Machine Learning Applications in Grid Computing

George Cybenko, Guofei Jiang and Daniel Bilar

Thayer School of Engineering
Dartmouth College
Hanover, NH 03755, USA
gvc@dartmouth.edu, guofei.jiang@dartmouth.edu

Abstract

The development of the World Wide Web has changed the way that we think about information. Information on the web is distributed, updates are made asynchronously and resources come and go online without centralized control. Global networking will similarly change the way we think about and perform computation. *Grid computing* refers to computing in a distributed networked environment in which computing and data resources are located throughout a network. Grid computing is enabled by an *infrastructure* that allows users to locate computing resources and data products dynamically during a computation. In order to locate resources dynamically in a grid computation, a grid application program consults a broker or matchmaker agent that uses keywords and ontologies to specify grid services. However, we believe that keywords and ontologies cannot be defined or interpreted precisely enough to make brokering and matchmaking between agents sufficiently robust in a truly distributed, heterogeneous computing environment. To this end, we introduce the concept of functional validation. Functional validation goes beyond the symbolic negotiation level of brokering and matchmaking, to the level of validating actual functional performance of grid services. In this paper, we present the functional validation problem in grid computing and apply basic machine learning theory such as PAC learning and Chernoff bounds to solve the sample size problem that arises. Furthermore, in order to reduce network traffic and speedup the validation process, we describe the use of Dartmouth D'Agents technology to implement a general mobile functional validation agent system which can be integrated into a grid computing infrastructures as a standard grid service.

1. Introduction

There are currently several efforts are underway to build *computational grids*, such as Globus [1], Infospheres [2] and the DARPA CoABS [3] . These projects are developing the fundamental technology that is needed to build computational grids, execution environments that enable an application to integrate geographically distributed instruments, sensors, data products, displays, and computational and information resources. Such grid computations may link tens or hundreds of these resources. The vision is that these grid infrastructures will connect multiple regional and national computational grids, creating a universal source of pervasive and dependable computing power that supports dramatically new classes of applications.

Acknowledgements: This work was partially supported by Air Force Office of Scientific Research grants F49620-97-1-0382, National Science Foundation grant CCR-9813744 and DARPA contract F30602-98-2-0107.

A fundamental capability required in such a grid is a directory service or broker that dynamically matches user requirements with available resource. On the web, this capability is provided by search engines that index web pages and implement retrieval services. Whereas humans are typically the consumers of web information, grid agents will be the producers and consumers of grid resources with humans occasionally steering or interpreting the computations.

A grid computation that needs a computational service, such as for example the solution to structured linear system or a multidimensional Fourier transform, will locate the required service by consulting a distributed object request broker or a matchmaker service. For example, CORBA is an infrastructure for implementing distributed applications and provides a broker as a key component [4]. Jini[5] is designed for deploying and using services in a network and enables the construction of dynamic, flexible, and robust systems from independent distributed components and it also can provide a nice lookup service. An object request broker (ORB) not only locates a component or object that performs the required service but also mediates communication between the client and the service. In standard terminology, a matchmaker is an ORB with reduced capability. A matchmaker merely locates remote services but does not mediate communications between client and server agents. In the matchmaker framework, a client and the remote service that it invokes communicate directly once their locations are made known by the matchmaker service.

2. Grid Computing Services and Functional Validation

A schematic of part of a grid computation is shown in Figure 1. Remote service providers will publish their services' catalog on ORB or matchmaker agents. When clients need some computing services, they will make a request to an ORB or matchmaker agent. The request is made based on some keywords and possibly parameter lists for invoking the remote computing resources. The ORB or matchmaker consults its service catalog and returns with several remote service candidates and specifies their returning value structures. Then clients can negotiate with these service candidates and ask for the appropriate computing service. Just like web search engines, ORB and matchmaker agents will use keywords and ontologies to specify remote computing services. Ontologies specify a domain and keywords specify functionality within that domain. For example, ontologies are envisioned for signal processing, ocean modeling, image processing, weather modeling and so on. Within an ontology, keywords such as "Fourier transform" and "linear system solver" will have possibly domain specific meanings. Several systems have been proposed for implementing such ontological matching [6][7].

