METHOD FOR ENABLING FASTER RECOVERY OF CLIENT APPLICATIONS IN THE EVENT OF SERVER FAILURE

Failure of server node

Fallover to backup node

Notification to client node

Termination of connection to server node

Fig. 2

Abstract: A system and method are provided for improving recovery times in fallover conditions in a multinode data processing system by sending notification of the failure of a server node, which is acting as server for a client application running on a client node, to the client application. In the present invention, this notification is provided by the fallover node acting as backup for the server node. When a client application receives no response from a server for a long time, it assumes that the server has failed and initiates reconection. The present invention speeds-up the reconnect initiated by the client application by having system level software proactively notify the client application about the server failure. This results in faster recovery for client applications.
Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— with international search report (Art. 21(3))
— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))
METHOD FOR ENABLING FASTER RECOVERY OF CLIENT APPLICATIONS IN THE EVENT OF SERVER FAILURE

Technical Field

The present invention is generally directed to multinode data processing systems and methods. More particularly, the present invention is directed to a system and method for improving recovery time in the event of a server failure. Even more particularly, the present invention is directed to a system in which a faullover node provides failure notification as opposed to waiting for a determination of failure based on the lack of performance by the failed node. As used herein and in the appended claims, the terms "node" and "server" are used interchangeably, though it is understood that a server may include several nodes.

Background of the Invention

In clusterware applications such as HACMP (High Availability Cluster Management Program), there exists many client applications which are intelligent enough to automatically reconnect to a backup server when the primary server fails. This is made possible by dynamically moving the IP (Internet Protocol) address of the primary server, with which a client was interacting, to the backup server. Accordingly, client applications do not have to be killed and/or restarted in response to a server failure. This capability of clusterware is referred to herein as "faullover." However, the problem is that, as soon as the primary server fails, the TCP (Transmission Control Protocol) connection existing between the client and the server takes some time to get terminated. This is primarily because client applications are unaware of the server failure and so the client application keeps retransmitting until the TCP retransmit counter expires, thus eventually terminating the TCP connection.

Accordingly, it is seen that it takes a considerable amount of time for client applications to recognize the server failure event before attempting a reconnect to backup server. From the above, it is therefore seen that there exists a need in the art to overcome the deficiencies and limitations described herein and above.
Summary of the Invention

The delay described above is avoided by timely notification of the failure event to the client. The client preferably reconnects as soon as it receives this notification about the server failure. The present invention preferably provides a mechanism for speeding up the process of reconnection, to one or more backup servers within cluster, by a client application. To quickly terminate all of the TCP connections existing between the server (the presumably failed node) and the client, the client preferably receives a RST (Reset Packet) for that TCP connection as soon as the server fails. Despite the failure of the node from which this RST packet is expected, the method of the present invention still preferably generates this RST packet. To achieve this, as soon as the fallover happens, the fallover node (backup server) preferably sends this RST packet to the client. In effect the fallover node pretends to be the failed node for the client.

In accordance with one embodiment of the present invention, a method for improving recovery in fallover conditions comprises the step of sending notification of the failure of a server node, which is acting as server for a client application running on a client node, to the client application. This notification is provided by the fallover node acting as backup for the server node.

 Preferably the fallover node masquerades as the server node in order to send the notification of failure. In accordance with a preferred embodiment, a data packet is received from the client application. This packet is accessed to determine transmission information and the transmission information is used to send notification of failure of the server node to the client application.

In accordance with a preferred embodiment, the data packet is one that has been retransmitted by the client application when no acknowledgement of the original transmission of the data packet to the server node was received by the client application. In one embodiment, the fallover node informs said client node that the fallover node has taken over for said server node.
Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

The recitation herein of desirable objects which are met by various embodiments of the present invention is not meant to imply or suggest that any or all of these objects are present as essential features, either individually or collectively, in the most general embodiment of the present invention or in any of its more specific embodiments.

