# Supporting decision-making in pervasive grids with advanced network distances

Julien Gossa[1], Jean-Marc Pierson[2], and Lionel Brunie[1]

[1] LIRIS INSA Lyon, 69621 Villeurbanne, FRANCE
{firstname.lastname}@liris.cnrs.fr,
WWW: http://liris.cnrs.fr
[2] Universite Paul Sabatier, F-31062 Toulouse, FRANCE
{lastname}@liris.cnrs.fr,
WWW: http://www.irit.fr

**Abstract.** The merging of grid and pervasive computing raises new challenges. Its main impact is a critical increase of the dynamicity, heterogeneity and unpredictability of the users, of the uses and even of the network topology. Moreover, emergency *scenarii* are one of the target applications of pervasive grids, with strong time constraints making the performances a critical point. In such a context, the decisions about the distribution become a key of the efficiency. However, existing grid middlewares have been designed for quite stable, homogeneous and predictable architectures. Our proposal is a distribution decision-making support allowing to make optimal decisions in an easy, usable and profitable way. We present a use case of pervasive grid and demonstrate why existing solutions are not adapted. We show how our method has been embedded in a grid service, namely the Network Distance Service (NDS) and illustrate its benefit on the main problems of distribution: replication, placement and selection of data, task, services... We show experimentation results and discuss how NDS can be useful to handle the new constraints involved by the merging of grid and pervasive computing. [3]

## 1 Introduction

One might wonder if the term "pervasive grid" is really relevant and if the underlying concept is really consistent. As V. Hinge et al. show in [1], grid computing and pervasive computing have a lot of divergent characteristics against few convergent ones: Grid computing is meant to support intensive computation and storage on large scale highly powerful and stable infrastructure with light time constraints whereas pervasive computing essentially concentrates on interoperability of light mobile devices for real-time delivery of adapted contents thanks to local interactions. Thus, the purpose of their merging is far from being obvious.

About the term "pervasive grid" itself, one can note that Foster, Kesselman and Tuecke used the term "the Grid" in the earlier works. As "the Web", "the

[3] This work is supported by the french ministry of research within the ACI "masse de donnees" (Grid for Geno-Medicine Project)

Grid" denotes a resource available from anywhere, at any time, by any mean and for any purpose. From this point of view, "the Grid" was in fact meant to be pervasive from the beginning. As a matter of fact, "the Grid" was abandoned for "grid computing" and this pervasive aspect have been put on one side. The "pervasive grid" can be perceived as a revival of this first approach.

First of all, we have to define what the physical and logical architecture of a pervasive grid should be.

## 1.1 A Pervasive Grid Architecture

As the Grid is designed to achieve very intensive computation and storage, it is nowadays unrealistic to consider that mobile devices will take part of the grid inner tasks due to their limited capabilities and autonomy. Currently, the most relevant way to integrate mobile devices into grid architectures is to allow mobile users to access grid resources. This can be allowed by extending the grid access points with the capabilities of proxies techniques which are widely used in pervasive computing. Thus, the mobile devices will be located at the edges of the grid only as shown in figure 1.2.

This approach has two big advantages from the grid software architecture point of view: (1) The current grid middlewares do not have to be implemented on mobile devices (which seems to be impossible regarding their size and complexity). Only grid clients have to be implemented on these devices. (2) The grid middleware still can be used inside the grid and thus do not have to be fully rethought.

The peculiar point we are interested in is: How will pervasive accesses impact on the existing inner activities of the grid? And which of the grid management mechanisms should be redesigned?

## 1.2 Pervasive Grid Use Cases

In this section, we identify two realistic use cases of pervasive grids.

The first use case concerns simple accesses to grid monitoring information from mobile devices. This allows administrators and users to check the state of their resources and jobs when they are out of office. This does not modify the constraints of any of the two computing areas and the problem is limited to interoperability and adaptation issues.

The second use case concerns the emergency situation described in [2]: "A crisis management team responds to a chemical spill by using local weather and soil models to estimate the spread of the spill, determining the impact based on population location as well as geographic features such as rivers and water supplies, creating a short-term mitigation plan (perhaps based on chemical reaction models), and tasking emergency response personnel by planning and coordinating evacuation, notifying hospitals, and so forth.".

This use case has not been described in more details. However, it implies a lot of interesting issues relevant in a pervasive grid context and should be detailled.

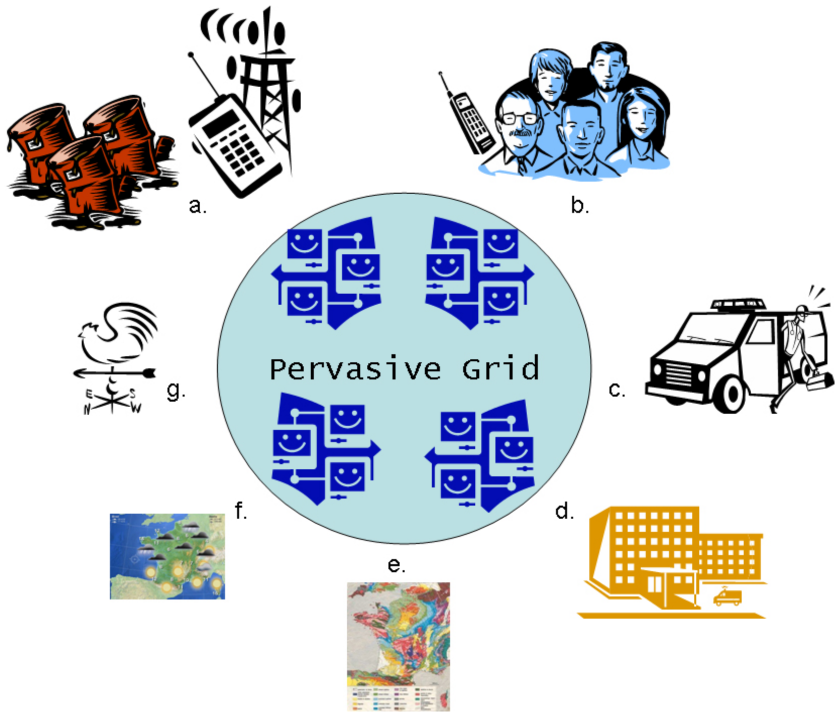

**Fig. 1.** A pervasive grid architecture for emergency use case

We can detail some of the interaction between the grid infrastructure and the external pervasive infrastructure:

a. Sensors must be deployed over the chemical spill scene, probably organized in had-oc network.
b. The local population must be located by their mobile phones and must be alerted with SMS.
c. The emergency units must be coordinated by their PDA.
d. The hospitals must be alerted and must provide the mitigation plan through the Internet.
e. A geological expertise center must provides geological maps and associated services able to estimate the spread of the spill.
f. The weather center must provide measures and predictions.
g. Local sensors, like weather vanes, must be used to obtain local accurate measurements.

This use case presents great interests and challenges:

– It is highly time-constrained while requiring very intensive computations.
– It involves safe, adapted and relevant content delivery to mobile devices.
– It involves on-demand integrations of scattered sensors devices.

Another use case of pervasive grid in emergency situation has been presented by Pierson in [3].

Obviously, existing grid solutions can not handle such use cases from many point of views. The peculiar aspect we are interested in is the distribution decision-making. In this use case, the most crucial decision concern the data and services provided by the geological center. Initially, it is either outside the grid in the center, or inside the grid at an arbitrary location, probably not adapted to the current situation. If these data and services are not optimally distributed and used over the grid, the estimation of the spread of the spill might take too much time, leading to irreparable environmental and sanitary consequences.

However, these decisions are very difficult to make because they depend on many parameters, for instance:

- The topology and capability of the grid and the characteristics of the services: Resource consuming services must be deployed on high-performance nodes, key services must be replicated as much as possible...
- The location, number and characteristics of the different actors: If the sensors are very numerous and produce very large sets of data, the services must be deployed close to them; If the data produced by the services are very large, the service must be deployed close to the consumers...
- The global size of the involved area: If the spill is very local, only one instance of the services might be the optimal solution; If the area is at country-scale the services might need to be replicated all over the network.
- The size of the used data: If the used maps concern a limited area, only one instance can be sufficient; If the maps are very large and fine grained, they might be not storable on a unique node, and thus must be distributed and replicated.

This (not exhaustive) list shows that this distribution decisions are numerous, crucial and very difficult.

### 1.3 Decision-Making Support In Pervasive Grid

Distribution decisions concerns all grid resources: data, databases, services, hosts... The three main problems are:

- Replication: How many instances of each resource is needed?
- Placement: Where these instances must be put?
- Selection: Which instance must be select when the resource is requested?

As the configuration can vary from a situation to another, the decision-making process can not be done beforehand by the developers and administrators, but rather at run time to suit actual specific constraints. Moreover, as number of highly heterogeneous devices must be instantaneously integrated, the grid infrastructure can be suddenly thrown of balance: parts of the architecture might need to be reorganized as fast and optimally as possible to suit the new grid configuration.

To tackle with this kind of situations, QoS management solutions are mandatory. But, even with a powerful QoS management, if the distribution decisions are not optimal then the inner activities of the grid are not optimized. Consequently, the time constraints might not be satisfied.

Moreover, in a pervasive grid, the number of services will dramatically increase. In this context, the optimization of the whole service oriented architecture will rely on the optimization of each service. Each of them will have to face high dynamicity and heterogeneity. Consequently the decision-making support will be need at each level of the architecture and must be adaptable to several contexts.

To achieve optimality, decision-making processes in a pervasive grid must take into account several aspects:

- The infrastructure topology and condition: communication, computation and storage capabilities...
- The characteristics of the task: communication, computation and storage needs...
- The objectives to be achieved: End-user response delay, load balancing, limitation of financial costs...

Moreover, decision-making supports have some constraints:

- Profitability: The ratio profit/cost must be satisfactory.
- Usability: It must be exploitable by the other services with the less effort possible.
- Adaptability: It must be able to face any need in any situation.
- Accuracy: Obviously.

As a matter of fact, existing grid middleware are designed to deal with quite static and stable resources with no real-time constraints. Consequently, they lack of support for distribution decision-making to handle the dynamicity, adaptativity, reactivity (and even proactivity) required by pervasive grid use cases.

In this paper, we propose a decision-making support service designed to handle all the presented issues.

The rest of this article is organized as follows. First in section 2, we present the existing decision-making supports provided by grid environments and their limitations. Second, we describe our proposal in section 3 and analyse some experimentations in section 4. Finally, we discuss our proposal in section and conclude our article with perspectives in section .

## 2 Related Work

Decision-making in existing grid environment is mainly supported by the monitoring systems.

## 2.1 Monitoring Systems

The identification of relevant metrics for grid environment and measurement methods has been made by the Network Measurements Working Group of the Global Grid Forum in [4]. Recent developments in grid infrastructure have lead to effective tools providing the indexing of all the hosts, data and services available in the network, such as the Monitoring and Discovery Service of Globus [5] which is based on the GMA.