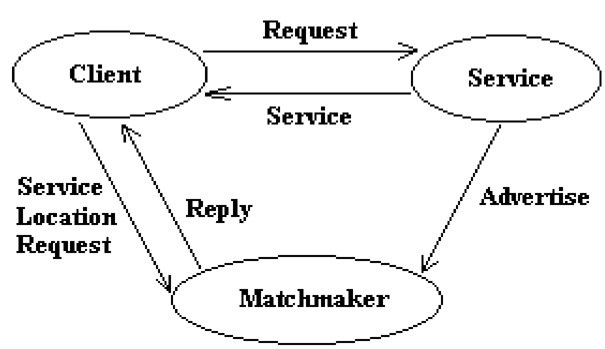

Figure 1: The prototype of grid computing services

Note however, there are literally dozens of different algorithms for implementing Discrete Fourier Transforms [8]. Different algorithms make different assumptions about the symmetries of the input vector and order the output in a variety of ways. Some algorithms may be only able to transform the input vector of certain dimensions. The actual numerical computations carried out vary from algorithm to algorithm so that different round-off errors are accumulated leading to slightly different answers. Moreover, different numerical implementations of some basic computations in an can lead to different computational speed, different accuracy and so on. The same is true of linear system solvers, other numerical algorithms and data products. In some complicated computational tasks, the possible situations are more challenging. For example, there are many different system modeling algorithms developed for different control systems such as ARMX or ARMAX systems, time variant or invariant systems, noisy or noiseless systems, linear or nonlinear systems, and so on. So in this case it is more difficult for keywords and ontologies to describe precisely the real functionality of the service provider's algorithms.

Keywords and ontologies cannot be defined and interpreted precisely enough to make brokering or matchmaking between grid services robust in a truly distributed, heterogeneous computing environment. Thus matching conflicts will exist between a client's requests and a service provider's responses. Some form of functional validation of computing resources will be required.

Functional validation means that a client presents to a prospective service provider a sequence of *challenges*. The service provider replies to these challenges with corresponding *answers*. Only after the client is satisfied that the service provider's answers are consistent with the client's expectations is an actual commitment made to using the service. This is especially important in mission critical applications. In fact we can find the same idea of functional validation in our daily lives. For example, a demo is often used to show the functionality of some software.

Our ongoing research on agent-based systems [9] has led us to the conclusion that brokering at the purely symbolic level will not be sufficient to implement truly distributed, heterogeneous multi-agent computing. Two steps are required before agents commit to each other:

1. Service *identification* and *location*;
2. Service *functional validation*.
3. *Commitment* to the service.

These steps are shown in Figure 2. Identification and location will be performed by ORB or matchmaker agents and is already an area of active research. However, functional validation of distributed components and agents is a new subject of research that is essential for the future success of truly heterogeneous, distributed computing grids.

3. Machine Learning Models of Functional Validation

Our approach to functional validation is to allow the client to challenge the service provider with some test cases, x_1, x_2, ..., x_k. The remote service provider offers corresponding responses/answers, $f_R(x_1)$, $f_R(x_2)$, ..., $f_R(x_k)$. The client may or may not have independent access to the correct responses/answers, $f_C(x_1)$, $f_C(x_2)$, ..., $f_C(x_k)$. Depending on the sequence of responses, the client may or may not commit to using (and therefore possibly paying for) the service provided.

