

Further objects, features and advantages of the present invention will become apparent to the ones skilled in the art upon examination of the following description in reference to the accompanying drawings. It is intended that any additional advantages
 5 be incorporated herein.

SUMMARY OF THE INVENTION

According to one aspect, the invention relates to a method of maintaining consistency between copies of a replicated file distributed over a plurality of nodes of a data processing system, the method comprising:

10 receiving an update request in a master server, the update request requesting that the replicated file be updated in a master database;

generating and storing a new version of the replicated file in a shared file system based on the update request;

15 providing a notification of availability of the new version of the replicated file to a plurality of slave nodes;

in response to receiving the notification of availability at each slave node, preloading a copy of the new version of the replicated file from the shared file system into a memory of each slave node, and upon completion of the preloading, acknowledging successful completion of the preloading; and

20 committing the data processing system to use the new version of the replicated file only if each slave node of the plurality of slave nodes acknowledges successful completion of the preloading to the master server. ~~the invention describes a method of maintaining consistency of replicated files distributed over a plurality of independent processing slave nodes part of an application tier of a multi-tier client/server data processing system, the replicated files being distributed from a master node of a master tier. The method comprises the following steps performed with at least a processor:~~

~~—receiving an update request in at least a master server of the master tier to update a master database of the data processing system;~~

30 ~~—based on said update generating and storing a new version of a replicated file stored in a shared file system of the master tier;~~

~~—providing a notification of availability of the new version of the replicated file to all slave nodes;~~

35 ~~—in each slave node, starting preloading from the shared file system the new version of the replicated file and upon completion of the preloading, acknowledging successful completion of the preloading;~~

•only if all slave nodes acknowledge successful completion of the preloading then performing the following steps:

- receiving at the master server a notification of preloading achievement;
- from the master server, updating the master database with data of the new version of the replicated file thus committing the use of the new version of the replicated file by the master tier;
- committing the use of the new version of the replicated file in a slave database of the application tier arranged for keeping track of all replicated file versions;
- forwarding to the master server a notification of commitment of all slave nodes;
- from master server, acknowledging achievement of the update in response to the received update request;

•if not all slave nodes acknowledge successful completion of the preloading, then receiving at the master node a notification of error without updating the master database and committing the use of the new version in the slave database.

Thus the invention provides an efficient solution for responding to a query for information in near real-time while ensuring a strong consistency over a possibly high number of nodes having the piece of information required for responding to the query. Therefore, the invention enables maintaining strong consistency, availability with a very short latency and enhanced scalability.

Optionally, the invention may comprise at least one of the following optional features:

In the present invention, what is designated as replicated files are complete files that are replicated from the master base and brought as such into the slave nodes to expedite the processing of the corresponding data by the applicative processes. They are not per se small pieces of data brought from a slower back end memory into a cache memory. In the context of the invention replicated files, also referred to as replica are indeed complete files on which an applicative process in a slave node can work without experiencing any miss of data to complete.

According to a particular but not limitative embodiment, the replicated files, also referred as replica, are cache files.

The step of providing a notification of availability of the new version of the replicated file to all slave nodes comprises the following steps:

- forwarding from the master server a notification of availability of the new version of the replicated file to a synchronizing slave node taken among the slave nodes;

- broadcasting from the synchronizing slave node said notification of availability to all other slave nodes.

The step of acknowledging successful completion of the preloading comprises a step wherein the slave nodes acknowledge successful completion of the preloading to a server process of the synchronizing slave node.

Once all slave nodes acknowledge successful completion of the preloading to the server process of the synchronizing slave node and before the step of receiving at the master server a notification of preloading achievement, the synchronizing slave node forwards to the master server said notification of preloading achievement.

Upon completion of the step of committing the use of the new version of the replicated file by the master tier, the master node forwards a commitment notification to the synchronizing slave node and then the synchronizing slave node triggers the step of committing the use of the new version of the replicated file in the slave database.

At beginning of a transaction initiated by a query for information from a user, at least in each slave node involved in said transaction an applicative process interrogates the slave database and is instructed whether to switch from a current version of the replicated file to the preloaded new version of the replicated file.

