ing to the object class. The agent instantiated in step 84 is distributed among the plurality of computer machines of the network, so the task instances and object instances, like the agent process itself, may similarly be distributed among the computer machines of the network. In a step 85, the object migrate method, task migrate method and agent migrate method are performed within the agent process. It should be noted that the methods performed during step 85 need not be performed in any particular order, and each may be performed multiple times, if desired. Moreover, only some of the migrate methods may be defined and performed, if desired.

Method Call Models

[0063] In an object-oriented language, an object defines a collection of data and operations called methods or procedures to manipulate that data. A method call invokes a method on some arguments. In a distributed object-oriented language like that used in the present invention, a method call may span machine boundaries: that is, the machine where the call is made may be different than the machine where the object containing the called method resides.

[0064] The present invention provides two different ways or protocols for executing methods. These two protocols derive from the fact that the caller of a method and the callee object may not be physically located on the same machine. Before describing these two calling protocols, it is useful to first explain fast and slow access modes to objects. In the fast access mode, an object field is accessed without checking and dereferencing the object's identity in the object space. Such an operation is only valid if the object is guaranteed to be present on the caller's base. In this case, the object is accessed through an ordinary addressing scheme. In the slow access mode, an object's global location must be checked via the object space and dereferenced every time one of its components is accessed.

[0065] In order to utilize these two access modes, the present invention defines the following two calling protocols:

(1) An "RPC Model" (remote procedure call model) utilizes the fast access mode. A method runs on the base where the *self* object that owns the method resides, so that field accesses of the object can always be done in fast mode. No dereferencing of the object's global identity is required.

(2) An "Invoker Model" realizes the slow access mode. A method runs at the caller base and does not require that the *self* object be on the same base, so field accesses require dereferencing before actually accessing. The invoker model allows code to be run at the calling or called location; that is, on different machines within the same agent.

[0066] Although the use of one or the other of these protocols impacts efficiency; neither of the two protocols influences program behavior.

[0067] The RPC Model and Invoker Model are illustrated in FIG. 7. A single agent spans two bases, Base1 and Base2. A method m0( ) is running on Base2. Method m0( ) calls a method x.m1( ) which is associated with an object x on Base1. Under the RPC model, computation in method m0( ) moves from Base2 to Base1 where object x and the associated method x.m1( ) resides. That is, the method x.m1( ) executes on Base1 though it was called by a method running on Base2. Field accesses, such as the statement "this.v1 = 0;", can be performed as a local computation requiring no communication between the bases. After method x.m1( ) is completed, control return to method m0( ) on Base2. Next, a second method, x.m2( ), associated with object x on Base1 is called under the invoker model. A remote reference to object x is created on Base2, and method x.m2( ) is run on Base2 rather than on Base1. Field accesses, such as the statement "this.y = 0;", require initiation of a communication event between Base1 and Base2.

[0068] Special cases for method calls include constructor methods or instance methods of a class implementing either an Anchored or a Remote interface. A constructor method is always called in the RPC model, since it might have location-dependent initialization. The instance native method to anchored objects must always be called in the RPC model, since the semantics of the instances are dependent on their locations. The interface method to a remote object is called in one of the bases on which the agent that has the actual object resides.

[0069] Programmers may also explicitly specify a base where an invoker method call should be executed using a method name with an '@[base]' expression, although the invoker method is executed in a caller base by default. In this call, the caller base, the base on which the instance resides, and the base on which the method is executed might be different, but this does not raise an error since the instance methods and fields are always accessed using the slow access mode.

[0070] The language of the present invention preferably assumes the following default behavior:

(1) Unless specified otherwise, methods are always called in the invoker model.

(2) When a programmer specifies an RPC method modifier to a method, the method directly accesses instance fields of *self* objects in the fast mode, though realization of this protocol requires execution to move to the base where the associated object resides.

(3) RPC modifiers can also be applied to static methods. In this case, static methods are called at the location where the class object is, and then the methods access static fields in the fast mode.