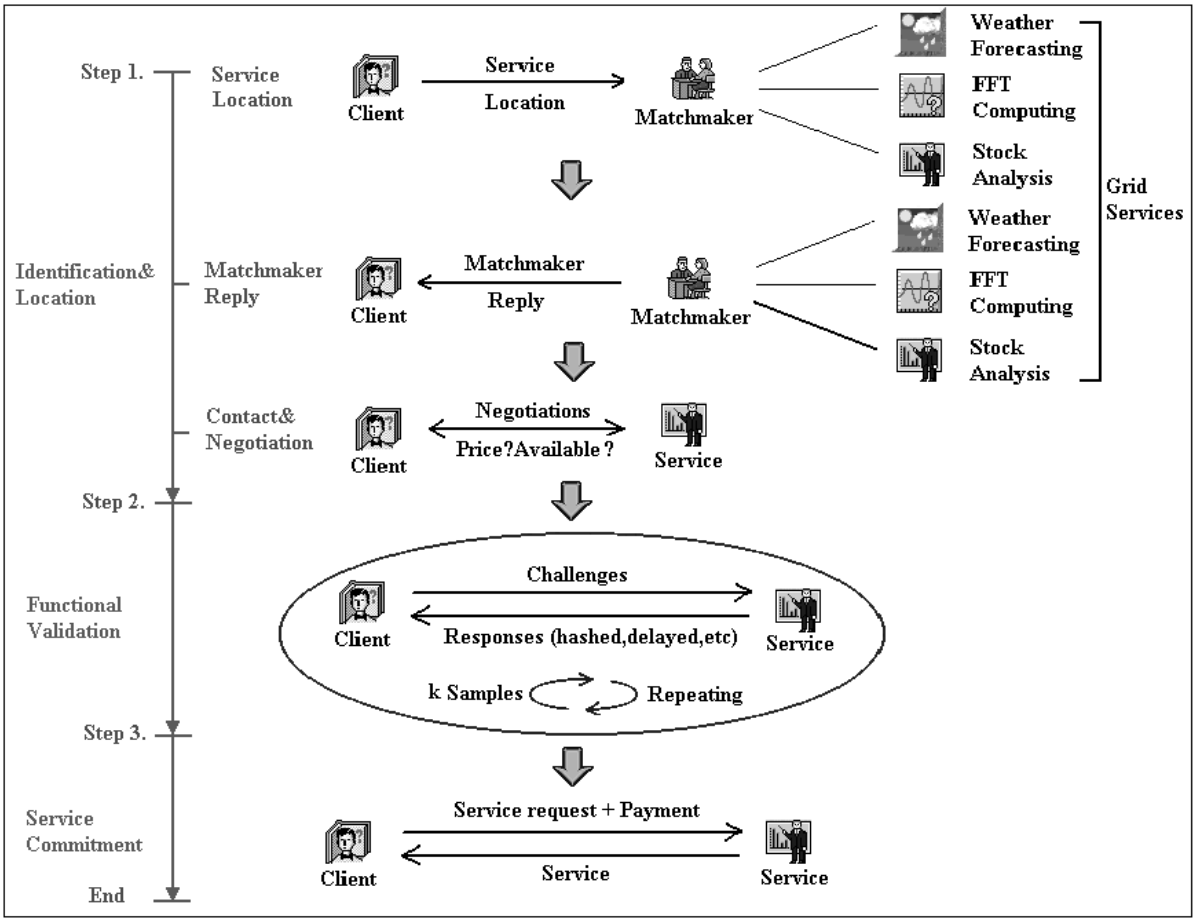

Figure 2: Grid services and functional validation

We will formalize the functional validation program for a computational and data service as follows. Denote the client's calling simulation by ***C*** and the remote service component by ***R***. ***C*** requires the evaluation of a function, $f_C(x)$, where x is the input parameter. Assuming compatibility of input and output parameter structures and types, which has already been checked by the ORB or matchmaker, the remote service is expected to provide $f_R(x)$. Table 1 shows the possible situations that can arise in this problem [10].

