

18 contain data 36 that is allocated to the different computer systems 12 #1, #2, and #3 in accordance with the partition data 32.

Referring now to FIG. 3, the sequential operations performed by processor 14 of one of the computer systems 12 to create the distributed file object 18 begin as indicated at a block 300 with creating a node group as indicated at a block 302. Remote system information is identified for each of the multiple computers systems 12 #1, #2, and #3, as shown in FIG. 1, as indicated at a block 304. A hash algorithm 31 to be used for the file object 18 and a partition distribution 32 for partitioning data among each of the multiple computers systems 12 are identified as indicated at a block 306. Various partitioning schemes can be used for partitioning data, for example, a default partitioning scheme can provide an even distribution of data among each of the computer systems. Alternatively, a user specified partitioning scheme can provide a selected percentage for particular ones of the computer systems 12, such as 50% of the data to be stored on computer system 12 #1 and 25% of the data on each of computer systems 12 #2 and #3. A file object is created and stored on the particular one of the computer systems 12 as indicated at a block 308. The file object contains the hash algorithm 31, partition distribution 32, and remote system information 34. A remote connection is established to another one of the multiple computer systems 12 and the file object is created on that particular system 12 as indicated at a block 310. Checking for other of the multiple computer systems 12 that have not been accessed is provided as indicated at a block 312. A next remote connection is established to another one of the multiple computer systems 12 and the file object is created on that particular system 12 until the file object has been created on all the computer systems 12.

Referring to FIG. 4, the sequential operations performed by processor 14 of one of the computer systems 12 begin as indicated at a block 400 with opening the distributed file object 18 as indicated at a decision block 402. A data record to be stored is received as indicated at a block 404. The hash algorithm 31 is applied to the received data as indicated at a block 406. Then the hash algorithm result is compared with the partition data 32 as indicated at a block 408. A remote system can be identified as indicated at a decision block 410. When a remote system is identified at block 410, then a connection to the identified system is established as indicated at a block 412. The data record 36 is inserted in the particular remote system as indicated at a block 414. Otherwise, when a remote system is not identified at block 410, then data record is inserted in the system that received the data record at block 414 to complete the operations as indicated at a block 416.

The result of this implementation is that when the distributed file object 18 is opened, and the user wants all the data, all of the remote distributed file objects 18 can immediately opened as well as the local distributed file object 18. Also, there is a very simple process involved, since the information is self-contained in the distributed file object 18. At open time, the open process knows that the file is a distributed object file 18, and can quickly find the communications information 34 needed to establish the remote connections. No other objects or external constructs need to be accessed in order to establish the remote connections. Another benefit results when data is added to the file object 18, or data is updated in the file object because the hash algorithm 31 applied to the new data is stored in each file object 18. Again, the entire process can be handled without accessing external programs, external catalogs, or any other

objects that describe this hashing information. Also, the above process works in reverse, when querying the file for specific values. When the user is querying for specific data, the hash algorithm 31 is applied to the desired data, and the hash result is compared to the partitioning data 32 to immediately identify which system 12 #1, #2, or #3 contains that data. Note that this can only be done when the user has provided a search predicate that involves a test for equality.

Referring now to FIG. 5, an article of manufacture or a computer program product 500 of the invention is illustrated. The computer program product 500 includes a recording medium 502, such as, a floppy disk, a high capacity read only memory in the form of an optically read compact disk or CD-ROM, a tape, a transmission type media such as a digital or analog communications link, or a similar computer program product. Recording medium 502 stores program means 504, 506, 508, 510 on the medium 502 for carrying out the methods of the preferred embodiment in the system 10 of FIG. 1.

A sequence of program instructions or a logical assembly of one or more interrelated modules defined by the recorded program means 504, 506, 508, 510, direct the computer systems 12 for implementing self-describing file objects of the preferred embodiment.

While the present invention has been described with reference to the details of the embodiments of the invention shown in the drawing, these details are not intended to limit the scope of the invention as claimed in the appended claims.

What is claimed is:

1. A method for implementing self-describing file objects comprising the steps of:

creating a node group to define multiple computer systems for storing the file object;

identifying a hash algorithm for applying to data records; identifying a partition distribution map for distributing data to each of said multiple computer systems utilizing a set of predetermined hash algorithm results;

identifying remote system information for each of said multiple computer systems; and

creating a file object in each of said multiple computer systems; each said file object including said hash algorithm, said partition distribution map, and said remote system information.

2. A method for implementing self-describing file objects as recited in claim 1 wherein said step of creating a file object in each of said multiple computer systems; each said file object including said hash algorithm, said partition distribution map, and said remote system information includes the steps of establishing a connection to each remote computer system and storing said file object in each said remote computer system.

3. A method for implementing self-describing file objects as recited in claim 1 wherein said step of identifying said partition distribution map includes the steps of utilizing a set of possible hash values for said identified hash algorithm and providing an equal distribution of said possible hash values to said multiple computer systems.

4. A method for implementing self-describing file objects as recited in claim 1 wherein said step of identifying said partition distribution map includes the steps of utilizing a set of possible hash values for said identified hash algorithm and providing a user selected distribution of said possible hash values to said multiple computer systems.

5. A method for implementing self-describing file objects as recited in claim 1 further includes the steps of receiving