Bridging the Gap Between Cluster and Grid Computing

Albano Alves and António Pina

[1] ESTiG-IPB, Campus Sta. Apolónia, 5301-857, Bragança-Portugal
albano@ipb.pt
[2] UMinho, Campus de Gualtar, 4710-057, Braga-Portugal
pina@di.uminho.pt

Abstract. The Internet computing model with its ubiquitous networking and computing infrastructure is driving a new class of interoperable applications that benefit both from high computing power and multiple Internet connections. In this context, grids are promising computing platforms that allow to aggregate distributed resources such as workstations and clusters to solve large-scale problems. However, because most parallel programming tools were primarily developed for MPP and cluster computing, to exploit the new environment higher abstraction and cooperative interfaces are required. $Ro_{c}me\mu$ is a platform originally designed to support the operation of multi-SAN clusters that integrates application modeling and resource allocation. In this paper we show how the underlying resource oriented computation model provides the necessary abstractions to accommodate the migration from cluster to multicluster grid enabled computing.

1 Introduction

Many researchers have contributed to important aspects of grid computing allowing applications to share and aggregate distributed resources that cross the cluster boundary. However, because most parallel platforms were built for MPP and cluster computing, programmers still don't have tools to exploit the grid in a straightforward way.

1.1 Distributed Parallel Applications

By its nature, the Grid seems to be the perfect platform to run application systems that integrate multiple cooperative parallel applications distributed across the Internet. Each parallel application should exploit local high performance hardware.

A parallel application should be able to launch other applications or instantiate specific remote components. In addition, components of distinct applications should be able to exchange data through message passing or by accessing remote memory. Parallel applications started independently, eventually running on different computing systems and under the control of different users, should be able to establish some kind of relationship. To achieve cooperation between distinct applications we need mechanisms to represent application entry points and to discover application components.

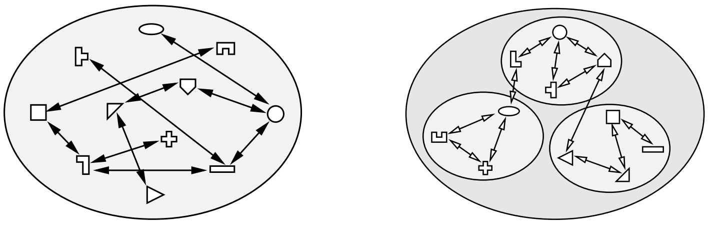
The concept of application system aims to aggregate collaborative application components spread among multiple architectures and machines. Independently of the resources used and the administrative domains and users involved, the programmer should be able to produce a unified system view of the application system. Moreover, different programmers/users would benefit from the existence of multiple distinct views shaped according to their different interests.

1.2 The $Ro_cme\mu$ Approach

$Ro_cme\mu$ combines the facilities of two systems: R_OCl [2] and $m_\mathcal{E}\mu$ [3].

R_OCl acts as a base layer interfacing low-level communication subsystems generally available in clusters. It constitutes a basic single system image, by providing interconnectivity among communication entities despite their location. Those application entities we name resources may exchange messages despite the underlying communication subsystems. The R_OCl dispatching mechanism is able to bridge messages from GM to M-VIA, for instance. A cluster oriented directory service allows programmers to announce and locate application resources thus turning R_OCl into a convenient platform to drive multi-SAN clusters that integrate multiple subclusters (Myrinet and Gigabit subclusters, for instance) interconnected by multihomed nodes.

$m_\mathcal{E}\mu$ was implemented over R_OCl and provides higher-level programming abstractions. Basically, it supports the specification of physical resources, the instantiation of logical resources accordingly to the real organization of physical resources and high-level communication operations between logical resources.



(a) R_OCl (b) $m_\mathcal{E}\mu$

Fig. 1. Resource interaction in R_OCl and $m_\mathcal{E}\mu$

Figure 1 depicts the main difference between R_OCl and $m_\mathcal{E}\mu$ abstraction models; while R_OCl (figure 1(a)) provides unstructured communication between resources, $m_\mathcal{E}\mu$ (figure 1(a)) offers mechanisms to organize resources and exploit locality.

