Market Requirements Document

Feature Name: Objectivity/DB and Apache Spark Integration

Version 1: 27-June-2014 Author: Leon Guzenda

1. Description of the Problem

Objectivity/DB has a number of features designed to make it easy and efficient at handling parallel workflows:

- Multithreaded processes and Session Pools
- Shared caches
- Parallel Query Engine
- Rich clients that access distributed data, lock and query agents.

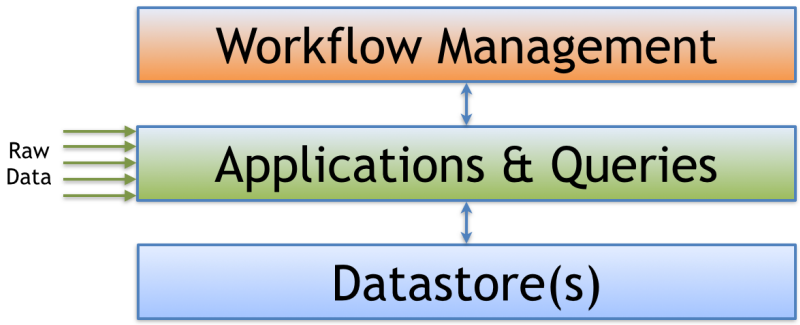
InfiniteGraph supplements the above features with a parallel, high speed ingest capability that avoids lock contention by using multiple pipelines and an eventually consistent transaction model.

Although Objectivity/DB and InfiniteGraph can sustain extremely large ingest, processing and query loads the industry focus has shifted from unique solutions to standard models. The leading technologies are Apache Hadoop, with its MapReduce workflow model, and a new community effort called Apache Spark that adds streaming and iterative process control to the batch process controls that Hadoop provides.

Our users could build applications that run under the control of Hadoop or Spark, but there are no tools to make this easier for them and it would be impossible for them to exploit some of the more powerful features of Spark, such as its distributed, shared memory caching. They would also need to build their own interfaces to exploit the powerful toolsets within Spark, such as the Machine Learning Library (MLlib).

2. Apache Spark

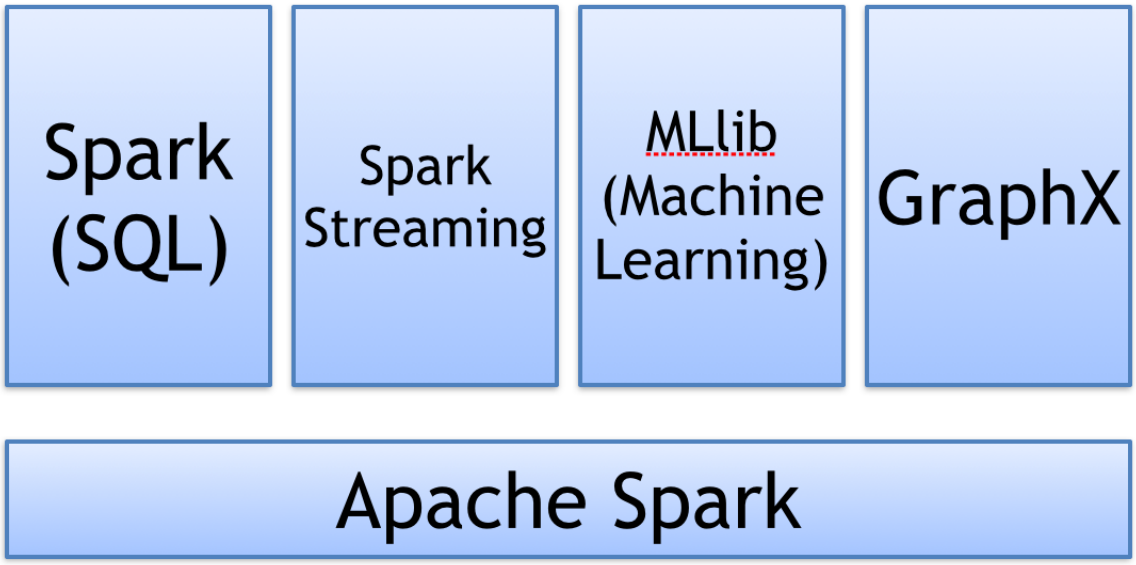
The diagram below shows the key functional components of a typical Big Data analytics environment. As an example, in Hadoop the Workflow Management component is MapReduce. There are tools such as Pig and Hive. The Datastore is HDFS.



