51 EXPERIMENTS ON GRID COMPUTING FOR VE-RELATED APPLICATIONS

Fábio R. Pinheiro; Ricardo J. Rabelo
Federal University of Santa Catarina
GSIGMA – Intelligent Manufacturing Systems Group
Florianópolis (SC), Brazil
{fjrp, rabelo}@das.ufsc.br

Companies have made invested increasingly in computing resources upgrades but this starts to be an obstacle to be supported by them and hence can hazard innovation activities. Most of computers are however underutilized at the companies so they could be used by external users if appropriate platforms could find them and manage them properly. This work aims at presenting some exploratory results of using Grid Computing as a supporting technology both to allow a company to use the computing resources of the other members of the Virtual Enterprise and to allow a flexible and smart selection of the most suitable resources to have access. A prototype has been developed and its results are discussed.

1. INTRODUCTION

Micro, Small and Medium-size Enterprises (MSMEs) have been facing enormous difficulties to acquire and to maintain enterprise systems, e.g. ERP and SCM/Virtual Enterprises (VE) management systems, regarding the required investment in technological, computing and human resources as well as in internal organization (Europe-EU, 05). This is particularly critical as around 99% of companies in Europe are formed by MSMEs and they are more and more dependent on investments in ICT to leverage their innovation and hence their competitiveness (Europe-EU, 05).

However, it is a paradox to see that, if from one hand enterprises have tried to increase their investments in ICT, from another hand the existing computing resources at the enterprises stay idle most of the time. According to IBM, desktop machines are busy less than 5% of the time (Berstis et al., 02).

A VE is defined as a dynamic, temporary and logical aggregation of autonomous enterprises that cooperate with each other as a strategic answer to attend a given opportunity or to cope with a specific need, and whose operation is achieved by a coordinated sharing of skills, resources and information, enabled by computer networks (Rabelo et al., 04). The VE paradigm has been seen as a strategic positioning of MSMEs in order to compete in the global market as stronger alliances.

The approach investigated in this work is based on the principle that that sharing of resources in a VE scenario has been so far exploited in a limited way. Actually, the fact of being a VE, i.e. a group of enterprises which are very much interested that every one can accomplish its business process in the most efficient way so that the whole team can make better profits, is underutilized. The proposed idea is to allow VE members to share the computing resources one of another and even applications one of another, so saving many investments and using the resources more rationally.

This corresponds to another level of cooperation, where other dimensions of virtualizations (e.g. equipments, software, people, etc.) can emerge and work concurrently with the business process execution. This cooperation is important as it strengthens partners' relationships and provides additional elements for the creation of trust building among VE's partners.

Grid computing (Foster, 01) seems to be a very promising and powerful technology to handle this scenario as it makes possible that interconnected and heterogeneous resources can be accessed, used and managed in a controlled and coordinated way seamlessly. Most of Grid applications has focused on high performance processing (Genoma, spatial applications, oil prospecting, etc.) requiring clusters of computers. This work, instead, focuses on the flexibility of resources allocation aiming at using idle computing resources.

This paper aims at discussing some experiments on Grid in the context of VE. It is organized as follows: Section 2 presents an overview about Grid Computing. Section 3 describes the proposal of this work. Section 4 shows some results of a software prototype. Section 5 discusses the results and points out next steps of this work.

2. GRID COMPUTING

The technology of Grid Computing offers a new approach for distributed systems, but with the difference of focus in large-scale resource sharing. A computational Grid is a hardware and software infrastructure – usually called as *platform* – that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities (Foster, 01). This is not primarily concerned with file exchange but rather with direct access to computers, software, data, and other resources, as it is required by a range of collaborative problem-solving. This sharing is highly controlled, with resource providers and consumers defining clearly and carefully what is shared, who is allowed to share, and the conditions under which sharing occurs.

There is a good number of Grid platforms being developed over the world, with different levels of maturity, and that have been applied on several areas. Examples of stable platforms are: Globus Toolkit (Globus, 2004), Condor (Condor, 2004), Legion (Legion, 2004), Grip (Grip, 2004) and Alchemi (Alchemi, 2004). For this work, Globus was adopted as it is becoming the *de facto* standard for Grid Computing. Globus is a community-based, open-architecture, open-source set of services and software libraries that support Grid and Grid applications (Foster, 02).