$Ro_cme\mu$, as an overall platform, is well suited to the development of applications aimed to exploit multi-SAN clusters that integrate multiple communication technologies such as Myrinet and Gigabit. The corresponding programming model has proved to be very useful to manage the distinct locality levels intrinsic to that system architecture. The approach provides the necessary abstractions at the application level to accommodate the evolution from cluster to multicluster grid enabled computing.

2 Resource Oriented Communication

R_OCl is an intermediate-level communication library that allows system programmers to easily develop higher-level programming environments. It uses existing low-level communication libraries to interface networking hardware, like Madeleine [4].

2.1 General Concepts

R_OCl defines three major entities: contexts, resources and buffers. Contexts are used to interface the low-level communication subsystems. Resources permit to model both communication and computation entities whose existence is announced by registering them in a global distributed directory service. To minimize memory allocation and pinning operations, R_OCl uses a buffer management system; messages are held on specific registered memory areas to allow zero-copy communication.

Every application should initialize a R_OCl context, register some resources, query the directory to find remote resources, request message buffers, address messages to resources previously found and retrieve received messages from a local queue.

R_OCl includes a fully distributed directory service where resources are registered along with their attributes; a directory server is started at each cluster node and all application resources are registered locally. A basic interserver protocol allows applications to locate remote resources; query requests are always addressed to a local server but, at any moment, servers may trigger a global search by broadcasting the request.

Resources are animated by application threads which share the communication facilities provided by contexts. On the other hand, R_OCl supports the simultaneous exploitation of multiple communication technologies being responsible for selecting the most appropriate communication medium to deliver messages to a specific destination, for aggregating technologies when the target resource of a message can be reached through distinct technologies and for routing messages at multihomed nodes that interconnect distinct subclusters. To provide the above facilities, R_OCl uses a multithreaded dispatching mechanism and includes native support for multithreaded applications.

2.2 Multicluster Operation

The evolution of R_OCl from the cluster environment (multisubcluster) to the grid environment (multicluster) is highly dependent on the capabilities offered by its distributed directory service. The ability to locate resources at cluster level is primarily based on broadcast which does not scale to the grid environment.

R_OCl was improved to include resource location in grid environment by means of a directory proxy installed at each cluster. A directory server unicasts query requests to all known proxies, whenever it fails to obtain replies from servers inside its cluster.

Resource lookup in a multicluster environment comprises the following stages: *(1)* the application sends a request to the server running in the same node, *(2)* that server searches its local database, *(3)* the request is broadcast to all cluster servers, *(4)* those servers search their databases, *(5)* the request is sent to all known proxies that represent the known remote clusters, *(6)* proxies broadcast the requests to servers, *(7)* those servers search their databases and finally *(8)* the replies are sent to the proxy of the cluster where the query process started. The query process may conclude after stages 2 or 4 depending on the location of the resource. It is also possible to specify the scope of a particular query (*node*, *cluster* or *multicluster*) thus bounding the lookup process.

The dispatching mechanism uses a similar approach in order to deliver messages to resources instantiated on nodes from remote clusters. In effect, the directory proxy also acts as a message forwarder; when the dispatching mechanism knows that a resource is from a remote cluster, it sends the message to the proxy of that remote cluster.

3 Unified Modeling and Exploitation

$m_\mathcal{E}\mu$ programing methodology includes three phases: *(1)* the definition and organization of the concepts that model the parallel computer - physical resources -, *(2)* the definition and organization of the entities that represent applications - logical resources - and *(3)* the mapping of the logical entities into the physical resources.

The programing interface is organized around six basic abstractions (see figure 2) designed for modeling both logical and physical resources: *(1)* domains - to group or confine a hierarchy of related entities; *(2)* operons - to delimit the running contexts of tasks; *(3)* tasks - to support fine-grain concurrency and communication; *(4)* mailboxes - to queue messages sent/retrieved by tasks; *(5)* memory blocks - to define segments of contiguous memory; *(6)* memory gathers - to create the illusion of global memory.

