

PlanetLab Overlay: Experimenting with Sensing and Actuation Support for Situated Autonomic Computing Services

Wail M. Omar^{1,2}

w.omar@soharuni.edu.om

A. Taleb-Bendiab¹

a.talebbendiab@livjm.ac.uk

Yasir Karam¹

cmpykara@livjm.ac.uk

¹ School of Computing and Mathematical Sciences

Liverpool John Moores University

Byrom Street

Liverpool, L3 3AF, UK

² Faculty of Applied Sciences

Sohar University

Sohar, P.C. 311

Sultanate of Oman

Abstract:

Situated autonomic computing requires systems to possess and/or be able to access feedback and context information, from their environment, including instrumentation and sensor data. This paper presents the motivation and development details of a sensor and actuator framework, together with its associated description language, developed for widely decentralized software systems. The framework was originally developed with web services and grid computing in mind. Hence, the paper argues that the developed approach is applicable to widely decentralised grids. To portray our proposed approach, the paper uses an illustrative example taken from an experimental case-study developed using the PlanetLab overlay. The latter is an open community research testbed and overlay for Planetary-scale services. In addition, the paper will present an evaluation of the work, and then conclude with general remarks and an indication of further works.

Keywords: Sensor and Actuator Framework, PlanetLab, self-management, instrumentation, autonomic computing.

1. Introduction: Planetary-Scale Overlay

The last few years have witnessed the emergence of new types of network and application services driven by the needs of modern business, including dynamic and virtual enterprises models. Such services include networked storage services [12], peer-to-peer file sharing [13], content distribution network [14] and robust routing overlays [15] to name but a few. Hence, advocates of service-oriented architecture envisage future Internet through serviceware technology to play a vital role in facilitating global computing (planetary-scale). This will be via on-demand composition of a

range of service-oriented applications, including business services and end-users. On the other hand, serviceware can be characterised as an overlay abstraction, upon existing global infrastructures (grids), offering end-users programming, interaction and control models to develop, deploy and manage their required applications, for the modern virtual organisation business model.

However, this vision engenders a range of long standing technical challenges including; scalability, complexity, availability and manageability of such systems. To explore the self-management issues of planetary-scale systems, we are conducting a case-study using the PlanetLab overlay and services [1, 11]. For completeness of the paper, we provide a very brief overview of the PlanetLab project as its full introduction is beyond the scope of this paper.

PlanetLab is an open platform for developing, deploying and accessing planetary scale services [1]. PlanetLab is a testbed for Internet, WAN network and global computing research [1], which depends on location partitioning or zoning where each region has a central unit known as a proxy. The latter is responsible for all the management, control and monitoring processes of the different components in this region. Each of which represent computer hosts (nodes), slices, slivers, Virtual Servers (VServer), infrastructures, communications, applications and users [1, 16]. Hence, the application of autonomic principles, including self-management and administration, applied to such an overlay can provide interesting insight, evaluation and/or validation of autonomic computing using very-large scale systems.

Thus, the experiment presented in this paper focuses on how to use the sensor and actuator framework and associated services, to provide intelligent monitoring and control of systems, leading to autonomic self-management utilities for the PlanetLab overlay, which incorporates thousands of

deployed sensors and actuators within its environments [2].

The remainder of the paper is structured as follows: Section 2 outlines related works followed by a software sensor and actuator overlay. Section 4 presents the structure of the Sensor and Actuation Description Language (SADL) followed by an illustrative example of using it. Model for Monitoring Sessions Description Language (MSDL) is demonstrated in Section 5. Finally, the paper concludes with a general summary.

2. Background

Much work focusing on systems monitoring, including tools and services for PlanetLab, can be found in the public domain [1]. Different types of sensors services and API such as; CoMon [7], Ganglia [9], iPerf [19] and IrisLog [20] are available for this environment. These are used to provide instrumentation data such as; processor, memory, bandwidth and other networking services.

