iPlant Semantic Web Platform uses SSWAP (Simple Semantic Web Architecture and Protocol) to enable Semantic Pipelines across Distributed Web and High Performance Computing Resources

Damian D. G. Gessler[1], Blazej Bulka[2], Evren Sirin[2], Yan Kang[2,*], Pavel Klinov[2,**], Hans Vasquez-Gross[4], John Yu[4], Jill Wegrzyn[4]

[1]The iPlant Collaborative, University of Arizona, Tucson, AZ, U.S.A.
dgessler@iplantcollaborative.org
[2]Clark and Parsia, Washington, D.C., U.S.A.
{blazej,evren}@clarkparsia.com
[4]University of California, Davis, CA, U.S.A.
{havasquezgross,jjsyu,jlwegrzyn}@ucdavis.edu

Abstract. The iPlant Semantic Web Platform uses transaction-time reasoning and just-in-time ontologies to allow users to drag-*n*-drop independent, distributed semantic web services into a semantic pipeline. The platform uses first-order, description logic reasoning to examine input data types, output data types, and service descriptions to discover services. At each step the user is presented with only those services which operate on the data at that stage in a pipeline. The platform coordinates third-party service invocations, including transaction-time reasoning at third-party web sites. It includes a Just-In-Time ontology editor to allow anyone to host ontologies on our servers; ontologies may also be hosted anywhere on the web. The platform enables TreeGenes, a large biological information resource, to offer single-click marshalling of data and service discovery to semantic pipelines engaging enterprise-class resources at the University of Arizona and high performance computing (HPC) resources at the Texas Advanced Computing Center. The HPC results are sent to other distributed semantic web services, such as phylogenetic tree visualization and University of California CartograTree geographical mapping resource.

Keywords: Semantic Web Services, SSWAP, iPlant, TreeGenes, DiversiTree, CartograTree, OWL

[*] Current address: University of Maryland, Baltimore, MD, U.S.A. kangyan1@umbc.edu
[**] Current address: University of Ulm, Ulm, Germany. pavel.klinov@uni-ulm.de

1 The Platform

Architecture The iPlant Semantic Web Platform is a web architecture of four distributed actors: *i*) providers of services; *ii*) consumers of services; *iii*) ontology severs; and *iv*) a semantic Discovery Server (pipeline-maker and match-maker). Data—be it unstructured, semi-structured, or structured (*e.g.*, as in relational database stores)—enters the system via a service interface layer; *i.e.*, the platform does not operate on raw data *per se*, but via service interfaces, the invocations of which yield access to, and transformations of, data. This service interface layer is key to enabling distributed data to be integrated "rationally" under a first-order description logic protocol.

SSWAP (Simple Semantic Web Architecture and Protocol) SSWAP is a 100% W3C OWL DL-compliant light-weight protocol of five classes and 12 properties [1]. It allows services to describe what they are, the types of data they consume, and the types of data they produce. The protocol's ontology in its entirety is at [2]. The five classes correspond to the: *i*) service Provider, *ii*) actual service itself, called a Resource, *iii*) a data structure head-node called a Graph, *iv*) input data (a Subject), and *v*) output data (an Object). SSWAP is the service analog of the RDF construct of mapping a subject to an object (via a service). Subject and Object instances may be URIs, thereby allowing for indirection and non-serialization of data, or they may be head-nodes of arbitrary OWL sub-graphs, with properties and serialized data. Instances of Resource, Subject, and Object may be annotated with user-defined ontologies and thus are "unlimited" in domain scope; the protocol simply defines the scaffold. Services may have multiple Subjects mapping to multiple Objects. A protocol description of a service is called an RDG (Resource Description Graph). A HTTP GET on the Resource URL of the RDG returns the RDG in W3C-compliant RDF/XML. An RDG with input data creates an RIG (Resource Invocation Graph). A HTTP POST of the RIG or a GET with ontology term=value assignments in the query string invokes the service. An RIG with output data is called an RRG (Resource Response Graph). Thus SSWAP creates an ecosystem of protocol graphs, all sharing a canonical model, with a common syntax (W3C RDF/XML), under a common services' semantic (SSWAP), amendable to customization by user semantics (adding ontology terms to the Resource, Subject, and Object). SSWAP is a wrapper technology, so it can semantically enable legacy and non-semantic services. Notably, a SSWAP service *description* yields the service amenable to semantic *discovery*, *invocation*, and *response*.

