

A METHOD OF OBTAINING DATA FOR A PLANT MONITORING

APPLICATION

TECHNICAL FIELD

- 5 This invention relates generally to data management and relates more particularly to a method of obtaining data for a plant monitoring application.

BACKGROUND

- Plant monitoring is a critical activity to ensure smooth operation of plants that comprise numerous devices running multiple processes, such as heat exchangers, transmitters, pumps, and so on. To monitor a plant efficiently, data from the various devices need to be obtained. Each device is normally provided with a sensor that has an output of data. Each sensor forms a node in the plant. Data from all nodes in the plant is typically transmitted to a server that collects and updates the data for each node, maintaining a proprietary database of data as a plant hierarchy for all the nodes in the plant. Data from the nodes can be in various forms including quantitative numerical values as well as qualitative values such as “on”, “off”, “normal”, “error” and so forth.
- 10
15

- To display the plant hierarchy on a client machine, a Client Access Library (CAL) on the client machine is normally used to retrieve all the data from the server. The server services data requests from a plant monitoring application on the client machine using an OPC (Object linking and embedding for Process Control) interface. Depending on a size of the plant, which may easily comprise thousands of nodes, the plant hierarchy may potentially be very large. Retrieving all the data in the server for display by the
- 20

application can be time consuming. In order for the application to display a current picture of the operation of the plant, all the data needs to be retrieved every time there is an update of data from any one of the nodes. The overall time taken to continually keep all nodes up to date at the client level can therefore be substantial for large plants. The
5 retrieval of the plant hierarchy may be at the expense of performance.

SUMMARY

The invention aims to provide a new and useful method of obtaining data for a plant monitoring application. Using the invention, the application may be provided with
10 current data in a time-saving manner.

In general terms, the invention proposes that a cache on a client machine is maintained to replicate data from a server on a server machine. After loading the cache with a set of data that was previously obtained from the server, subsequent retrievals of data from the
15 server comprise retrieving only data that was updated in the server within a predetermined time period. Retrieving only updated data takes less time compared to retrieving a complete set of data from the server each time there is an update to only one or some of the data in the server.

20 A specific expression of the invention is a method of obtaining data for a plant monitoring application on a client machine, the method comprising updating data stored in a cache on the client machine and associated with a timestamp indicative of a specified time by the steps of a server identifying one or more updated data in the server, each updated data being data updated in the server after the specified time; the client

machine retrieving the identified updated data from the server; loading the identified updated data into the cache on the client machine by substituting the identified updated data for a corresponding data; and updating the timestamp.

5 BRIEF DESCRIPTION OF THE DRAWINGS

In order that the invention may be fully understood and readily put into practical effect there shall now be described by way of non-limitative example only exemplary embodiments, the description being with reference to the accompanying illustrative drawings.

10

In the drawings:

FIG. 1 is a flowchart of a method of obtaining data for an application;

FIG. 2 is an exemplary system architecture diagram for performing the method of FIG. 1 for one server; and

15 FIG. 3 is another exemplary system architecture diagram for performing the method of FIG. 1 for multiple servers.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

As shown in the figures, there is provided a method 100 of obtaining data of a plant 10
20 for a plant monitoring application 12 on a client machine 14. Data of the plant 10 may be maintained by a server processing module 16 on a server machine 18 in the form of a proprietary database 20. The database 20 preferably comprising a plant hierarchy of data obtained from nodes in the plant 10, each node corresponding to a device or process in the plant 10. Data for the application 12 may be obtained from one server

processing module 16 (as shown in system architecture 2 of FIG. 2) or from a plurality of server processing modules 16a, 16b, 16c on corresponding server machines 18a, 18b, 18c (as shown in system architecture 3 of FIG. 3) for display by the application 12. The server processing module 16 services data requests from the plant monitoring application 12 on the client machine 14 using an OPC (Object linking and embedding for Process Control) interface 22. For each server machine 18a 18b, 18c there will be a database 20a, 20b and 20c as well as an OPC 22a, 22b and 22c.

