Applications of the YarcData Urika in Drug Discovery and Healthcare

Robert Henschel
Indiana University
Research Technologies
Bloomington, Indiana, USA
henschel@iu.edu

Abhik Seal
Indiana University
School of Informatics and Computing
Bloomington, Indiana, USA
abseal@indiana.edu

Jeremy J. Yang
Indiana University
School of Informatics and Computing
Bloomington, Indiana, USA
jejyang@indiana.edu

David J. Wild
Indiana University
School of Informatics and Computing
Bloomington, Indiana, USA
djwild@indiana.edu

Ying Ding
Indiana University
School of Informatics and Computing
Bloomington, Indiana, USA
dingying@indiana.edu

Abhinav Thota
Indiana University
Research Technologies
Bloomington, Indiana, USA
athota@iu.edu

Scott Michael
Indiana University
Research Technologies
Bloomington, Indiana, USA
scamicha@iu.edu

Matt Gianni
Cray, Inc.
Life Sciences
Pleasanton, CA, USA
mattg@cray.com

Jim Maltby
YarcData LLC
Pleasanton, CA, USA
jmaltby@yarcdata.com

***Abstract*—The Cheminformatics & Chemogenomics Research Group (CCRG) at Indiana University has been working on algorithms and tools for large scale data mining of drug discovery, chemical and biological data using semantic technologies. The work includes finding new gene targets for drugs, identifying drug/drug interactions and pinpointing the cause for drug side effects. CCRG uses sematic web technologies like Resource Description Framework (RDF), triple stores, and the SPARQL Protocol and RDF Query Language (SPARQL). The YarcData Urika appliance promises to radically speed up this specific type of research by implementing a SPARQL endpoint on specialized hardware. In this paper, we are describing our first steps in testing if a Urika system could be integrated into our workflows.**

***Keywords*-sematic web; RDF, triple store; SPARQL;**

I. Introduction

Semantic Web technologies are becoming important when analysing large and complex data sets as it becomes more and more common that the analysis is no longer limited to one data set but relies on connecting multiple data sets. As once siloed data sets are becoming available for analysis, it is possible to derive better insights from connecting a larger number of data sets. Traditional relational database systems are very good at analysing large datasets, as long as the schema is well designed and the data conforms to this schema. However, relational databases are not well suited for connecting different datasets that were organized for different purposes using different schemas.

In contrast, this is where Semantic Web technology excels. Triple stores are used in scenarios where the schema is less well defined and particularly where there is a wide range of different types of entities that need to be stored. Triple stores also support an evolving set of relationships between entities. In the flexible, schema-free environment that a triple store provides, dealing with this kind of data is much easier. Triple stores are well suited to integrating data from a wide range of sources without having to force that data into a single normalized schema.

The principal technologies of the Semantic Web fit into a set of layered components. The current components are the Resource Description Framework (RDF) Core Model, the RDF Schema language, the Web Ontology Language (OWL) and the SPARQL query language for RDF.

The RDF model[1] based upon the idea of describing statements about resources in a triple. A triple in RDF terminology is an association of the form (subject, predicate, object), loosely describing a relation between the subject and the object. The subject of an RDF statement is a resource identified by a Uniform Resource Identifier (URI)[2]. The predicate is denoting a specific property of the subject. The object, which can be a resource or a string literal, represents the value of this property. For example, the fact that “The drug Atorvastatin is given for type II diabetes“ can be represented in an RDF triple by denoting the subject as “Atorvastatin“ with a predicate “is given“ and the object “type II diabetes“.

RDF Schema (RDFS)[3] and the Web Ontology Language (OWL)[4] are used to explicitly represent the meanings of the resources and how they are related. When connecting data sources that conform to OWL, the meaning of relations is predefined and uniform in all data sources. It is no longer

necessary to adapt and map the various schemas to one another.