Table 1: Possible situations arising in functional validation and their solutions

Client *C*	**Server *R***	**Machine Learning Model**
"knows" $f_C(x)$	provides $f_R(x)$	PAC-learning and Chernoff bounds
"knows" $f_C(x)$	doesn't provide $f_R(x)$	Zero-knowledge proof
doesn't "know" $f_C(x)$	provides $f_R(x)$	Simulation-based and reinforcement learning

In this paper , we discuss the simplest case and assume that the client ***C*** itself has the correct value $f_C(x)$ for *the selected samples* and the candidate service provides responses $f_R(x)$ directly after ***C*** challenges ***R*** with the sample inputs. So the basic question in the functional validation problem is that in order to know whether the service provider can offer the "correct" service, how large should the sample size k be?

4. General Mathematical Framework

In general, we can formalize the functional validation problem and answer it using PAC learning theory. The role of PAC learning in this context is to use as few samples as possible, and as little computation as possible to pick a service (hypothesis) that is, with sufficiently high probability, a close approximation to the desired functionality as expressed by the test samples.

Here assume the input space X is a fixed set, either finite, countably infinite, $[0,1]^n$, or E^n (Euclidean n-dimensional space) for some $n \geq 1$. In the functional validation problem, we are concerned with whether the service provider can offer the "correct" service , so we define a concept to be a boolean mapping $c: X \to \{0,1\}$, with $c(x)=1$ indicating that x is a positive example of c, i.e. the service provider offers the "correct" service for challenge x, and $c(x)=0$ indicating that x is a negative example, i.e. the service provider does not offer the "correct" service for challenge x. A concept class C over X is a collection of concepts (that is, boolean functions) c over X. More precisely, given an error tolerance, $\gamma > 0$, define $c(x)$=1 if $\|f_C(x) - f_R(x)\|_X \leq \gamma$ (where γ is the allowable computational error tolerance) for all x, and define $c(x)$=0 if $\|f_C(x) - f_R(x)\|_X \geq \gamma$.

Let P be a fixed probability distribution on X and assume that examples are created by drawing challenges independently and randomly according to P. Define an index function

$$F(x) = \begin{cases} 1 & if\ \|f_C(x) - f_R(x)\| \leq \gamma \\ 0 & otherwise \end{cases} \tag{1}$$

Then the client can randomly pick $S_m = \{(x_1, F(x_1)), (x_2, F(x_2)), \cdots, (x_m, F(x_m))\}$ m samples to learn a hypothesis $h \in H$ about whether the service provider offers the "correct" service, where H is the hypothesis space and usually is the concept class C itself. Here let $A_{C,H}$ denotes the set of all learning functions $A: S_m \to H$. We claim that $A \in A_{C,H}$ is consistent if it agrees with the samples, that is $h = A(S_m)$. Thus based on the PAC learned hypothesis, the client can conclude whether the service provider can offer the "correct" service with the desired level of confidence.

Now consider the problem of how many samples or challenges are needed to make a decision about whether the hypothesis is a good enough approximation to the real target concept or not. Define the error between the target concept c and the hypothesis h as

$$error(h) = Prob_{x \in P}[c(x) \neq h(x)] \tag{2}$$

where $Prob_{x \in P}[.]$ indicates the probability with respect to the random drawing of x according to P. Then mathematically we can formalize the above problem as follows: How large must the number of challenges, m, be so that

$$Prob^m\{error(h) \leq \varepsilon\} \geq 1 - \delta \tag{3}$$

where ε is the accuracy parameter and δ is the confidence parameter.

Blumer et. al. [11] solved this problem with the following powerful result:

Theorem 1 (Blumer et al.[11]) *Let H be any well-behaved hypothesis space of finite Vapnik-Chervonenkis dimension d contained in 2^X, P be any probability distribution on X and the target concept c be any Borel set contained in X. Then for any $0 < \varepsilon, \delta < 1$, given*

$$m \geq \max\left(\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{13}{\varepsilon}\right) \tag{4}$$

independent random examples of c drawn according to P, with probability at least $1-\delta$, every hypothesis in H that is consistent with all of these examples has error at most ε.