To obtain data from a server processing module 16, the client machine 14 is provided with a client access library (CAL) 24 for retrieving data from the server 12 and loading the data into an internal data cache 26 on the client machine 14 for display by the application 12. Where data is to be retrieved from multiple server processing modules 16a, 16b, 16c, multiple internal data caches 26a, 26b, 26c are preferably provided, one for each server respectively. Each internal data cache 26 or alternatively 26a, 26b, 26c is preferably initialised as a simple in-memory data table of rows and columns. A date-time variable for each internal data cache 26 or alternatively 26a, 26b, 26c is preferably also initialised as a timestamp having a minimum allowable value of 01/01/0001. To cater for data being retrieved from multiple server processing modules 16a, 16b, 16c, a plurality of timestamps are maintained in Coordinated Universal Time for the multiple server processing modules 16a, 16b, 16c respectively. The server processing module 16 is configured to record the time when data for a node in its database 20 is updated, in this way maintaining a history of all data updates in the server processing module 16.

The application 12 comprises a specialised software component called an “asset

service” that initiates a polling thread for each server processing module 16 that data is to be retrieved from. The polling thread performs a polling loop that periodically polls the server processing module 16 for updates in data at intervals of a predetermined time period. The intervals may be of equal duration. Because the polling is performed in the
5 polling thread which is a separate thread, a main user interface of the application 12 can continue to be fully responsive.

When the application 12 is launched (step 101), the asset service is instantiated (step 102). The internal data cache 26 for that server is also initialized (step 104). The asset
10 service preferably checks a local file system 28 for any data previously stored therein (step 106) before beginning the polling loop to periodically poll the server processing module 16 for updates in data for the various nodes in the plant 10.

When the application 12 is first launched, there is no previously stored data in the local
15 file system 28. Upon finding no stored data, the asset service retrieves data from the server processing module 16 (step 110) comprising all data in the database 20 and loads the data into the cache 26 for that server processing module 16 (step 112). The date-time variable is recorded as an initial timestamp (step 114) associated with a specified time upon loading data into the cache 26. Polling of the server processing module 16 is
20 initialized (step 115). After a predetermined time period has elapsed (step 116), the time period being a period of time starting from the specified time associated with the initial timestamp, e.g. 10 minutes, the server processing module 16 identifies one or more data in its database 20 that have been updated after the specified time (step 117) by checking its history of data updates. The CAL is queried to retrieve only the updated data (step

118) from the server processing module 16. Data in the server processing module 16 that has not been updated within the time period are not retrieved by the CAL.

The retrieved updated data is then loaded into the cache 26 (step 120) by substituting
5 the updated data for their corresponding data previously in the cache 26. An updated timestamp is recorded by updating the date-time variable (step 122) upon loading the updated data into the cache 26. Data previously in the cache 26 that has not been updated in the server remain as it was in the cache 26. The polling thread is configured to put itself to “sleep” after each polling of the server processing module 16 and to
10 “wake up” after each time period to repeat the polling loop of the server processing module 16.

The polling thread preferably also checks if there is an attempt to close the application (step 124). If there is no attempt to close the application, after a time period starting
15 from the updated timestamp (step 116), the server processing module 16 is again polled for the CAL 24 to retrieve data that has been updated within that time period (step 118) and to load the updated data into the cache 26 (step 120). By continually updating the timestamp (step 122) upon each loading of updated data into the cache 26 (step 120), and retrieving only data that has been updated within a time period starting from a
20 specified time associated with the updated timestamp that was last recorded (step 118), the cache 26 can be maintained as a collection of current data identical with the latest data in the server processing module 16 while reducing the overall time taken to retrieve data from the server processing module 16.

Preferably, with every retrieval of updated data (step 118), a notification is sent by the server processing module 16 to a graphical user interface of the application 12 (step 126) so that a user may be alerted to any changes in data in the plant hierarchy.