Preferably, at beginning of a transaction initiated by a query for information from a user, in each slave node involved in said transaction an applicative process reads a tag of a current version of the replicated file, said tag indicating if a new version of the replicated file is available. If the tag indicates that a new version of the replicated file is available, then the applicative process interrogates the slave database and is instructed whether to switch from the current version of the replicated file to the new version of the replicated file.

The tag is set at successful completion of preloading of a new version of the replicated file and is reset at first reading by the applicative process.

The received update request comprises at least an update and the steps of generating and storing a new version of a replicated file in the shared file system is an incremental process including, in the master server, the steps of:

- retrieving from the shared file system a current version of a replicated file and converting to an appropriate format the current version of the replicated file to be updated;
- applying the update onto the current version and converting and storing the new version of the replicated file into the shared file system.

The replicated files are compressed when stored in the shared file system.

The replicated files stored in the shared file system are decompressed and converted into an appropriate format by the master server when performing the incremental update and by the slave nodes upon preloading.

5

If not all slave nodes acknowledge successful completion of the preloading, then the master node sends a notification of error to an administrator that sent said update request.

10 The master database can be stored in a single machine or in a plurality of machines of the master tier. The slave database can be stored in a single machine or in a plurality of machines of the application tier.

The shared file system is a network-attached storage NAS.

15 According to another aspect, the invention relates to a computer program product comprising a non-transitory computer-readable storage medium including program instructions that, when executed by one or more processors, cause the one or more processors to:

20 receive an update request in a master server, the update request requesting that a replicated file be updated in a master database;

generate and store a new version of the replicated file in a shared file system based on the update request;

provide a notification of availability of the new version of the replicated file to a plurality of slave nodes;

25 in response to receiving the notification of availability at each slave node, preload a copy of the new version of the replicated file from the shared file system into a memory of each slave node, and upon completion of the preloading, acknowledge successful completion of the preloading; and

30 commit the data processing system to use the new version of the replicated file only if all slave nodes of the plurality of slave nodes acknowledge successful completion of the preloading, stored in a non-transitory computer-readable memory medium and performing the method according to any one of the preceding features.

35 According to another aspect, the invention relates to a method of maintaining consistency between copies of a replicated file containing business rules, the copies of

the replicated file being distributed over a plurality of nodes of a data processing system, the method comprising:

receiving an update request in a master server to update business rules stored in a master database;

5 generating and storing a new version of the replicated file in a shared file system based on the update request;

providing a notification of availability of the new version of the replicated file to a plurality of slave nodes;

10 in response to receiving the notification of availability at each slave node, preloading a copy of the new version of the replicated file from the shared file system into a memory of each slave node, and upon completion of the preloading, acknowledging successful completion of the preloading; and

15 committing the data processing system to use the new version of the replicated file only if all slave nodes of the plurality of slave nodes acknowledge successful completion of the preloading.

20 ~~a method of maintaining consistency of replicated files containing business rules, said replicated files being distributed over a plurality of independent processing slave nodes forming a part of an application tier of a multi-tier client/server data processing system, the replicated files being distributed from a master node of a master tier, wherein the method comprises the following steps performed with at least a data processor:~~

~~—receiving an update request in at least a master server of the master tier to update business rules stored in a master database of the data processing system;~~

25 ~~—based on said update generating and storing a new version of a replicated file stored in a shared file system of the master tier;~~

~~—providing a notification of availability of the new version of the replicated file to all slave nodes;~~

30 ~~—in each slave node, starting preloading from the shared file system the new version of the replicated file and upon completion of the preloading, acknowledging successful completion of the preloading;~~

~~•only if all slave nodes acknowledge successful completion of the preloading then performing the following steps:~~

~~—receiving at the master server a notification of preloading achievement;~~

35 ~~—from the master server, updating the master database with data of the new version of the replicated file thus committing the use of the new version of the replicated file by the master tier;~~

~~if not all the slave nodes acknowledge successful completion of the preloading, then receiving at the master node a notification of error.~~

Optionally, the invention may comprise at least one of the following optional features.

5 Advantageously, each business rule comprises a set of criterion and a content. Preferably, each criterion is associated to a weight, allowing thereby a search engine to identify the most relevant rule when a search is launched.

10 The set of criterion comprises at least one of: a point of sale, a country, a group of countries, a country of origin of a flight, a destination of a flight, a customer profile.

 The multi-tier client/server data processing system is part of an inventory of a travel provider.