In the above theorem, the Vapnik-Chervonenkis dimension (VC dimension) is a combinatorial measure of concept class complexity which assigns to each concept class C a single number that characterizes the sample size needed to PAC learn C. See its detailed definition in [11].

Thus by determining the boolean concept's VC dimension over X and selecting an accuracy parameter ε and a confidence parameter δ, according to the above theorem , the client can pick m samples to PAC learn a hypothesis which is probabilistically close enough to the real target concept, thereby deciding whether the service agent can offer the "correct" services.

5. Simplified Theoretical Results

PAC learning theory formalizes an ideal mathematical framework for the functional validation problem. However, the above theoretical result can not be practically applied in grid computing easily because it is difficult to compute the VC dimension for generic problems and the sample size bound will be very large in general. We simplify the complexity of the functional validation problem and assume that with regard to some concepts, all test cases have the same probability about whether the service provider can offer the "correct" service. Then Chernoff bounds [12] theory can be used to solve the sample size problem.

Theorem 2 (Chernoff Bounds): *Consider independent identically distributed samples $x_1, x_2, \cdots, x_m$ from a Bernoulli distribution with expectation p. Define the empirical estimate of p based on these samples as*

$$\hat{p} = \frac{\sum_{i=1}^{m} x_i}{m} \tag{5}$$

Then for any $0 < \varepsilon, \delta < 1$, if the sample size

$$m \geq \frac{\ln \delta - \ln 2}{-2\varepsilon^2} \tag{6}$$

then the probability

$$prob^m\left[\left|\hat{p} - p\right| \geq \varepsilon\right] \leq \delta \tag{7}$$

During the functional validation process, the client only continues with the next sample if it knows that the service provider offers the "correct" response for all previous samples. Once the client encounters a negative example, i.e. the service provider offers a wrong reply for the current sample, the client will stop the validation process with this service provider and proceeds to negotiate with other possible service provider candidates. So with regard to this special situation, the client only needs to know how many samples are needed if all tested samples are positive examples (otherwise it just gives up). Thus the empirically estimated $\hat{p}$ should be equal to one. Then because $0 < p < 1$ and $\hat{p} = 1$, with the same sample size m defined in inequality (6), the inequality (7) can be updated by the following inequality:

$$prob^m\left[(1-p)\geq\varepsilon\right]\leq\frac{\delta}{2} \tag{8}$$

Then according to Theorem 2, it is straightforward to have the following corollary:

Corollary 2.1: *For the functional validation problem described above, given any* $0<\varepsilon,\delta<1$*, if the sample size*

$$m\geq\frac{\ln\delta}{-2\varepsilon^2} \tag{9}$$

then the probability

$$prob^m\left[(1-p)\geq\varepsilon\right]\leq\delta \tag{10}$$

In the above inequality, ε is the accuracy parameter which bounds the difference between the estimated probability and the true probability, while δ determines the confidence in this approximation. However, the goal of the functional validation process is to have a high confidence that the service will provide the correct response on the next request. Given a target probability, say *P*, the client agent needs to know how many consecutive positive samples, *m*, are required so that the next request to the service will be correct with probability *P*. We can combine the accuracy parameter, ε, and confidence, δ, together into this one parameter P. Using the definitions of δ and ε from above, it is easy to see that if the inequality

$$P\leq(1-\varepsilon)(1-\delta) \tag{11}$$

is satisfied, then the next sample will have probability P of being correct as well, given the Bernoulli model and other assumptions from above. Note that in inequality (9), ε and δ are both needed to compute the sample size m. So after a client agent selects a P, we still need to figure out how to compute the sample size m under the constraint inequality (11). We can formulate this problem as the following nonlinear optimization problem:

$$m=\min_{\varepsilon,\delta} f(\varepsilon,\delta)=\min_{\varepsilon,\delta}\left(\frac{\ln\delta}{-2\varepsilon^2}\right) \tag{12}$$
$$\text{s.t.: } (1-\varepsilon)(1-\delta)\geq P \text{ and } 0<\varepsilon,\delta<1.$$

