Intel® Graph Builder for Apache Hadoop* Software v2

Answers to Common Questions

Intel® Graph Builder for Apache Hadoop* Software v2 simplifies creation of graph data models, enabling data scientists to focus on solving business problems, rather than formatting data. Prebuilt libraries automate workflows for cleaning, transforming, and constructing graph models with cost-effective parallel computation throughput using Apache Hadoop. Once built, the graph model can be operated on using a wide variety of graph databases, analytic engines, and visualization tools. By automating previously laborious, custom workflows and substantially removing the complexities of cluster computing for constructing large graphs from Big Data, Graph Builder helps speed the time to insight for data scientists using powerful graph analytics.

What is a "graph"?

"Graph" refers to a way of modeling data as networks of relationships. (In data science, data entities are known as "vertices," and the links between entities are known as "edges.") You can think of a "graph" as a data structure that is naturally visualized as a network or tree, where each section has associated values. Data is stored in a way that makes it easy to traverse the links. This is in contrast to traditional databases, which are naturally visualized as a table or spreadsheet of rows and columns. Traditional databases are optimized to look up data using specific row and column parameters, and calculate statistics based on rows or column attributes. Graph databases are good for seeing what data is linked to other data. This includes traversing many degrees of arbitrary dependence and precedence, easily answering complex queries, and extracting other useful algorithmically derived information about relationships, such as the shortest path or central influencers of separation.

What is graph construction?

Like the ETL process for data warehouses, graph construction begins with extracting data from the original sources; cleaning it of missing data points or erroneous data; and extracting, transforming, and reformatting raw data into specific parameters of interest (known as "features" in data science) .

But there is an important distinction between ETL for data warehouses and graph construction. For ETL, the choice is which data to pull. Then, there is a frequently tedious process of pulling from different sources and dealing with different formats, missing values, incorrect outliers, and other tasks associated with "cleaning" the data. When building a graph, all of these challenges also exist, but with the added task of choosing which features to connect together to form the graph data points. How to do this may not be obvious—multiple contexts may be used to relate the data. This is part of the discovery process and the "art" of data science—finding which model yields the most impactful insight. One of the advantages of Graph Builder is that it simplifies the actual steps needed to construct the model, making it easier to try different models that ultimately enhance the insight.

What is Intel Graph Builder?

Intel Graph Builder is a set of libraries for constructing large-scale graphs. It helps automate this process with prebuilt tools to offload the many traditionally manual and tedious steps. For example, to make sure the data is formatted properly, graph connections are assembled properly. We do this using MapReduce*, so it can be done with high throughput on Big Data. Writing a MapReduce program can be complicated for new users and time-consuming for experienced ones. Graph Builder offloads much of that complexity from data scientists, allowing them to focus on writing specific dataset parsing routines with Graph Builder, making it executable as a parallel program for high throughput.

What is graph analytics?

Graph analytics is the application of algorithms to graph data to find answers to business or computing problems. Graph analytics include important metrics that can be "mined" from the graph. For example, the shortest path between two points is a classic statistic of interest in graph data. This is something users experience with social networks to see how they are connected to other users. Centrality is another; this measure the importance of a node along the paths of other data points. It also includes the application of machine learning algorithms executed across the graph data iteratively calculating values of interest. For example, Google page rank is a graph-learning algorithm that iterates across each hyperlink on the World Wide Web—a form of graph data—to calculate a relevance score that is propagated across all of the links. Examples of graph-learning algorithms include belief propagation and page rank.

Categories of optimized tools for using graph analytics include:

Graph databases:
Platforms optimized for storing, querying, and transacting with data in graph form. They are considerably faster for traversing relationships and dependencies in linked data sets than relational databases. Examples of Graph databases include Neo4J*, the Aurelius Titan* database, the Franz AllegroGraph*, the Teradata Aster SQL-GR*, and Oracle NoSQL* graph.

Graph engines:
Optimized for iteratively processing algorithms by updating the weight of vertices or edges, such as the page rank algorithms. Examples of graph engines include the GraphLab and Apache Giraph* open source projects.

Graph visualization tools:
Enable visual interaction with graph data that is typically stored in a graph database. Most graph databases also include some basic visualization capabilities. Graph visualization may also be included in business intelligence applications. The Gephi open source platform is an example of a graph visualization application.

