Distributed Objects with NGrid applied to SMO

JOANNÈS VERMOREL

Computational Biology Group, École des Mines de Paris, 35 rue Saint-Honoré, 77305 Fontainebleau cedex, France
`http://www.bioinfo.ensmp.fr` and `http://ngrid.sourceforge.net`

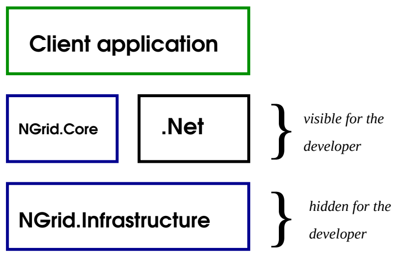
ABSTRACT

Very often software engineering is a major blocking factor when it comes to distributed systems. Most of the existing distributed computing environments (MPI for example) are discouraging and obscure for non-specialists. On the contrary, distributed objects are simple and natural extensions of the object-oriented programming paradigm that can be used to tackle distributed systems.

`NGrid` is a transparent object-oriented distributed programming framework developed at the Computational Biology Group of the École des Mines. `NGrid` is designed to facilitate the distribution of machine learning algorithms such as Sequential Minimization Optimization (SMO).

1 Introduction

1.1 NGrid architecture



`NGrid.Core` is an open-source LGPL `.Net` library written in `C#`. The infrastructure is independent from the client code.

1.2 Quick NGrid `C#` syntax

Distributed objects inherit the `GObject` class.

```
public class HW : GObject {
  public void Hello() {
    Console.WriteLine("Hello World");
  }
}
```

The proxy must be retrieved at instantiation.

```
HW obj = (HW) (new HW()).GetProxy();
```

Distributed threads are similar to local ones.

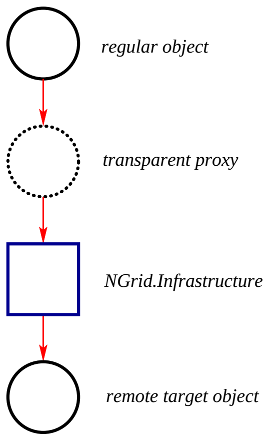
```
GThread thread = new GThread(obj.Hello);
thread = thread.GetProxy();
thread.Start();
```

A distributed lock ensures thread safety.

```
using(new Lock(obj)) {
  // exclusive execution here.
}
```

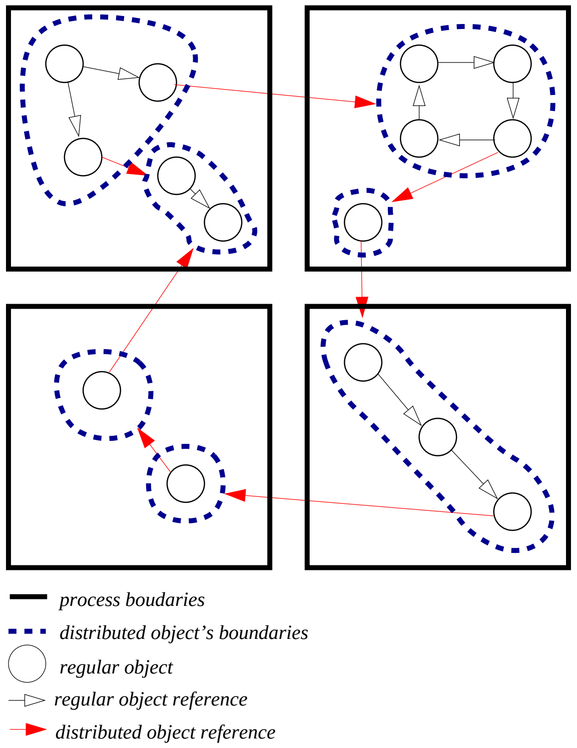
2 Distributed Objects Semantic

2.2 The notion of transparent proxy



With transparent proxies, programming with distributed objects is totally transparent for the developers.

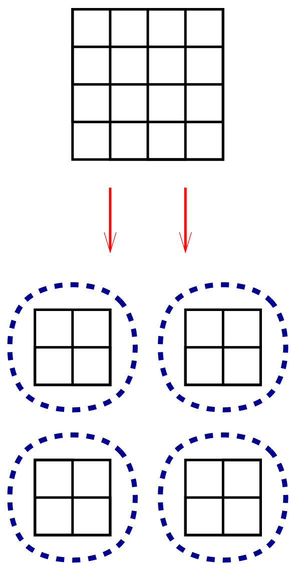
2.1 Mixing regular and distributed objects



Distributed objects can migrate between processes. When crossing a process boundary, regular objects are transmitted by value, whereas distributed objects are transmitted by reference.

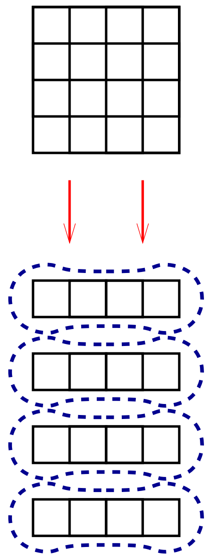
3 SMO acceleration with NGrid

3.1 Gram matrix parallel computation



The first phase of the SMO consists of computing the Gram matrix. With NGrid, splitting the Gram matrix into small sub-matrices and running concurrent tasks on each sub-matrix is straightforward.

3.2 Exploiting the locality of reference in SMO



The second phase of the SMO consists of iterations over the lines of the Gram matrix. It's simple with NGrid to redefine the distributed object to take advantage of the locality of reference to speed-up even a sequential algorithm.

4 Other NGrid features

- Distributed Garbage Collection.
- Distributed Object Replication.
- Attributes for performance tuning.
- Multiple physical grids available.

5 Conclusions

`NGrid` is still a very preliminary project, yet the base implementation is already functional. Empirical evaluations of SMO acceleration through `NGrid` (among other widely used algorithms such as K-Means, or Near Neighbor Search algorithms) are currently under way.

6 Future directions

- Common machine learning algorithm implementations targeting `NGrid`.
- "Smart" load balancing strategies for migration and replication.
- Strong mobility for distributed objects.
- Distributed `IO` subsystem.
- ...

References

[1] John C. Platt, "Fast training of support vector machines using sequential minimal optimization", *Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA, USA* (1999),

[2] Denis Caromel and Ludovic Henrio, "A Theory of Distributed Objects", *Springer-Verlag* (2005).

[3] Joannes Vermorel, "Sketch-based Distributed Garbage Collection, Theory and Empirical Elements", *to be published* (2005).