The objective function $f(\varepsilon,\delta)$ has two variables. From the constraint inequality, we require

$$\delta\leq 1-\frac{P}{1-\varepsilon} \tag{13}$$

Obviously when δ is larger, the sample size m is smaller. So if we choose $\delta=1-\frac{P}{1-\varepsilon}$ and think about the constraint $0<\varepsilon, 1-\frac{P}{1-\varepsilon}<1$, we can transfer the above two-dimensional function optimization problem to the following one-dimensional function optimization problem:

$$m=\min_{\varepsilon} g(\varepsilon)=\min_{\varepsilon}\left(\frac{\ln(1-\frac{P}{1-\varepsilon})}{-2\varepsilon^2}\right) \tag{14}$$
$$\text{s.t. } 0<\varepsilon<1-P.$$

So when ε approaches the edge point 0 or $1-P$, the function $g(\varepsilon)$ will approach positive infinity. Unfortunately, we have not obtained a closed-form analytical solution because of the log function. However, we use elementary nonlinear function optimization methods to compute the minimum sample size, m, such as the steepest descent method. Figure 3 shows how the sample size changes with the given probability P.

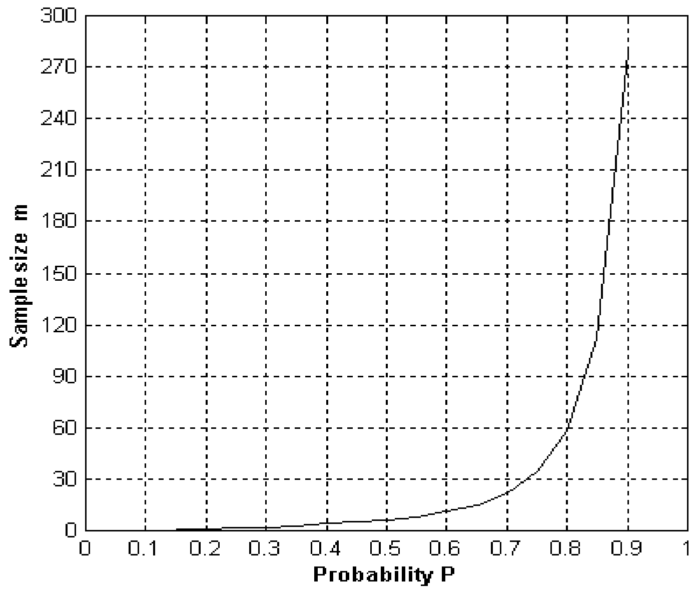

Figure 3: sample size vs. probability P curve

6. Mobile Functional Validation Agent

In addition to the above analytic results, we have implemented basic functional validation protocols for a grid computing infrastructure. We have implemented a test bed for the validation protocols, using the Dartmouth D'Agents system [9][13] to create mobile functional validation agents (MFVA) for grid computing.

D'Agents is a mobile-agent system developed at Dartmouth College. A mobile agent is a program that can migrate from machine to machine in a heterogeneous network. The program can choose when and where to migrate. It can suspend its execution at an arbitrary point, transport to another machine and resume execution on the new machine. Mobile agent have several advantages over the traditional client/server model: mobile agents consume fewer network resources and communicate efficiently; mobile agents do not require a continuous connection between machines; mobile agents hide the communication channels but not the location of the computation. So people can use mobile agent systems as a single, unified framework to implement a wide range of distributed applications easily and efficiently.