Other PlanetLab tools and services such as, Node List [10], Trumpet [18], and SWORD [21] are used to provide availability and status information of available nodes (hosts), which are registered with the PlanetLab environment. Yan *et al.* [17] presented an algebraic approach for adaptive scalable overlay network monitoring. Matthew *et al.* [22] presented the design, implementation, and evaluation of Ganglia, a scalable distributed monitoring system for high-performance computing systems. Brent *et al.* [11] describe the initial

implementation of PlanetLab, including the mechanism used to implement virtualization, and the collection of core services used to manage PlanetLab.

However, the authors argue that more work is still required to support sensor and actuator middleware services to provide for instance; a seamless generation, deployment and discovery of typed on-demand sensors [2]. This will be further detailed in Section 4.

3. Using PlanetLab Overlay

The developed sensor and actuator framework runs over the PlanetLab environment providing monitoring facility, in that, a set of deployed sensors and actuators monitor PlanetLab components -- nodes, proxies, slices. For example, as portrayed in Figure 1, the sensor and actuator framework scenario contains three major actors namely:

- Sensor provider: deploys and exposes a specific sensor and its meta-model via the framework for use by other PlanetLab agents.
- Sensor container: responsible for storing and managing the sensors' information, which is deployed by the sensor provider in this framework.
- Consumer: sends a request for a sensor to be injected in the targets. The consumer can ask for more than one sensor at a time to be injected in more than one target.

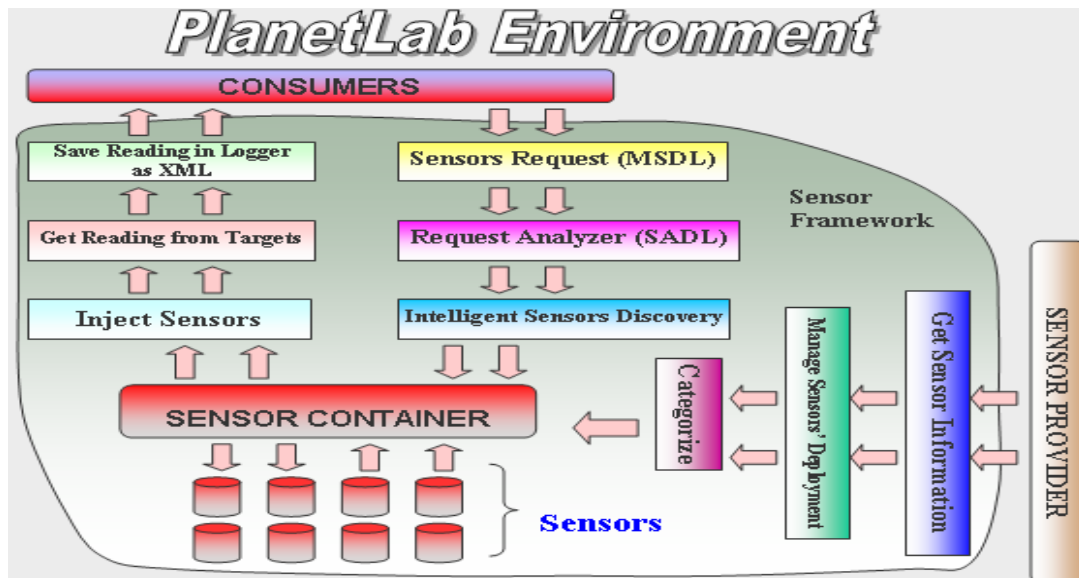


Figure 1. Sensor Framework for PlanetLab Environment.

In other words, the sensor provider supplies the framework with the required sensors, which are able to read different types of data. The process of the

framework starts by gathering information concerning deployed sensors. Such information includes types of sensors, environment, application,

category of sensor, location, etc. Such information is described using the developed Sensor and Actuation Description Language (SADL) [2]. The next section illustrates the use of SADL with the PlanetLab environment. The Sensor framework manages the lifecycle deployment of the sensors according to their type, category and application. In view, of the large number of sensors likely to be deployed with the framework, an indexing scheme has been introduced to improve the discovery and selection of sensors. The sensor container stores all the sensors' information in its registry. Such information is used by the sensor framework to inject sensors to monitor a given target.

