The OrbitOutlook Data Archive

Michael Czajkowski, Andrew Shilliday, Nicholas LoFaso, Asif Dipon, David Van Brackle
Lockheed Martin Advanced Technologies Laboratory

Abstract

In this paper, we describe and depict the Defense Advanced Research Projects Agency (DARPA)'s OrbitOutlook Data Archive (OODA) architecture. OODA is the infrastructure that DARPA's OrbitOutlook program has developed to integrate diverse data from various academic, commercial, government, and amateur space situational awareness (SSA) telescopes. At the heart of the OODA system is its world model – a distributed data store built to quickly query big data quantities of information spread out across multiple processing nodes and data centers. The world model applies a multi-index approach where each index is a distinct view on the data. This allows for analysts and analytics (algorithms) to access information through queries with a variety of terms that may be of interest to them. Our indices include: a structured global-graph view of knowledge, a keyword search of data content, an object-characteristic range search, and a geospatial-temporal orientation of spatially located data. In addition, the world model applies a federated approach by connecting to existing databases and integrating them into one single interface as a "one-stop shopping place" to access SSA information. In addition to the world model, OODA provides a processing platform for various analysts to explore and analytics to execute upon this data. Analytic algorithms can use OODA to take raw data and build information from it. They can store these products back into the world model, allowing analysts to gain situational awareness with this information. Analysts in turn would help decision makers use this knowledge to address a wide range of SSA problems. OODA is designed to make it easy for software developers who build graphical user interfaces (GUIs) and algorithms to quickly get started with working with this data. This is done through a multi-language software development kit that includes multiple application program interfaces (APIs) and a data model with SSA concepts and terms such as: space observation, observable, measurable, metadata, track, space object, catalog, expectation, and maneuver.

1. Introduction

There are more than a half million pieces of manmade debris orbiting the Earth. In the past decade, the SSA community has developed many different networks incorporating many sensors. Handling large quantities of data and fusing them together into a common picture has proven difficult. DARPA's OrbitOutlook program is working towards making this easier. The program has created the OrbitOutlook Data Archive (OODA) as the data persistence and retrieval framework for all data providers and algorithms to use to create this common picture.

OODA includes three key functional layers: a messaging framework, a process management infrastructure, and a distributed database called the world model (WM) (see Fig. 1). OODA is integrated with a SSA data model as a lingua franca for all system component interaction. The messaging framework layer provides communication for all components within the OrbitOutlook system by enabling messaging over multiple protocols. Its features include robust and guaranteed message passing through a "request for service" protocol and the capability for components to broadcast information to a variety of topic streams to allow other components to subscribe to. The process management layer provides the capability to schedule, maintain, and monitor the lifecycle of all system components. OODA's process management ensures that the components are allocated with the correct resources they need to successfully execute their goals.

The world model is OODA's distributed scalable database that supports multiple indices unified by a single standardized query and ingest handler. Data that comes in to the world model is stored in an object store that persists terms found in a SSA data model. The ingested objects also are sent to a myriad of indexers that provide a custom view of the data accessible via a rich set of query operands. The world model supports indices that view the data as a dynamic graph, geospatial-temporally located, as attribute characteristics, and as unstructured text. The world model is also capable of forwarding data and queries to other external data base back ends through custom indexers. These indexers can transform the OrbitOutlook data structures to different formats as required. The scalability of the world model comes from its back end, hosted through three open source big data technologies: Apache Hadoop, Apache Zookeeper, and Apache Accumulo [1,2,3]. These popular big data technologies provide capabilities for service synchronization, storage on a distributed file system, and a big-table data retrieval methodology [4]. The world model uses these to scale out by adding more processing nodes as needed to handle larger velocity and volumes of data ingest.

Each of these infrastructure layers interoperates with a SSA data model that includes several terms. The data model also allows the tracking of pedigree to determine what data provider or algorithm created the data.