Consider the functional validation process for grid computing. In a traditional setting, during this process a client submits sample data, x, to the service provider and waits for the reply $f_R(x)$ from the remote service provider. For some difficult computing task, it may take the service provider a long time to compute the result. After the client gets the response $f_R(x)$, it must compare $f_C(x)$ with $f_R(x)$ to see whether the service provider offers an acceptable result for the sample. If $f_R(x)$ is acceptable, the client will send out the next sample data and repeat the above procedure. If $f_R(x)$ is unacceptable, the client will close the current communication channel and open a new one with another possible service candidate and then repeat the same validation process. So during this process, the client and the service provider can generate significant network traffic.

Using the D'Agents system, we have created a mobile functional validation agent to make the whole functional validation process more efficiently. Figure 4 shows how the mobile functional validation agent works in the grid computing infrastructure. After the ORB

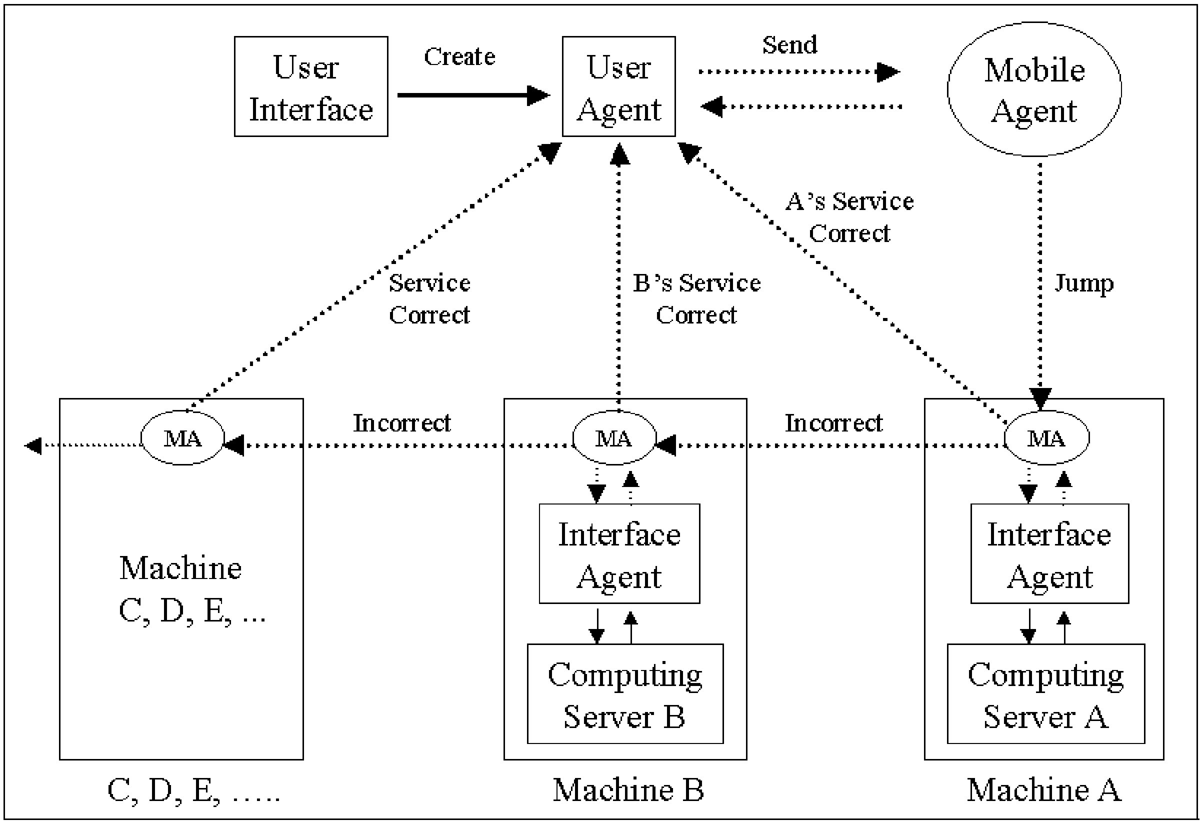

Figure 4: mobile functional validation agent

or matchmaker returns the machine list of possible service provider candidates, the client uses the simplified theoretical results in section 5 to compute the sample size m suitable for a desired probability of correctness. Then it creates a mobile agent that encapsulates the machine list, the m sample list and the client's actual computing request. The mobile functional validation agent jumps to the first machine that is listed on the machine list and executes the functional validation process on that local machine. During the validation process, if the mobile agent gets an unacceptable reply from the local service provider, it will autonomously give up this service provider, jump to the next machine on the machine list and initiate a new validation process on that machine. Otherwise it will continue to submit samples to the service provider and repeat the validation procedure. If it continuously gets correct responses for all m samples, it will ask for the client's actual computing service from the service provider. After it gets the result, it will autonomously jump back to the client machine and return the result to the client.

So using the mobile functional agent, the functional validation process is executed on the service provider machine and all communication is done between programs that are running on the same machine. This reduces network traffic and makes the validation process faster and more robust. Moreover, without the client's involvement, a mobile agent can autonomously complete the whole validation process and return the correct result to the client.

