SCIENCE WORKFLOW MANAGEMENT SYSTEM FOR THE TERRESTRIAL OBSERVATION AND PREDICTION SYSTEM (TOPS)

Petr Votava[1,2], Robert Morris[1], Jennifer Dungan[1], Lina Khatib[1,3]

[1]NASA Ames Research Center, Moffett Field, CA 94035, USA

[2]University Corporation at Monterey Bay, Seaside, CA 93955, USA

[3]SGT, Inc., Greenbelt, MD 20770, USA

1. BACKGROUND

The Terrestrial Observation and Prediction system (TOPS)[1] is a data and modeling software system designed to seamlessly integrate data from satellite, aircraft, and ground sensors, and weather/climate models with application models, to expeditiously produce operational nowcasts and forecasts of ecological conditions. TOPS provides reliable data on current and predicted ecosystem conditions through automation of data retrieval, pre-processing, integration, and modeling steps, allowing TOPS data products to be used in an operational setting for a range of applications. As the number of TOPS users and applications grows, there is an increased need to manage not only the data, but also to provide an easy way to design and manage custom processing pipelines that integrate Earth science data with processing components and models in order to meet the operational requirements of users and customers. This paper describes a workflow management system that provides capabilities for definitions and execution of science workflows within an extensible data and process management framework accessible both through a java API as well as through web service invocation.

2. APPROACH

The science workflow management system consists of three main components – an XML-based goal specification language (GoalML), a workflow processing engine and a plug-in framework integrating data acquisition, storage and processing components. The GoalML language enables users to define high-level science campaigns consisting of a set of goals, where each goal is composed of a number of data acquisition and processing tasks. The goal specification also contains spatial and temporal constraints that are imposed on the data acquisitions within each goal, but can also be inherited throughout the goal hierarchy. These constraints are encoded using OGC GML[2] format. The goal specification expressed as a GoalML document serves as a blueprint for the workflow and it is the input to the workflow processing engine. These blueprints together with their run-time instantiations are stored in a database and can be later searched, loaded and re-executed reproducing the original results and providing means for data provenance. During execution, the GoamML document is first translated into a finite state machine (FSM)[3] using the Labeled Transition System Analyzer (LTSA)[4]. The finite state machine defines the order of tasks to be executed, including tasks that can be executed in parallel as well as synchronization states, such as the beginning of a process that requires all the data acquisitions to be completed prior to execution. The workflow engine then executes the FSM, coordinating the resources, local or remote, used to carry out data acquisition and processing, and monitoring of the execution of individual tasks as well as responding to execution failures. The individual tasks are performed by the data and process "manager" components – frameworks that manage a set of plugins implementing a number of data acquisition and processing protocols. Examples of these protocols are FTP, OGC SOS[5] or a custom database interface for data acquisition, and OGC WPS[6] or SensorML[7] for data processing. This architecture abstracts the interfaces to the data acquistion and processing components, so the workflow engine is executed against these unified interfaces and the actual implementation is handled by the protocol-specific plugins. This provides a flexible way to extend the system to a wide range of protocols ranging from simple database interface to arbitrarily complex protocols, even ones requiring complex authentication. The entire system is designed in a very modular fashion with minimal dependencies among the components and the architecture translates very well into the SOA[8] paradigm. We are providing both REST[9] and SOAP[8] style interfaces to the data and processing framework as well as to the workflow engine. Only the workflow engine depends on the GoalML language, but the data and process web services are independent and could be accessed through either a direct web services invocation or through another processing system, such as a BPEL[10] processing engine.

3. CONCLUSIONS

We have designed and implemented a science workflow management system that provides a mechanism for defining science campaign specifications, translating them into concrete workflows and executing them in the context of the Terrestrial Observation and Prediction System (TOPS). We have designed a goal specification language that facilitates science goal descriptions, a workflow processing engine that translates this language into a finite state machine for optimized execution, and a flexible plugin architecture that is capable of handling large number of complex protocols. We are integrating this system with the TOPS production to improve management of several dozens processing pipelines as well as over 100TB of Earth science datasets.

4. REFERENCES

[1] R.R. Nemani, R.R. et al., *Terrestrial Observation and Prediction System (TOPS): Developing ecological nowcasts and forecasts by integrating surface, satellite and climate data with simulation models*, Taylor & Francis Book Series, London, 2007.

[2] OpenGIS Geography Markup Language (GML) Encoding Standard, http://www.opengeospatial.org/standards/gml

[3] D. Martin, R. Sigal, E.J. Weyuker, *Computability, Complexity, and Languages and Logic: Fundamentals of Theoretical Computer Science* (2nd ed.), Academic Press, San Diego, 1994.

[4] J. Magee, and J. Kramer, *Concurrency: State Models and Java Programs, 2nd Edition*, Wiley, July 2006.

[5] OpenGIS Sensor Observation Service, http://www.opengeospatial.org/standards/sos

[6] OpenGIS Web Processing Service, http://www.opengeospatial.org/standards/wps

[7] OpenGIS Sensor Model Language, http://www.opengeospatial.org/standards/sensorml

[8] D. Chappell and T. Jewell, *Java Web Services*, O'Reilly, 2002.

[9] R.T. Fielding and Taylor, Richard N. "Principled Design of the Modern Web Architecture", ACM Transactions on Internet Technology (TOIT), New York, 2005.

[10] B. Margolis, *SOA for the Business Developer: Concepts, BPEL, and SCA*, McPress, 2007.