Title: SYSTEM AND METHOD FOR SUPPORTING A COMPLEX MESSAGE HEADER IN A TRANSACTIONAL MIDDLEWARE MACHINE ENVIRONMENT

(57) Abstract: A flexible transactional data structure can be used to store message header in a transactional middleware machine environment. The flexible transactional data structure can have dynamic numbers of fields and is accessible via specific IDs. The message header can include a first data structure that stores address information for accessing a client using a first message queue, and a second data structure that stores address information for accessing a client using a second message queue. The first type of server operates to use only the first data structure to obtain the address information for accessing the client using the first message queue. The second type of server operates to obtain a key from the first data structure first, and then use the key to obtain from the second data structure the address information for accessing the client using the second message queue.
SYSTEM AND METHOD FOR SUPPORTING A COMPLEX MESSAGE HEADER IN A TRANSACTIONAL MIDDLEWARE MACHINE ENVIRONMENT

Copyright Notice:

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Field of Invention:

The present invention is generally related to computer systems and software such as middleware, and, is particularly related to supporting a transactional middleware machine environment.

Background:

A transactional middleware system, or a transaction oriented middleware, includes enterprise application servers that can process various transactions within an organization. With the developments in new technologies such as high performance network and multiprocessor computers, there is a need to further improve the performance of the transactional middleware. These are the generally areas that embodiments of the invention are intended to address.

Summary:

Described herein is a system and method for supporting a complex message header in a transactional middleware machine environment. The complex message header includes a first data structure that stores address information for accessing a transactional client using a first message queue, and a second data structure that stores address information for accessing the transactional client using a second message queue. The first type of transactional server operates to check the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue. The second type of transactional server operates to obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

Brief Description of the Figures:

Figure 1 shows an illustration of a transactional middleware machine environment that supports a complex message header, in accordance with an embodiment of the invention.
[0006] Figure 2 illustrates an exemplary flow chart for supporting a complex message header in a transactional middleware machine environment, in accordance with an embodiment of the invention.

[0007] Figure 3 shows an illustration of a complex message header in a transactional middleware machine environment, in accordance with an embodiment of the invention.

Detailed Description:

[0008] Described herein is a system and method for supporting a transactional middleware system, such as Tuxedo, that can take advantage of fast machines with multiple processors, and a high performance network connection. A flexible transactional data structure can be used to store message header in a transactional middleware machine environment. The flexible transactional data structure can have dynamic numbers of fields and is accessible via specified IDs. The message header can include a first data structure that stores address information for accessing a client using a first message queue, and a second data structure that stores address information for accessing a client using a second message queue. The first type of server operates to use only the first data structure to obtain the address information for accessing the client using the first message queue. The second type of server operates to obtain a key from the first data structure, and then use the key to obtain from the second data structure the address information for accessing the client using the second message queue.

[0009] In accordance with an embodiment of the invention, the system comprises a combination of high performance hardware, e.g. 64-bit processor technology, high performance large memory, and redundant InfiniBand and Ethernet networking, together with an application server or middleware environment, such as WebLogic Suite, to provide a complete Java EE application server complex which includes a massively parallel in-memory grid, that can be provisioned quickly, and can scale on demand. In accordance with an embodiment, the system can be deployed as a full, half, or quarter rack, or other configuration, that provides an application server grid, storage area network, and InfiniBand (IB) network. The middleware machine software can provide application server, middleware and other functionality such as, for example, WebLogic Server, JRockit or Hotspot JVM, Oracle Linux or Solaris, and Oracle VM. In accordance with an embodiment, the system can include a plurality of compute nodes, IB switch gateway, and storage nodes or units, communicating with one another via an IB network. When implemented as a rack configuration, unused portions of the rack can be left empty or occupied by fillers.

[0010] In accordance with an embodiment of the invention, referred to herein as "Sun Oracle Exalogic" or "Exalogic", the system is an easy-to-deploy solution for hosting middleware or application server software, such as the Oracle Middleware SW suite, or Weblogic. As
described herein, in accordance with an embodiment the system is a "grid in a box" that comprises one or more servers, storage units, an IB fabric for storage networking, and all the other components required to host a middleware application. Significant performance can be delivered for all types of middleware applications by leveraging a massively parallel grid architecture using, e.g. Real Application Clusters and Exalogic Open storage. The system delivers improved performance with linear I/O scalability, is simple to use and manage, and delivers mission-critical availability and reliability.