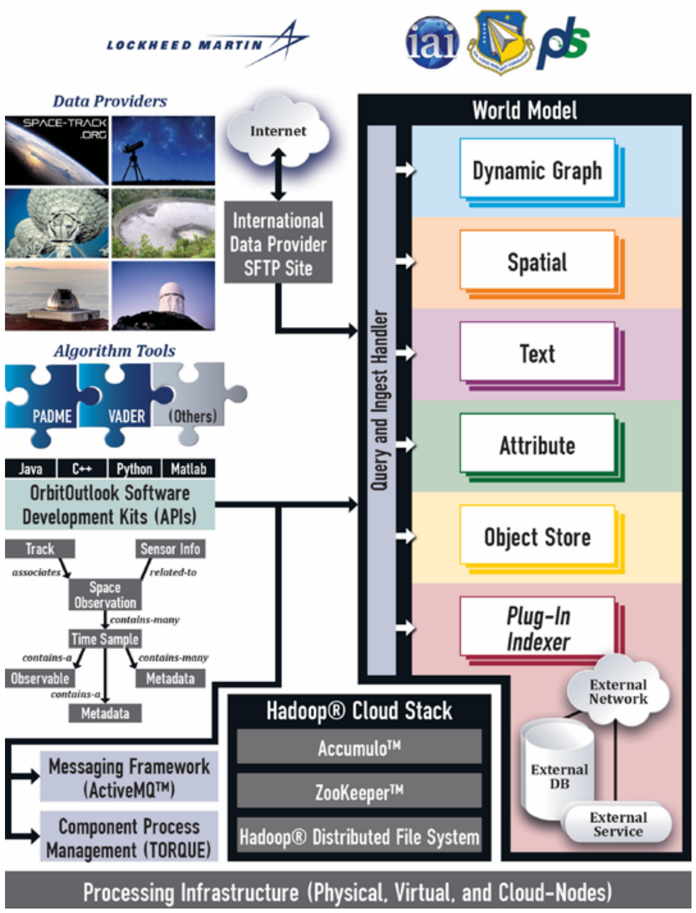

Fig. 1 The OrbitOutlook Data Archive Overview

OODA provides access to its various services and this data model through a set of APIs found in the OrbitOutlook software development kit (O2SDK). The APIs "hide" the underlying OODA system so that data providers and algorithm developers don't have to deal with the nuances of deployment and execution of a software system on its underlying hardware.

The rest of this paper is organized as follows. Section 2 goes into the world model in more depth. Section 3 covers the data model and the O2SDK APIs in further detail. Section 4 addresses an application created to allow international access to storing and retrieving data within OODA's world model. Section 5 discusses OODA's current implementation and Section 6 provides conclusions.

2. The World Model

The OODA world model is the foundational unified data and information store for the OrbitOutlook system. The world model is designed to be highly distributed and scalable to support massive multi-dimensional data at high ingest and query rates. Internally, the WM employs existing and homegrown data indexing systems that operate independently. Each indexer structures its information according to a particular dimension or view into the data. The world model supports queries that scan the entire known history or alert a component when data arrives that matches the query. The latter is called a 'standing query.'

A central information broker called the world model daemon (WMD) maintains modularity and standardization of independent and individually scalable indexers. The WMD provides the unified access layer through which clients (OrbitOutlook algorithms or data providers) can commit new or updated data to the WM or execute multi-dimensional, multi-index queries. The WMD handles all interaction with underlying indexers by distributing new data and data requests out to individual index instances (each indexer class may have multiple instances running). The WMD does this according to replication and scalability strategies to maximize throughput and availability. Currently, the WM indexer suite consists in classes of indexers covering four primary dimensions of the data: graph, geospatial-temporal, text, and attributes.

The dynamic graph indexer stores relationships between data objects, as <subject, predicate, object> triplets as seen in ontologies. It allows queries to "walk through" a graph to determine the structure of data by relationships. The graph indexer employs Apache Accumulo for its underlying data store and organizes relationships into table according to the Rya architecture [3,5]. Each subject-predicate-object relationship is stored into Accumulo tables as three keyed records: one in subject/predicate/object order, another in predicate/object/subject order, and the third in object/subject/predicate order. In this way, the graph indexer can quickly answer a large range of graph queries with fast direct key lookups.

The spatial indexer extracts space and time attributes from the objects and indexes them in a way suitable for spatiotemporal queries against the data. The back-end database system for the spatial indexer is a custom high-performance disk-based tree that supports the following spatial query types: contains, crosses, disjoint, equals, intersects, overlaps, touches, within, and within-distance. Spatial indexing supports custom plugins for processing individual data types. Plugins can be created independently of the rest of the system and are dynamically loaded by the indexer at runtime. The spatial indexer scales by having multiple instances be fed a part of the data and all queries go to all index instances to seek data.

The text indexer provides for data searches of unstructured text attributes within data objects. The indexer utilizes Apache Lucene as the basis of a data store and allows for keyword-based queries and regular expression searches [6]. The text indexer scales also by having multiple instances be fed a part of the data and all queries to go to each instance to seek data.