SPARQL[5] is a query language for RDF. A SPARQL query is represented by a graph pattern to match against the RDF graph. Graph patterns contain triple patterns, which are like RDF triples, but with the option of query variables in place of RDF terms in the subject, predicate or object positions. For example, the query composed of the triple pattern ("Atorvastatin", "is given", ?disease) matches the triple described above and returns "type II diabetes" in the variable disease.

On the YarcData web pages[6], the Urika is described as an enterprise-ready graph appliance, that once fed with data, can generate insights fast. This suggests convenient commodity-computing. The vision is implemented by having the Urika comply to all of the above outlined standards. By exposing a standard SPARQL endpoint, it is straight forward to include the appliance in workflows that rely on such technology.

As we looked for past experiences with the performance of Urika, we found only limited published results. [7] has a nice comparison of different graph databases, but very little detail on them and the performance comparison in this paper does not include the Urika system. [8] is a Urika specific paper and compares the Urika on two specific problems to a SQL database and another SPARQL endpoint. Unfortunately, no hardware information is provided for those two systems making it difficult to compare them. [9] is an in-depth paper that discusses performance of a Cray XMT system. The XMT was renamed to Urika after the paper was published, so the results continue to be relevant. The paper focuses on the technical details and the performance of implementing a triple store on the XMT. The YarcData website[6] lists a number of use cases for Urika, but the whitepapers are more focused on outlining what is possible with a Urika, as opposed to performance details.

The Cheminformatics and Chemogenomics Research Group (CCRG) [10] in the School of Informatics and Computing at Indiana University, Bloomington is a world-leader in the development of large-scale semantic networks of data for drug discovery and biomedicine[11][12][13]. Major contributions include: Chem2Bio2RDF[14], the first large-scale semantic network of drug discovery data, and a wide range of network prediction, association finding and modeling tools that work on this data. Of particular note are the association search tool[15] that integrates pathfinding and literature-based association finding algorithms to identify literature supported association paths between drugs, genes, diseases, side effects and biological pathways, and the highly novel and validated Semantic Link Association Prediction (SLAP) approach[16] which makes drug-gene association predictions based on statistical analysis of the subnetworks which exist between them.

This paper describes our first steps of evaluating if a Urika system could be used by researchers at Indiana University, it is not intended as a thorough performance comparison. The paper describes out experiences when moving a popular dataset from the Virtuoso triple store onto a Urika system and initial performance numbers. The paper is organized in the following way. Section II outlines an interesting use case for combining data from multiple data sources, motivating our research into this topic. Section III briefly describes the hardware that we have used in this paper. Section IV outlines the methodology of our tests and section V describes the results of our testing. Section VI concludes the paper, outlining what we found and what we plan on working next.

II. Use Case

An example that can easily be solved using SPARQL queries is discovering drug-drug interactions and drug side effects. This is very important for patients taking more than one drug. For example, a patient with high blood pressure and a history of cardiovascular disease, which includes heart attacks and bypass surgery, may take 5 or more drugs at the same time. For this example, lets assume those drugs are: Fenofibrate (lowers LDL/VLDL, triglycerides, increases HDL), Levothyroxine (Synthetic thyroid hormone used to treat hypothyroidism), Valsartan/Hydrochlorothiazide (Angiotensin II antagonist for hypertension), Aspirin (as a blood thinner) and Rosuvastatin/Crestor (LDL lowering drug). Those drugs target certain proteins in the body and once those targets are identified, one can actually go a step further and also compute what other drugs effect the same target, but have different side effects. All the relevant data sources are available in RDF and can easily be loaded into a triple store.

```
PREFIX sider:
<http://chem2bio2rdf.org/sider/resource>
SELECT *
FROM <http://chem2bio2rdf.org/sider>
WHERE
{
 ?sider sider:drug_name ?drug_name .
 FILTER regex(?drug_name,
 "Fenofibrate|Aspirin|Rosuvastatin|
 Levothyroxine|Valsartan","i" ) .
 ?sider sider:side_effect ?side_effect .
}
```

Figure 1. SPARQL query to retrieve the side effects of 5 drugs.