To this end, the sensors are deployed with the sensor framework. Consumers use Monitor Session Description Language (MSDL) [2] to send requests for sensors. MSDL contains complete information for the session task describing the interval of reading, targets information, and contract information. SADL analyses the consumer request to find the most suitable sensors, according to the consumer's requests and contract. After finding the required sensors, the system injects them into the required targets and starts collecting measurement data, which is stored in a log file and delivered to the consumers as an XML file.

The next sections outline the use of the sensor and actuator framework together with its description languages using the PlanetLab testbed.

4. Sensor & Actuation Description Language and PlanetLab Overlay

Previous research has focused on the deployment of a range of sensors to detect and record various systems' activities in the PlanetLab environment. Such readings are used to supply the self-management and control services with contextual data necessary for the adjustment of systems operation and behaviour. Consequently, SADL is used to provide adequate information regarding the deployed sensors.

SADL was originally developed to provide standard methods for exchanging information between different actors in the sensor framework to support our grid computing middleware [2]. In this paper we show its applicability to global computing using PlanetLab environment. SADL is used to deploy the PlanetLab sensor overlay with our sensor framework. Different types of sensors are available to be deployed with our framework such as; CoMon [7], Ganglia [9], Node List [10] and others. A SADL process consists of two parts: (i) sensor information - - deployed by the provider, and (ii) process discovery -- from a given consumers' requirements.

Figure 2, presents an example of using SADL with PlanetLab. In this example CoMon sensor [7, 8] is selected to be deployed, which provides data regarding processor status, memory state and transmission bandwidth. The deployed information consists of: basic sensor information, contract information, which describes Service Level Agreements (SLA), a user interface for requesting sensors (if possible), environment information, and required resources as shown in Figure 2.

5. Monitor Session Description Language

Monitor Session Description Language (MSDL) provides a meta-description of a consumer's monitoring requirements, which is used by the sensor and actuator services as described in Section 3. Figure 3 presents an example of MSDL to access and interact with PlanetLab Sensors. The consumer sends information using MSDL format, which covers the application (PlanetLab, Grid, Connected-Home machine and others), duration of requesting information, client or target information and requested sensors. SADL uses this information to find the requested sensors and inject them into the target and start to obtain readings. The data is then stored in logger, which is responsible for converting the readings into a standard format, based on XML, and delivering it to the consumers.

```

<SADL>
  <Sensor>
    <SensorID>10</SensorID>
    <SensorName>CoMon</SensorName>
    <SensorDescription>To get information from the PlanetLab
regarding processor, memory, and bandwidth</SensorDescription>
    <Application>planetlab</Application>
    <SensorInformation>
      <Type>Performance</Type>
      <DataType>string</DataType>
      <DataStorageType>XML</DataStorageType>
      <Host>cambridge.intel-research.net</Host>
      <Container>Cambridge</Container>
      <Execution />
      <Status>On-Line</Status>
      <Category>Memory</Category>
      <Location>planetlab1.cambridge.intel-
research.net</Location>
    </SensorInformation>
    <Contract>
      <ContractID>16</ContractID>
      <ContractName>Wail</ContractName>
      <ContractLease>1/1/2006</ContractLease>
    </Contract>
    <Interface>
      <InterfaceName>PlanetLab Sensor</InterfaceName>
      <InterfaceLocation>www.cmpwomar.livjm.ac.uk/
PlanetLabSensor.exe</InterfaceLocation>
    </Interface>
    <Environment>
      <Platform>Any</Platform>
      <Middleware>.Net</Middleware>
      <MaximumUsers>1000</MaximumUsers>
      <CurrentUsers />
    </Environment>
    <Resources>
      <Processor>PI233</Processor>
      <Memory>32M</Memory>
      <Framework>Windows-Unix</Framework>
    </Resources>
  </Sensor>
</SADL>