Spark is a more performant and capable framework than Hadoop, for example:

- It has an interactive shell that can be used by non-programmers to access large datasets immediately.
- It improves workflow control by adding streaming and iterative process typos to the batch ones supported by Hadoop.
- It overcomes the high latency of the HDFS filesystem by providing a distributed, shared in-memory cache.

The diagram below shows the main components of the current (1.0) Spark release.

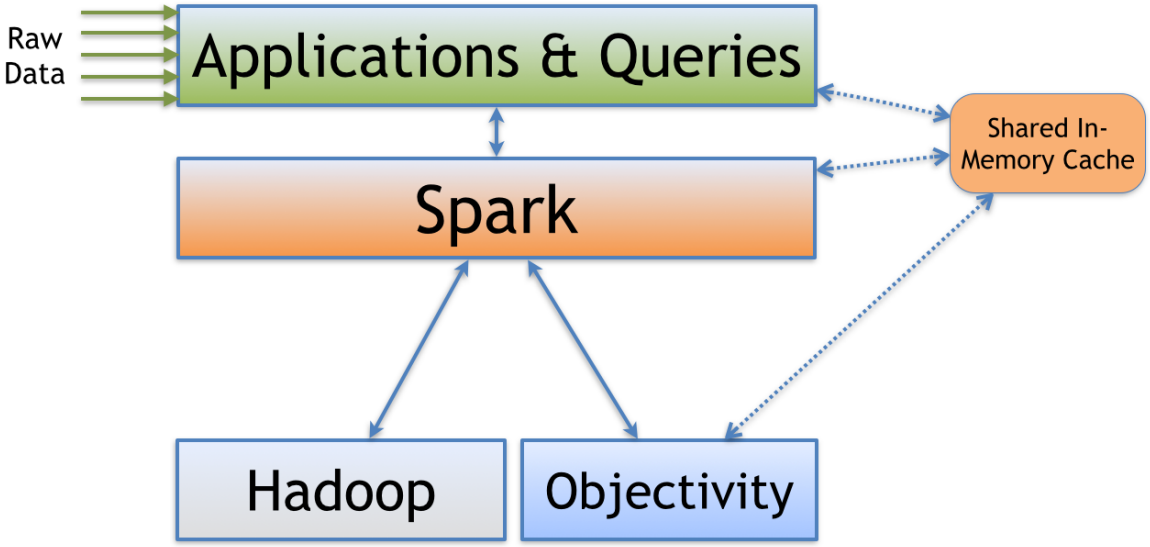


3. Justification for Integrating Objectivity/DB and Spark

Spark has limited graph processing capabilities, but it is not a data and connection store for highly interconnected structures, unlike Objectivity/DB and InfiniteGraph. it relies on a message-based, distributed graph algorithm that is similar to Google's Pregel and the open source Giraph algorithms. It is fairly efficient at traversing in-memory structures, but, like SAP's HANA, is limited by the total amount of available RAM.

Conversely, combining Spark's workflow, analytic and shared, distributed cache capabilities with Objectivity/DB will improve both pieces of technology with very little functional overlap.

The diagram below shows a high level version of the integrated systems. The initial goal is to exploit the workflow capabilities. A subsequent releases would exploit the Spark Shared, In-Memory Cache. [Note: Spark is shown at the center for convenience. It controls the workflow.] There is a more detailed depiction of a typical high throughput system in Appendix A.



The combined approach leverages three major technologies:

- Hadoop HDFS for fast ingest and file management
- Objectivity/DB for complex analytics
- Spark for workflow management and interactive queries and analytics across hybrid datastores.

This approach will empower system integrators, such as Wynyard Group, to improve their current offerings with high throughout, advanced analytic functionality, such as machine learning. It will also allow them to provide systems with improved scalability and elasticity while shortening the time to deployment.

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

Although Objectivity/DB customers have been running systems in grid environments since 1998 and cloud environments since 2007 we are not aware of any production usage in a Hadoop or Spark environment. This is mainly because of the current features described previously, but it is also probably inhibited by the lack of Hadoop or Spark integration tools from Objectivity.