The SPARQL query for retrieving the side effect of those five drugs is shown in figure 1. The query works on the *sider* database, first retrieving all drug names and filtering it down to the relevant drugs. Then, the side effects of those drugs are retrieved. This will yield a complete list of side effects for

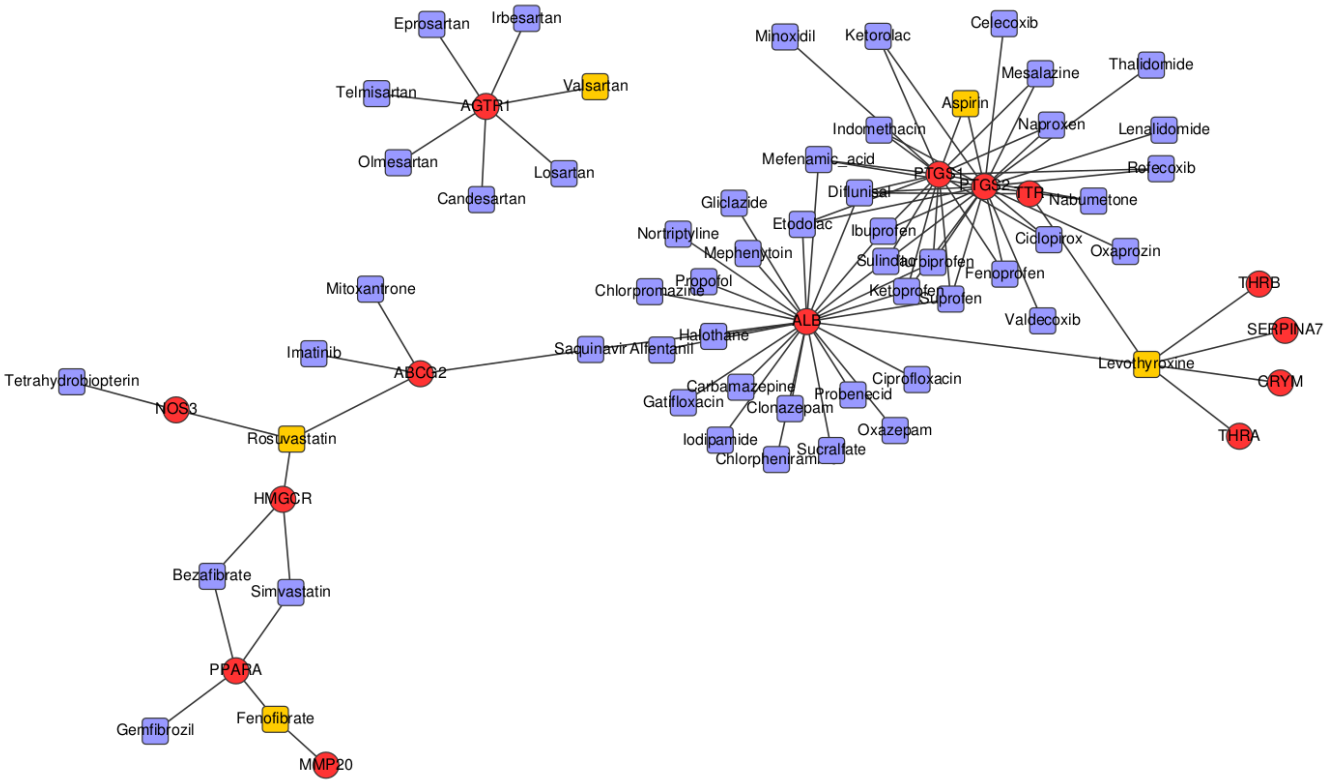

Figure 2. A graph showing the proteins that are targeted by the drugs Fenofibrate, Aspirin, Rosuvastatin, Levothyroxine and Valsartan. The proteins are colored in red circles, the dugs in yellow squares and the other drugs that target the same proteins in blue squares.

those drugs. In a similar manner, the actual protein targets of the drugs can be retrieved. With this information, other drugs can be found that target the same protein, but may have different side effects. Figure 2 shows the graph that was computed using this information. In the graph, the five original drugs are represented using yellow squares. The proteins that are targeted by those drugs are connected by an edge to the drug and are shown as red circles. Other drugs that target the same proteins are shown in blue squares.