Brief Description of the Drawings

Preferred embodiments of the present invention will now be described, by way of example only, and with reference to the following drawings:

FIG. 1 is a block diagram illustrating the connections between a basic node configuration environment in which the present invention is employed and also illustrates a sequence of events that occurs in the operation of a preferred embodiment of the present invention; and

FIG. 2 is a flow chart illustrating a sequence of events that typically occurs in the process of a preferred embodiment of the present invention.

Detailed Description

As previously discussed, a client may be communicating with a server using, for example, the TCP protocol. If the server fails, the client may not notice that this has happened. In the absence of any acknowledgements to the packets the client sends out, the client will keep retransmitting until the TCP retransmit counter expires. This could be quite some time later. It is desirable therefore to force the connection between the client and the server to be terminated and reset. The TCP protocol has a RESET (RST) option. When two parties are communicating, if the first party notices that the other party has stopped communicating, then the first party may send an RST to force the other party to reset the connection. The point here is however that the client may not realize for quite some time that there is a
problem. When the client does eventually realize that there is a problem, the client will then attempt to reconnect and will be connected to a fallover node.

The invention, in accordance with a preferred embodiment, makes use of a backup (or fallover) server which monitors a server node with which the client is communicating. When the fallover node notices that the server node has failed, the fallover node takes over the address of the server node. At this point the backup server looks at all traffic on the taken over IP address to see if any is destined for (i.e. addressed to) the application that used to run on the server node, but has been restarted on the backup node along with the IP address. If it finds any it then 'masquerades' as the server node.

The fallover node however has no knowledge of the connections that were held by the server node. The fallover node doesn't know who held them or what type of connections they held. The fallover node therefore needs to be able to deduce who the server node had a connection with and to then 'encourage' (using the RST option) such clients to connect to the fallover node. Naturally this would be transparent to any such clients who would think they were re-connecting to the server node.

For this to work, the client has to believe that the RST packet came from the server node. In a TCP connection every packet has a sequence number and "ACK" number which insures that data is properly ordered at the end of the transmission at the receiver. At any point of time, if a TCP receives a packet with incorrect sequence number or ACK number, then it immediately discards that packet. So, for any application "A" to be able to send packets on a TCP connection existing between two other applications "B" and "C," application "A" has to determine the sequence number, ACK number, and TCP header information for the next packet in the TCP sequence to be sent to the client.

Similarly, for the backup server (fallover node) to be able to send a RST (Reset) packet to the client it has to determine appropriate TCP header information for that RST packet. In a TCP connection, when either of the communicating parties dies without terminating the connection by sending a FIN packet (which is used in TCP to indicate an end of transmission) or RST packet, the other party begins to retransmit the old packets again.
In the preferred embodiment, the fallover node has taken over the address of the server. The fallover node therefore receives some retransmitted packets from the client as soon as it acquires (by activating) the IP address from the failed node. The backup server (fallover node) obtains TCP header information of the next packet in the sequence using the retransmitted packet it just received from the client. This TCP header information enables the backup server to determine for example what sequence number the RST packet should use and where the RST packet should be sent.

Using raw IP sockets, any application can handcraft a packet and send it on a TCP connection existing between two different applications. This newly crafted packet appears to be a genuine packet for the receiver since it thinks that it is sent by its peer application.

In the present invention, clusterware such as HACMP handcrafts a RST packet using raw IP sockets and send it to the client. This RST packet breaks the old TCP connection immediately and hence a client can reconnect again quickly.

In this invention, there is no need to change either the existing client applications or the TCP stack itself. Irrespective of whether the TCP connection is terminated by TCP connection time-out or by RST packet, the client application views it as abrupt termination and attempts a reconnect.

Figure 1 illustrates, in accordance with a preferred embodiment, a sequence of events or situations in the use of the present invention. In particular, the "1" in the circle indicates an initial state in which client node 100 is in communication with server node 200. The "2" in the circle indicates the event in which server node 200 fails and there is a fallover to fallover node 300. The "3" in the circle indicates the event in which fallover node 300 communicates to client node 100 that server node 200 has failed and that it is taking over the communication function. The "4" in the circle indicates a final status in which client node 100 is now connected to fallover node 300. The illustrated process avoids the necessity of an application running on client node 100 having to wait for an indication of failure for node 200 which is long in coming since application level software is not typically aware of node failures. The fallover process, running at system level speeds and priorities occurs much
faster and this fact along with the protocol structure of TCP connections allows a much faster notification to a client based server application.