The industry attention given to Hadoop (and Spark now) has also probably cost us business as people selected alternative NoSQL solutions, even though they are often more difficult to install and maintain.

4. Description of the Requested Feature

The integration of Objectivity/DB and Spark will provide three main groups of features:

- Tools and wrappers for installing Objectivity/DB in a Spark environment and for running client applications, lock servers, Advanced Multithreaded Servers (AMS) for "remote" data access and query agents under Spark.
- Analytic and visualization tools for analyzing Spark process logs.
- Exploitation of the Spark shared, distributed in-memory cache by client processes, in effect placing the Objectivity/DB internal cache into that address space in situations where it will improve performance or resilience.

6. What languages must support this capability?

- .Net for C# (because of project Mustang)
- Java
- C++
- SQL++ (eventually, in support of the Spark SQL module).

7. Which platforms must be supported?

- Windows 7 (because of project Mustang)
- Linux
- Mac OS X

8. Do any competitors already have this feature?

No.

9. Customers who require this feature

- Wynyard Group - for Project Mustang.
- There may also be interest from customers in the telco and process control equipment, Big Science and Intelligence Community markets.

10. Related Material

We will also need:

- New Quality Assurance material.
- Updated documentation, training and web based training.
- Marketing and sales collateral.

11. Implementation Recommendations

Goals

There are three primary goals:
1. Safe, low cost, fastest time to market.
2. Minimal disruption to existing products, especially the Objectivity/DB kernel.
3. Enhancing the usability, applicability and power of Objectivity/DB applications, such as ACA built on Xerem/Objectivity.

Phases

Phase 1: The Objectivity/DB shared processes (lock server, AMS data servers and PQE query agents) will be enclosed in Spark wrappers so that they can be started, accessed, monitored and stopped via the Spark workflow mechanisms and interactive shell. It is not yet clear whether choosing iterative or streaming process types will yield the highest performance and resilience. This interim release will be easier to use with…

Phase 2: This will provide:
- Sample wrappers (that can be run as Tasks by a Spark Executor on a node under the control of a Cluster Manager driven by a SparkContext) for Objectivity/DB client applications (such as ACA and Xerem/Objectivity).
- Spark scripts for deploying and controlling them.
- Extensions to the Placement Manager to enable it to utilise the Spark environment.

Phase 3: This will add additional monitoring capabilities to Objectivity/DB, in particular:
- Coupling the Kernel Event Mechanism and the Lock Server monitoring output with Spark logging, monitoring and analysis tools.
- Adding monitoring outputs to the Advanced Multithreaded Server and parallel query agents and coupling them to the Spark logging, monitoring and analysis tools.

Phase 4: This will make data in any Objectivity/DB federated database accessible to standard Spark tools, such as MLlib and SQL. It is not yet clear whether this is possible via the streaming source interfaces, database adaptors, emulation of Spark structures or some other mechanism. Hopefully, there will be more details of datastore replacement capabilities in the upcoming release. The issue will be addressed as soon as possible.

Phase 5: This will integrate the Objectivity/DB front end caching mechanism (upcoming release) with the Spark shared, distributed, in-memory cache. This capability will be configurable as some processes, such as simple Extract-Translate tasks, probably won't benefit from the capability.

APPENDIX A - Spark and Objectivity/DB - Big Data Use Case

The diagram below shows a typical three-stage Big Data analytics system, perhaps supported by machine learning algorithms. This is the data/process flow:

- Raw streaming data or input from other systems is handled by Extract-Translate streaming jobs that store incoming streams and files in the Hadoop HDFS filesystem, perhaps with the assistance of a Key-Value store.
- The data fusion or filtering and correlation tasks (such as ACA modules) run in Spark Iterative jobs that access existing data in Objectivity/DB and create additional objects and relationships there. These two sets of applications are combined under a single, tightly coupled workflow controller, shown as Task 1. Note that this actually equates to a bunch of programs running on a node under the control of an Executor, with access to a shared cache.
- Queries are supported via the Spark interactive shell and deep search (data-mining and analytic) tasks are run as Spark batch processes (Task 2).