[0011] In accordance with an embodiment of the invention, Tuxedo is a set of software modules that enables the construction, execution, and administration of high performance, distributed business applications and has been used as transactional middleware by a number of multi-tier application development tools. Additionally, a transactional middleware system, such as a Tuxedo system, can take advantage of fast machines with multiple processors, such as Exalogic middleware machine, and a high performance network connection, such as an Infiniband (IB) network.

[0012] In accordance with an embodiment of the invention, a transactional middleware system can exchange messages between a local machine and a remote machine using Remote Direct Memory Access (RDMA) protocol to achieve short latency in a manner like a local message transfer, e.g. bypassing the bridge process and preventing single point bottleneck. Exchanging messages between a local machine and a remote machine using RDMA protocol is disclosed in U.S. Application No. 13/415,760 filed March 8, 2012, entitled "SYSTEM AND METHOD FOR PREVENTING SINGLE-POINT BOTTLENECK IN A TRANSACTIONAL MIDDLEWARE MACHINE ENVIRONMENT," which application is incorporated herein by reference in its entirety.

Supporting a complex message header

[0013] In accordance with an embodiment of the invention, a transactional middleware machine environment can use a complex message header in order to support bypassing the bridge processes and prevent single-point bottleneck in transferring messages among machines. The complex message header can use a flexible data structure for supporting different message queues in the transactional middleware machine environment, such as a RDMA message queue and a System V Inter-process Communication (IPC) message queue.

[0014] Figure 1 shows an illustration of a transactional middleware machine environment that supports a complex message header, in accordance with an embodiment of the invention. As shown in Figure 1, a transactional client 101 can send different service request messages, Message A 104 and Message B 105, to different type of transactional servers, Server A 102 and Server B 103. Sever A can be a first type of server that uses an IPC queue, and Sever B can be
a second type of server that uses a RDMA queue. Additionally, a server can listen to both an IPC 
queue and a RDMA queue at the same time. Server A 102 and Server B 103 may be hardware 
compute nodes. Server A 102 and Server B 103 may be provided in a rack of a middleware 
machine.

[0015] In accordance with an embodiment of the invention, a complex message header can 
use a flexible data structure in order to support bypassing the bridge process and prevent single 
point bottleneck. As shown in Figure 1, the Message Header A 106 includes a first data structure 
A 108 and a second data structure A 110, while the Message Header B 107 includes a first data 
structure B 109 and a second data structure B 111. Each of the first data structure A 108 and 
the first data structure B 109 can be a simple data structure that stores address information for 
accessing a client using the IPC queue. Additionally, the second data structure B 111 in the 
Message header B 107 can be a data buffer that stores address information for accessing a 
client using a second message queue, while the second data structure A 110 in the Message 
header A 106 can be left empty, not in existence, or not created initially.

[0016] In the example as shown in Figure 1, Server A 102, a server that prefer to use a 
System V IPC queue A 122, can use only the simple data structure 108 to obtain the address 
information for accessing the client using the IPC message queue C 116, via the bridge process 
A 120 and the bridge process C 118. On the other hand, Server B 103, a server that prefers to 
use a RDMA queue, can use the simple data structure 109 to obtain a key 113, e.g. a faked IPC 
queue address with a negative long value. Furthermore, Server B 103 can use the key 113, or a 
specified field name, to look up the data buffer to obtain information about the address B 114 for 
accessing the client using the second message queue.

[0017] In accordance with an embodiment of the invention, the client can maintain an IPC 
queue and a RDMA queue at the same time. Before the client send out a service request 
message to a target server, the client can first determine the type of a target server based on the 
information the client receives previously. If the target server is determined to be preferable of 
using an IPC queue, then the client can send a service request message with a message header 
in a format similar to Message Header A 106 as shown in Figure 1. Otherwise, if the target 
server is determined to be preferable of using a RDMA queue, then the client can send a service 
request message with a message header in a format similar to Message Header B 107 as 
shown in Figure 1.