Figure 2 illustrates a typical sequence of events that occurs in the process of the present invention. The process is triggered by the failure of server node 200 (step 101). System level software detects this failure in a relatively rapid fashion and provides a fallover to fallover (or backup) node 300 (step 102). Fallover node 300 then provides notification to client node 100 (step 103). Client node 100 then terminates the connection to node 200 (step 104).

The present invention also encompasses software in the form of machine readable instructions for carrying out the process recited herein. Furthermore, the present invention encompasses data processing systems having such instructions disposed within the memories thereof for carrying out the recited process.

While the invention has been described in detail herein in accordance with certain preferred embodiments thereof, many modifications and changes therein may be effected by those skilled in the art. Accordingly, it is intended by the appended claims to cover all such modifications and changes as fall within the spirit and scope of the invention.
CLAIMS

1. A method for improving recovery in failover conditions, said method comprising the step of:
   ending notification of failure of a server node, acting as server for a client application running on a client node, to said client application, said notification being provided by a failover node acting as backup for said server node.

2. The method of claim 1 further including the step of terminating connection to said server node.

3. The method of claim 1 or 2 further including the step of activating at the failover node the IP address originally associated with the server node.

4. The method of claim 3 further including the step of receiving, at said failover node, retransmitted packets at said client node.

5. The method of claim 4 further including the step of obtaining, at said failover node, TCP header information.

6. The method of claim 5 in which said header information includes packet sequence number.

7. The method of claim 5 or 6 further including the step of constructing a Reset packet at said failover node.

8. The method of claim 7 in which said Reset packet construction uses raw sockets.

9. The method of claim 7 or 8 further including the step of transmitting said Reset packet to said application running on said client node.
10. The method of claim 7, 8 or 9 further including the step of terminating connection from said client node to said server node.

11. The method of claim 10 further including the step of attempting reconnection by said application.

12. The method of any preceding claim in which said server node and said client node are in communication via Transmission Control Protocol.

13. The method of any preceding claim in which said fallover node informs said application running on said client node that said server node has failed.

14. The method of claim 13 in which said fallover node is operable to inform said client node that the fallover node has taken over for said server node.

15. The method of any preceding claim in which said method is triggered by failure of said server node.

16. The method of any preceding claim in which said notification is provided earlier than notification provided through a Transmission Control Protocol connection.

17. The method of any preceding claim, wherein the fallover node masquerades as the server node in order to send the notification of failure, the method comprising:

   receiving a data packet from the client application; and

   accessing the data packet to determine transmission information, wherein the step of sending notification of failure comprises:

   using the transmission information to send notification of failure of the server node to the client application.

18. The method of claim 17 wherein the data packet is one that has been retransmitted by the client application when no acknowledgement of the original transmission of the data packet to the server node was received by the client application.
19. A multinode data processing system including at least three nodes containing program instructions in memory for said nodes for sending notification of failure of a server node, which is acting as server for a client application running on a client node, to said client application, said notification being provided by a fallover node acting as backup for said server node.

20. The system of claim 19 further including means for terminating connection to said server node.

21. The system of claim 19 or 20 further including means for activating at the fallover node the IP address originally associated with the server node.

22. The system of claim 21 further including means for receiving, at said fallover node, retransmitted packets at said client node.

23. The system of claim 22 further including means for obtaining, at said fallover node, TCP header information.

24. The system of claim 23 in which said header information includes packet sequence number.

25. The system of claim 23 or 24 further including means for constructing a Reset packet at said fallover node.

26. The system of claim 25 in which said Reset packet construction uses raw sockets.

27. The system of claim 25 or 26 further including means for transmitting said Reset packet to said application running on said client node.

28. The system of claim 25, 26 or 27 further including means for terminating connection from said client node to said server node.
29. The system of claim 28 further including means for attempting reconnection by said application.

