Optimizing Semantic Data Transformation using High Performance Computing Techniques

José Antonio Bernabé-Díaz[1], María del Carmen Legaz-García[2], José M. García[1], and Jesualdo Tomás Fernández-Breis[1]

[1] Faculty of Informatics, University of Murcia, 30100 Murcia, Spain
{joseantonio.bernabe1, jfernand, jmgarcia}@um.es
[2] Biomedical Informatics and Bioinformatics Platform, IMIB-Arrixaca, Calle Luis Fontes Pagán, n 9, 30003 Murcia, Spain
mcarmen.legaz@ffis.es

Abstract. The growth of the Life Science Semantic Web is illustrated by the increasing number of resources available in the Linked Open Data Cloud. Our SWIT tool supports the generation of semantic repositories, and it has been successfully applied in the field of orthology resources, helping to achieve objectives of the Quest for Orthologs consortium. However, our experience with SWIT reveals that the time required for the generation of datasets is longer than desired. In this work we present the application of High Performance Computing techniques, mainly memory optimization and parallelization, to speed up SWIT.

Keywords: Semantic Web, Data transformation, High Performance Computing

1 Optimizing SWIT

The Semantic Web Integration Tool (SWIT) [1] transforms and integrates heterogeneous biomedical data for generating open semantic repositories by defining mapping rules between an input schema and a OWL ontology. The SWIT inputs are:

- Data instances: The tool is able to process XML and relational data.
- OWL Ontology: It supplies the domain knowledge for the transformation and the constraints applied for the generation of logically consistent data.
- Transformation rules: They define how the content of the input dataset is transformed into a semantic format: (1) mapping rules provide the links between the entities of the input data schema and the entities of the OWL ontology; and (2) identity rules prevent the creation of redundant entities by defining what makes an individual unique.

The application of SWIT to the transformation of large datasets has revealed that despite the computational complexity increases linearly with the number of entities to be transformed, it is slower than expected, and we have identified

some limitations to the performance: (1) use of an interpreted language; (2) inefficient memory management; (3) the execution of identity rules using SPARQL queries create an execution bottleneck; and (4) the sequential execution of the transformation. Hence, a code modernization process was carried out[3]:

1. The SWIT kernel has been re-implemented in C/C++.

2. We have optimized the way SWIT manages the ontology individuals during the transformation process. We use now two hash maps composed of pointers to individuals and not copies, so keeping the coherence and reducing memory consumption. One map grants that no failure happens when searching for an individual, if this exists, while the other map could miss in some searches since it acts as a greedy algorithm. The speed up of the execution of searches with the two maps is up to 2x.

3. The optimization of memory management also affects the process of identifying equivalent individuals using identity rules. Identity rules are defined using *AND* and *OR* conditions, and the new method uses one hash map of vectors for each type, where pointers to hashed individuals are stored. The hashing is carried out differently depending on the map. For *AND* conditions, a hash is performed along all the properties of the entity. Contrariwise, for *OR* conditions, the entity properties are hashed separately, having several pointers to the same individual along the hash map.

4. The SWIT parallelization is done by using *gnu parallel* tool[4]. The parallel design consists in setting one input file and one SWIT instance per core, so the parallelization only works when multiple files are established. When transforming a single large file, we need to split it in several smaller files to enable parallelization. It provides the largest speed-up.

2 Results

Our tests show a speed-up of 1000x, 4100x and 7800x in the *InParanoid* [2] datasets *E.coli*, *H.arabidopsidis* and *H.sapiens* respectively. The executions were tested on a high performance server that provides 2 chips of Intel® Xeon® E5-2698 v4 with 20 cores each (2 hyper-threading), making a total of 40 physical cores or 80 virtual cores, running at 2,2 GHz and 128 GB RAM DDR4.

References

1. Legaz-García, M.D.C., Miñarro-Giménez, J.A., Tortosa, M.M., Fernández-Breis, J.T.: Generation of open biomedical datasets through ontology-driven transformation and integration processes. J. Biomedical Semantics **7** (2016) 32
2. O'brien, K.P., Remm, M., Sonnhammer, E.L.: Inparanoid: a comprehensive database of eukaryotic orthologs. Nucleic acids research **33**(suppl_1) (2005) D476–D480

[3] https://software.intel.com/en-us/articles/what-is-code-modernization

[4] http://www.gnu.org/s/parallel