Market Requirements Document

Feature Name: Objectivity/Parallel_Ingest

Version: 0.1 Date Submitted: 1/4/16 Completed By: Brian Clark

Description of the Problem

As part of the Objy30 initiative there is a need to support parallel multi-source ingest of data and relationships. However without taking extra steps it is very likely that lock conflicts and even deadlocks will occur, thus severely restricting ingest performance.

Background

For data with a high degree of cross connectivity (like a graph) there are problems with just going multi-threaded or multi-process. Because of the nature of the graph like connectivity lock conflicts and even deadlocks are likely to occur. Multiple lock servers may be OK for data that has some natural clustering like trees or hierarchies, or some other segmentation like all data for a particular device stored with that device and limited cross device connectivity, but it does not work for the more general graph use case where any object could be connected to any other objects. This situation was observed in the InfiniteGraph use case where the pipelining and pipeline agents turned out to be a reasonable solution. Another use case showed up with a Wynyard use case. In this case Wynyard implemented a pipeline in the application code.

Description of the Requested Feature

As a database developer I want to be able to ingest in parallel from multiple streams of data and relationships without having to write extra code to re-organize the data and relationships prior to ingest.
As a database developer I want to be able to ingest in parallel from multiple streams of data and relationships without having to write extra code to handle lock conflicts and deadlocks.
As a database developer I want to be able to ingest in parallel from multiple streams of data and relationships using the standard database API for creating objects and relationships.
As a database developer I want to be able to parallelize using multiple threads within the same ingest application or multiple processes, for distributed processing, using the same API.
As a database developer and user I want to be able to control placement of newly created objects and relationships across multiple locations, whether multiple disks on a single machine, or multiple disks across multiple servers, or in the cloud. In some environments this may be similar to splits/splitting streams.

As a database developer and user I want to tune and/or control the parallel ingest declaratively or via external properties.
As a database developer and user I want to use the same API and tools whether dealing with batch processing of data or streaming data.
As a database developer I want programmatic control of input data processing prior to creating new objects and relationships.
As a database user I want the database to provide both data and relationship ingest and data and relationship creation functionality.
As a database user I want to be able to specify full ACID consistency or eventual consistency.
As a database developer and user I would like to be able to vary the ingest batch size and direct specific batches to specific parts of the database.
As a database developer and use I want to have control over the latency between data/ relationship creation and availability to the query system.
As a database developer and user I want to have some control of the balance between required rate of ingest and the availability of processing resources, to ingest data without affecting data availability and to maintain flexibility in the volumes of data ingested.

Part of an existing feature or does it require another feature, if so, which one?

- New feature.

How is this problem being solved now, and why isn't that acceptable?

- Customers have to write their own code to re-organize the data in a more suitable order for parallel ingest. This is error prone and time consuming. Usually involves multiple stages such as pre-sorting and some sort of pipeline processing.

What languages must support this capability?

Initially Java for ThingSpan but eventually all languages will benefit, C++, Java, C# .Net, Python.

Which platforms must be supported?

- Should run on Linux, Windows and Mac OS X.

Do any competitors already have this feature?

- Most databases have some sort of parallel ingest support.

Customers who require this feature

- Although not ThingSpan, Wynyard implemented their own database ingest framework including pipeline capabilities.

Revenue at risk, or which could be won

- Could lead to more early adopters.

When is this required?

- First implementation (assuming phased implementation) next version of ThingSpan.

Additional Notes

Some examples:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1304ingestcmd/

https://accumulo.apache.org/user_manual_1.3-incubating/High_Speed_Ingest.html

http://www.jigsawsecurityenterprise.com/#!Stateful-Parallel-Ingest/jp1q9/5653740c0cf29e70f2240e1c

1. **Implementation notes:**
    a. Goals:
        i. Safe, high quality, low cost, fastest time to market;
        ii. High performance, massive scalability and optimal physical storage efficiency.
        iii. Minimal disruption to the existing Objectivity/DB kernel.

2. **Related Material**
    We will also need:
        Field Training.
        Quality Assurance.

3. **Software requirements**

4. **Hardware Requirements**