The attribute indexer organizes data according to individual attributes within the data object. Clients can execute range queries on numeric and string-based attributes (e.g., a query for all radio frequency (RF) observations with frequencies between X and Y MHz). As the OrbitOutlook data model is specified as an XML schema, the attribute indexer supports querying within lists of values for a given attribute.

As each indexer is concerned only with the dimension of the data for which it's responsible, no indexer preserves the complete object within its index. The graph indexer, for example, only cares about the existence of a node and relationships that hold between them. The specific details and attributes within a node are stripped out and dismissed. When clients execute queries and the WMD in turn queries the individual indexers, the indexers return identifiers (IDs) of objects that match against the query. The WMD then turns to the object store to retrieve the complete details of the object to be returned to the client.

The world model can also connect to existing data stores that exist outside our system by plugging them in as new indexers. This detail is hidden from the analytics that gain immediate access to new information. For example, one such database might be a source available on a service-oriented architecture (SOA). These pluggable indexers handle data translation for ingest and query translation for access to the data contained within.

Underlying the entire world model is a Hadoop-based cloud stack containing three key technologies used to store and retrieve its data as well as facilitate communication between its internal components [1]. Hadoop's distributed file system is a software framework for distributed storage and processing of very large data sets across many nodes in a cloud. The world model uses this as its back-end storage for indexers and the object store to persist big data volumes of data model objects. Apache ZooKeeper is a distributed management tool whose job it is to maintain state of an application between multiple instances running on different nodes and provide service discovery capabilities to systems that communicate with it [2]. The world model uses ZooKeeper to understand the state of its indices and WMD instances during execution. Finally, Accumulo is a big-table implementation that excels in storing vast quantities of sparse-record data on top of the Hadoop distributed file system [3,4]. The world model uses Accumulo to power its object store, graph, and attribute indexers for scaling and fast data retrieval.

3. The OrbitOutlook Data Model and Software Development Kit

Interoperability between components in any system is important in order for data providers, algorithms, and human-machine interfaces (HMIs) to talk to one another. To achieve this end, the program developed a data model to capture the more popular terms found in the SSA domain. The data model contains detailed descriptions of terms such as space observation, observable, metadata, measurable, sensor information, track, space object, catalog, and two-line element set (TLE) (see Fig. 1). For example, the term space observation is how the data model records raw data. Space observations contain a list of time slices that are observed quantities and measurements with associated metadata for an observation setup. The data model also captures pedigree, which links a component to the source data it takes in and the product it generates. Using pedigree, the world model can be queried to find "chains" that link every object ingested to where it came from.

The data model is defined in the XML schema definition language (XSD) [7]. These schemas are used to generate Java and C++ objects using Java Architecture for XML Binding (JAXB) and CodeSynthesis XSD tools respectively [8,9]. The data model is split up into core schemas and domain specific schemas. This separation allows OODA to be built off of the core and also give the ability to add additional objects through schema updates for different operating environments. Extensibility is achieved this way without requiring recompilation of the entire system.

A software development kit has been created to contain all of these data model objects along with a set of APIs to allow access to OODA and other components in the OrbitOutlook system. This kit is called the OrbitOutlook software development kit (O2SDK) and is implemented in Java, C++, Python, and Matlab with communication interfaces generated through Apache Thrift [10,11]. This allows developers to build components in any of these languages while being able to communicate to components written in other languages through the Thrift interfaces. This capability can be expanded to several other languages with minimal effort. Additionally, the O2SDK has thorough documentation, samples, and a reference implementation to make it easier to create algorithms to integrate with the system.

4. International OODA

An objective of the OrbitOutlook program is to allow selected international parties to provide observation data and to have access to collected observation data that OODA is aware of. The OODA team supports these international data providers by providing access to a server using Secure Shell (SSH). This server is simply a data store that is separate from an instance of OODA.

In order for data to make it into the world model, it first must be put in a common, accessible location known only to the data provider. This is accomplished using private drop-box folders for each provider. As data is generated these providers, the owners can upload the data to the drop-box manually or through an automated script. The OODA team monitors the various drop-boxes and applies a custom translator to each to form O2SDK data objects. Several existing data formats are currently supported, including electro-optical space situational awareness (EOSSA) [12]. The O2SDK data model is extensible so that new terms and concepts can be included without invalidating data from other providers. New data sent to OODA is carefully tagged using pedigree and placed into the world model. Downstream components can then validate and verify the data as being useful or not.

