ESPRIT Project No. 25 338

Work package G

Service Deployment

Requirements

ID:	DG2 V. 1.0	Date:	30.01.98
Author(s):	L. Amsaleg, M. Billot, M. Le Nouy	Status:	
Reviewer(s):		Distribution:	


FollowMe

Change History

Document Code	Change Description	Author	Date
DG2 Requirements	First version of document. No changes.	LA, MB, MLN	30.Jan.98

1 Introduction

The architecture of FollowMe raises new issues related to distributed services. New opportunities are based on agent technology, which provides mobile code. In work package G, we investigate some methods aimed at taking advantage of these new opportunities in order to provide load-balancing tools. We do not plan to provide an exhaustive compilation of the available techniques, however we expect to exhibit some useful examples. As an example, we consider the deployment of Etel++ over the architecture of FollowMe. This application, described in DJ1 exhibits properties such as distributed computing, extensive usage of storage and data transfers. Therefore, it requires efficient load-balancing policies relying on accurate monitoring tools.

The main goals of work package G are as follows:

- To allow applications to monitor the performance of hosts. As an example, this monitoring may deliver numbers reflecting the resource consumption of an agent or the available bandwidth between two hosts.
- To provide a set of tools which allow extracting useful information of the large amount of numbers the monitoring delivers. These tools exhibit tendencies, meaningful numbers (average, deviation, and mean square, …) and also predictive values.
- To implement a load-balancing policy dedicated to Etel++.

2 Architecture

The proposed architecture of the work package G is based on two levels:

- The monitoring tools, which rely on measurement, on the notions of target, history and filters. These features are described hereafter.
- A load-balancing policy based on the proposed monitoring tools which should be detailed by means of documents DJ3 (Requirements of Etel++) and DG4 (Design of Service Deployment). However, we provide some basic use-cases in order to illustrate the utilization of the proposed monitoring tools.

Monitoring tools

Sampling and history of measurements

We already described in document DG1 the resources we intend to monitor. This monitoring should typically reflect the utilization of CPU, disk, memory and network. The utilization of a resource such as a disk is described by means of basic measurements reflecting for example the used storage space. These measurements are also attached to the notion of cluster, that is, the measurement may reflect the utilization of storage space of a cluster or a set of cluster. Therefore, we intend to provide the monitoring of hardware resources for various targets like hosts, clusters or sets of clusters.

Applications also rely on more complex measurements. An application monitoring the bandwidth available on the network of a given host has to collect measurements of bandwidth available through various paths. These elementary samples can be compiled in a complex measurement reflecting more accurately the bandwidth of a given host. Work package G has therefore to provide a notion of aggregate of basic measurements.

In order to obtain reliable values useful for mid-term to long-term decision taking, we also need to accumulate successive values in order to build an history of the behavior of the monitored resource(s). Dealing with the large number of values stored in a history is a difficult task based on the detection of representative values and tendencies (see also DG1: survey on monitoring tools and data mining techniques). The computation of such meaningful values requires filtering of the history of measurements.

Using the History

There are two basic ways to use the histories of measurements:

1. REQUEST-based mode. In this case, an application gets values upon explicit requests. A typical request needs the resource-name and a time range as arguments, and may return for example the average load placed on that resource during the specifyed period.
2. NOTIFICATION-based mode. In this case, an application gets values when an application-defined condition becomes true. To use an history in this mode, the application has to specify the name of the resource, a time range, and a condition. For example, the application can ask to be notifyed when the average load placed on that resource for the period becomes greater than a specific threshold value.

In general, the tradeoffs for chosing between these two modes are analogous to the ones that exist for the delivery of messages either upon request or via a notification service.

To increase the power of the tools provided by this work-package, several filters will be included as features. A filter allows applying a (possible) complex operation on a set of raw-values (i.e., elementary samples) in order to compute higher level data more directly relevant to applications. Typical filters are for example:

- Average, minimal or maximal value over an interval of time, mean-square, etc…
- Linear interpolation, high- or low-gain filters…,
- Prediction of tendencies.

In general, we intend to support arbitrary composition of filters, as the opportunity to extend the set of filters.

Low-Level Interface (draft)

We outlined in the previous section the need for composition rules of basic measurements in order to built measurements better suited to more complex requirements. The composition rules are described hereafter by mean of a short grammar indrocucing the notions of `Target` and `MeasurementAggregate`.

```
HardwareResource :: Disk | CPU…
ClusterAggregate :: Cluster+
Target :: Host | ClusterAggregate
BasicMeasurement :: (HardwareResource, Target)
MeasurementAggregate :: (MeasurementAggregate | BasicMeasurement)+
```

Basically, the interface of work package G relies on two classes:

- MeasurementHistory gathers actual measurements.
- Filter is applied on a MeasurementHistory or a Filter. Filter provides an evaluation of measurements after processing such as interpolation or estimation.

These classes are briefly described hereafter:

```
interface Filter
{
public Filter (Filter f);

public getEvaluation(Date d) throws NoEvaluationException;
}

Class MeasurementHistory implements Filter
{
Public MeasurementHistory (MeasurementAggregate m, SamplingFrequency s);
}
```

Some uses-cases are given in the following section.

Load-balancing policies

Exploitation

In the document named DG1, Section 4.1, we presented the three entities on which ETEL++ is constructed (i.e., the Ouest-France Server, several secondary servers and finally user terminals). We also presented the three different possibilities for computing the electronic newspaper, the computation taking place either in one of the previous entities, or in any combinations. We also mentioned the tradeoffs that might help choosing where the computations should take place. In this paper, we use this example again in order to illustrate how the interfaces provided by WP-G might be used.

Use-cases

In order to determine whether a host is able to support an Etel‘s agent or not, we create a MeasurementHistory which will monitor disks:

```
measurement = new basicMeasurement (DISK, HOST);

history = new MeasurementHistory (measurement, frequency);

/*

EtelMax value corresponds to the utilization of storage space by Etel’s agent
(This value could have been computed on another host). AvailableSpace corr e-
sponds to the storage capacity of the disk.

*/

Filter diskSpace = new MaxFilter (new EstimationFilter (new InterpolationFilter
(history)));

If ( EtelMax <= Available - diskSpace.max() ) {

      // Etel decides to place an agent on HOST

}
```