Globus provides four basic types of components (Foster, 01):

- *Grid Resource Allocation Management* (GRAM): it is at the core of the Globus remote program execution infrastructure;
- *Information Service*: it provides information about Grid resources for use in resource discovery, selection, and optimization;
- *Data Management*: it supports access to and manipulation of distributed data that are stored in databases or files.
- *Authentication*: it supports some security access control upon resources.

However, considering the objective of this work, both Globus and the other current grid platforms present some drawbacks:

I. Grid platforms are not designed to be used by ordinary users:

   1. they are very complex to be deployed;
   2. they require IT specialists to use it;
   3. they are not transparent to the users.

II. Existing software systems are not designed to use grid computing:
   1. both traditional monolithic systems and even modern component-based systems don't use to be designed in/as separated parts that can run concurrently;

III. Lack of flexibility to define which computing resources are the most suitable ones to host a given processing:
   1. the user should know in advance the execution software's requirements;
   2. the user should know/specify in advance the computers where he wants to execute the software (or some of its parts).
   3. the user should know in advance which are the available computers.

Actually, the basic reason for these constraints is that Grid platforms are a new area and it has primary focused on high performance at the server side. In this work we are primary interested to focus on the client side, allowing a more rational and flexible resources utilization but transparently to the user as much as possible. Seeing in the literature, only two related works in that desired focus were found. However, they provided only an easier way of submitting jobs. The first one was performed as part of the Globus Project (Laszewski, 02). It introduced a Grid service that combines the ability of serving as an information service and as a job execution service, demonstrating a significant simplification of the architecture while treating job submissions and information queries. The second work was developed in the Gridway project (Gridway, 04) to provide an easier and more efficient execution of jobs in dynamic Grid environments. It automatically performs all the job scheduling steps, provides fault recovery mechanisms, and adapts job scheduling and execution to the changing Grid conditions. There are also other works being done that are independent of any particular grid platform that use the concepts of scheduler, meta-scheduler and workflow (Joseph et al., 04). This aims to offer some facilities at the server's side in terms of execution coordination as well as of inference about all grid providers in order to try to identify the best resources. However, this is done purely looking at software and hardware aspects, without any other complementary qualitative analysis.

3. PROPOSAL

Regarding those Grid drawbacks mentioned before, this work aims at contribution for the solution of the aspects I.2, I.3 and III. For that, this work proposes a smart, thin and transparent client architecture and software tool / "meta-interface" that makes possible to an ordinary end user to have access to several idle/available resources over the Internet in a transparently way. Figure 1 presents the proposed architecture.

The basic idea is that this tool can automatically evaluate when to use grid and which computers should be involved in the execution of a given application. This tool checks if there are enough resources to run an application locally as soon as it detects the application is to be executed. This is made checking the needed application's requirements against the existing computing resources in the local environment.

The architecture has two layers. Grid Independent layer is the one developed in this work and that contains supporting modules to help users to use Grid. Grid Dependent layer comprises modules that are specializations of Grid platform's implementation classes (APIs), i.e. that execute grid services themselves. At the server's side, Globus must be deployed in the companies that are going to share their resources. There must be one node in the net that should host the Information Service structure. Seen as a whole, these distributed resources from the members of a given VE create a (instance of a) Grid Resources Federation.