Semantic Querying A service's protocol description encapsulates the information needed for its discovery and invocation. Thus one can consider any putative RDG as a query graph (called an RQG: Resource Query Graph) into a knowledge base of all RDGs. For semantic querying, we find all services for

which the RQG's: *i*) Resource is a subclass, and *ii*) Subject is a super-class, and *iii*) Object is a subclass, of any service in the knowledge base. Subsumption reasoning covers arbitrary complex, inferred, anonymous classes. The resultant services, and only these services, are guaranteed to be of the type of service queried (or more specialized), to operate on the input data (or generalizations of it), and return data of the requested output data type (or specializations of it). This allows us to use a reasoner for match-making based on the output of one service being logically sufficient for the input of another.

Constructing semantic pipelines Pipelines are built on-demand by using transaction-time reasoning to aid the user in building a workflow of services.

Start with a lexical search Users at `http://sswap.info` may search for services using keywords. Upon selecting a service and adding it to a new pipeline via web-based drag-*n*-drop, we present the user with all downstream services that can operate on the upstream service via semantic querying as described above. In this manner the user can build a pipeline of services. For each service, we reason over the service's Resource to determine its necessary and sufficient conditions, and based on this construct on-demand a custom user dialog that allows the user to enter the service's required and optional parameters, if any. In a similar manner the Subject is examined, and the user may upload data to be ontologically tagged via the RDG.

Start with data launched from a web site We provide a Javascript snippet that allows any web master to add a "sswap.info" button to their web pages. We call this Web Discovery. We provide a service to allow the web master to package or reference the data using JSON (see `/api`)[1]. Upon pressing the button, the JSON is sent to our Discovery Server, where we translate it into an RDF/XML RQG, perform semantic querying, and present the user with a new pipeline preloaded with their data and the semantic results of all candidate downstream services.

Start with the results from previous pipelines Because the last service in a pipeline returns a standard RRG, this can be used to start a new pipeline. In this manner, a pipeline can be extended from other pipelines. Data is private, but pipelines can be publicly published, so when someone makes a pipeline, it can be made available for anyone to use. In this manner, we grow a database of user-built combinations of web distributed services; this has deep social networking value. We note that public exposure of pipelines does not imply unregulated execution: any service is free to gate-keep resources with logins, HTTPS, etc.

Pipeline invocation is orchestrated, but execution is distributed RDGs represent published SSWAP services that are offered by third-parties anywhere on the web. When the user initiates a pipeline, we coordinate the invocation and callback of services, but do not ourselves execute the services: the services run independently, asynchronously on their host machines. Downstream services

[1] Relative URLs are RESTful endpoints on `http://sswap.info/`

retrieve the upstream RRG and convert it to an RIG without passing through our servers, so we are not privy to non-serialized data being transferred between services, maintaining an important privacy safe-guard.

Transaction-time reasoning SSWAP graphs (RDGs, RIGs, RRGs, and RQGs) are small documents of a few dozen lines of OWL DL RDF/XML that expand to a few thousand triples after first-order reasoning. We use reasoning in four places: *i*) when Providers publish their RDGs with us, we resolved ontology terms by dereferencing them on the web; we then infer over the closure RDG and store the resulting inferred graph in a triple-store [3]. We use a combination of transaction-time reasoning at publication time and offline processing to maintain the knowledge base; *ii*) when users initiate Web Discovery from a web site by sending us an RRG, we similarly resolve the RRG, convert it to a RQG, and execute transaction-time semantic querying; *iii*) when users build pipelines we reason during the transaction process to satisfy semantic querying and other pipeline duties; *iv*) when third-party services receive an RIG they need to process the request and return a RRG that complies with the logical contract of their RDG. We provide a kit (`/sdk`) that allows third-parties to run their own servlet reasoner to handle transaction-time reasoning to process requests.

Pipeline management Control is architected as three separate components: *i*) we use Vaadin [4] to offer a RIA (Rich Internet Application) enabling an intuitive, drag-*n*-drop user experience; *ii*) communication to the backend is performed by a 100% RESTful JSON API, making heavy use of idempotent HTTP GETs and PUTs. This means that a user may start building a pipeline, bookmark it, close their browser, and open it anywhere, anytime, and continue their work. It means that users may begin long-running pipelines, and return at their convenience with a desktop or mobile tablet; *iii*) the pipeline manager communicates with the Discovery Server via a RESTful API.

Platform APIs We wrote ~135,000 lines of open-source Java to build a platform and a Java API (`/javadocs`). We use this API internally, and package it as part of our SDK (Software Development Kit) so anyone may write their own SSWAP services (`/sdk`). Many developers are fluent in JSON, but not OWL RDF/XML, so we wrote a RESTful translator that allows SSWAP graphs and user ontologies to be written in JSON and then translated to OWL RDF/XML (`/api`; see also `/make` and [5]). We expose Discovery Server engagements as RESTful endpoints (`/wiki/api`).

2 Ontologies

A challenge for semantic web services is how to enable and incorporate distributed ontologies. We enable the use of arbitrary user-defined OWL

ontologies to allow service providers to describe their data, and to allow clients to query and engage services.