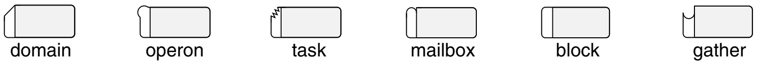


Fig. 2. $m_\mathcal{E}\mu$ entities

3.1 Representation of Resources

As an instantiation of the first phase of the methodology, figure 3(a) shows how the physical resources of a parallel machine made of distinct technological partitions may be modeled by a hierarchy of domains. At the top-level, the domain *Cluster* has as its direct descendants three subclusters *Quad Xeon*, *Dual PIII* and *Dual Athlon*. At each subcluster, nodes are modeled by *Node* A_x, *Node* B_x and *Node* C_x.

The second phase is illustrated in figure 3(b) by a high-level specification of a simple distributed web crawler. The crawling process is modeled by a top domain that represents the application whose descendants are a certain number of *Robots* modeled by operon resources. Further refinement introduces three different sorts of tasks (*Download*, *Parse* and *Spread*) to model the three well known crawling stages: *(1)* downloading of pages from the web, *(2)* parsing the pages to obtain new URLs and *(3)* distribution of URLs for future crawling, according to a certain partitioning scheme.

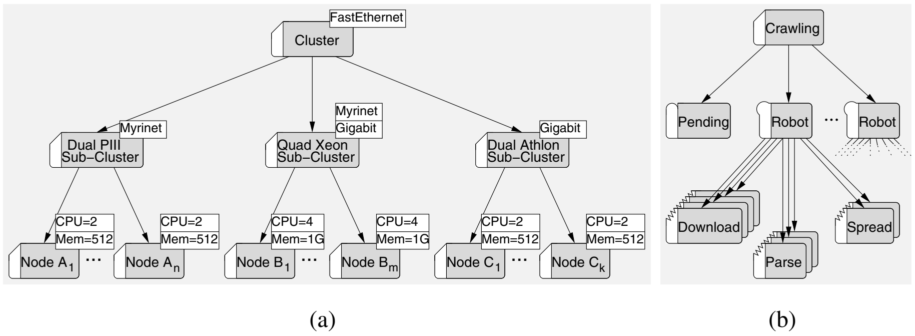


Fig. 3. Logical and physical resources

Finally, *Pending* models a general global accessible mailbox used by *Download* tasks to store/retrieve pending URLs.

Resources are named and globally identified entities to which we may attach lists of properties. In figure 3(a), *FastEthernet*, *Myrinet* and *Gigabit* are properties that qualify *Cluster* and *Subcluster* domains, while *CPU=2* and *Mem=512* are another sort of properties used to quantify characteristics of nodes A_x and C_x.

$m_\mathcal{E}\mu$ resources accumulate properties inherited from their chain of ancestors. Inheritance allows resource properties to be propagated top-down from ancestors to descendants, while synthesis is the reverse mechanism. $m_\mathcal{E}\mu$ also introduces the concept of resource alias which is a virtual resource node, used as a proxy to one or more existent nodes. In addition to the ancestor and descendants, an alias also has one or more originals, which share with it their own full set of accumulated properties. Aliases are represented as dashed shapes and the dashed arrow coming from each original in the direction of an alias represents the mechanism that allows for the sharing of the original accumulated properties with the target resource (see figure 4).

3.2 Integration of Hierarchies

To map the abstract logical representation, derived from the second phase, into physical resources, programmers should use $m_\mathcal{E}\mu$ primitives. Next, we describe the mapping process used to produce the hierarchy presented in figure 4, which corresponds to the fusion of two hierarchies - the hierarchy that represents the cluster (figure 3(a)) and the hierarchy that represents the application (figure 3(b)).

First, the application identifies the target physical resources - the subclusters *Quad Xeon* and *Dual Athlon*. Then, the alias domain *Crawling* is created to express the effective selection of physical resources. This domain represents a particular view of the available physical resources. Subsequently, operons *Robot* may be launched by specifying the alias *Crawling* as the target. The $m_\mathcal{E}\mu$ runtime system will automatically select a node among the nodes of subclusters *Quad Xeon* and *Dual Athlon*, instantiate an operon on that node and create an alias under the domain *Crawling*.