[0018] Figure 2 illustrates an exemplary flow chart for supporting a complex message 
header in a transactional middleware machine environment, in accordance with an embodiment 
of the invention. As shown in Figure 2, at step 201, a first data structure is provided to store 
address information for accessing a transactional client using a first message queue. Also, at 
step 202, a second data structure is provided to store address information for accessing the
transactional client using a second message queue. Then, at step 203, a first type of transactional server can check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue. Also, at step 204, a second type of transactional server can obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

Carrying Message Queue Information in Tuxedo

In accordance with an embodiment of the invention, a Tuxedo application can take advantage of the RDMA protocol and use RDMA queues in order to bypass the bridge processes associated with the System V IPC queues. The bridge processes associated with the System V IPC queues can be the single-point bottleneck in the Tuxedo environment. Using the RDMA queues, the remote message transfer between different Tuxedo machines can achieve a shorter latency in a manner similar a local message transfer.

Tuxedo client and server can be compiled in different versions and can be deployed in different machines. There is a possibility that not every machine supports the bypass bridge process feature. The system can keep the message compatibility among them. For example, a version of Tuxedo server can access the correct offsets of all fields in Tuxedo message header, although the Tuxedo server does not support the bypass bridge feature. Thus, applications running on Tuxedo from different versions can recognize messages sent from each other.

Figure 3 shows an illustration of a complex message header in a transactional middleware machine environment, in accordance with an embodiment of the invention. As shown in Figure 3, the complex message header contains several Tuxedo message headers in C-structures 301 and 302 and a Tuxedo message header in a FML32 typed buffer 303.

A client can store System V IPC queue address information in the C-structures. In order to solve the interoperability issue, the C-structures can be kept unchanged at the same location in the message header, even when the bypass bridge feature is activated. Additionally, a client can store its full RDMA message queue information into the flexible transactional header, for example in the FML32 typed buffer header 303, with IDs that are not used for System V IPC queue address.

The RDMA message queue information in the FML32 typed buffer generally contains more bytes than the System V IPC queue address information in the C-structures. For example, the C-structure in a Tuxedo request message header contains an eight (8)-bytes "long" variable, while the RDMA message queue address is an array of 128 bytes.

The FML32 typed buffer is a field-indexed flexible transactional data structure. Every
entry in the FML32 typed buffer can be retrieved by a specified field name. If a process gets an FML32 typed buffer, the process does not cause any trouble if the process does not get a value via a field name it does not know. Furthermore, it is not likely that any process may try to get a value via a field that it does not know.

[0025] In accordance with an embodiment of the invention, dynamic numbers of fields 304 and 305 can be put into the same buffer that is accessible via specified IDs. When the bypass bridge feature is not activated, the process does not look into the FML32 header, and the process only have access to the System V IPC queue, since the C-structure header is not changed.

[0026] When the bypass bridge feature is activated, a client can use the RDMA message queue instead of the System V IPC queue to send messages to server, and can store its reply RDMA message queue address in the request message so that the server can send the response back.

[0027] As shown in Figure 3, when a Tuxedo server process receives a message from a Tuxedo client, the Tuxedo server process can first get the IPC queue address in the C-structures, e.g. the Tuxedo message header X 302. If the Tuxedo server process finds out that the entry has a negative long value, then the Tuxedo server process can get the RDMA message queue address via specified field name from the Tuxedo message header Y 303, which is in a FML32 typed buffer.

[0028] Furthermore, a Tuxedo client can get the version information from the server before sending the request. Thus, for the servers compiled without the bypass bridge feature, only System V IPC queue addresses can be put into the request message header, and the FML32 typed buffer is not touched.

[0029] When the bypass bridge feature is disabled, there is no RDMA message queue address field in the FML32 buffer 303. The queue address in message buffer (C-structure part) can be a positive IPC queue address, which indicates a System V IPC queue address. Thus, when Tuxedo gets this queue address from message buffer, the system does not look into the FML32 buffer.

[0030] Other embodiments of the invention comprise a system for supporting a complex message header in a transactional middleware machine environment, comprising a first data storing unit that stores address information for accessing a transactional client using a first message queue; and a second data storing unit that stores address information for accessing the transactional client using a second message queue, a first type of transactional server that operates to check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue, and a second type of transactional server that operates to obtain a key from the first data
structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

[0031] Another embodiment comprises a system wherein the transactional client operates to send a message to a transactional server using the complex message header.

