CIM in a Triple-Store Database for Distribution Power Flow

The U.S. Department of Energy has funded the development of an open-source platform to support the development of advanced distribution planning and operation applications, called GridAPPS-D. Our goal is to reduce the time and cost of developing, integrating and deploying data-driven, system-independent advanced applications for electric power distribution. These applications may be distributed and multi-layered. The sample applications under development by universities and national labs are of low technology readiness level (TRL), and not included in commercial products. Standards like the CIM are a key part of GridAPPS-D; new applications written to use the CIM should be easier to integrate with a variety of other, more complete and robust operational platforms. GridAPPS-D platform code, CIM interfaces and sample applications are all open-source, under a Berkeley-style license. The first release cycle (RC1) ended in April 2017, with two more planned to end in October 2017 and April 2018.

We used approximately 10% of the CIM classes in RC1, to support unbalanced distribution power flow solutions in two open-source distribution simulators, PNNL's GridLAB-D and EPRI's OpenDSS. In RC1, we used the MySQL relational database, but it was difficult to design SQL queries to extract power flow models from CIM. The CIM is heavily polymorphic. For example, there are four ways to define the impedance of an ACLineSegment, and two of those associate a PerLengthImpedance, which might actually be PerLengthPhaseImpedance or PerLengthSequenceImpedance. Polymorphism can be represented in SQL using join tables or possibly other ways, but this complicates the learning curve for new CIM developers, which we had thought to mitigate by implementing a custom query builder. One of our secondary objectives in GridAPPS-D, as the means to an end, is to encourage more widespread use of the distribution CIM. As a temporary measure, in RC1 we simply exported CIM XML back out from the MySQL database, and converted to power flow models using legacy Jena toolkit code. Concurrently, we evaluated other database technologies as described in a companion presentation, and chose a triple-store implementation for RC2.

The RC2 triple-store implementation has been completed, using the open-source Blazegraph database server with SPARQL queries, along with updated Java classes and Jena code. The processing performance has been better than in RC1. More importantly, the SPARQL query development time has been drastically reduced. We were able to develop a complete set of SPARQL queries in less time than it took to prepare just a few sample SQL queries for the database technology evaluation in RC1. SPARQL syntax has some similarities to SQL, but the main body of a WHERE clause comprises a set of triples (subject-predicate-object or node-edge-node) that are simply "stacked up" to specify and filter results. The UML generalizations, attributes and associations translate directly to SPARQL triples. In practice, this means a developer can design SPARQL queries directly from the UML, without the complications of SQL JOIN. There are no auxiliary tables or relationships not shown on the UML, and no need for a custom query builder. Based on this experience, we recommend that others consider a triple-store database for their CIM work.