III. Hardware and Software

For our testing we used a standard rack mounted server for running the Virtuoso triple store and a Urika-64 system for comparison. Details of the hardware and software configuration of both systems are described in the next two sections.

A. Urika

1) Hardware: YarcData offers the Urika in different sizes, labeled after the number of Threadstorm processors in the system. For our tests, we had access to two systems of the smallest configuration, Urika-64. The Urika appliance includes two distinct types of nodes, compute and service. Compute nodes run application programs. Each Urika compute node consists of two Threadstorm 4.0 application-specific integrated circuit (ASIC), DIMM memory, and a SeaStar2 chip. All compute nodes in a logical system use the same processor type. Service nodes handle support functions such as user login, I/O, and network management. Each service node contains an Opteron processor, DIMM memory, and a SeaStar2 chip. In addition, each service node may be configured with one or two PCIe Ethernet Network Interface Cards (NIC).

The system fits in a single cabinet, and consists of 64 compute nodes and a number of service nodes depending on local configuration. Four compute nodes fit on a single blade for a total of 16 compute blades. The compute blades are actually Cray XT-5 blades, with the AMD Opteron processors replaced with Threadstorm 4.0 ASICs. The blades are interconnected using the Cray SeaStar2 interconnect.

The Threadstorm 4.0 is custom-designed compute processor designed to provide uniform performance across a large shared memory architecture despite memory and network latency. The Threadstorm processor is organized into a series of streams, which the hardware uses to execute a single thread. Each Threadstorm chip provides 128 streams with 31 general purpose 64-bit registers, 8 target registers, and

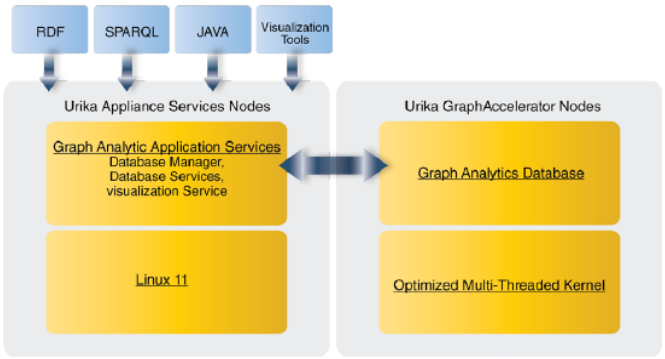

Figure 3. The Urika software stack[17].

a status word that includes the program counter as well a flag indicating the ready state of the individual stream. At each instruction cycle, Threadstorm switches contexts to the next stream that is indicating a ready state. Streams making memory requests are placed into a non-ready state until the memory references are fulfilled, allowing the Threadstorm processor to skip over them to continue processing steams in the ready pool. The large number of streams allows each processor to avoid stalls due to memory requests to a much larger extent than commodity microprocessors[9].

The Seastar-2 ASIC chip is the message processor for the Urika system. SeaStar offloads communications functions from the Threadstorm processor by integrating six high-speed serial links and a 3D router on each compute node. The network architecture includes a message passing interface, which provides a data path from an application to memory. Portions of the interface are implemented in SeaStar firmware, which transfers data directly to and from user memory without operating system intervention.

A Urika-64 system is usually equipped with a total of 2 TByte of globally shared main memory. The memory has uniform access across the system, in a data cache-free architecture. The system is booted using a single system image, allowing for an application to use all 2 TByte of memory and 64 processors with 8192 threads at once. Overall, the hardware configuration of a Urika system is very similar to a Cray XT5, with the exception of having Threadstorm compute processors instead of AMD Opterons.