Just-In-Time ontologies We used Smart GWT [6] to write an application that allows anyone to host their ontologies on our servers [5]. Users register for a free iPlant Collaborative account and may create and administer new ontologies (called "namespaces"). Users build ontologies term-by-term using the JSON syntax, translate them to RDF/XML, and publish them on-demand. Terms are separately dereferenceable and immediately available to anyone on the web.

Support for "small" legacy ontologies When we read a SSWAP graph, we extract ontology terms and dereference them to retrieve their OWL statements. If these documents themselves contain terms, we dereference those, and continue this cascade until closure is achieved, subject to byte and time limits.

Support for "large" legacy ontologies: module extraction with BioPortal BioPortal [7] is a major repository funded by the National Center for Biomedical Ontology. It contains over 320 ontologies, and over 180 OWL ontologies. We use the method of [8,9] to process each OWL ontology offline to generate "atoms," such that at transaction-time we can compute the subset of statements (called a "module") that are necessary and sufficient for complete entailment over any subset of terms. Importantly, for large ontologies the module is often much smaller than the ontology itself [8], thus lending it as a key approach to bringing large, legacy ontologies to transaction-time applications in the semantic web. This research won the 2011 ISWC Best Student paper; it is being rolled into production, and is available as a service at `/modularize`.

3 Integrating Enterprise and HPC into the Semantic Web

Enterprise resources TreeGenes [10] is a large biological resource serving over 2500 forest geneticists from over 800 organizations. It contains data from 15 yrs on over 1200 species, including genomic, phenotypic, and other data. We wrote 11 SSWAP services to expose slices of this data and added SSWAP Web Discovery to TreeGenes' DiversiTree [11]. This allows scientists to select data and launch directly into iPlant's Semantic Pipeline with a single button press. For geographically-oriented tree scientists, we wrote a mapping tool called CartograTree [12]. We enabled CartograTree with SSWAP Web Discovery so that scientists can launch directly into semantic discovery.

High Performance Computing The iPlant Collaborative is a cyberinfrastructure for the plant sciences. iPlant serves over 7500 scientists with enterprise-class and high performance computing resources, petabyte-scale storage, and other resources. We wrote semantic pipeline support to engage the HPC resources and used SSWAP to semantically wrap 10 resources in the domain of multiple sequence alignment and phylogenetic tree reconstruction.

Putting it all together: a real-world demonstration A worked example is at `/wiki/example`. Visit [11], select 'Contig' on the left and enter "copper ion binding" for 'GO Term.' Click Display. Select the top 15+/- records and press *sswap.info*. A JSON RRG with the data will be sent to `sswap.info`; an on-demand semantic query will ensue and a pipeline will be initiated. Add 'Treegenes' Tree Sample Service' to the pipeline and press Play. When the pipeline has run (~30 secs), output data is available for rendering. The RDG declares CartograTree as a viewer, so clicking Display Data makes a RESTful call to CartograTree with the data. Based on the geographical locations, select a subset of the data (*e.g.*, the two trees in California and Mexico). Select both records from the data table, choose Common Amplicon in the combobox and Proceed. Select an amplicon (*e.g.*, copper ion binding amplicon #406 *UMN_CL18Contig1_08*), Submit, and then send to *sswap.info*. In the new pipeline, register or login to gain access to HPC services. Drag in 'Treegenes' MultiFasta Service' to retrieve sequences based on the CartograTree data, then ClustalW2 to align sequences, FastTree to build a tree, and TreeViz to visualize. Play. Depending on the HPC queue, execute time may be minutes to hours. Data will travel from UC Davis, to TACC (HPC Lonestar), to TACC (HPC Ranger), to U. Arizona. Display phylogenetic tree as output from VizTree. At any time, bookmark the pipeline and close your browser, or search for pipelines by logging-in and entering "user:*login*" as the search phrase. Exact sequence of steps is subject to change; see `/wiki/example`.

Acknowledgements This work was supported by NSF grants for the iPlant Collaborative (#DBI-0735191) and SSWAP (#0943879).