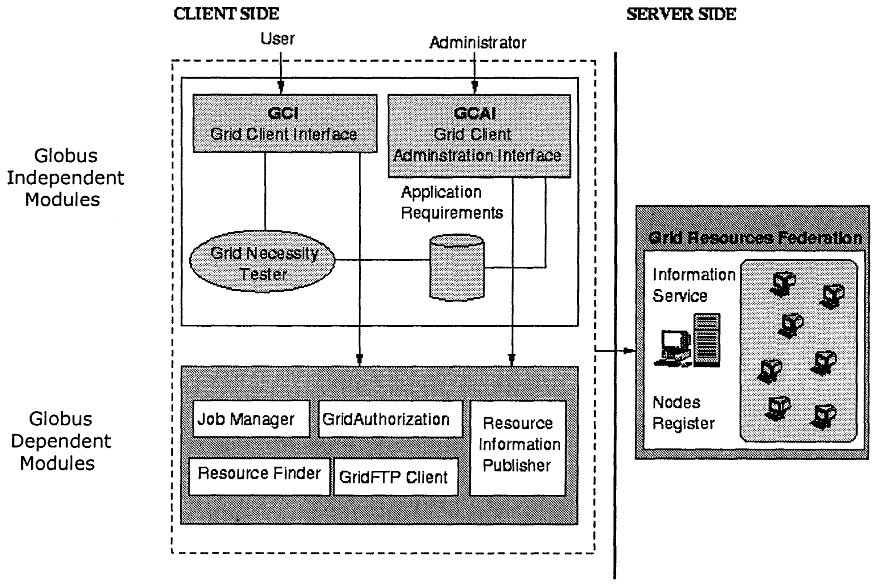


Figure 1 – Proposed Grid Thin Client Architecture

In general, the functioning of this tool runs as follows: i) every application that can make use of Grid must be registered in the platform by a local system administrator; ii) when the user executes a "gridable" application via the GCI module, GCI detects and compares the application's requisites with the current status (memory, hard disk, processor, operating system and architecture) of the local computer; iii) the need for the Grid is then evaluated by the Grid Necessity Tester (GNT) module; iv) if there is a need for Grid the user is authenticated via setting a login/password; v) the platform accesses the Globus' Index Service in order to find out the resources that can match the application's requirements. This service provides information about all the resources available in the VE members (or eventually in external and authorized providers), which were previously configured when the Grid platform was deployed. This information is stored in a structure called *Information Service*. The GNT module receives the list of resources and applies some selection criteria in order to elect the most suitable computer(s) to run the application; vi) the application and input data are sent to the computer(s); vii) the application, or part of it, is executed in the computer(s); viii) the result is returned to the Client / user after the execution of the application. It is important to mention that the user must require a certificate (signed by the Grid administrator) before executing applications.

3.1 Globus Independent Layer

3.1.1. Grid Client Interface to the Grid (GCI)

This module is responsible for creating the interface with the end user, and for abstracting the communication with the other modules. Its main features are:

- The user selects the application (s)he wants to execute, providing some essential input parameters/files for the execution.
- The user has a direct but transparent access to the modules that make use of Grid services.

3.1.2. Grid Client Administration Interface (GCAI)

This module has administrative purposes. It has two basic functions:

- Registering the applications that are candidates to use the Grid infrastructure. In order to accomplish this task, the user must have a minimum knowledge about the application requirements, such as executable, arguments, processor, architecture, operating system, amount of memory and hard disk;
- Publishing the resource's information in the case the user decides to turn his local machine a Grid resource. Thus, it is necessary to provide the information necessary to locate this resource, such as: architecture, operating system, amount of memory and hard disk available.

3.1.3. Grid Necessity Tester (GNT)

This module evaluates the current status of the local node and compares it with the application requirements. According to the results, it decides if Grid is necessary. This evaluation is immediately triggered when the user is going to launch an application. GNT acts only at the local machine. The parameters used for the comparison are:

- memoryFree: amount of free memory (megabytes) at the machine;
- freeHardDisk: amount of free hard disk space (megabytes) at the machine;
- freeProcessor: percentage of free processor.

Another main function of this module is to select the most suitable resource out of the Federation of Grid Providers. In the resource selection, if more than one node is selected, those attributes are also compared in order to choose the most effective one. For instance, if the application demands a large amount of data to be transferred, it could be preferred to choose a machine in the same LAN (Local Area Network).

3.1.4. Applications Requirements

The evaluation made by GNT module depends on the data (about the resource) provided. This data must be defined in a well structured and interoperable way. XML (eXtend Markup Language) (W3C, 2000) has become the standard for these purposes.

For each application (job) registered in the tool, two files are generated. The first one describes a job and it is used by the Globus container (Sandholm, 2003) to make a submission. For the job specification, it is used an XML based language, the RSL (Resource Specification Language) Schema (Globus, 2004a).