2) Software: While the overall hardware architecture of a Urika is very similar to a Cray XT5, the system software is considerably different. Users can certainly write their own applications that run on the compute nodes of the system, since compilers and libraries are provided, however, much of the functionality of a Urika system is achieved by making use of the already provided RDF triple store and SPARQL endpoint. The system is marketed as an appliance, not as general purpose machine.

YarcData has implemented a proprietary W3C compliant tripe store for the Urika, optimized for the architecture of the shared memory. Figure 3 shows a schematic overview of the software architecture of the Urika.
The triple store can be loaded using either data that is available via HTTP or on a local Lustre file system. Making use of the Lustre file system will allow for a quick load of large data sources. Results of SPARQL queries can be retrieved using HTTP, or can be saved to the Lustre file system. Due to the integration of Lustre as a way to load and store data, a Urika system can be integrated relatively easily into a center that already has Lustre available for other HPC systems. GPFS and Panassas are also supported.

B. Standard Server

1) Hardware: The standard server that was used for comparison is a Dell PowerEdge R510. The server is equipped with two Intel Xeon X5550 quad core processors running at 2.67Ghz. Hyper-threading was enabled, turning the eight physical cores into 16 logical cores. The server was equipped with 26 GB of main memory.

2) Software: The server was run using Red Hat Enterprise Linux Server release 5.10. Virtuoso version 06.01.3127-pthreads was used, as available from http://virtuoso.openlinksw.com/.

IV. Methodology

Dataset	**Triple Count**
http://chem2bio2rdf.org/omim	17,251
http://chem2bio2rdf.org/kegg	245,997
http://chem2bio2rdf.org/reactome	15,849
http://chem2bio2rdf.org/ctd	4,933,479
http://chem2bio2rdf.org/chebi	5,812,141
http://chem2bio2rdf.org/dcdb	20,780
http://chem2bio2rdf.org/bindingdb	1,191,201
http://chem2bio2rdf.org/hprd	477,697
http://chem2bio2rdf.org/hgnc	1,720,541
http://chem2bio2rdf.org/medline	480,716,135
http://chem2bio2rdf.org/kidb	744,738
http://chem2bio2rdf.org/pubchem	15,439,873
http://chem2bio2rdf.org/qsar	32,206
http://chem2bio2rdf.org/bindingmoad	252,938
http://chem2bio2rdf.org/matador	269,656
http://chem2bio2rdf.org/pharmgkb	512,361
http://chem2bio2rdf.org/pdb	95,925
http://chem2bio2rdf.org/ttd	116,767
http://chem2bio2rdf.org/chembl	85,156,878
http://chem2bio2rdf.org/medline_lite	56,212,993
http://chem2bio2rdf.org/dip	1,113,871
http://chem2bio2rdf.org/sider	127,755
http://chem2bio2rdf.org/uniprot	1,994,607
http://chem2bio2rdf.org/drugbank	436,283
http://chem2bio2rdf.org/chemogenomics	7,327,361
Total	**664,985,283**

Table I
TRIPLE COUNT FOR THE CHEM2BIO2RDF DATA SETS.

Our testing focused mostly on functionality, and only to a small extent on performance. We were interested if we can migrate an existing workflow relying on a SPARQL endpoint

