HTCondor backed WPS Service

Extended Abstract

Teodora Selea
West University of Timișoara
Blvd. Vasile Pârvan 4
Timișoara, România 300223
teodora.selea@e-uvt.ro

Marian Neagul
West University of Timișoara
Blvd. Vasile Pârvan 4
Timișoara, România 300223
marian.neagul@e-uvt.ro

Silviu Panica
West University of Timișoara
Blvd. Vasile Pârvan 4
Timișoara, România 300223
silviu.panica@e-uvt.ro

ABSTRACT

The area of GIS (Geographical Information System) witnessed in recent decades an fast growing number of geospatial processing and information retrieval algorithms and tools. This growth created the need for a standardized mechanism for controlling geospatial processing services, resulting in the Web Processing Sevices (WPS) standard. In this paper we propose a novel WPS implementation running on top of the HTCondor[2] workload manager.

KEYWORDS

WPS, Zoo, 52°North, HTCondor, batch scheduling, WPS server, WPS implementation, OGC

The Web Processing Service (WPS) is a standard for defining a common mechanism for describing and publishing GeoSpatial services. The WPS standard provides both the means for modelling a geospatial processing (specifying it's input and output arguments, their data types) and for managing the life-cycle of the various processing tools. By standardizing the modelling and execution mechanisms, the WPS standard effectively decouples the backend processing implementation from the WPS exposed interface, hiding the implementation details from the WPS clients. From the client perspective, the standardized execution lifecycle management allows the interoperability with multiple client implementations.

A WPS compliant implementation provides clients with a set of the operations aimed for managing the processing service. This operations are defined in WPS standard [6] and include operation like *GetCapabilities* for processing discovery or *Execute* for triggering execution

In [5] and [7] the authors have identified three WPS implementations: 52°North[1], Zoo-project [4] and PyWPS [3]. 52°North is a mature project, developed in Java that has a variety of already implemented processes. Zoo-project has its internal core written in C++, but its main advantage over 52°North is that it offers the end-user the possibility to develop its own processes in a variety of languages: C, Python, PHP, Java, C# and JavaScript. In addition to this, it has a mechanism to load and handle the requested processes dynamically. PyWPS is Python WPS server that was designed to offer an easy-to-use WPS platform: it is easy to install, maintain, to add new process and has a "native support for GRASS-GIS python scripting"[3]. Both 54°North and Zoo-project support WPS 2.0.0, while PyWPS is still compliant only with WPS 1.0.0.

All the afore mentioned WPS implementations concentrate on providing different ways for user to develop their own services and on the variety of supported libraries, however **none** of them tackle the **problem of job scheduling**. Providing a mechanism to manage the computational resources and a job scheduling policy means a quicker job result and the ability to schedule more jobs.

The solution we propose aims to expose WPS 2.0 services and manage the **effective execution of the backend processing tools** using a specialised workload management systems. Although we initially aim to provide support for the HTCondor[2] workload manager, we envision a pluggable interface allowing multiple workload management systems. HTCondor manages the available resources and schedules job execution in order to minimize the waiting time for a job's result. HTCondor is unique due to its ClassAd mechanism. A job is submitted for execution with a ClassAd containing its system and architecture requirements. Then, through a match-making process between the resources' ClassAds and the job's ClassAds, HTCondor find the most suitable solution for the execution of a job.

A simplified life-cycle for a processing session might be:

1. The server recives an WPS execution request using *Execute(Process,Data)*;
2. Based on the received request the WPS server queues a new batch job inside the workload manager;
3. The workload manager schedules the job, reserving the required resources;
4. At the completion of a job, its output data is sent back to the WPS server, which in turn expose the data back to the client.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

WOODSTOCK'97, July 1997, El Paso, Texas USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

REFERENCES

[1] 2017. 52°North WPS. http://52north.org/communities/geoprocessing/wps/. (2017). (Accessed on 04/26/2017).
[2] 2017. HTCondor - Home. https://research.cs.wisc.edu/htcondor/. (2017). (Accessed on 04/27/2017).
[3] 2017. PyWPS • Home. http://pywps.org/. (2017). (Accessed on 04/26/2017).
[4] Gérald Fenoy, Nicolas Bozon, and Venkatesh Raghavan. 2013. ZOO-Project: the open WPS platform. *Applied Geomatics* 5, 1 (2013), 19–24.
[5] J Garnett and G Fenoy. 2011. WPS Shootout. In *Tech Session presentation at FOSS4G Conference in Denver.*
[6] M Mueller and B Pross. 2015. OGC WPS 2.0 Interface Standard. *OpenGeospatial Consortium Inc, OGC* (2015), 14–065.
[7] V Rautenbach, S Coetzee, M Strzelecki, and A Iwaniak. 2012. Results of an evaluation of the orchestration capabilities of the ZOO project and the 52 North framework for an intelligent geoportal. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2012).