The second file describes the resources requirements. This file is used by the GNT module to perform the evaluation, as well as to search for resources. The initial idea was to use the same RSL file employed by the job submission since it provides some fields necessary to this function and a well defined structure. However, some additional information are needed. For this reason, an extension of this RSL file was created, adding the following elements: operating system, architecture, processor, clock, free hard disk, memory, band width (Figure 2).

These requirements are used in three different stages: in the job submission; in the Grid evaluation; and in the resource searching.

```
<gram:resource>
    <gram:operatingSystem><rsl:path>
        <rsl:stringElement value="Linux" />
    </rsl:path></gram:operatingSystem>
    <gram:memory><rsl:path>
        <rsl:integer value="256" />
    </rsl:path></gram:memory>
    <gram:freeHardDisk><rsl:path>
        <rsl:integer value="50" />
    </rsl:path></gram:memory>
</gram:resource>
```

Figure 2 - RSL Extension example

3.2. Globus Dependent Layer

3.2.1. Resource Information Publisher

Each node that integrates the VE must have a way to be found out. In the Globus Information Service component, it is possible to query any data that is registered in that component. The resources that will be available in the Grid must publish their own information in a way they can be found. In this sense, this module provides an easy way to get this information and to publish it in the Information Service. The publication is made through a Grid Service, in the Service Data (Tuecke, 2003), which has meta-information associated with the Grid Service. This Grid Service is deployed in the local machine, and after that, those data are aggregated in the central server where the Information Service is located.

3.2.2. Grid Authorization

In order to allow the user to submit jobs or transfer files to remote nodes, some kind of permission for this access is necessary. This access is obtained via a certificate required by the user, (creating a login/password), and then signed by the Certificate Authority (Tuecke, 2001). These operations are supported by this module.

3.2.3. Resource Finder

Once the need of a Grid has been detected, the next step is to find the resource(s) that supplies the application requirements. For that, this client provides access to the Information Service (Czajkowski, 01), where all nodes have their information published. When the resource is found, the Information Service returns the address of the selected to the client's node. This address is used by the Job Manager and GridFTP Client modules, which are described below.

3.2.4. Job Manager

This is the module that makes the job submission, in a direct connection with the Globus Container (Sandholm, 2003) of the selected nodes. This submission is made using the RSL file (described above), through the GRAM services.

3.2.5. GridFTP Client

It is a client of the GridFTP server, which makes the upload of the application and other essential files (e.g. input/output files) for the execution. The user's credential is also required to get permission to write files in a remote node.

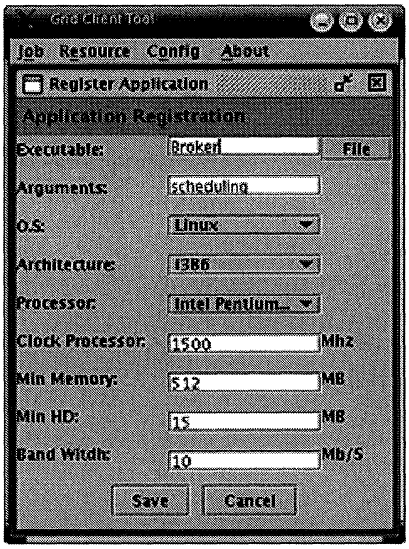
4. PROTOTYPE

Globus Toolkit version 3.2 has been used in the prototype implementation. It provides a comprehensive Java API, which offers programming facilities to both server and client sides. Using Globus, servers should be Unix machines so applications should be written according to. At the client side, there is the CoG Kit (CoG), which makes easier some programming aspects related to the Grid services access. This led us to implement the prototype in Java. From the server point of view, Globus platform is obligatory to be installed in the machines that will act as grid resources, i.e. that will make available computing resources to the others. The client needs the CoG's APIs.

Only some attributes (memory, processor and hard disk) have been considered in as selection criteria out of the available resources. Other more "qualitative" attributes, such as trust levels, historical performance and use, costs safe and job's deadline, were not implemented in this first prototype. Other useful features, such as the possibility to set up precise values for the computing resources that every node is interested to make available, is so far not supported by any existing grid platform. In practice it means that a given computer stay either "totally" available for the others or it simply can't be accessed.

