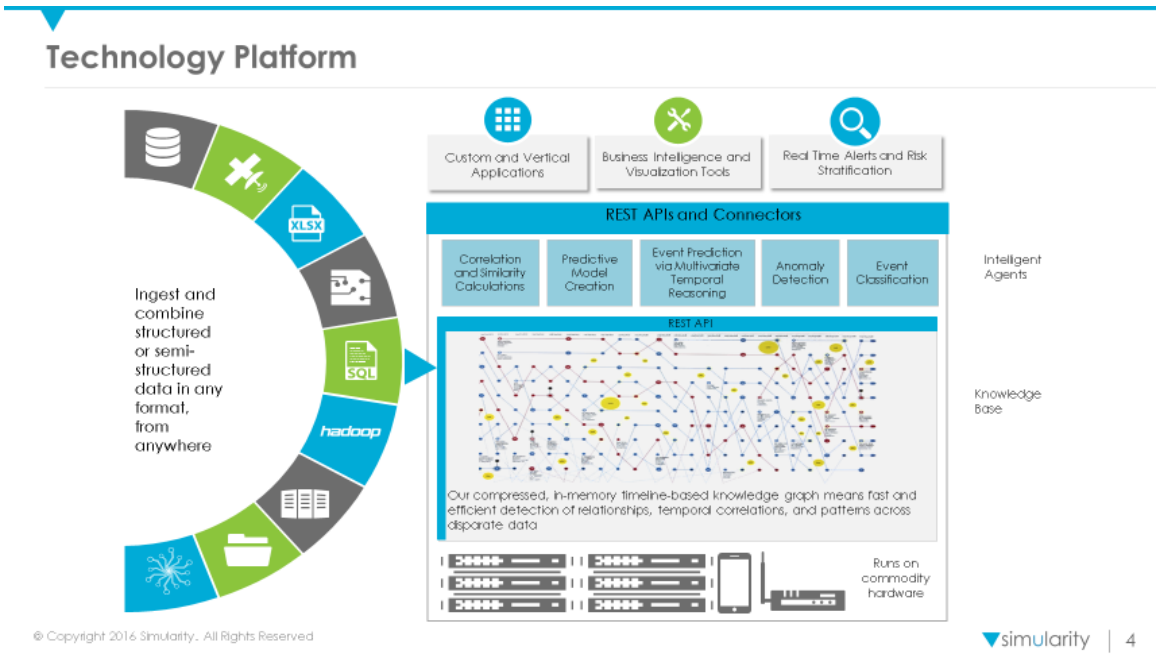
The Simularity AI Operational Paradigm



The Simularity AI consists of two major components, the segment (or a set of segments that act as a single segment) and a set of agents, which reason over the segment. New agents may be created, either by the user or by Simularity, in any programming language which supports HTTP POST and JSON. Simularity primarily uses Prolog as an implementation language for agents, and a Prolog interpreter and support modules for Prolog agents are supplied with the Simularity AI package. Support modules for Java and Ruby are forthcoming.

A common question about the Simularity AI is "what models do you support?". The answer to this question is complex as the "model" is encapsulated in the agent – if you want a new or different model, simply write or acquire an agent that implements that model. Simularity provides a few standard agents with every distribution, these agents implement a Hidden Markov model for discrete event prediction, and Neuroevolution for continuous event prediction and anomaly detection. These agents all benefit from the *segment*, which provides a time series database which can be queried for temporal correlations, among other things. Many of

the problems with implementing a model over an actual time series are solved by using the segment and its associated code.

The Segment

The segment is a time-series database that stores information like sensor readings, events, and other information on timelines, where each timeline is associated with a context (an identifier for the real-world item the timeline is about, such as an employee ID or machine number). This data is intended to be dynamically updated, and all operations on the segment allow for the fact that the operations happen concurrently with, and considering dynamic updates to the segment which may happen at any time. These timelines can be queried in a Set-Theoretic Query Language which allows complex time-based questions to be answered as either correlations (what are the most correlated things on the timelines covered by the query), or as chains of items on the timeline which correspond to the events (context + timestamp) specified by the query.

A query can ask questions like "What is most correlated with the 5 minutes surrounding a new network connection?", "What is most correlated to the time period preceding the first time a customer has an online interaction when that customer previously had in-person interactions?", or "What is the time series for the temperature sensor in the 1 hour preceding every machine failure?". Queries may be arbitrarily complex, and can include many set operations such as union or symmetric difference. It >should be noted that a Query specifies a set of events, and those events can then be used to determine correlations, or enumerate values on the timeline at those events.

A correlation simply is a measurement of difference between the likelihood that a timeline item occurs in the Query set, as opposed to the likelihood that it occurs in the universal set (for practical reasons, the total occurrence in the segment). Things that occur more often in the Query set are positively correlated and things that occur less often are negatively correlated. A number of metrics can be applied to determine correlation:

- Log Likelihood: a comparison of the of the Query Set to the Universal set
- Cosine – the cosine of the angle of the Event vectors of the correlated item vs the query. Very easy to compute
- Predictiveness: The probability that the correlated item "predicts" the Query

- Rate Ratio: The number of times more likely the correlated item is in the Query set than it is in the universal set

There are numerous others, they mostly have special purpose application to certain scientific and mathematical fields.

Data Ingestion

Data is ingested into the segment by a program called a *Loader*. Loaders may be written in any programming language which can perform an HTTP POST and interpret JSON. The loader reads the data in its original form, and translates it into the timeline format required by the segment. This timeline format is a general way to represent many kinds of data (it is not a universal data representation scheme like XML or JSON – it is designed to represent time series data only). The basic components of ingested data are:

- The Timeline: This is the type of the timeline, like employee, cell_phone, or extruder.
- The Context ID: This is an integer which specifies which of the type of things specified by the timeline we are referring to with this entry on the timeline
- The Timestamp: The time (in nanoseconds since Jan 1, 1970) that this entry on the timeline occurred
- The Instance Class: This is either object, reading, or text, meaning the data for this entry is an integer, floating point number or string, respectively
- The Instance Type: This is the type of the entry. It specifies what this entry is, i.e. temperature_sensor, salary, etc.
- The Instance Value: The integer, floating point number or string that is the data for this entry

Data can be added by transforming the data to the above fields and executing a RESTful transaction, or by using the sim_interact tool provided with the system.

Agents

The real utility of the Simularity AI is realized when *Agents* are run on top of a segment. Agents use the segment's RESTful API to perform queries which allow for predictive and other analytic operations to be performed on the data in the segment, as that data is updated in real time. Several agents

are supplied with the Simularity AI, and more are being created by Simularity, however, it is easy to construct new agents, which use the Simularity RESTful API to interact with the segment (and with other agents). Agents need not run on the same node as the segment, as all communication is via RESTful interfaces,

An agent typically implements a particular machine learning algorithm, or model. For instance, the Predictive Archetypes Agent uses the temporal query mechanism of the segment to create a compound Hidden Markov Model predictor. Neural networks can be implemented easily as the Query Language easily allows expression of training sets. The Event Signatures Agent uses a Neuroevolutionary Neural Network model to predict events based on the history of a set of sensor time series. The Anomaly Detector Agent uses a similar Neuroevolutionary Model to detect anomalies in a set of sensor time series. Many other models are possible and can be written in any programming language that supports RESTful interaction and JSON.

To discover how the Simularity AI can help your enterprise, contact us today. Partner enquiries welcome.