The Grid Monitoring Architecture (GMA) responds to the main constraint of grid context which is users and resources dynamicity. The GMA-compliant grid resource monitors differ by the kind of entities they manage. MDS [5] and R-GMA [6] allow to monitor physical resources like CPU power and storage when other monitors like SCALEA-G [7] allows managing software resources too. They differ also by some implementation considerations. First, the stored information at the top level of the architecture: In MDS, only resource availability is mentioned. Users interested by an available resource must then request more information from the local levels of monitoring. The second difference is about the storage format and query language. In R-GMA, storage is implemented like a virtual relational database queried with an SQL-Like language, when in SCALEA-G[7], information is stored in an XML format and queried using Xquery language. MDS stores resource information in an LDAP-like directory.

Another approach is adopted by the Network Weather Service [8]. It is able to capture the condition of both network and hosts. It can provide the raw measurements of the classical metrics as well as forecasts based on aggregations of the set of raw measurements. NWS is not a service: Its integration into the grid service oriented architecture is limited as well as its usability.

Nevertheless, those monitoring system are designed to produce and deliver measurements efficiently. They are too low-level to be really usable by any service as a support for decision-making: Developers have to deal with raw metrics such as latencies or CPU availabilities which are far from their real and immediate concerns. Using these metrics separately is inadequate to take adapted complex and accurate decisions, taking into account all the aspects presented before. This implies a considerable effort from the developers and thus presents expertise issues. From our point of view, raw network metrics are well relevant to low-level tasks like packet routing that have simple goals well identified. In grid environment, the tasks are in fact *services invocations*. But these *services* have complex and diverse goals and behaviours.

Moreover, in most cases, several monitoring system must be used to obtain a full view upon the network. For instance MDS and NWS must be used together to have a complete monitoring of both hosts and communications. Unfortunately, each of these systems is very expensive in term of resource consumption.

To conclude, monitoring systems are not usable enough and too expensive to be fully profitable.

## 2.2 Network Distances

An alternative to monitoring system providing several raw metric measurements, is the concept of *distance*. Distances are a wonderful tools to support decision-making: they are simple and refer of real life concept. Thus they are widely usable by anyone in intuitive ways, facilitation decision-makings.

Several Distance Vector protocols have been implemented for routing of packet-switched networks in computer communications, as in, for example, the Routing Information Protocol for Internet traffic. Here, the concept of *distance* is actually the number of hops from an end point of the network to another. Since then, this concept has often be reused, not only for routing (RIV v2, IGRP, EIGRP, OSPF,...) but also for very different purposes such as data management, network topology discovering, resource brokering, nodes clustering, etc. We do think that the popularity of this concept comes from its similarity with our real world. So it constitutes a precious help in the understanding of the network and thus in the elaboration of decision making process.

A good illustration of distance usefulness for decision-making is presented by Karlsson et al. in [9]. It proposes an evaluation framework for replica placement algorithms and a rather complete survey on this topic with several algorithms comparison. They define the distance as "a metric such as network latency, number of network hops, or total link *cost*". Most of the listed algorithms use a notion of distance. But this notion is always restricted to a representation by a single raw network metric. An use case of distance in grid environment is presented in [10]. The authors present a method for automatic nodes clustering. Distances are represented by the minimum RTT and are used to map the hosts to a geometric space.

The main topic using distance concerns Distance Map Services. They aim at providing a map of the physical network architecture. These maps are graphs where the vertices are the hosts of the network and the weights of the edges are computed thanks to a distance function. The main goal of these works are to define scalable software architectures meant to provide an estimation or a prediction of the distance between all the hosts of a network while minimizing the number and cost of measurements (which can be very expensive). Most of them base their distance computation on network latency which is the easiest and one of the most inexpensive to measure. For instance: Francis et al. with IDMaps in [11], Eugene et al. with the Global Network Positioning (GPN) in [12] and Coates et al. in [13]. Few works are able to integrate other network metrics. For instance IDMaps can integrate the bandwidth "when possible", but do not give any further details on how this should be done.

An observation is related to the use of the metrics that are not related to the network itself but rather to the characteristics of the hosts. As a matter of fact, metrics such as CPU or disk or memory are never considered in any of these works although they are important in many cases. Obviously, CPU capabilities and loads have a major impact on computing tasks performances. Moreover, some other tasks, such as database query optimisation, need advanced monitoring information such as disk throughput or memory capacity and load.

Advanced Distance Map Service should be able to take these host metrics into account in the distances they provide.

To conclude, Distance Map Services focus on measurement accuracy and delivery effectiveness and are based on a single metric, most of the time the latency. Unfortunately, they are not enough adaptable to fully support any decision-making process and thus to be fully profitable.

To sum up, both raw monitoring systems and Distance Map Services are inadequate to fully support decision-making processes as needed in a pervasive grid environment. Our proposal is a new service able to provide fully adaptable distances in an easily usable way, well integrated into service oriented architectures and thus ensuring the most profitability possible.

## 3 NDS: The Network Distance Service

Our purpose is to define a method to design made-to-measure network distances for any given *task* to support any distribution decision-making in pervasive grid environment. We call *task* an interaction between hosts of a network. Generally, it corresponds to the invocation of one service. But it may be more basic tasks such as data retrieval or storage. Such distances are meant to be more usable than raw metrics and more relevant than the distance provided by Distance Map Services.

The computation of our distances is based on the measurement of different raw metrics that can be provided by any monitoring systems. This computation has been embedded in a Web Service developed with Globus Toolkit 4: the Network Distance Service.

### 3.1 Metric Model

According to the GGF NM-WG in [4], a metric is a quantity related to the performance and reliability of the Internet. More precisely, a metric is a primary characteristic of the Internet, or of the traffic on it. A metric is the characteristic itself, not an observation of that characteristic. A measurement is an observation of a metric. Measurements may be either raw or derived. Raw measurements are something that can be measured directly, such as measuring latency using ping. Derived measurements are measured indirectly, and might be an aggregation or estimation based on a set of low-level measurements. For convenience purposes, we consider that a *metric* is a primary characteristic measured with a particular method. For instance, the last observation of the latency is one metric and the mean of all the available observations of latency over the past day is another metric.