15 More generally, the slave nodes are connected to at least one of: an inventory of an airline, a departure control system of an airline, a revenue management system of an airline, a revenue accounting system of an airline, an electronic ticket server of an airline.

 Another aspect of the invention is a distributed data processing system comprising:

20 a master tier including a master server and a shared file system; and
an application tier including a plurality of slave nodes, each slave node having a memory and being communicatively coupled to the master tier, wherein the master server is configured to provide a notification of availability of a new version of a replicated file in the shared file system to the plurality of slave nodes;

25 each slave node of the plurality of slave nodes is configured to, in response to receiving the notification of availability, preload a copy of the new version of the replicated file from the shared file system, and upon completion of the preloading, acknowledge successful completion of the preloading; and

30 the master server is further configured to only commit the data processing system to use the new version of the replicated file in response to all slave nodes acknowledging successful completion of the preloading.~~distributed multi-tier client/server data processing system comprising an application tier and a master tier, the application tier comprising a plurality of independent processing slave nodes. The master tier comprises a master node arranged for distributing replicated files to the slave nodes, a master database and a master server. The master tier comprises a~~
35 ~~shared file system. The master server is arranged for receiving an update request to~~

update the master database, to generate a new version of a replicated file stored in the shared file system and to store said new version in the shared file system. The system according to the invention also comprises a slave database connected to all slave nodes and arranged for keeping track of all replicated file versions. The system is configured for:

- providing a notification of availability of the new version of the replicated file to all slave nodes;
- in each slave node, starting preloading from the shared file system the new version of the replicated file and upon completion of the preloading, acknowledging successful completion of the preloading;
- only if all slave nodes acknowledge successful completion of the preloading then performing the following steps: receiving at the master server a notification of preloading achievement ; from the master server, updating the master database with data of the new version of the replicated file thus committing the use of the new version of the replicated file by the master tier ; committing the use of the new version of the replicated file in the slave database ; forwarding to the master server a notification of commitment of all slave nodes; from master server, acknowledging achievement of the update in response to the received update request;
- if not all the slave nodes acknowledge successful completion of the preloading, then receiving at the master node a notification of error without updating the master database and without committing the use of the new version in the slave database.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 illustrates an example of a standard distributed 3-tier data processing system.

CLAIMS

1. A method of maintaining consistency of replicated files (250) distributed over a plurality of independent processing slave nodes (210, 210', 210'') part of an application tier (200) of a multi-tier client/server data processing system, the replicated files (250, 150) being distributed from a master node (110) of a master tier (100), the method characterized in that it comprises the following steps performed with at least one data processor:
- receiving (1) an update request (142) in at least a master server (112) of the master tier (100) to update a master database (120) of the data processing system;
 - based on said update generating and storing (2) a new version (150) of a replicated file stored in a shared file system (160) of the master tier (100);
 - providing (3, 4) a notification of availability of the new version (150) of the replicated file to all slave nodes (210, 210', 210'');
 - in each slave node (210, 210', 210''), starting preloading (51) from the shared file system (160) the new version (150) of the replicated file and upon completion of the preloading, acknowledging (52) successful completion of the preloading;
 - only if all slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading then performing the following steps:
 - receiving at the master server (112) a notification of preloading achievement (6);
 - from the master server (112), updating (7) the master database (120) with data of the new version (150) of the replicated file thus committing the use of the new version (150) of the replicated file by the master tier (100);
 - committing (9) the use of the new version (150) of the replicated file in a slave database (260) of the application tier (200) arranged for keeping track of all replicated file versions (150, 250);
 - forwarding to the master server (112) a notification of commitment (10) of all slave nodes (210, 210', 210'');
 - from master server (112), acknowledging (11) achievement of the update in response to the received update request (142),
 - if not all the slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading, then receiving at the master node (110) a notification of error.
2. The method according to the previous claim wherein the step of providing (3, 4) a notification of availability of the new version (150) of the replicated file to all slave nodes (210, 210', 210'') comprises the following steps:

– forwarding (3), from the master server (112), a notification of availability of the new version (150) of the replicated file to a synchronizing slave node (210) taken among the slave nodes (210, 210', 210'');

– broadcasting (4) from the synchronizing slave node (210) said notification of availability to all other slave nodes (210', 210'').