- 5 If the polling thread finds that the application 12 is to be closed, all data currently in the cache 26 are preferably stored as data in the local file system 28 on the client machine 14 (step 128). The date-time variable is recorded as a closing timestamp (step 129) and preferably incorporated into a filename of a file containing the data stored in the local file system 28.

10

- When the application 12 is next launched (step 101), the polling thread is again initiated by the asset service. The local file system 28 is checked for any data previously stored therein (step 106) before beginning the polling loop to periodically poll the server processing module 16 for data updates. Upon detecting the plurality of data stored in the local file system 28, the asset service loads the data from the local file system 28 into the cache 26 (step 132) and initialises the date-time variable (step 112) with the closing timestamp that was recorded when the data was stored in the local files system 28 (step 128) when the application 12 was last closed (step 130). The server processing module 16 is then polled (step 114) and the CAL 24 retrieves only data in the server processing module 16 that has been updated within a time period (step 118) starting from the closing timestamp. Upon loading the updated data into the cache (step 120), the date-time variable is again updated as an updated timestamp (step 112). Each subsequent interval between polling of the server processing module 16 is the time period starting from the updated timestamp of a last loading of updated data into the cache 26.
- 15
- 20

Whilst there has been described in the foregoing description exemplary embodiments of the present invention, it will be understood by those skilled in the technology concerned that many variations in details of design, construction and/or operation may be made
5 without departing from the present invention. For example, server processing module 16 may be configured to asynchronously notify any external system such as the plant monitoring application 12 on the client machine 14 of changes in data in its database 20.

CLAIMS

1. A method of obtaining data for a plant monitoring application on a client machine,
the method comprising updating data stored in a cache on the client machine and
associated with a timestamp indicative of a specified time by the steps of:
 - 5 (a) a polling thread of the application polling a server processing module for
updates in data in a server;
 - (b) the server processing module identifying updated data in the server, each
updated data being data updated in the server after the specified time;
 - (c) the client machine retrieving the identified updated data from the server;
 - 10 (d) loading the identified updated data into the cache on the client machine by
substituting the identified updated data for a corresponding data; and
 - (e) updating the timestamp.
2. The method of claim 1, wherein the data stored in the cache was loaded into the
15 cache from the server before retrieving the updated data.
3. The method of claim 2, wherein the recorded timestamp is an initial timestamp
recorded upon loading the data from the server into the cache when there was no
data stored in the cache.
- 20 4. The method of claim 1, wherein the data stored in the cache was loaded into the
cache from a local file system before retrieving the updated data and the recorded
timestamp is a closing timestamp recorded when the data was stored in the local
file system when the application was last closed.

5. The method of any one of the preceding claims, further comprising at least once repeating the set of steps of (a) to (e), with a predetermined time interval between successive said sets of steps.
- 5 6. The method of any one of the preceding claims, further comprising storing all data in the cache as data in a local file system when the application is to be closed and recording a closing timestamp for data stored in the local file system.
7. The method of claim 4 or claim 6, wherein the closing timestamp is incorporated
10 into a filename of a file containing the data stored in the local file system.
8. The method of any one of the preceding claims, further comprising the server notifying the application whenever data in the server is updated.
- 15 9. The method of any one of the preceding claims, further comprising supplying the application with data from the cache.
10. The method of any one of the preceding claims, wherein a plurality of caches is provided for loading updated data into each one of the plurality of caches and
20 supplying the application with data retrieved from a respective one of a plurality of servers.
11. The method of claim 10, wherein a plurality of timestamps are maintained in a Coordinated Universal Time for retrieving data from the plurality of servers

respectively.

12. A system for obtaining data for a plant monitoring application on a client machine using the method of any one of claims 1 to 11, the system comprising:
- 5 a cache on a client machine for storing data therein, each datum being associated with a timestamp indicative of a specified time;
- a server configured to maintain a database of data therein and to identify one or more data in the server updated after the specified time;
- the client machine being configured to retrieve the identified updated
- 10 data from the server and to load the identified updated data into the cache on the client machine by substituting the updated data for a corresponding one or more of the data, the client machine being further configured to update the timestamp.