We call either *observation* or *measure* the actual value related to one particular metric. We note $M$ the set of metrics handled by the NDS server.

The metrics we use in this article are:

– the bandwidth: $BW$

- the latency: $L$
- the CPU capacity: $CPUc$
- the CPU availability: $CPUa$

The observation of the network metric $m$ from the host $i$ to the host $j$ is noted $m_{i,j}$. The observation of the host metric $m$ for the host $i$ is noted $m_i$.

One can note that monitoring services can not provide observations of network metrics from one host to itself. These peculiar values must be configured by the user: Here, we decide to have

$$\forall i,\ BW_{i,i} = \infty \text{ and } L_{i,i} = 0$$

because we want to consider that the cost of local data transfer is null (in term of network usage) with the compound metrics we define in section .

**Integration into NDS:** NDS must be able to use the measures provided by any monitoring tools, while being easily usable. Thus monitoring tools are accessed through command lines execution. Then, no wrapper has to be developed and the integration of a new metric, or even a new tool, is done through a simple JNDI configuration file. The only requirement is that the monitoring tools must by queryable from the execution host of NDS. A metric is described by:

- Its name
- Its description
- The description of its parameters
- The command line executed to retrieve its measurement
- The type of the measurement
- The value if $m_{i,j}$ for network metrics

Here is the declaration of the TCP latency measured by NWS:

```
<parameter>
   <name>
      metrics
   </name>
   <value>
      NWSlatencyTcp
      The amount of time required to transmit a TCP message
      Parameters: %P1, %P2 = IP or name of targets, use %Src and %Dest
      nws_extract latencyTcp -n1 -h0 -fMeasure -N NWSnameServerIP %P1 %P2
      double
      0
   </value>
</parameter>
```

### 3.2 Compound Metric Model

As a matter of fact, service invocations are generally composed of three steps:

1. The client sends a request to the service with its problem data.
2. The service computes the response.
3. The service returns the response data to the customer.

Actually, this is well adapted to a large diversity of computation tasks, such as multimedia contents adaptation or data mining or database request. The steps 1 and 3 correspond to data transfer across the network while the step 2 corresponds to a computation by a host (or a cluster hidden behind this host).

According to the task, one step can be ignored in the distance computation. For instance a simple database "select" query does not involve important request data and thus the step 1 might be irrelevant. On the other hand for a basic data transfer, the step 2 and 3 should be ignored. One can note that if the task is very simple, for instance a time service, none of these three steps are important. As a matter of fact in that case, our method is practically useless.

Now we will see how we can model simple compound metrics representing the cost of a data transfer, the cost of a computation and the cost of any service invocation.

**DTC:** First, we have to build a compound metric called $DTC$, for *Data Transfer Cost*. It is involved in the steps 1 and 3 and must reflect the cost to transfer a piece of data from one host to another according to its size $x$ and the network capacity and condition. This time can vary according to many parameters, such as the protocol used, the MTU, the client and server configuration... We will define a basic metric here only based on the main parameters and on a raw TCP/IP data transfer. According to [14], "the Raw Bandwidth model using NWS forecasts can be used effectively to rank alternative candidate schedules". Then, we decide to base our function on it:

$$\forall (i,j) \in \mathcal{V} \times \mathcal{V},\ DTC_{i,j}(x) = \frac{x}{BW_{i,j}}$$

Actually, we observe that the size of the data does not influence only the distance itself, but also the relevance of the different metrics: Latency is the key factor about small size data, whereas Bandwidth is the key factor about large ones. The integration of the latency in our distance function is done according to the TCP/IP protocol: Opening and closing the connection needs 3 round trips (respectively $SYN \rightarrow SYN/ACK \rightarrow ACK$ and $FIN \rightarrow FIN/ACK \rightarrow FIN$). Then we obtain:

$$\forall (i,j) \in \mathcal{V} \times \mathcal{V},\ DTC_{i,j}(x) = 3 \times (L_{i,j} + L_{j,i}) + \frac{x}{BW_{i,j}}$$

**CTC:** Second, we have to build a compound metric representing the *Computation Task Cost*, noted $CTC$. It is involved in step 2 and must take into account the complexity of the computation and the computing capacity of the target host. Basically, we can represent this cost by:

$$\forall i \in \mathcal{V},\ CTC_i(x) = \frac{x}{CPUc_i \times CPUa_i}$$

where $x$ is the number of computation cycle needed by the task. One more time, this representation is very basic and must be refined to fit more accuracy needs as it depends on several host characteristics such as architecture, OS, bus frequencies, buffers size, scheduler configuration, etc.

**SIC:** To be the more close possible to the concerns of the developers and administrators, we have built another compound metric representing one *Service Invocation Cost*, noted *SIC*. It is defined by:

$$\forall (i,j) \in \mathcal{V} \times \mathcal{V},\ SIC_{i,j}(x,y,z) = DTC_{i,j}(x) + CTC_j(y) + DTC_{j,i}(z)$$

where $x$ is the size of the query to the service, $y$ is the number of computation cycle needed by the task, and $z$ is size of the response.

The relevance of this cost is far from being obvious. Indeed, it assumes that *CTC* and *DTC* are comparable (from the same magnitude)... But this assumption has no insurance. Nevertheless, we will see that even without this insurance, our solution presents accurate results.