An application related to the partners' search and selection problem was developed to test the feasibility of the proposed approach. This application is a module of a wider VE management system called SC^2 (Rabelo, 2004) that supports some steps in the creation and operation phases of the VE life cycle. This module was designed having Grid philosophy in mind so it has parts that can run separately / concurrently. The part tested is the scheduling functionality, which is responsible for generating all the scheduling possibilities of VE for a given business and to select the best schedule. As it is CPU-bound, it is evaluated the situation where it is realized that the required computing resources are not available in the local machine and then external resources should be sought using Grid facilities. A given resource is further selected and the scheduling application and its data are sent to it and executed on it.

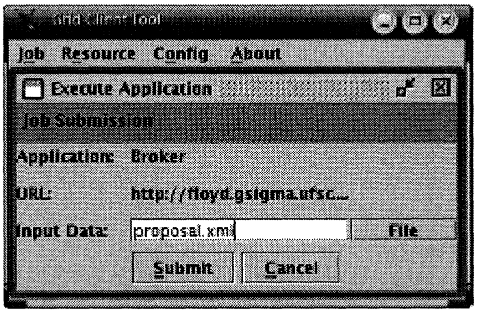
As said in the section 3.1.2, "gridable" applications must be previously registered and their requirements specified in order to make use of Grid platform's facilities. This is made by the system's administrator. Figure 3 shows this process for the case of that scheduling application.



Executable: Broker
Arguments: scheduling
O.S: Linux
Architecture: I386
Processor: Intel Pentium...
Clock Processor: 1500 Mhz
Min Memory: 512 MB
Min HD: 15 MB
Band Witdh: 10 Mb/S

Figure 3 - Application Registration

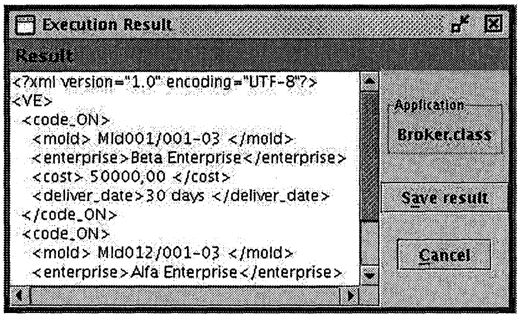
Figure 4 shows another interface through which the user specifies the input data's file name that should be sent to the (previously) selected resource's URL. Thus, the job can be indeed submitted/executed.



Application: Broker
URL: http://floyd.gsigma.ufsc...
Input Data: proposal.xml

Figure 4 - Job Submission

Figure 5 shows the result of the VE scheduling algorithm/functionality. The result is generated as a XML file in this prototype as it is to be easily interoperable with other functionalities. However, for the user visualization, a more user friendly layout should be developed although it is a matter of how the functionality was implemented. This result shows the list of the selected enterprises (i.e. the VE), showing their identification, production cost and delivery date.



```
<?xml version="1.0" encoding="UTF-8"?>
<VE>
  <code_ON>
    <mold> Mld001/001-03 </mold>
    <enterprise>Beta Enterprise</enterprise>
    <cost> 50000,00 </cost>
    <deliver_date>30 days </deliver_date>
  </code_ON>
  <code_ON>
    <mold> Mld012/001-03 </mold>
    <enterprise>Alfa Enterprise</enterprise>
```

Figure 5 – Scheduling Result

5. CONCLUSION

This work proposed an approach to allow smart and flexible utilization of computing resources that are idle to overcome local software and hardware restrictions when executing applications. This execution is done remotely, seamlessly and by means of the support of a grid (Globus) platform and of a client interface to leave this transparent to the user as well as to make a more rational use of the resources.

It also realized that all VE members can benefit from making available their resources to the others and that richer qualitative metrics can be added in the future for smarter selection of computing resources.

The developed prototype has offered elements for preliminary conclusions that Grid computing is a feasible technology for dealing with the problem of the increasing complexity of enterprises' systems and that it can be useful to help MSME in their investments on IT.

The experiments involved several types of PCs, both powerful ones and old machines. In both cases the system worked out, with natural differences in terms of time processing.