```

Figure 2. SADL example for PlanetLab environment.

```

<MSDL>
  <MonitorSession Session_ID="10">
    <Application>PlanetLab</Application>
    <Duration>
      <DurationStart>11:05:49 AM</DurationStart>
      <DurationEnd>11:07:49 AM</DurationEnd>
      <Interval>30000</Interval>
    </Duration>
    <TargetInformaion>
      <Client_ID>544</Client_ID>
      <HostName>princeton_codeen</HostName>
      <WebAddress>128.112.139.71</WebAddress>
      <UserName>Wail</UserName>
      <Password>XXXXXX</Password>
    </TargetInformaion>
    <ContractInformation>
      <ContractID>16</ContractID>
      <ContractName>Wail</ContractName>
      <LeaseTime>1/1/2006</LeaseTime>
    </ContractInformation>
    <Sensors>
      <ID>7</ID>
      <Name>TCP Queries</Name>
    </Sensors>
    <Sensors>
      <ID>10</ID>
      <Name>CoMon</Name>
    </Sensors>
  </MonitorSession>
</MSDL>

```

Figure 3. MSDL example for PlanetLab consumers.

Figure 4 outlines the format of the readings (instrumentation data) accessed from the log service (repository) for CoMon sensor. The parameters include:

- CTX: the slice's context ID, a sort of user number
- TX1: transmit bandwidth in Kb/s over 1 minute
- TX15: transmit bandwidth in Kb/s over 15 minutes
- RX1: receive bandwidth in Kb/s over 1 minute
- RX15: receive bandwidth in Kb/s over 15 minutes
- #PR: the number of processes owned by this slice
- PMEMMB: the amount of physical memory (in MB) used by this slice
- VMEMMB: the amount of virtual memory (in MB) used by this slice
- %CPU: what fraction of the CPU is being consumed by this slice
- %MEM: what fraction of memory is being consumed by this slice, and
- NAME: the slice's name.

```

<Logger>
<Readings>
  <CTX>524</CTX>
  <TX1>107</TX1>
  <TX15>115</TX15>
  <RX1>113</RX1>
  <RX15>124</RX15>
  <PR>58</PR>
  <PMEMMB>82.1</PMEMMB>
  <VMEMMB>144.6</VMEMMB>
  <CPU>6.2</CPU>
  <MEM>8.0</MEM>
  <Name>princeton_codeen</Name>
  <TimeOfReading>2/16/2005 11:15:30 AM</TimeOfReading>
</Readings>
<Readings>
  <CTX>524</CTX>
  <TX1>107</TX1>
  <TX15>115</TX15>
  <RX1>113</RX1>
  <RX15>124</RX15>
  <PR>58</PR>
  <PMEMMB>82.1</PMEMMB>
  <VMEMMB>144.6</VMEMMB>
  <CPU>7.9</CPU>
  <MEM>8.0</MEM>
  <Name>princeton_codeen</Name>
  <TimeOfReading>2/16/2005 11:16:01 AM</TimeOfReading>
</Readings>
</Logger>

```

Figure 4. Logger Example

6. Implementation

The current implementation of the framework and middleware services is achieved using the .Net framework and Visual Studio, with the sensor and actuator services developed as web services (Figs. 5, 6). SOAP is used to invoke PlanetLab services and achieve distributed service programming as the PlanetLab environment currently does not support Windows OS or the .Net framework. Figures 5 and 6, illustrates a proof-of-concept interface for the general management of the sensor and actuator overlay. The latter is used for the self-management of the overlay.

7. Conclusions

The paper presented an overview of a developed sensor and actuator framework for gathering information to support monitoring and control utility for autonomic services – systems' self-management. To study scalability issues, a usage scenario of the developed sensor and actuator framework was conducted using the PlanetLab environment. A prototype of the Sensing and Actuation Description Language (SADL) is designed for the purpose of deploying, discovering, and exchanging information concerning different types of PlanetLab sensors (Sec. 4). A prototype for Monitor Session Description

Language (MSDL) (Sec. 5) has been developed to present a standard method for sending monitor job schedules from the consumer to the sensor framework. A standard format of passing information, from the logger inside the sensor framework to the consumers, is demonstrated in this paper.

To this end, the sensor framework is tested using a large scale system such as; PlanetLab. The results are promising and expected to provide a template for generating, deploying, and discovering different types of sensors for different types of environment. Moreover, this highlights the use of middleware, with its services and functions, for the PlanetLab and WAN environments.