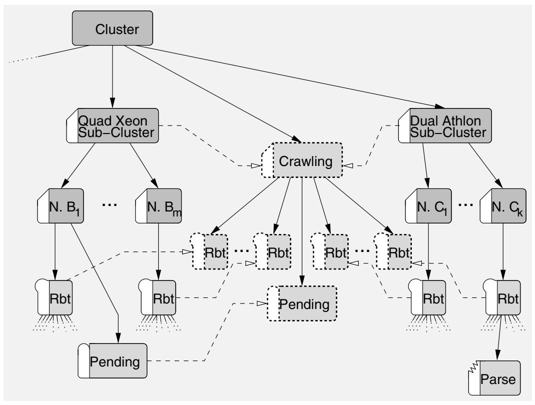


Fig. 4. Resource mapping

It is important to note that the integration of hierarchies preserves the original organization of the logical hierarchy through aliases but introduces a system view which combines physical and logical resources. The system view does not take into account aliases but allows to reach any component without knowing the application structure.

3.3 Multicluster Exploitation

In a grid environment, the physical hierarchy of the application example depicted in figure 4 will be extended to include, at least, an additional top-level domain. The new root domain would have multiple descendant domains representing each of the clusters.

The multicluster environment enforces the need to control the access to resources. $m_{\mathcal{E}}\mu$ allows to specify access control lists for each resource which are used to validate *(1)* the creation of descendants and aliases and *(2)* the access to resource properties. The selection/allocation of physical resources will thus be denied for unauthorized users as well as the access to logical resources (messaging and memory access).

$m_{\mathcal{E}}\mu$ offers high-level structured message passing that allows any received message to be implicitly broadcast to the totality of the active descendants of a logical domain. This is a valuable feature of the resource oriented paradigm particularly suited for a cluster environment where broadcast can be efficiently implemented at the hardware/software level. In a multicluster environment programmers must be aware of potential bottlenecks caused by logical domains that traverse distributed clusters.

Resource selection, through the creation of an alias that aggregates multiple physical domains, requires several $m_{\mathcal{E}}\mu$ lookup operations. Since those operations are implemented using global $R_O Cl$ queries, the use of this functionality must be occasional.

4 Deploying Applications

$Ro_c me\mu$ constitutes a very simple approach for the development and execution of applications in some particular grid scenarios - multicluster systems where the nodes at each cluster may be interconnected by Myrinet, Gigabit and/or FastEthernet. Cluster administrators are responsible for installing $Ro_c me\mu$ libraries and services while programmers use $m_{\mathcal{E}}\mu$ programming interface to develop applications. Users are also responsible for defining a virtual cluster to support the execution of the application.

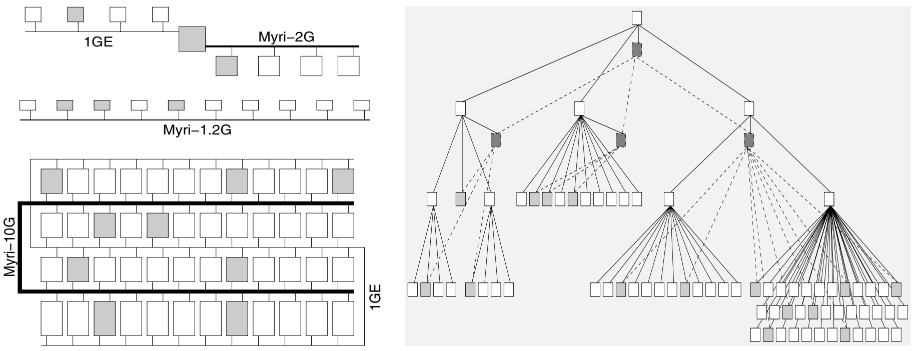
4.1 Virtual Clusters

Users may define virtual clusters – subsets of the totality of multicluster resources – using a web portal where the nodes of each cluster are graphically represented. Figure 5(a) depicts the reservation of 15 nodes spread through three clusters.