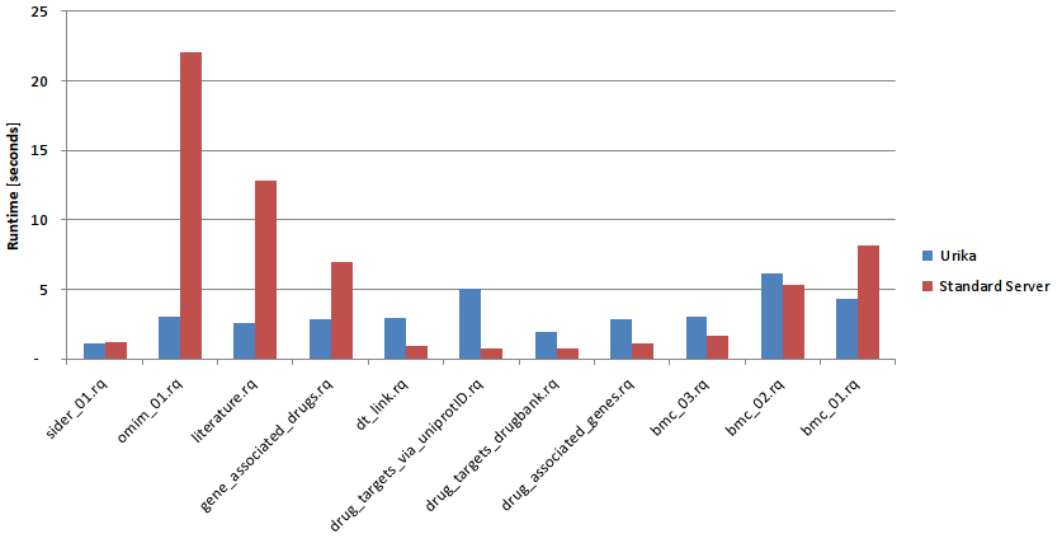

Figure 4. Runtime comparison of selected queries on the Urika and the standard server.

to the Urika. For this test, we imported the chem2bio2rdf data sources[18] into the Urika triple store. The triple count for the individual datasets is shown in table I.

During import, we found a few thousand triples that could not be imported on the Urika, due to spaces in the URI. We believe that those won't effect performance or functionality for our testing. Besides this, importing the data went without a problem. Data can be imported in two ways, using a command line utility or using the web-GUI. The data can be provided via the HTTP protocol or can be imported from the file system that is available on the Urika.

Once the data is imported, the Urika will combine the data into a "database". Multiple such "database" can exist on a Urika, but only one can be active. For our testing, we inserted all the data sources into one database.

Then we ran queries against those datasets. We picked a subset of the queries[19] that are published for the chem2bio2rdf datasets. On the Urika, queries can be submitted directly to the SPARQL endpoint from the command line or using a web-GUI. In addition, since the Urika SPARQL endpoint is build on the widely used Apache Jena[20] framework, applications that can connect to Jena should work out of the box. In the beginning we used the web-GUI to make sure the queries ran without an issue. The web-GUI will also format the output in a nice way, so that one can click on the results for further analysis. For detailed timing, we used a Java application that connected to the SPARQL endpoint directly.

V. Results

The first and most important result from our testing is that the Urika can be used for the kind of work that we are interested in. Due to relying on industry standards, existing triple store based workflows can easily be reconfigured to run on a Urika system. YarcData has created a solution that combines a custom and proprietary hardware platform with a standard software platform.

Figure 4 shows the runtime of the queries on the Urika and the standard server. Queries were run 5 times and the runtimes were averaged. No easy conclusion can be drawn from the chart. There are queries where the standard server is faster and there are queries where the Urika is faster. The queries we picked were certainly not designed to test the various features of SPARQL provides. Overall, it should be noted that the workload is very small, compared to what a Urika was designed for. The memory required for the dataset was nowhere near the 2TByte of main memory that were available on the Urika.

When loading a "database" on the Urika, quite a number of optimizations are performed to distribute the data across the compute nodes. Those optimizations are performed for all dataset sizes, consistent with the design goal of the Urika. When running a query the same methodology is applied. This means that even small queries are distributed across the compute nodes to exploit maximum parallelism. For small queries, the overhead cannot be amortized by the available parallelism in the query. In addition, the Urika has a certain irreducible overhead latency due to communications between the Opteron front end nodes and the Threadstorm based compute nodes. This is not a problem for the complex queries over large databases it was designed for, but it is noticeable on simple queries. The standard server we tested has a much smaller latency for simple queries.