The program also provides data that has been collected for use by others. This data is carefully separated from other international data partners so that only the program's data is made available. To ensure this happens, queries are done on the pedigree chains as earlier described. The program data is made visible on the same SSH server in a drop-box designed for download. The data is provided in a flattened comma separated value (CSV) format including headers on how to parse the data. The program is evaluating other possible standard formats to support.

5. Implementation

An instance of OODA has been operational since early 2015 in a custom-created OrbitOutlook Data Center (OODC) containing seven servers. Each has 64 GB of RAM and 5+ TB of hard disk space. The aforementioned Hadoop cloud stack and the world model occupy six servers, leaving the seventh as a staging "gateway" for program use.

Data being ingested into the OODC follows a fairly typical pattern. The producer of the data creates a component or uses a library of translators to bring their data in. This raw sensor data is turned into space observations as defined in the O2SDK. A track assembly component then receives these space observations via a world model standing query (see Section 2). The track assembler turns the space observations into tracks that link together associated observables. Algorithms use another standing query to consume both tracks and the raw data to process. All results are then published back in to the world model for further analysis.

The OrbitOutlook program is concluding an integration phase of component development at the time of writing this paper. The team is preparing for a realistic demonstration of the technology using live sensor data in late 2016 and early 2017. Live data (typically a few weeks' worth of data) will be collected. The analytic components of the program will process this sensor data in real time and provide an analyst with a view into the results. System effectiveness will be judged on a number of factors, primarily on the ability of the algorithms to comprehend the data properly. However, additional metrics such as the ability of OODA to scale to the velocity and volume of data will be measured.

6. Conclusion

With more sensors and networks of sensors coming online each and every year, the SSA community has a big-data integration problem. The OrbitOutlook program has been using OODA as a means to store large volumes of data and handle the velocity of data that arrives. The heart of OODA is the world model, a scalable distributed heterogeneous data store that provides access to stored data through a myriad of query types. OODA also provides a single location in which data providers, algorithms, and analysts can work together to explore the data to determine what it all could mean. In 2016 and 2017, live sensors from academic, commercial, government, and other sources will be streaming data into OODA and the world model running in the OODC. The future of having a common repository to use to evaluate vast quantities of data is taking form today and OODA is playing a key role.

Acknowledgements

The OODA team would like to thank the Defense Advanced Research Projects Agency (DARPA) who sponsored much of this work. The team recognizes the hard work done by Dr. Channing Chow who carefully constructed the OrbitOutlook data model through listening to many, having great patience, and paying attention to detail.

Only a few people can write a short paper on the OODA world model, but many more have spent significant time constructing it. The following people should be acknowledged for their contributions: Nicholas Morizio, Brendan Drew, Daniel Donavanik, Stephen Jameson, Alexei Samoylov, Robert Chevalier, Alison Craig, Thomas Damiano, Christopher Oster, and Jason Isabel.

The views, opinions and/or findings expressed are those of the authors and should not be interpreted as reflecting the official views or policies of the Department of Defense or the U.S. Government.

References

1. *Apache Hadoop*: http://hadoop.apache.org
2. *Apache ZooKeeper*: http://zookeeper.apache.org
3. *Apache Accumulo*: http://accumulo.apache.org
4. Chang F., et al, *Bigtable: A Distributed Storage System for Structured Data*, in proceedings of Operating Systems Design and Implementation 2006, Seattle, WA, 2006
5. Punnoose, R., Crainiceanu, A. and Rapp, D., *Rya: A Scalable RDF Triple Store for the Clouds* in proceedings of the 1st International Workshop on Cloud Intelligence, Istanbul, Turkey, 2012
6. *Apache Lucene*: http://lucene.apache.org
7. *XML Schema Definition*: http://www.w3.org/standards/xml/schema
8. *Java Architecture for XML Binding*: http://www.oracle.com/technetwork/articles/javase/index-140168.html
9. *Codesynthesis XSD*: http://www.codesynthesis.com/products/xsd
10. *Matlab*: http://www.mathworks.com/products/matlab/
11. *Apache Thrift*: http://thrift.apache.org
12. Payne, T. Mutsclher, S., and Shine, N., *A Community Format for Electro-Optical Space Situational Awareness (EOSSA) Data Products* in proceedings of Advanced Maui Optical and Space, Maui, Hawaii, 2014