Why would I use graph analytics tools?

Graph tools provide a more intuitive way to create predictive models and articulate queries. It is natural for the human mind to work with tree-structured networks when conceptualizing interactions. Building a model using a graph structure is often much easier. For example, in contrast to a graph database, capturing multiple degrees of separation with relational databases requires multiple table joins, a process that is very slow and results in gigantic tables that are hard to work with.

Likewise, visualizing data is easier using graph structures. When searching, querying, or exploring data, easily being able to zoom in and out of dependencies makes data analysts more productive, and graph statistics enable tools to automatically focus the visualized data on the local points of interest.

Graph analytics improves machine learning solutions by incorporating parameters about relationships between objects, versus just the objects themselves. For example, a website that wants to suggest similar items for consideration could examine similar attributes of the items in the catalog, but is much more powerful when it includes information about relationships of the items and the users. Users of LinkedIn, Amazon, or Netflix usually receive high-quality, highly relevant suggestions, because they incorporate graph data into the machine learning tools to create these recommendations.

How does a data scientist interact with Graph Builder?

Most programs written using Graph Builder can be scripted with Apache Pig*, a much simpler programming environment than writing Java* code to MapReduce APIs. This means that the routines operate at cluster scale out-of-the-box, eliminating additional complex programming. Graph Builder provides libraries for cleaning and transforming data and constructing many types of graph. Graph construction is usually problem specific, so Graph Builder helps to substantially reduce the code-writing effort to focus only on the problem-specific areas, such as parsing a unique data source, while making it easy to script the rest of the parsing, cleaning, transforming, and graph construction routines using prebuilt library functions.

Users can extend Graph Builder functionality with their own Pig user-defined functionality, altering the underlying Java MapReduce code that Pig generates, or by modifying the open source routines.

What output formats does Intel Graph Builder generate?

Graph Builder can output graphs in files containing resource description framework (RDF), a format for capturing lists of edges and their properties. These are standard formats that can be read by most graph databases.

In addition, Graph Builder includes routines to load the Aurelius Titan database in parallel (instead of sequential) form. Because Titan is scalable and can operate over multiple nodes, this can include loading at cluster scale. This further speeds up the graph processing pipeline into a fully scalable solution.

Because Graph Builder is open source, it is possible to add optimizations for parallel loading into other graph database solutions as well.

How is Intel® Graph Builder for Apache Hadoop* Software v2 different from the previous Graph Builder open source release?

Graph Builder 1 demonstrated the opportunity to use scalable graph construction as part of a complete pipeline for performing graph analytics on Big Data. Its functionality was limited and users were required to develop a substantial amount of MapReduce code to achieve a useful application. Intel® Graph Builder for Apache Hadoop* Software v2 takes a major step forward in building out the scope of functionality, greatly simplifying the effort for the data scientist to write the graph construction MapReduce program.

COMPARING GRAPH BUILDER 1 AND INTEL® GRAPH BUILDER FOR APACHE HADOOP* SOFTWARE V2

FUNCTIONALITY	GRAPH BUILDER 1	INTEL® GRAPH BUILDER FOR APACHE HADOOP* SOFTWARE V2
Hadoop* Platform Requirements	Hadoop 1 (MapReduce*)	Hadoop 1.2.1 (MapReduce, Apache Pig* 0.12.0, HBase* 0.94.12)
Programming Environment	Java* coding using MapReduce APIs	Apache Pig scripts and/or Java UDF extensions in Apache Pig
Input Format Parsing	XML (+ user defined)	XML, CSV, TSV, JSON, (+user defined)
Types of Graphs Supported	Unlabeled directed graphs	Directed and undirected graphs, multirelational (i.e., property) graphs with vertex and edge properties and labeled edges
Data Cleansing		Extensive PIG libraries for: ▪ String manipulation ▪ Null checks ▪ Table manipulations ▪ Common math operators
Output Format	Edge list (text on HDFS)	Edge list (text on HDFS) RDF triples (on HDFS)
Graph Database Connector (Output)		Aurelius Titan*

For more information:

Explore Intel® Graph Builder for Apache Hadoop* Software v2 and graph tools, and look for downloads and resources at: intel.com/graph

Learn more about Intel® Datacenter Software tools at: intel.com/datacentersoftware.