A Note on Distributed Computing summary

This paper looks at the differences between non-distributed and distributed systems, and specifically interactions between objects. It also talks about how to deal with these differences.
The paper differs between local and distributed systems by what address spaces they make calls to. Local is to single, and distributed can make to other address spaces that can be located on other computers. In distributed calls the client does not know anything about the hardware or language of the recipient.

Vision of unified objects
There is a vision of removing the distinction of distributed object from the programmers view, through interfaces. Such a vision would make it possible for programmers to follow a single type object oriented design where development decisions are made without regards to what kind of computational model being used. For this to be a reality, several aspects need to be considered

First of all the underlying communication protocols play an important part in distributed computing, and the development of these protocols have taken one of two paths. One focuses on the current language model. Another focuses on solving problems inherit from distributed computing. Every few years people working on the language side of the problem realise that the current programming model is outdated. This leads to dispute between the two paths, and the creation of a new distributed computing paradigm. These iterative upgrades, might end up unifying distributed and local computing, or it might go on forever due to the fact that local and distributed computing are too different to be unified.

Local and Distributed Computing
There are four major differences between local and distributed computing. Latency, memory access, concurrency and partial failures

The difference of latency on local and remote machines, are four to five times the magnitude of each other. This obviously leads to problems if you treat all invocation calls as if they were done locally. A good program, that does distributed computing, should be able to differ objects that could be clustered together and the ones that could be made remote. Things that could fix the latency problem without doing drastic changes to an application, is improvement in hardware, or tools that can properly see communication patterns in an application.

Problems related to memory access may arise when you treat pointers in a remote address space the same way as you treat pointers in a local address space. One could either have the underlying system take care of all remote memory access, or have the programmer figure out a solution for the different types of access.

Masking the differences in local and distributed computing might be a solution for latency and memory access. This solution does however not seem possible for partial failure and concurrency.

Taking the difference seriously
Merging local and distributed computing by having one follow the other will unlikely succeed due to the major differences in latency, memory access, partial failures and concurrency. Having one model follow the other would lead to faults or big unnecessary changes. One should rather be aware of these differences through every step of the development process.

Instead of having one computational model follow the other, one could create a middle ground. This could be achieved by creating a object that differs from local and distributed objects. This would be objects that are guaranteed to stay on the same machine, while beeing in different address spaces. These objects will then serve as the middle ground.