Based on user's selection, *(1)* the system manages to create a global user account, *(2)* the $m_{\mathcal{E}}\mu$ physical hierarchy is updated to include the right access control lists and *(3)* some $m_{\mathcal{E}}\mu$ aliases are created in order to produce a suitable view (see figure 5(b)).

4.2 Multicluster Wide Access

In a multicluster environment, the first problem to solve is TCP/IP connectivity, because of private networks, firewalls, etc. Our approach uses OpenVPN to provide full connectivity among multicluster nodes despite the composition of each virtual cluster.



(a) User selection (b) Corresponding $m\varepsilon\mu$ hierarchy

Fig. 5. Creating a virtual cluster

Since $Ro_c me\mu$ allows to dynamically start application components, we also needed to extend *rsh/ssh* to support multicluster operation. $Ro_c me\mu$ maintains a pool of shadow accounts at each cluster and thus the virtual global account created when the user reserves multicluster nodes is mapped into multiple (real) shadow accounts.

Finally, access to binaries and application data requires some kind of multicluster file system. Our approach is to extend NFS by introducing appropriate proxies. This idea has already been investigated and deployed in the context of PUNCH [7], a wide-area networking computing environment.

5 Discussion

In the last decade PVM and MPI have dominated the field of parallel computing. The paradigm still remains the main choice for grid programming as attested by current ports like MPICH-G2 [8] and other platforms to run unmodified PVM/MPI applications [9].

Message passing is more adequate than the connection based approach for exploiting the parallelism of distributed memories in grid and web services [6]. However, fine grain synchronization is not easily achieved in a grid environment, meaning that grids cannot be programmed as flat huge machines made of a larger number of processors.

$Ro_c me\mu$ offers programmers the possibility of structuring the communication between resources in a multicluster environment following the same principles that lead to the organization of a cluster into subclusters. Taking into account the hierarchy of a multicluster grid and the available interconnection technologies, programmers may take advantage of locality at: *(1)* SMP nodes, where processors share memory, *(2)* subclusters, where nodes are interconnect by a sole high performance communication technology, *(3)* clusters, where subclusters are interconnect by multihomed nodes and *(4)* multicluster grid, where clusters are interconnected by Internet links.

Programmers need the power of grids to build and deploy applications that break free of single resource limitations to solve large-scale problems. Although much work have been done in the context of Globus [5] to exploit grid resources, $Ro_c me\mu$ offers as a unique feature the integration of: physical resource specification, application modeling and logical components instantiation, providing much of the operations identified in [1] as key functionality for developing grid applications.

References

1. G. Allen, T. Goodale, M. Russell, E. Seidel, and J. Shalf. *Grid Computing: Making the Global Infrastructure a Reality*, chapter Classifying and enabling grid applications. John Wiley & Sons, Ltd., 2003.
2. A. Alves, A. Pina, J. Exposto, and J. Rufino. RoCL: a Resource oriented Communication Library. In *Euro-Par 2003*, number 2790 in LNCS, pages 969–979. Springer, 2003.
3. A. Alves, A. Pina, J. Exposto, and J. Rufino. $m\varepsilon\mu$: unifying application modeling and cluster exploitation. In *SBAC-PAD'04*, pages 132–139. IEEE Computer Society, 2004.
4. O. Aumage, L. Bougé, A. Denis, J.-F. Méhaut, G. Mercier, R. Namyst, and L. Prylli. Madeleine II: A Portable and Efficient Communication Library for High-Performance Cluster Computing. In *CLUSTER'00*, pages 78–87, 2000.
5. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. In *HPDC'01*, pages 181–194, 2001.
6. G. Fox. Messaging Systems: Parallel Computing the Internet and the Grid. In *Euro-PVM/MPI'03 (invited talk)*, 2003.
7. N. Kapadia, R. Figueiredo, and J. Fortes. Enhancing the Scalability and Usability of Computational Grids via Logical Accounts and Virtual File Systems. In *HCW'01*, 2001.
8. N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Parallel and Distributed Computing*, 63(5):551–563, 2003.
9. D. Royo, N. Kapadia, and J. Fortes. Running PVM Applications in the PUNCH Wide Area Network-Computing Environment. In *VECPAR'00*, 2000.