**Integration into NDS:** The important aspect is that any new compound metric can be integrated in our service in the same way as raw metrics. It can use both raw metrics and other compound metrics. Here is the declaration of *DTC*:

```
<parameter>
   <name>
      compoundMetric
   </name>
   <value>
      NWSDTC
      Data Transfer Cost based on NWS measurements
      Parameters: %P1, %P2: src and dest; %P3: size of the data
      3*( NWSlatencyTcp(%P1,%P2) + NWSlatencyTcp(%P2,%P1) )
       + %P3/NWSbandwidthTcp(%P1,%P2)
      3
      double
   </value>
</parameter>
```

### 3.3 NDS Queries

The NDS queries are composed of three parts:

**Sources and Destinations:** The set of hosts involved in the decision. The sources represent the clients and the destinations represent the resources.

**Task Properties Set:** The set of parameters involved in the computation of the distances that are not pure metrics. It is just a set of named values:

$$TPS = \{< name[= value] >, ..., < name[= value] >\}$$

**Distance Function:** The real-valued function that gives the final value of the distance between two nodes. We note it:

$$df : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$$

It is a nonlinear combination of observations of metrics and values in $TPS$. This function is the core of the distance computation. It must be relevant to the aspects the user wants to assess. Generally, its result must tend to zero when the reflected performances tend to the perfection, as distances generally represent *costs* and algorithms generally tend to minimize it.

**Example of query for the selection of a matrix multiplication service**
.

One service of the host `alice.dummy.fr` must select between two available services ran on the hosts `bob.dummy.fr` and `carol.dummy.fr`.

– The source is {`alice.dummy.fr`}, the destinations are {`bob.dummy.fr`, `carol.dummy.fr`}.
– The properties of this task are $n$, $m$ and $p$ where $n \times m$ and $m \times p$ are the size of the multiplied matrices. For instance, the $TPS$ should be:

$$TPS = \{< n = 1000 >,\ < m = 500 >,\ < p = 2000 >\}$$

– Then, according to the matrix multiplication algorithm and the size of the result matrix, the distance function should be:

$$df(i, j) = SIC_{i,j}(n \times m + m \times p,\ n \times m \times p,\ n \times p)$$

The corresponding client side code in JAVA is:

```
// Declaration of the invocation message
CSrcDestTaskPropertiesDistanceFunction ftsidcf
  = new CSrcDestTaskPropertiesDistanceFunction();

// Declaration of sources and destinations
String sF1[]={"alice.dummy.fr"};
String sT1[]={"bob.dummy.fr, carol.dummy.fr"};
CSrcDestSet fts = new CSrcDestSet(sF1, sT1);
ftsidcf.setSrcDestSet(fts);
```

```
// Declaration of TPS
CNamedValue ids[] = new CNamedValue[3];
ids[0] = new CNamedValue("n", 1000);
ids[1] = new CNamedValue("m", 500);
ids[2] = new CNamedValue("p", 2000);
ftsidcf.setTaskProperties(new CTaskProperties(ids));

//Declaration of df
ftsidcf.setDistanceFunction("SIC(n*m+m*p, n*m*p, n*p)");

// Retrieving of the distances
CDistance ftd[] = ndsPT.getDistances(ftsidcf).getDistance();

int bestDest;
if ( ftd[0].getValue() < ftd[1].getValue() ) bestDest=0;
else bestDest=1;
System.out.println(ftd[i].getSrc()+" is closer to "+ ftd[i].getDist());
```

Please note that this code is above all verbose due to WSDL specification and that it will be no more complex with a very complex distance: The declaration of $TPS$ and *df* values are rather simple actually and the rest can be simply pasted.

## 4 Experimentation

These experimentations illustrate how NDS can be used as an actual decision-making support for any given service, leading to optimal decisions in the three problems presented before: Replication, Placement and Selection.

### 4.1 Context

Our experimentations are made on the Grid5000 [15] with 19 nodes from 7 sites distributed all over France: `Lyon` (3+3 nodes), `Rennes` (2 nodes), `Orsay` (2 nodes), `Sophia-Antipolis` (2+2 nodes), `Toulouse` (2 nodes), `Nancy` (2 nodes) and `Lille` (1 nodes). The sites `Lyon` and `Sophia-Antipolis` have nodes from two different clusters, with slightly different capacities.

$\mathcal{V} = \{$ `node-36.lille`,
`node-2.lyon`, `node-31.lyon`, `node-37.lyon`,
`sagittaire-13.lyon`, `sagittaire-18.lyon`, `sagittaire-44.lyon`,
`grillon-20.nancy`, `grillon-39.nancy`,
`gdx0039.orsay`, `gdx0077.orsay`,
`paravent74.rennes`, `paravent99.rennes`,
`helios51.sophia`, `helios52.sophia`,
`node-56.sophia`, `node-85.sophia`,
`node-25.toulouse`, `node-6.toulouse` $\}$

Our experimentation are based on $SIC$ values, which is the compound metric closest to services concerns. $SIC$ takes three parameters. As we want to remain as general as possible, these parameters have been set to all the possible combinations of the values in $\{1000,\ 100000,\ 10000000\}$.

For instance $SIC(1000,\ 100000,\ 10000000)$ represents the cost of the invocation of one service taking 1000 bytes of input, having 100000 cycles of computation and having 10 Mbytes of output, just like a simple DB select query: the query itself is short, its treatment is not very CPU expensive, but the returned data is very large.

Thus, all services behaviours have been tested (few IO but a lot of computation, a lot of input but few output, etc). This allows to analyse the accuracy of our solution without being limited to one specific case.

The objective of the decisions is to optimize the response time. The replication and placement are optimized assuming that the distribution of the requests over the hosts is uniform and that the services can be placed on any of them.

