Kemari
Sender/Receiver protocol
Version 0.5a (Nov. 17th, 2010)

Yoshiaki Tamura, Kei Ohmura(NTT)

Copyright © 2009-2010 Nippon Telegraph and Telephone Corporation
Definition of words.

- **Sender**
  - QEMU that send the snapshot of VM for fault tolerance.
- **Receiver**
  - QEMU that receive the snapshot of VM for fault tolerance.
- **Initialization mode**
  - Mode to synchronize all the VM information before starting fault tolerant synchronization. The fault tolerant synchronization is done with transaction mode.
- **Transaction mode**
  - Mode to send only the difference between events. If this mode started, QEMU will send the difference of VM snapshot only when some events occurs.
- **FT mode.**
  - Special transfer mode of migration to realize fault tolerance.
  - When sender start migration with this mode option, its state will become initialization mode and transaction mode, accordingly.
Message sequence chart and state transitions of sender and receiver

**Sender**

- **Initialization mode**
  - Init sync. (starting migration with FT option)
  - Finish init sync
  - Check ready
  - Transaction

**Receiver**

- Live migration (starting QEMU in –incoming w/ ft_mode option)
- Synchronize VM same as live migration
- To notify receiver get ready.
- Send ACK
- Transaction
- FT mode sync.
- All the init sync data received.
Data format from sender to receiver. (Initialization mode)

Whole Transferred data

<table>
<thead>
<tr>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
</table>

(1) QEMU header

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>QEMU_VM_FILE_MAGIC</td>
<td>QEMU_VM_FILE_VERSION</td>
</tr>
</tbody>
</table>

Completely same as live migration

(2) Payload

Device information

(3) QEMU footer

<table>
<thead>
<tr>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>QEMU_VM_EOF</td>
</tr>
</tbody>
</table>

Completely same as live migration.
Data format from sender to receiver. (transaction mode)

<table>
<thead>
<tr>
<th>Whole Transferred data</th>
<th>(1)</th>
<th>(2)</th>
<th>...</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) can exist more than once.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (1) Transaction header

<table>
<thead>
<tr>
<th>QEMU_VM_TRANSACTION_BEGIN</th>
<th>Transaction id (ID)</th>
<th>seq</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Payload contains device information, the same as live migration. Conjunction of all payloads of each transaction can be loaded with the loadvm function.

### (2) Transaction payload

<table>
<thead>
<tr>
<th>QEMU_VM_TRANSACTION_CONTINUE</th>
<th>ID</th>
<th>seq</th>
<th>Payload length</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>Payload</td>
</tr>
</tbody>
</table>

Used to line up series of payload. This value start from 0 in each transaction, and be increased every time (2) is sent.

### (3) Transaction footer

<table>
<thead>
<tr>
<th>QEMU_VM_TRANSACTION_COMMIT</th>
<th>Transaction id (ID)</th>
<th>seq</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

### Special purpose formats
Those can be sent any time instead of (1), (2), or (3).

#### (A) Quit transaction

<table>
<thead>
<tr>
<th>QEMU_VM_TRANSACTION_CANCEL</th>
<th>Transaction id (ID)</th>
<th>seq</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

#### (B) Atomic transaction (for small payload)

<table>
<thead>
<tr>
<th>QEMU_VM_TRANSACTION_ATOMIC</th>
<th>ID</th>
<th>seq</th>
<th>Payload length</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>Payload</td>
</tr>
</tbody>
</table>
Data format from receiver to sender.

Initialization mode

- Nothing will be sent in this mode. Its behavior is the same as live migration.

Transaction mode

<table>
<thead>
<tr>
<th>Transaction Type</th>
<th>Format Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) Accept transaction</td>
<td>QEMU_VM_TRANSACTION_ACK</td>
</tr>
<tr>
<td></td>
<td>Transaction id (ID)</td>
</tr>
<tr>
<td></td>
<td>seq</td>
</tr>
<tr>
<td></td>
<td>Payload length</td>
</tr>
<tr>
<td>(B) Found error during transaction</td>
<td>QEMU_VM_TRANSACTION_NACK</td>
</tr>
<tr>
<td></td>
<td>Transaction id (ID)</td>
</tr>
<tr>
<td></td>
<td>seq</td>
</tr>
<tr>
<td></td>
<td>Payload length</td>
</tr>
<tr>
<td>(C) Quit transaction</td>
<td>QEMU_VM_TRANSACTION_CANCEL</td>
</tr>
<tr>
<td></td>
<td>Transaction id (ID)</td>
</tr>
<tr>
<td></td>
<td>seq</td>
</tr>
<tr>
<td></td>
<td>Payload length</td>
</tr>
</tbody>
</table>

This ID correspond to each transaction id.

Difference between (B) and (C) is error or not.
State transition diagram of sender (normally)

- Start migration with FT option
  - Init sync.
    - Live migration finished
      - Waiting receiver ready
        - ACK received
          - Waiting event
            - Got event
              - Begin transaction
                - QEMU_VM_TRANSACTION_BEGIN sent.
                  - Committed
                    - QEMU_VM_TRANSACTION_COMMIT sent.
                      - Sending device info.
                        - Send device info. with QEMU_VM_TRANSACTION_CONTINUE
                          - QEMU_VM_TRANSACTION_ACK received.
                            - Sending device info.
Init sync.

Waiting receiver ready

Waiting event

Begin transaction

Sending device info.

Committed

QEMU_VM_TRANSACTION_NACK, or QEMU_VM_TRANSACTION_CANCEL received.

Closing

Connection closed

Finalizing

All the FT related things such as dirty-bitmap mode become clear.

QUIT (sender terminated)
State transition diagram of Receiver (normally)

- **Start QEMU with -incoming option**
- **Live migration finished**
- **QEMU_VM_TRANSACTION_BEGIN received.**
- **QEMU_VM_TRANSACTION_CONTINUE received.**
- **QEMU_VM_TRANSACTION_COMMIT received.**

**States:**
- **Live migration**
- **FT mode prepare**
- **Waiting event**
- **Loading VM image**
- **Reading device info.**
- **Device info. committed**
State transition diagram of Receiver (to cancel FT mode)

Live migration

FT mode prepare

Waiting event

Canceling

Reading device info.

QUIT (waiting for live migration)

QEMU_VM_TRANSACTION_CANCEL received.

Connection closed