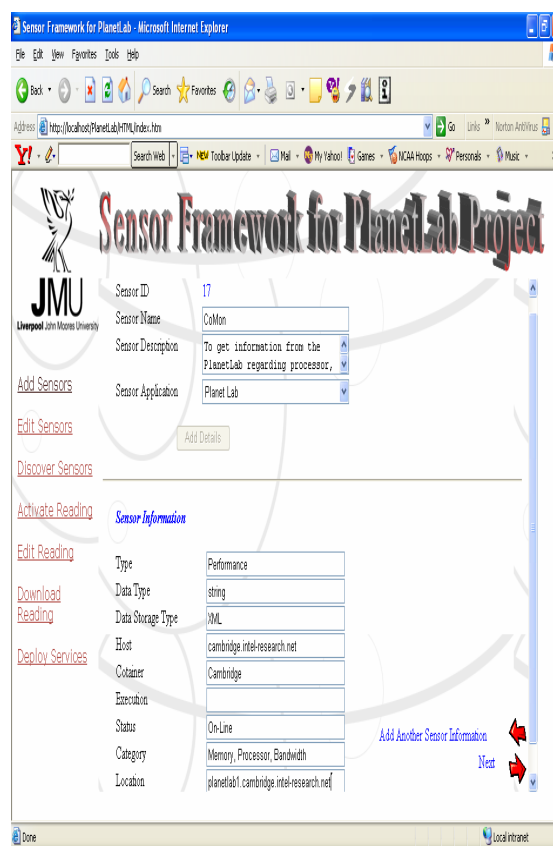


Figure 5. Deployed Sensor



Figure 6. Discover Sensor

8. References

- [1] <http://www.planet-lab.org>
- [2] Wail M. Omar, Bassam A. Ahmad, A. Taleb-Bendiab and Yasir Karm, "A Software Framework for Open Standard Self-Managing Sensor Overlay For Web Services".
- [3] R. Pollock, T. Lane and M. Watts, "A Kohonen Self-Organizing Map for the functional classification of proteins based on one-dimensional sequence information", 2001.
- [4] N. N. Schraudolph and T. J. Sejnowski, "Competitive Anti-Hebbian Learning of Invariants", 1992.
- [5] T. Joachims, "Text Categorization with Support Vector Machine: Learning With Many Relevant Features", ECML-98. 10th European Conference on Machine Learning, Heidelberg, Germany, 1998.
- [6] C. Hsu, C. Chang, and C. Lin, "A Practical Guide to Support Vector Classification", 2003.
- [7] <http://comon.cs.princeton.edu/>
- [8] <http://codeen.cs.princeton.edu/>
- [9] <http://planetlab.millennium.berkeley.edu/>
- [10] <http://www.planet-lab.org/db/nodes/nodelists.php>
- [11] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services. January 2003", *Appears in ACM Computer Communications Review, vol. 33, no. 3, July 2003, a special issue on tools and technologies for networking research and education.*
- [12] J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage", *In Proceedings of the Ninth International Conference on Architectural Support for Programming Language and Operating Systems (ASPLOS 2000)*, Nov. 2000.
- [13] F. Dabek, M..F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS" , *In Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP_)*, Chateau Lake Louise, Banff, Alberta, October 2001.
- [14] L. Wang, V. Pai and L. Peterson, "The effectiveness of Request Redirection on CDN Robustness" , *In Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI)*, Boston, MA, Dec. 2002.
- [15] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The End-to-End Effects of the Internet Path Selection" , *In Proceedings of the ACM SIGCOMM Conference*, Cambridge, MA, Sep. 1999.
- [16] <http://www.planet-lab.org/doc/UsersGuide.php>
- [17] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An Algebraic Approach to Practical and Scalable Overlay Network Monitoring", *ACM SIGCOMM*, 2004.
- [18] <http://jabber.services.planet-lab.org/>
- [19] <http://www.planet-lab.org/logs/iperf/>
- [20] <http://www.intel-iris.net/irislog/irislog.php>
- [21] <http://www.swordrd.org/>
- [22] Matthew L. Massie, Brent N. Chun, and David E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience", *Parallel Computing, Vol. 30, Issue 7*, July 2004.
- [23] Darlington, R. B. (1990), "Regression and linear models", New York: McGraw-Hill.