### 4.2 Validation

The validation of the decisions proposed by NDS is done by a comparison the real times needed to achieve the corresponding operations. We have implemented a *Fake Service* to obtain these times. It is placed on each hosts and takes three parameters corresponding the parameters of $SIC$: $x$, $y$ and $z$. The *Fake Client* sends $x$ random bytes to the *Fake Service*; Then this one makes $y$ divisions of double typed variables; Finally, $z$ random bytes are returned to the client. The time needed to achieve this operation represents the time needed to achieve a real operation with a real service having the same kind of behaviour. The time needed to complete the whole operation is designed by *Experiment Invocation Time*.

As we use 19 nodes, the matrices we use ($L$, $BWS$, $cf$, ...) are too large to be presented here (and to be comfortably readable). Consequently, the results are presented within figures.

### 4.3 K-centers problem

Actually, the problem of placement is a classical graph problem, namely the "k-centers problem". This problem can be defined as:

Given a set $V$ of points in a metric space endowed with a metric distance function $df$, and given a desired number $k$ of resulting clusters, partition $S$ into non-overlapping clusters $C_1, \ldots, C_k$ and determine their "centers"

$$\mu = \{\mu_1, \ldots, \mu_k\} \subset V \text{ so that } score = \sum_{i \in \mathcal{V}} min_{j=1}^{k}\ df(i, \mu_j) \text{ is minimized.}$$

In our scenario, $k$ is the number of instances of the service we want to place, while $\mu_1, \ldots, \mu_k$ are their optimal location. We show in [16] how properties of the distance produced by NDS can be validated to ensure that we have a "metric distance".

NDS imbeds a trivial algorithm to solve the k-centers problem and computes the "score" of each possible solution.

## 4.4 Replication

The first of our problems is to decide how many instances of the service is optimal. This is achieved by comparing the best result of the k-servers algorithm for each possible value of $k$. The figure 2 shows these results for two different $TPS$:

(1) a CPU expensive configuration: $x = 1000$, $y = 10000000$, $z = 1000$
(2) a communication expensive configuration: $x = 10000000$, $y = 1000$, $z = 10000000$

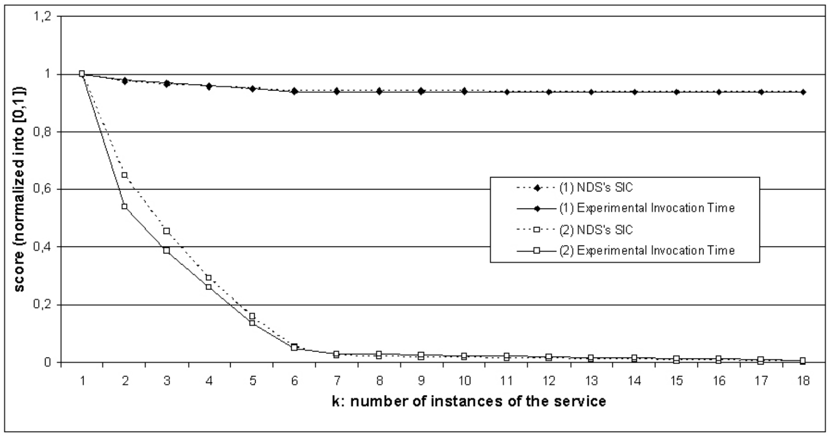

**Fig. 2.** K-servers best scores according to $k$

One can make some observations on the figure 2:

– The results based on NDS's $SIC$ distances are very representative of the real results in both cases.
– The optimal number of instances of communication expensive services is 7. More replicas do not lead to any improvement. This corresponds to the number of sites.
– CPU expensive services (or rather communication cheap services) do not need to be replicated.

Actually, the interesting aspect here is that one can decide to deploy, for instance, only 4 services in the case (2) because it represents a sufficient improvement in the given situation. Moreover, in any situation the improvement from $k = 7$ (the number of sites) to $k = 6$ might not be enough important to justify another deployment. This is due to the fact that `Lille` have only one host and thus does not need one dedicated service.

## 4.5 Placement

The second of our problems is to decide where to place the different instances. The table 1 shows the decisions indicated by NDS and the best experimental decisions in the two cases presented before for $k = 1$ in the case (1) and $k = 6$ in the case (2). The accuracy is computed by the difference of the experimental scores of the solution proposed by NDS and of the real best solution, normalized to obtain a percentage.

**Table 1.** placement decision accuracy

| case | accuracy | $\mu_i$ by NDS's $SIC$ | Experimental $\mu_i$ |
|---|---|---|---|
| (1) | 100.00% | `sagittaire-18.lyon` | `sagittaire-18.lyon` |
| (2) | 99.68% | `node-25.toulouse`, `node-56.sophia`, `paravent99.rennes`, `gdx0077.orsay`, `grillon-39.nancy`, `sagittaire-44.lyon` | `node-6.toulouse`, `helios51.sophia`, `paravent74.rennes`, `gdx0077.orsay`, `grillon-20.nancy`, `node-31.lyon` |

One can make some observations based on the table 1:

- The decisions proposed by NDS are very accurate.
- The differences concerns hosts inside the same site. Consequently, they can not be considered as real “mistakes”.
- The site excluded from the deployment is `Lille`, as expected.

More globally, this placement decision method has been tested for all the possible combinations of $TPS$ and all $k \in [1, 19]$. Our method achieves a global mean accuracy of 94.26%. It ranges from 100.00% to 21%, which might seem to be rather bad. Actually, all the worst accuracies concerns $TPS$ involving the lowest size 1000. More especially $x = y = z = 1000$ shows the very worst results. In fact, the execution time of this kind of task is around $5ms$: at this level, even the timing is not accurate and results are in fact quite random. Actually, distribution optimizations are not relevant at such a granularity.