[0032] Another embodiment comprises a system wherein the transactional client operates to get information on the transactional server before sending the message.

[0033] Another embodiment comprises a system wherein the transactional server operates to send a response message to the transactional client based on the address information in the complex message header.

[0034] Another embodiment comprises a system wherein the first data structure is a C data structure and the second data structure is a typed buffer data structure.

[0035] Another embodiment comprises a system wherein the transactional client operates to communicate with a first type transactional server through a local bridge process.

[0036] Another embodiment comprises a system wherein the transactional client operates to communicate with a second type transactional server directly via a high performance network.

[0037] Another embodiment comprises a system wherein the high performance network is an Infiniband (IB) network that uses a remote direct memory access (RDMA) protocol.

[0038] Another embodiment comprises a system of claim 1, wherein a first type transactional server do not understand address information in the second data structure in the complex message header.

[0039] Another embodiment comprises a system wherein the second data structure is a flexible transactional data structure with dynamic numbers of fields, each of which is accessible via a specified ID.

[0040] Another embodiment comprises an apparatus for supporting a complex message header in a transactional middleware machine environment, comprising means for providing a first data structure that stores address information for accessing a transactional client using a first message queue; and means for providing a second data structure that stores address information for accessing the transactional client using a second message queue, means for allowing a first type of transactional server to check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue, and means for allowing a second type of transactional server to obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

[0041] Another embodiment comprises an apparatus further comprising means for allowing
the transactional client to send a message to a transactional server using the complex message
header.

[0042] Another embodiment comprises an apparatus further comprising means for allowing
the transactional client to get information on the transactional server before sending the
message.

[0043] Another embodiment comprises an apparatus further comprising means for allowing
the transactional server to send a response message to the transactional client based on the
address information in the complex message header.

[0044] Another embodiment comprises an apparatus wherein the first data structure is a C
data structure and the second data structure is a typed buffer data structure.

[0045] Another embodiment comprises an apparatus further comprising means for allowing
the transactional client to communicate with a first type transactional server through a local
bridge process.

[0046] Another embodiment comprises an apparatus further comprising means for allowing
the transactional client to communicate with a second type transactional server directly via a
high performance network.

[0047] Another embodiment comprises an apparatus wherein the high performance network
is an Infiniband (IB) network that uses a remote direct memory access (RDMA) protocol.

[0048] Another embodiment comprises an apparatus further comprising means for allowing
a first type transactional server to not understand address information in the second data
structure in the complex message header.

[0049] Another embodiment comprises an apparatus wherein the second data structure is a
flexible transactional data structure with dynamic numbers of fields, each of which is accessible
via a specified ID.

[0050] The present invention may be conveniently implemented using one or more
conventional general purpose or specialized digital computer, computing device, machine, or
microprocessor, including one or more processors, memory and/or computer readable storage
media programmed according to the teachings of the present disclosure. Appropriate software
coding can readily be prepared by skilled programmers based on the teachings of the present
disclosure, as will be apparent to those skilled in the software art.

[0051] In some embodiments, the present invention includes a computer program product
which is a storage medium or computer readable medium (media) having instructions stored
thereon/in which can be used to program a computer to perform any of the processes of the
present invention. The storage medium can include, but is not limited to, any type of disk
including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks,
ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or
optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0052] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.
What is claimed is:

1. A system for supporting a complex message header in a transactional middleware machine environment, comprising:
   a first data structure that stores address information for accessing a transactional client using a first message queue; and
   a second data structure that stores address information for accessing the transactional client using a second message queue,
   wherein a first type of transactional server operates to check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue, and
   wherein a second type of transactional server operates to obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

2. The system of claim 1, wherein the transactional client operates to send a message to a transactional server using the complex message header.

3. The system of claim 2, wherein the transactional client operates to get information on the transactional server before sending the message.

4. The system of claim 2, wherein the transactional server operates to send a response message to the transactional client based on the address information in the complex message header.

5. The system of claim 1, wherein the first data structure is a C data structure and the second data structure is a typed buffer data structure.

6. The system of claim 1, wherein the transactional client operates to communicate with a first type transactional server through a local bridge process.
7. The system of claim 1, wherein the transactional client operates to communicate with a second type transactional server directly via a high performance network.