Because different service providers have different communication protocols, in Figure 4, interface agents are used to translate these protocols to a standard agent communication protocol. Our mobile functional validation agent system is a very general and standard system. For different computing tasks, a client only needs to change the sample lists in the user agent program and a separate result comparison class. So we believe this general mobile functional validation agent system can be integrated into grid computing infrastructures as a standard service, such as the DARPA CoABS grid infrastructure.

7. Conclusions

Grid computing is an emerging infrastructure that will fundamentally change the way we think about and use computing. In a grid computing infrastructure, a broker or matchmaker agent will use keywords and ontologies to specify grid services. However, keywords and ontologies cannot be defined and interpreted precisely enough to make brokering or matchmaking between resource agent services sufficiently robust in a truly distributed, heterogeneous computing environment. This creates matching conflicts between client's requested functionality and grid service's actual functionality. Functional validation is proposed and studied in this paper. Some machine learning theory is applied to solve the sample size problem that arises in our approach. Furthermore we have used the Dartmouth D'Agents technology to implement a general mobile functional validation agent system which can be integrated into grid computing infrastructures as a standard grid service.

References

[1] Foster, I., and Kesselman, C., 1998. *The Grid Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, San Francisco, California.

[2] http://www.infospheres.caltech.edu

[3] http://coabs.globalinfotek.com

[4] Natan R.B.,1995. *Corba-A Guide to Common Object Request Broker Architecture*, McGraw-Hill, New York.

[5] Edwards, W. K. , 1999. *Core Jini*. Prentice Hall.

[6] http://logic.Stanford.edu/kif/specification.html

[7] http://www.cs.umbc.edu/kqml/

[8] Brigham, E.O., 1988. *The Fast Fourier Transform and Its Applications,* Prentice Hall, Englewood Cliffs, New Jersey.

[9] http://actcomm.dartmouth.edu and http://www.cs.dartmouth.edu/~agent/

[10] Cybenko, G. and Jiang, G., 1999. *Matching conflicts: Functional validation of agents.* 1999 AAAI workshop on Agents' Conflicts, pp. 14-19, Orlando, Florida.

[11] Blumer, A. et al., 1989. *Learnability and the Vapnik-Chervonenkis Dimension*. Journal of the Association for Computing Machinery 36(4):929.

[12] Kearns, M.J., and Vazirani, U.V., 1994. *An Introduction to Computational Learning Theory,* The MIT Press, Cambridge, Massachusetts.

[13] Kotz,D., Gray,R., Nog,S., Rus D., Chawla,S. and Cybenko,G. 1997. *Agent Tcl: Targeting the needs of mobile computers*. IEEE Internet Computing, 1(4):58.