VI. Conclusion

In this paper we outlined our initial testing of running chem2bio2rdf workflows on a Urika. We moved the required datasets to the system and ran a few queries to validate the results and get a better understanding of how to interact with the system. Since our tests were not defined to specifically stress a triple store, we are not drawing any performance conclusions. However, it seems that incorporating a Urika into our work is possible and we are looking forward to continue our testing with larger datasets.

Acknowledgment

The authors would like to thank Mario Vale from the Swiss National Supercomputing Center (CSCS) for his help in setting up a test environment on their Urika system. We would also like to thank YarcData for access to a Urika system for testing.

References

[1] RDF Primer. W3C Recommendation. http://www.w3.org/TR/rdfprimer (2004)

[2] Davies, J. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley and Sons, 2006.

[3] Dupre, J. *The Disorder of Things: Metaphysical Foundations of the Disunity of Science*. Harvard University Press. 1993.

[4] OWL 2 Web Ontology Language Document Overview. W3C. 2009-10-27.

[5] Rapoza, J. (2 May 2006). *SPARQL Will Make the Web Shine*. eWeek. Retrieved 2007-01-17.

[6] http://www.yarcdata.com/ Retrieved 2014-03-17.

[7] McColl, R. C., Ediger, D., Poovey, J., Campbell, D., Bader D.A.*A performance evaluation of open source graph databases*. In Proceedings of the first workshop on Parallel programming for analytics applications (PPAA '14). ACM, New York, NY, USA, 11-18. DOI=10.1145/2567634.2567638 http://doi.acm.org/10.1145/2567634.2567638

[8] Sukumar S.R., Bond N, *Mining Large Heterogeneous Graphs using Cray's Urika*. Computational Data Analytics Workshop - CDAW 2013. http://cda.ornl.gov/workshops/2013/cdaw/SukumarBondCDAW13.pdf

[9] Goodman E. L., Jimenez E., Mizell D., Al-Saffar S., Adolf B., Haglin D., *High-performance computing applied to semantic databases*. In The Semanic Web: Research and Applications, pages 3145. Springer, 2011.

[10] *Cheminformatics and Chemogenomics Research Group at Indiana University School of Informatics and Computing*. http://ccrg.soic.indiana.edu. Retrieved 2014-03-17.

[11] Wild, D.J. *Mining large heterogenous data sets in drug discovery*. Expert Opinion on Drug Discovery. 2009; 4(10), pp 995-1004

[12] Wild, D.J., Ding, Y., Sheth, A.P., Harland, L., Gifford, E.M., Lajiness, M.S. *Systems Chemical Biology and the Semantic Web: what they mean for the future of drug discovery research*, Drug Discovery Today, 2012, 17, 469-474.

[13] Dumontier, M., Wild, D.J. *Linked data in drug discovery*. IEEE Internet Computing, 2012, 16(6), 68- 71

[14] Chen, B., Dong. X., Jiao, D., Wang, H., Zhu, Q., Ding, Y., Wild, D.J. *Chem2Bio2RDF: a semantic framework for linking and data mining chemogenomic and systems chemical biology data*. BMC Bioinformatics 2010, 11, 255.

[15] He, B., Tang, J., Ding, Y., Wang, H., Sun, Y., Shin, J.H., Chen, B., Moorthy, G., Qiu, J., Desai, P., Wild, D.J., *Mining association paths in relational biomedical data PloS One*, 2011, 6(12), e27506.

[16] Chen, B., Ding, Y., Wild, D.J. Assessing Drug Target Association using Semantic Linked Data. PLoS Computational Biology, 2012, 8(7), e1002574.

[17] YarcData, *YarcData uRiKA Technical White Paper*

[18] *chem2bio2rdf - individual sources*. http://chem2bio2rdf.wikispaces.com/individual+sources. Retrieved 2014-03-17.

[19] *chem2bio2rdf - multiple sources*. http://chem2bio2rdf.wikispaces.com/multiple+sources. Retrieved 2014-03-17.

[20] *Apache Jena Home*. http://jena.apache.org/. Retrieved 2014-03-17.