30. The system of any of claims 19 to 29 in which said server node and said client node are in communication via Transmission Control Protocol.

31. The system of any of claims 19 to 29 in which said fallover node informs said application running on said client node that said server node has failed.

32. The system of claim 31 in which said fallover node comprises means for informing said client node that the fallover node has taken over for said server node.

33. The system of any of claims 19 to 32 in which said method is triggered by failure of said server node.

34. The system of any of claims 19 to 33 in which said notification is provided earlier than notification provided through a Transmission Control Protocol connection.

35. The system of any of claims 19 to 34, wherein the fallover node masquerades as the server node in order to send the notification of failure, the system comprising:
   means for receiving a data packet from the client application; and
   means for accessing the data packet to determine transmission information,
wherein the means for sending notification of failure comprises:
   means for using the transmission information to send notification of failure of the server node to the client application.

36. The system of claim 35, wherein the data packet is one that has been retransmitted by the client application when no acknowledgement of the original transmission of the data packet to the server node was received by the client application.
37. The system of any preceding claim in which said server node and said application running on said client node are initially connected through a Transmission Control Protocol connection.

38. A machine readable medium containing instructions thereon for sending notification of failure of a server node, which is acting as server for a client application running on a client node, to said client application, said notification being provided by a fallover node acting as backup for said server node.

39. The machine readable medium of claim 38 in which said server node and said application running on said client node are initially connected through a Transmission Control Protocol connection.

40. The machine readable medium of claim 38 or 39, wherein the fallover node masquerades as the server node in order to send the notification of failure, the machine readable medium comprising:

   instructions for receiving a data packet from the client application; and

   instructions for accessing the data packet to determine transmission information,

wherein the instructions for sending notification of failure comprises:

   instructions for using the transmission information to send notification of failure of the server node to the client application.

41. The machine readable medium of claim 40, wherein the data packet is one that has been retransmitted by the client application when no acknowledgement of the original transmission of the data packet to the server node was received by the client application.

42. A computer program comprising program code means adapted to perform the method of any of claims 1 to 18 when said program is run on a computer.
Fig. 2

101 Failure of server node
102 Failover to backup node
103 Notification to client node
104 Termination of connection to server node
**INTERNATIONAL SEARCH REPORT**

**International application No**
PCT/EP2009/052803

### A. CLASSIFICATION OF SUBJECT MATTER

**INV. H04L29/14 H04L29/08**

According to International Patent Classification (IPC) or to both national classification and IPC

### B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data

### C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No</th>
</tr>
</thead>
</table>

**X** Further documents are listed in the continuation of Box C

**X** See patent family annex

*Special categories of cited documents*

- **A** document defining the general state of the art which is not considered to be of particular relevance
- **E** earlier document but published on or after the international filing date
- **I** document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- **O** document referring to an oral disclosure, use, exhibition or other means
- **P** document published prior to the international filing date but later than the priority date claimed

**T** later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

**X** document of particular relevance, the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

**Y** document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

**S** document member of the same patent family

<table>
<thead>
<tr>
<th>Date of the actual completion of the international search</th>
<th>Date of mailing of the international search report</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 August 2009</td>
<td>18/08/2009</td>
</tr>
</tbody>
</table>

Name and mailing address of the ISA/
European Patent Office, P B 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel (+31-70) 340-2040, 340-3016
Fax (+31-70) 340-3016

Authorized officer
Huber, Oliver
**DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>EP 1 231 756 A (HEWLETT PACKARD CO [US]) 14 August 2002 (2002-08-14) claim 1</td>
<td>1,19,38, 42</td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>EP 1892921 A</td>
<td>27-02-2008</td>
<td>NONE</td>
</tr>
<tr>
<td>US 2005086342 A1</td>
<td>21-04-2005</td>
<td>NONE</td>
</tr>
<tr>
<td>US 2007198710 A1</td>
<td>23-08-2007</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2002319963 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002107966 A1</td>
</tr>
</tbody>
</table>