4 References

1. Gessler DDG, Schiltz GS, May GD, Avraham S, Town CD, Grant D, Nelson RT. SSWAP: A Simple Semantic Web Architecture and Protocol for semantic web services. BMC Bioinformatics, 10:309 2009. doi:10.1186/1471-2105-10-309.
2. `http://sswapmeet.sswap.info/sswap`
3. Pellet + Stardog; see `http://stardog.com`
4. `http://vaadin.com`
5. `http://sswapmeet.sswap.info`
6. `http://smartclient.com`
7. `http://bioportal.bioontology.org`
8. Del Vescovo, C., Gessler DDG, Klinov P, Parsia B, Sattler U, Schneider, T, and Winget A., "Decomposition and Modular Structure of BioPortal Ontologies," In: ISWC, LNCS, 7031: 130-145. 2011.
9. Klinov, P, C Del Vescovo, T Schneider Incrementally updateable and persistent decomposition of OWL ontologies. In: Proceedings of OWL: Experiences and

Directions Workshop 2012. P. Klinov and M Horridge (eds). Heraklion, Crete, Greece, May 27-28, 2012. CEUR Workshop Proceedings 849 CEUR-WS.org 2012
10. http://dendrome.ucdavis.edu
11. http://dendrome.ucdavis.edu/DiversiTree
12. http://dendrome.ucdavis.edu/cartogratree

5 Appendix[2]

Minimal requirements

1. *The application has to be an end-user application ...* sswap.info is an end-user application available to anyone, without registration. For access to HPC services, register for a free iPlant account.
2. *The information sources used:*
   - *should be under diverse ownership or control* Both data and services are under the control of independent, non-affiliated entities: U. Arizona, U. Texas, U. of California, as well as third-party ontologies.
   - *should be heterogeneous (syntactically, structurally, and semantically)* Raw data is syntactically heterogeneous (JSON, RDF/XML etc.); structurally heterogeneous (*e.g.*, it originates from, and is marshaled through, various servers and serializations); semantically heterogonous (utilizes numerous independent ontologies at [5] and elsewhere).
   - *should contain substantial quantities of real world data (i.e. not toy examples)* The data is real-world (TreeGenes 900,000 sequences and 24 million genotypes) and is acted upon by real-world services.
3. *The meaning of data has to play a central role.*
   - *Meaning must be represented using Semantic Web technologies.* 'Meaning' is represented using OWL 2 (DL) user-defined ontologies and SSWAP, a 100% OWL-compliant protocol.
   - *Data must be manipulated/processed in interesting ways to derive useful information* Data flow travels from lexical IDs, to DNA sequences, to phylogenetic trees, to geographical mapping and tree visualization.
   - *This semantic information processing has to play a central role in achieving things that alternative technologies cannot do as well, or at all;* We use transaction-time reasoning, just-in-time ontologies, and semantically-enabled distributed integration to achieve an information flow unparalleled by alternative technologies.

[2] Per http://challenge.semanticweb.org/2012/criteria.html

Additional Desirable Features

- *The application provides an attractive and functional Web interface ...* `sswap.info` uses state-of-the art RIA (Rich Internet Application) UI.
- *The application should be scalable ...* The application is web-scalable in both data and services: for data, the application does not require data to be serialized: arbitrary amounts of data in any format may be referenced by URIs, so there is no data limit *per se*. For services, protocol graphs are small and amenable to reasoning; we process dozens of services, thousands of pipelines, and in excess of 10^5 triples. Metrics may scale much larger (*e.g.*, thousands of services; $>10^6$ triples) by load-balancing across servers.
- *Rigorous evaluations have taken place ...* The platform has been rigorously tested with structured release testing; has been publicly beta tested for over 1 yr; TreeGene application has being submitted for scientific peer-review.
- *Novelty, in applying semantic technology to a domain or task that have not been considered before* Web-based, on-demand semantic pipelines at this scale, in this domain, and as a robust production platform as presented here do no exist elsewhere: this application is novel.
- *Functionality is different from or goes beyond pure information retrieval* Our pipelines achieve distributed, on-demand, reasoner-assisted data and service integration; information retrieval is but the first step.
- *The application has clear commercial potential and/or large existing user base* The immediate user base is scientists of The iPlant Collaborative and TreeGenes (approx. 10000 users). Commercial potential exists in the platform's agnostic technology: domain specifics are encapsulated in user ontologies and services, not in the platform itself.
- *Contextual information is used for ratings or rankings* Search result service prioritization is based on Subject and Object subsumption matching: services that can connect to a larger number of other services rank higher.
- *Multimedia documents are used in some way* Search on "image" for examples of services that process PNG, JPEG etc. as input data output data.
- *There is a use of dynamic data (e.g. workflows) ...* Application is highly dynamic, heavily using transaction-time reasoning and just-in-time ontologies for real-time construction of semantic workflows.
- *The results should be as accurate as possible (e.g. use a ranking of results according to context)* Accuracy is verifiable with over 2000 open-source unit tests on thousands of combinations of semantic queries.
- *There is support for multiple languages and accessibility on a range of devices* The platform supports any UTF-8 encoded ontologies and services in multiple languages that may reside on a diverse range of hardware and operating systems. Current ontologies and services are in English. The end-user web front-end is WebKit mobile compatible (*e.g.*, tablets; results vary).