Modularization for Lockdep

冯博群 Boqun Feng (Microsoft)
Pain points of lockdep

- The warnings (part 1)

WARNING: possible circular locking dependency detected
...
the existing dependency chain (in reverse order) is:
...
-> #4 (&sbi->s_writepages_rwlock){++++}-{0:0}:
...
-> #3 (mapping.invalidate_lock){++++}-{3:3}:
...
-> #2 (&sb->s_type->i_mutex_key#8){++++}-{3:3}:
...
-> #1 (&journal->j_checkpoint_mutex){++.+.}-{3:3}:
...
-> #0 (&journal->j_barrier){++.+.}-{3:3}:
Pain points lockdep (cont.)

- The warnings (part 2)

Chain exists of:
&journal->j_barrier --> mapping.invalidate_lock --> &sbi->s_writepages_rwsem

Possible unsafe locking scenario:

<table>
<thead>
<tr>
<th>CPU0</th>
<th>CPU1</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>lock(&amp;sbi-&gt;s_writepages_rwlock);</td>
<td>lock(mapping.invalidate_lock);</td>
</tr>
<tr>
<td>lock(&amp;sbi-&gt;s_writepages_rwlock);</td>
<td>lock(&amp;journal-&gt;j_barrier);</td>
</tr>
</tbody>
</table>
Pain points lockdep (cont.)

- "I'm testing XYZ, but warnings keep coming up for ABC".
Pain points lockdep (cont.)

- Lockdep itself creates synchronization points.
- Debug lockdep itself.
- Change deadlock detection algorithm.
- ...
Lockdep today

Per-task lock stack
Per-task lock stack
Per-task lock stack
deps
Dependency graph
Deadlock detector
Lockdep frontend-backend

Lockdep frontend
- Per-task lock stack
- Per-task lock stack
- Per-task lock stack

Lockdep pipeline
- Dependency graph

Lockdep backend
- Deadlock detector
- Pretty printer
Frontend

- Bookkeep the lock holding information.
- Work even if lockdep is off (behavior change)
Frontend (cont.)

- Able to run frontend alone.
  - `lockdep_assert_*( )` will continue to work.
  - No sync point unless `lock_class` allocation.
• The grapch
• The good old detector (based on the graph)
• The pretty printer or a reporter
Backend

- Able to implement in userspace (and in another language).
Pipepline

- Piping the information from the frontend to the backend.
- (Maybe) Maintain another dependency graph to save the traffic to backend
- Multiple modes
Pipepline (cont.)

- Just function calls
  - In this case, the whole lockdep behaves the same as today.
Pipeline (cont.)

- Files (or ?) exposed to userspace

![Diagram of pipeline components including Lockdep frontend and Lockdep backend (userspace) with connections and labels for dependency graph, deadlock detector, and pretty printer]
Pipeline (cont.)

- Serial consoles outputing to other machines
Future work

• Start upstream process
  ○ Frontend isolation will come first.
  ○ configs?
• Pipeline protocol?
• Multiple backend?