functioning of data in motion (current data) and data at rest (historical data) analysis, operating on massive volumes, varieties, and velocities of data. Integration of IBM's data-in-motion (InfoSphere Streams) and data-at-rest (BigInsights) platforms addresses three main application scenarios:

- **Scalable data ingest** — Continuous ingestion of data through Streams into BigInsights. For example, unstructured text data from social media sources, such as Twitter and Facebook, usually need to be ingested to extract sentiment and leads of various kinds. In this case, it is much more effective if the text extraction is done on data as it is being ingested, and irrelevant data such as spam is eliminated early. This integration can enable companies to avoid huge unnecessary storage costs.
- **Bootstrap and enrichment** — Historical context generated from BigInsights to bootstrap analytics and enrich incoming Streams data. BigInsights can be used to analyze data it has assimilated and integrated from various continuous and static data sources over a large time window. Results from this analysis provide contexts for various online analytics and serve to bootstrap them to a well-known state. Returning to the scenario of social media applications, an incoming Twitter message only has the ID of the person posting the message. However, historical data can augment that information with attributes, such as influencer, creating an opportunity for a downstream analytic to treat the sentiment expressed by this user appropriately.

- **Adaptive analytics model** — Models generated by analytic operations, such as data mining, machine learning, or statistical modeling, on BigInsights. These can be used as basis for analytics on incoming data on Streams and updated based on real-time observations.

The data-in-motion and data-at-rest parts of the IBM big data platform can be integrated via three main types of components:

- **Common analytics** — The same analytics capabilities can be used on Streams and BigInsights.
- **Common data formats** — Streams formatting operators can transform data between the Streams tuple format and data formats used by BigInsights.
- **Data-exchange adapters** — Streams `Source` and `Sink` adapters can be used to exchange data with BigInsights.

Conclusion

Helping companies manage, analyze, and benefit from big data is the key area of focus for the IBM big data platform. In this article, you were introduced to InfoSphere Streams, IBM's software platform for storing and analyzing data in motion (streaming data). You also got an overview of how to integrate InfoSphere Streams with BigInsights, IBM's software platform for storing and analyzing data at rest, to enrich your ability to achieve more-complex analytics. Many companies recognize that capitalizing on big data is an important information management tool for providing unique business value and advantages. If you're ready to get started with InfoSphere streams, see Resources for free training materials and software.

Resources

Learn

- Visit the InfoSphere Streams website.
- Explore InfoSphere Streams resources IBM InfoSphere Streams on developerWorks.
- Access the IBM InfoSphere Streams Information Center.
- Reac *IBM InfoSphere Streams: Assembling Continuous Insight in the Information Revolution* (Chuck Ballard et al., IBM Redbooks, October 2011) to learn about the positioning, functions, capabilities, and advanced programming techniques for IBM InfoSphere Streams.
- Learn more about InfoSphere BigInsights, a platform for the analysis and visualization of Internet-scale data volumes. BigInsights is powered by Apache Hadoop and is designed to help IT professionals quickly get started with big data analytics using Hadoop.
- Learn more about IBM InfoSphere Streams.
- Manage and analyze massive volumes of structured and unstructured data at rest with IBM InfoSphere BigInsights, IBM's mature Hadoop distribution for big data analytics.
- Learn more about big data in the developerWorks big data content area. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with developerWorks technical events and webcasts.
- Follow developerWorks on Twitter.

Get products and technologies

- Get a trial version of IBM InfoSphere Streams.
- Get a trial version of IBM InfoSphere BigInsights and manage and analyze massive volumes of structured and unstructured data at rest.
- Build your next development project with IBM trial software, available for download directly from developerWorks.

Discuss

- Participate in the discussion forum for this content.
- Check out the developerWorks blogs and get involved in the developerWorks community.

About the author

Sherif Sakr

Dr. Sherif Sakr is a senior research scientist in the Software Systems Group at National ICT Australia (NICTA), Sydney, Australia. He is also a conjoint senior lecturer in the School of Computer Science and Engineering at University of New South Wales. He received his doctorate in computer science from Konstanz University, Germany, in 2007. His bachelor's and master's degrees in computer science are from Cairo University, Egypt. In 2011, Dr. Sakr held a visiting research scientist position in the eXtreme Computing Group (XCG) at Microsoft Research, in Redmond, Wash. In 2012, he held a research MTS position in Alcatel-Lucent Bell Labs.