3. The method according to the previous claim wherein the step of acknowledging (52) successful completion of the preloading comprises a step wherein the slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading to a server process (220) of the synchronizing slave node (210).

4. The method according to the previous claim wherein once all slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading to the server process (220) of the synchronizing slave node (210) and before the step of receiving at the master server (112) a notification of preloading achievement (6), the synchronizing slave node (210) forwards to the master server (112) said notification of preloading achievement (6).

5. The method according to any one of the three previous claims wherein upon completion of the step of committing the use of the new version (150) of the replicated file by the master tier (100), the master node (110) forwards a commitment (8) notification to the synchronizing slave node (210) and then the synchronizing slave node (210) triggers the step of committing (9) the use of the new version (150) of the replicated file in the slave database (260).

6. The method according to any one of the five preceding claims wherein at beginning of a transaction initiated by a query for information from a user, at least in each slave node (210, 210', 210'') involved in said transaction an applicative process (240, 240', 240'') interrogates (264) the slave database (260) and is instructed whether to switch (12) from a current version (250) of the replicated file to the preloaded new version (150) of the replicated file.

7. The method according to any one of claims 1 to 5 wherein at beginning of a transaction initiated by a query for information from a user, in each slave node (210, 210', 210'') involved in said transaction an applicative process (240, 240', 240'') reads a tag of a current version (250) of the replicated file, said tag indicating if a new version (250) of the replicated file is available, and wherein if the tag indicates that a new version (150) of the replicated file is available, then the applicative process (240, 240', 240'') interrogates (264) the slave database (260) and is instructed whether to switch (12) from the current version (250) of the replicated file to the new version (150) of the replicated file.

8. The method according to the previous claim wherein the tag is set at successful completion of preloading (51) of a new version (150) of the replicated file and is reset at first reading by the applicative process (240, 240', 240").
9. The method according to any one of the preceding claims wherein the received
5 update request (142) comprises at least an update and wherein the steps of generating and storing a new version (150) of a replicated file in the shared file system (160) is an incremental process including, in the master server (112), the steps of:
- retrieving from the shared file system (160) a current version (250) of a replicated
file and converting (152) to an appropriate format (114) the current version of the
10 replicated file to be updated;
- applying the update (142) onto the current version and converting and storing (154) the
new version (150) of the replicated file into the shared file system (160).
10. The method according to the previous claim wherein the replicated files (150,
250) are compressed when stored in the shared file system (160) and wherein the
15 replicated files stored in the shared file system (160) are decompressed and converted
into an appropriate format by the master server (112) when performing the incremental
update (152), and by the slave nodes (210, 210', 210") upon preloading (51).
11. The method according to claim 1 wherein if not all slave nodes (210, 210', 210")
acknowledge (52) successful completion of the preloading, then the master node (110)
20 sends a notification of error to an administrator that sent said update request (142).
12. The method according to any one of the preceding claims wherein the shared file
system (160) is a network-attached storage (NAS).
13. The method according to any one of the preceding claims wherein the slave
database (260) only stores data related to the version of replicated files.
- 25 14. Computer program product stored in a computer-readable memory medium and
performing the method according to any one of the preceding claims.
15. A distributed multi-tier client/server data processing system comprising an
application tier (200) and a master tier (100), the application tier (200) comprising a
plurality of independent processing slave nodes (210, 210', 210") and the master tier
30 (100) comprising a master node (110) arranged for distributing replicated files (250,
150) to the slave nodes (210, 210', 210"), a master database (120) and a master
server (112), characterized in that: the master tier (100) comprises a shared file system
(160), the master server (112) being arranged for receiving (1) an update request (142)
to update the master database (120), to generate a new version (150) of a replicated
35 file stored in the shared file system (160) and to store (2) said new version (150) in the
shared file system (160); in that it comprises a slave database (260) connected to all

slave nodes (210, 210', 210'') and arranged for keeping track of all replicated file versions (150, 250); and in that the system is configured for:

- providing (3, 4) a notification of availability of the new version (150) of the replicated file to all slave nodes (210, 210', 210'');
- 5 • in each slave node (210, 210', 210''), starting preloading (51) from the shared file system (160) the new version (150) of the replicated file and upon completion of the preloading, acknowledging (52) successful completion of the preloading;
- only if all slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading then performing the following steps: receiving at the master server (112)
- 10 a notification of preloading achievement (6); from the master server (112), updating (7) the master database (120) with data of the new version (150) of the replicated file thus committing the use of the new version (150) of the replicated file by the master tier (100); committing (9) the use of the new version (150) of the replicated file in the slave database (260); forwarding to the master server (112) a notification of commitment (10) of
- 15 all slave nodes (210, 210', 210'') ; from master server (112), acknowledging (11) achievement of the update in response to the received update request (142),
- if not all the slave nodes (210, 210', 210'') acknowledge (52) successful completion of the preloading, then receiving at the master node (110) a notification of error.

CLAIMS

1. A method of maintaining consistency between copies of a replicated file distributed over a plurality of nodes of a data processing system, the method comprising:
 - 5 receiving an update request in a master server, the update request requesting that the replicated file be updated in a master database;
 - generating and storing a new version of the replicated file in a shared file system based on the update request;
 - providing a notification of availability of the new version of the replicated file to a
 - 10 plurality of slave nodes;
 - in response to receiving the notification of availability at each slave node, preloading a copy of the new version of the replicated file from the shared file system into a memory of each slave node, and upon completion of the preloading, acknowledging successful completion of the preloading; and
 - 15 committing the data processing system to use the new version of the replicated file only if each slave node of the plurality of slave nodes acknowledges successful completion of the preloading to the master server.
2. The method of claim 1 wherein providing the notification of availability of the new
- 20 version of the replicated file to the plurality of slave nodes comprises:
 - forwarding, from the master server, the notification of availability of the new version of the replicated file to a synchronizing slave node selected from the plurality of slave nodes; and
 - broadcasting from the synchronizing slave node the notification of availability to
 - 25 the other remaining slave nodes of the plurality of slave nodes.

3. The method of claim 2 wherein each slave node acknowledges successful completion of the preloading to a server process of the synchronizing slave node.

4. The method of claim 3 further comprising:

5 in response to all the other remaining slave nodes of the plurality of slave nodes acknowledging successful completion of the preloading to the server process of the synchronizing slave node, forwarding a notification of preloading achievement from the synchronizing slave node to the master server.

10 5. The method of claim 2 wherein committing the data processing system to use the new version of the replicated file comprises:

 forwarding a commitment notification to the synchronizing slave node from the master server; and

 in response to receiving the commitment notification, triggering with the
15 synchronizing slave node the use of the new version of the replicated file in a slave database.

6. The method of claim 5 wherein triggering the use of the new version of the replicated file in the slave database comprises:

20 updating a version table in the slave database with a version number of the new version of the replicated file.

7. The method of claim 1 further comprising:

 in response to a transaction initiated by a query for information from a user,
25 interrogating a slave database at each slave node involved in the transaction; and

based upon the interrogation, instructing each slave node involved in the transaction whether to switch from a current version of the replicated file to the preloaded new version of the replicated file.

- 5 8. The method of claim 7 wherein the slave database only stores data related to a version of replicated files.

9. The method of claim 1 further comprising:

in response to a transaction initiated by a query for information from a user,
10 reading in each slave node involved in said transaction a tag of a current version of the replicated file, the tag indicating if a new version of the replicated file is available;

in response to the tag indicating that a new version of the replicated file is available, interrogating a slave database of each slave node involved in the transaction;
and

15 in response to being interrogated, instructing the interrogating slave node whether to switch from the current version of the replicated file to the new version of the replicated file.

10. The method of claim 9 wherein the tag is set at successful completion of preloading
20 of the new version of the replicated file, and is reset at a first reading by the slave node.

11. The method of claim 1 wherein the received update request includes an update for the replicated file, generating and storing the new version of the replicated file in the shared file system is an incremental process in the master server comprising:

25 retrieving from the shared file system a current version of the replicated file;

converting the current version of the replicated file to an appropriate format to be updated;

applying the update to the current version of the replicated file to generate the new version of the replicated file; and converting the new version of the replicated file

5 for storage into the shared file system; and

storing the new version of the replicated file into the shared file system.

12. The method of claim 11 wherein the replicated file is compressed when stored in the shared file system, and converting the current version of the replicated file to the

10 appropriate format comprises:

decompressing the current version of the replicated file stored in the shared file system with the master server; and

preloading the copy of the new version of the replicated file includes decompressing the new version of the replicated file with the each slave node.