## 4.6 Selection

The third of our problems is to decide which of the instances must be used for a given invocation from a given client host. This problem can be solved by sorting the $SIC$ distances from the requesting host to the target instances: The lowest distance points out the best one to invoke. The figures 3 and 4 show the ranked $SIC$ values and the corresponding experimental invocation times from `paravent74.rennes` to all the hosts in both cases (1) and (2).

One can make some observations:

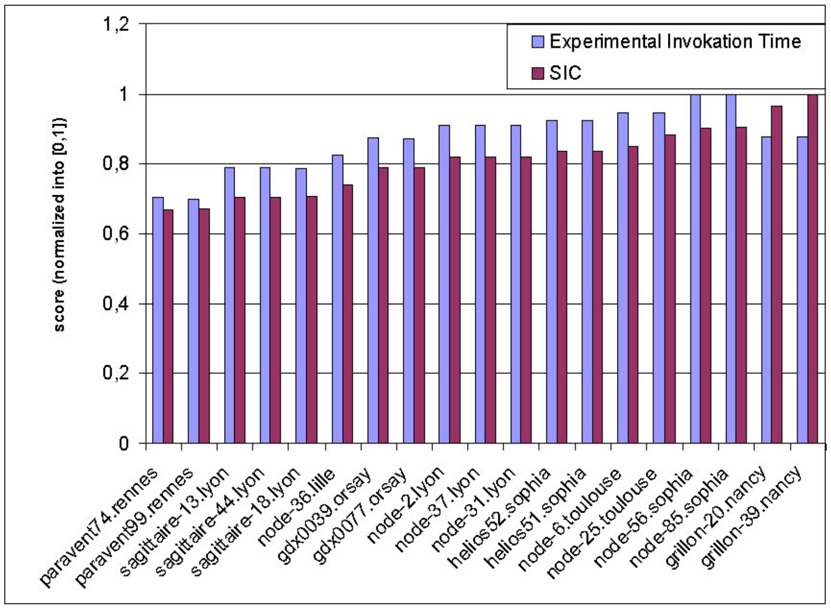

**Fig. 3.** Experimental Invokation Time and NDS's SIC distances from `paravent74.rennes` in case (1)

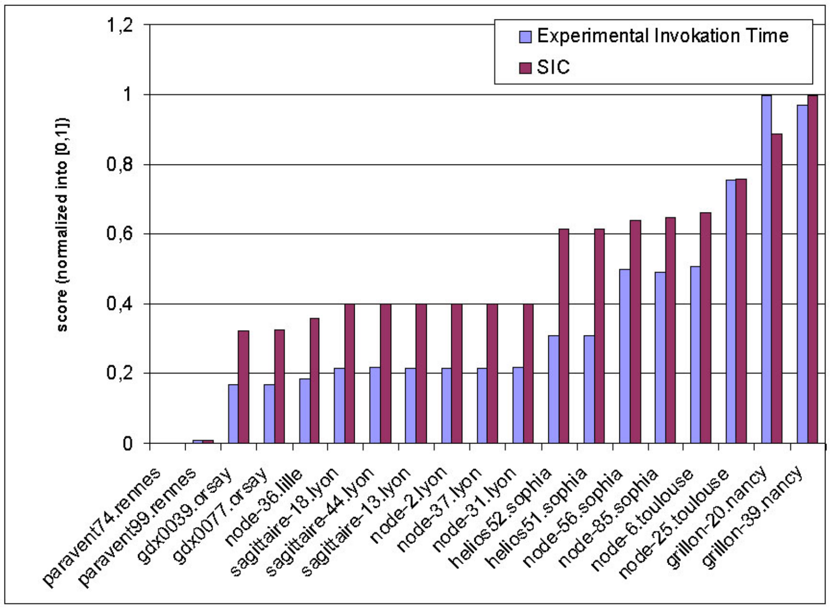

**Fig. 4.** Experimental Invokation Time and NDS's SIC distances from `paravent74.rennes` in case (2)

– The ranking given the $SIC$ values corresponds very well to the real ranking. More especially, the best server hosts are perfectly selected.
– With CPU expensive services (1), the results are very close. This is due to the homogeneity of the hosts in term of computation capability. In spite of this, the ranking based on $SIC$ are very well representative of the real performances.
– Moreover in this case (1), the hosts from `Nancy` are under-rated by NDS: This might be due to $CTC$ which considers CPU frequencies only, whereas other characteristics are influencing, like cache size or bus frequency. Although $CTC$ is sufficient in the general case, another more detailed compound metric can be declared if necessary, as soon as the raw metrics are available.
– With communication expensive services, the results are more varied and the ranking near to the perfection.
– The ranking is not the same in the two case (1) and (2). For instance, the hosts from `Orsay` are better ranked in the case (2) than in the case (1): This is because the hosts from `Orsay` are less powerfull but better connected than the hosts from `Lyon`. This is perfectly detected by our method whereas this kind of situation is hardly undetectable by either intuitive means or the existing decision supports.

## 5 Discussion

First of all, one can make some observations about the compound metrics: Actually, even if their accuracy has been shown, they are quite simple. For instance $CTC$ implies that the behaviour of the task is deterministic and predictable. As a matter of fact, a lot of algorithms, and more particularly approximation ones, are not so predictable. Nevertheless, our method allows deciding the best behaviour to model. As soon as one has minimal information on the complexity of the used algorithm, one can decide to assess a mean (or maximum or minimum) cost and to model it. On the other hand, one can have complete information about the algorithm behaviour, but this one is complex and involves several parameters. In this case, it is absolutely possible and easy to model it into an NDS's distance, even with non-functional parameters like financial aspects. In our future work, we will investigate whether it is possible to use Web Service Distributed Management to define the distance function to use and whether these distance functions can be built automatically.