However, some limitations could be observed. For example, so far applications can only run on Unix/Linux operating system, respecting the restrictions of current versions of existing Grid platforms. Besides that, only "pre-prepared" applications can benefit from Grid technology. Systems haven't been developed with the notion of concurrency among their submodules, which inhibits the achievement of most of grid advantages. A number of recommendations can be followed in this direction, as the ones proposed in (Ferreira, 04). More recent approaches for software development, like SOA (services oriented architecture) can naturally help this as it makes natural the "decomposition" of an application in several "parts" (services) that can run everywhere. There is also a reasonable difficulty in the resources matching. In a heterogeneous multi-institutional environment such as the Grid, it is difficult to impose common syntax and semantics for a resource description. An evolution on this can be implemented with the use of ontologies, as described in (Tangmunarunkit, 03).

Next steps of this work includes dealing with security (resources access policies; trust clients, etc.), extending the model to Microsoft OS as soon as Globus platform can support it. It is also important to better investigate "clever" meanings for "idle" resources. Another part of the intended research on this area comprises the possibility of a given company to offer its applications to the VE members, acting as an ASP (Application Service Provider) model once legal aspects related to software licenses are overcome.

ACKNOWLEDGMENTS

This work was partially supported by FAPEAL – Research Support Foundation of State of Alagoas (www.fapeal.br), and it has been developed in the scope of the Brazilian IFM project (www.ifm.org.br) and the European IST FP6 IP ECOLEAD project (www.ecolead.org).

6. REFERENCES

Alchemi .Net Grid Computing Framework, URL: http://www.alchemi.net, 2004.

Berstis, V., Fundamentals of Grid Computing, IBM Redbooks Paper, in http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf, 2002.

Condor High Throughput Computing, http://www.cs.wisc.edu/condor, 2004.

Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C., 2001, Grid Information Services for Distributed Resource Sharing, Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press.

Europe-EU, http://www.eubusiness.com/topics/SMEs, accessed in 21/05/2005.

Ferreira, L., Thakore, A., Brown, M., Lucchese, F., RuoBo, H., Lin, L., Manesco, P., Momtaheni, N., Hernandez, O., Grid Service Programming and Application Enablement, IBM Redbooks, 2004.

Foster, I., Kesselman, C. and Tuecke, S., 2001, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Int. Journal of High Performance Computing Applications, 200-222.

Foster, I., Kesselman, C., Nick, J. and Tuecke, S., 2002, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.

Globus, 2004, The Globus Alliance, URL: http://www.globus.org.

Globus, 2004, Managed Job Service - Resource Specification Language (RSL), URL: http://www-unix.globus.org/toolkit/docs/3.2/gram/ws/developer/mjs_rsl_schema.html.

Grip: Grid Interoperability Project, 2004, http://www.grid-interoperability.org.

Gridway, 2004, http://www.gridway.org.

Joseph, J., Fellenstein, C., Grid Computing, Prentice Hall, IBM Press, 2004.

Laszewski, G. Von., Foster, I., Gawor, J., Schreiber, A., Pena, C., InfoGram: A Grid Service that Supports Both Information Queries and Job Execution, Proceedings of the 11th IEEE International Symposium on High-Performance Distributed Computing, 2002.

Legion: A Worldwide Virtual Computer, 2004, URL: http://www.cs.virginia.edu/~legion.

Rabelo, R. J.; Pereira-Klen, A. A.; Klen, E. R.; Effective Management of Dynamic and Multiple Supply Chains. In: Int. J. of Networking and Virtual Organizations, V 2 N 3, p. 193-208, 2004.

Sandholm, T., Gawor, J., Globus Toolkit 3 Core – A Grid Service Container Framework.

Tangmunarunkit, H., Decker, S., Kesselman. C., Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web, Proc. of the Second Int. Semantic Web Conference, 2003.

Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C, Maguire T., Sandholm, T., Snelling, D., Open Grid Services Infrastructure (OGSI) Version 1.0, Global Grid Forum, 2003.

Tuecke, S., Engert, D., Foster, I., Thompson, M., Pearlman, L. and Kesselman, C. Internet X.509 Public Key Infrastructure Proxy Certificate Profile, IETF, 2001.

W3C, Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, URL: http://www.w3.org/TR/REC-xml, 2000.