15

13. The method of claim 1 further comprising:

in response to at least one of the slave nodes of the plurality of slave nodes failing to acknowledge successful completion of the preloading, sending an error notification from the master server to a sender of the update request.

20

14. The method of claim 1 wherein the shared file system comprises a network-attached storage (NAS).

15. The method of claim 1 wherein committing the data processing system to use the

25 new version of the replicated file comprises:

updating the replicated file in the master database to the new version of the replicated file;

updating data in a slave database configured to keep track of replicated file versions used by the plurality of slave nodes, the data identifying a version of the replicated file to be used by the plurality of slave nodes; and

in response to updating the data identifying the version of the replicated file in the slave database, receiving in the master server a notification of commitment of each slave node of the plurality of slave nodes to use the new version of the replicated file.

10 16. The method of claim 15 further comprising:

in response to each slave node of the plurality of slave nodes acknowledging successful completion of the preloading, receiving a notification of preloading achievement in the master server, wherein updating the replicated file in the master database to the new version of the replicated file is in response to receiving the notification of preloading achievement.

17. A computer program product comprising:

a non-transitory computer-readable storage medium including program instructions that, when executed by one or more processors, cause the one or more processors to:

receive an update request in a master server, the update request requesting that a replicated file be updated in a master database;

generate and store a new version of the replicated file in a shared file system based on the update request;

provide a notification of availability of the new version of the replicated file to a plurality of slave nodes;

in response to receiving the notification of availability at each slave node,
 preload a copy of the new version of the replicated file from the shared file system into
 a memory of each slave node, and upon completion of the preloading, acknowledge
 successful completion of the preloading; and

5 commit the data processing system to use the new version of the replicated file
 only if all slave nodes of the plurality of slave nodes acknowledge successful
 completion of the preloading.

18. A method of maintaining consistency between copies of a replicated file containing
 10 business rules, the copies of the replicated file being distributed over a plurality of
 nodes of a data processing system, the method comprising:

receiving an update request in a master server to update business rules stored
 in a master database;

generating and storing a new version of the replicated file in a shared file
 15 system based on the update request;

providing a notification of availability of the new version of the replicated file to a
 plurality of slave nodes;

in response to receiving the notification of availability at each slave node,
 preloading a copy of the new version of the replicated file from the shared file system
 20 into a memory of each slave node, and upon completion of the preloading,
 acknowledging successful completion of the preloading; and

committing the data processing system to use the new version of the replicated
 file only if all slave nodes of the plurality of slave nodes acknowledge successful
 completion of the preloading.

25

19. The method of claim 18 wherein each business rule comprises a set of criteria and

a content.

20. The method of claim 19 wherein each criterion of the set of criteria is associated to a weight.

5

21. The method of claim 19 wherein the set of criteria comprises a point of sale, a country, a group of countries, a country of origin of a flight, a destination of a flight, or a customer profile.

10 22. The method of claim 18 wherein an inventory of a travel provider comprises the data processing system.

23. The method of claim 18 wherein at least one of the slave nodes is connected to an inventory of an airline, a departure control system of an airline, a revenue management
15 system of an airline, a revenue accounting system of an airline, or an electronic ticket server of an airline.

24. The method of claim 18 wherein committing the data processing system to use the new version of the replicated file comprises:

20 committing the use of the new version of the replicated file in a slave database configured to keep track of all replicated file versions;

forwarding a notification of commitment of all slave nodes to the master server;
and from the master server, acknowledging achievement of the update request in response to the received update request.

25

25. A distributed data processing system comprising:

a master tier including a master server and a shared file system; and

an application tier including a plurality of slave nodes, each slave node having a memory and being communicatively coupled to the master tier, wherein the master server is configured to provide a notification of availability of a new version of a

5 replicated file in the shared file system to the plurality of slave nodes;

each slave node-of the plurality of slave nodes is configured to, in response to receiving the notification of availability, preload a copy of the new version of the replicated file from the shared file system, and upon completion of the preloading, acknowledge successful completion of the preloading; and

10 the master server is further configured to only commit the data processing system to use the new version of the replicated file in response to all slave nodes acknowledging-successful completion of the preloading.