8. The system of claim 6, wherein the high performance network is an Infiniband (IB) network that uses a remote direct memory access (RDMA) protocol.

9. The system of claim 1, wherein a first type transactional server do not understand address information in the second data structure in the complex message header.

10. The system of claim 1, wherein the second data structure is a flexible transactional data structure with dynamic numbers of fields, each of which is accessible via a specified ID.

11. A method for supporting a complex message header in a transactional middleware machine environment, comprising:

    providing a first data structure that stores address information for accessing a transactional client using a first message queue; and
    providing a second data structure that stores address information for accessing the transactional client using a second message queue,
    allowing a first type of transactional server to check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue, and
    allowing a second type of transactional server to obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

12. The method of claim 11, further comprising allowing the transactional client to send a message to a transactional server using the complex message header.

13. The method of claim 12, further comprising allowing the transactional client to get information on the transactional server before sending the message.

14. The method of claim 12, further comprising allowing the transactional server to send a response message to the transactional client based on the address information in the complex message header.
15. The method of claim 11, wherein the first data structure is a C data structure and the second data structure is a typed buffer data structure.

16. The method of claim 11, further comprising allowing the transactional client to communicate with a first type transactional server through a local bridge process.

17. The method of claim 11, further comprising allowing the transactional client to communicate with a second type transactional server directly via a high performance network.

18. The method of claim 6, wherein the high performance network is an Infiniband (IB) network that uses a remote direct memory access (RDMA) protocol.

19. The method of claim 11, further comprising allowing a first type transactional server to not understand address information in the second data structure in the complex message header.

20. The method of claim 11, wherein the second data structure is a flexible transactional data structure with dynamic numbers of fields, each of which is accessible via a specified ID.


22. A non-volatile storage media that stores the program of claim 21.
Providing a first data structure that stores address information for accessing a transactional client using a first message queue

Providing a second data structure that stores address information for accessing the transactional client using a second message queue

Allowing a first type of transactional server to check only the first data structure in the complex message header to obtain the address information for communicating with the transactional client using the first message queue

Allowing a second type of transactional server to obtain a key from the first data structure in the complex message header, and use the key to obtain the address information from the second data structure in the complex message header for communicating with the transactional client using the second message queue.

FIGURE 2
FIGURE 3
**INTERNATIONAL SEARCH REPORT**

**INTERNATIONAL APPLICATION**

- **International Application No.**
  - PCT/US 12/57121

**CLASSIFICATION OF SUBJECT MATTER**

- **IPC(8)**: G06F 15/173 (2012.01)
- **USPC**: 709/238

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

**FIELDS SEARCHED**

- Minimum documentation searched (classification system followed by classification symbols)
  - USPC: 709/238

- Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
  - USPC: 709/238, 709/224, 709/247, 709/242, 709/246, 709/244, 709/245

**SEARCHED ELECTRONIC DATA BASE**

- PatBase: Google Patents; Google Scholar
- Search Terms Used: message header, complex data structure, middleware, key address, client server, queue, transaction, c, local bridge

**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>Y</td>
<td>US 2011/0078214 A1 (MICHAEL et al.) 31 March 2011 (31.03.2011), para [0001]-[0024], [0044], [0093]</td>
<td>1-20, 21(1 1-20), 22</td>
</tr>
<tr>
<td>Y</td>
<td>US 2004/0015622 A1 (AVERT) 22 January 2004 (22.01.2004), para [0009], [0033]-[0034], [0046], [0054]-[0057], [0060]-[0061]</td>
<td>1-20, 21(1 1-20), 22</td>
</tr>
</tbody>
</table>

- Further documents are listed in the continuation of Box C.

- 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 application or patent but published on or after the international filing date
  - "L" 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

- 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

- 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

- 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

- document member of the same patent family

**Date of the actual completion of the international search**

- 14 November 2012 (14.11.2012)

**Date of mailing of the international search report**

- 11 DEC 2012

**Name and mailing address of the ISA/US**

- Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
- P.O. Box 1450, Alexandria, Virginia 22313-1450

**Facsimile No.**

- 571-273-3201

**Authorized officer:**

- Lee W. Young

- PCT Helpdesk: 571-272-4300
- PCT OSP: 571-272-7774

- Form PCT/ISA/210 (second sheet) (July 2009)