Second, one can make some observations about our experimentations: They may seem to be very peculiar due to the set of assumptions (uniform requests pattern, sizes fixed to 3 static values). Actually, these parameters do not influence the relevance of the produced distance, but rather the feature of the used k-centers algorithm. Whereas taking into account varied request probabilities for each host does not seems to be very difficult, tackling with a wide range of continuous sizes might be difficult. This implies to be able to compute a primitive of the distance function and is an important part of the future works.

Moreover, Grid 5000 is quite homogeneous both in term of computing and communication capacities. Actually, decision-making are easier in very heterogeneous infrastructures because the different decisions lead to very different performances. The more the infrastructure is homogeneous, the more the performances are close, making the distribution decisions very difficult to be optimal and increasing the risk of mistakes. Indeed, as our decision-making support can lead to optimal decisions in homogeneous environment, there is no doubt that it will lead to optimal decisions in heterogeneous environment.

Third, another observation is about the cost of using our service: As it uses the measurements from monitoring systems like MDS and NWS, it might be considered as expensive to use as them. Actually, the invocation of NDS itself is not expensive, especially compared to the cost of the underlying monitoring systems. Nevertheless, NDS is much more usable than classical monitoring systems, particularly the ones like NWS which are not integrated to the grid. Moreover, NDS is much closer to the concerns of the administrators and developers. Thus it is more in position to be used by all the other services at each level of service oriented architectures. As a matter of fact, one of the advantages of NDS is that it enhances the profitability of the used monitoring systems. Moreover, this usability has been reinforced by the development of a JAVA GUI: While developers can invoke NDS through the classical WS API as shown is section 3.3, the administrators can use the GUI to make the replication and placement decisions.

Finally, we have only tested one-to-one task. Obviously, our method can be used to optimize complex task involving several hosts like, for instance, content adaptation chains. But although the distances we produce are comparable and rankable, it is possible that the sum of several distances along a path leads to an accuracy loss.

## 6 Conclusion and Future Work

We have presented a novel distribution decision-making support service called the Network Distance Service. This service has been designed to allow the inner activities of grids to fit the requirement of pervasive computing: heterogeneity, dynamicity and unpredictability. The importance of these requirements has been presented on an emergency use case involving critical time-constrained and intensive computations. Existing solutions do not fit onto this kind of context. NDS enhances monitoring tools by providing distances adapted to a large range of applications. It is meant to be profitable, usable, adaptable and accurate. The embedment of a k-centers algorithm allows easily solving the main problems of distribution: Replication, Placement and Selection. The accuracy and benefits of our approach have been proved with some experimentation showing a concrete use in a real grid environment.

The future works are threefold. First we will investigate how we can compute the primitives of the distance functions in order to get independent from the values of $TPS$. Second, we will investigate the accuracy of our solution in many-to-many interactions. Finally, we will investigate how the compound metrics can be embedded into Web Service Distributed Management and how the distance functions can be built automatically.

## References

1. Hinge, V., Joshi, A., Finin, T., Kargupta, H., Houstis, E.: Towards a pervasive grid. In: 17th International Parallel and Distributed Processing Symposium. (2003)
2. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. The International Journal of High Performance Computing Applications **15** (2001) 200–222
3. Pierson, J.M.: A pervasive grid, from the data side. RR LIRIS-08-2006 (2006)
4. Lowekamp, B., Tierney, B., Cottrell, L., Hughes-Jones, R., Kielmann, T., Swany., M.: A hierarchy of network performance characteristics for grid applications and services. In: Global Grid Forum. (2004)
5. Globus_Alliance: (Monitoring and discovery service) http://www.globus.org/mds/.
6. et al., A.C.: The relational grid monitoring architecture: Mediating information about the grid. Journal of Grid Computing (2004)
7. Truong, H.L., Fahringer, T.: Scalea-g: a unified monitoring and performance analysis system for the grid (2004)
8. Wolski, R., Spring, N.T., Hayes, J.: The Network Weather Service: a distributed resource performance forecasting service for metacomputing. Future Generation Computer Systems **15** (1999) 757–768
9. Karlsson, M., Mahalingam, M.: We need replica placement algorithms in content delivery networks? In Boulder, ed.: 7th International Web Content Caching and Distribution Workshop (WCW 2002). (2002) 117–128
10. Qiang Xu, J.S.: Automatic clustering of grid nodes. In IEEE/ACM, ed.: 6th IEEE/ACM International Workshop on Grid Computing, Seattle, IEEE/ACM (2005) 227–233
11. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global Internet host distance estimation service. IEEE/ACM Transactions on Networking **9** (2001) 525–540
12. Ng, T.S.E., Zhang, H.: Predicting internet network distance with coordinates-based approaches. Volume 1 of Proceedings IEEE INFOCOM 2002., (IEEE Computer Society) 170–179
13. Coates, M., Castro, R., Nowak, R., Gadhiok, M., King, R., Tsang, Y.: Maximum likelihood network topology identification from edge-based unicast measurements. Volume 30, 1 of SIGMETRICS Performance Evaluation Review., (ACM Press)
14. Faerman, M., Su, A., Wolski, R., Berman, F.: Adaptive performance prediction for distributed data-intensive applications. In: ACM/IEEE SC99 Conference on High Performance Networking and Computing, Portland, OR, USA (1999)
15. Grid5000. (https://www.grid5000.fr/)
16. Gossa, J.: Evaluation of network distances properties by nds, the network distance service. In: 3th International Workshop on Networks for Grid Applications (GRIDNETS'06), IEEE/Create-Net (2006)