Charm++ - Bug #1531
Main (scheduler) thread can suspend, but that confuses QD, where other thread suspends don't

04/23/2017 01:59 PM - Phil Miller

Status: Merged
Priority: High
Assignee: Phil Miller
Category: 
Target version: 6.8.0

Start date: 04/23/2017
Due date:
% Done: 0%
Estimated time: 0.00 hour
Spent time: 0.00 hour

Description
If I remove the [threaded] tag from the attached code, it hangs at the first QD, while it doesn't with threaded in place. I'll test that this is specifically QD, and that it doesn't come up if I explicitly use a reduction or some other control flow.

Related issues:
Related to Charm++ - Support #1548: Reassess whether the primary scheduler th...

History
#1 - 04/25/2017 04:26 PM - Phil Miller
- Assignee set to Phil Miller

#2 - 04/25/2017 04:34 PM - Phil Miller
- Priority changed from Normal to High

#3 - 04/27/2017 10:42 AM - Phil Miller
Confirmed that a callback to resume the thread works fine. QD has an issue with this.

#4 - 04/27/2017 12:18 PM - Phil Miller
Here's a very minimal reproduction case

```c
// .ci
mainmodule simplering {
    mainchare Main {
        entry Main(CkArgMsg *m);
            entry void run_main();
    }
}

// .C
#include "simplering.decl.h"

struct Main : public CBase_Main {
    Main(CkArgMsg* m) {
        thisProxy.run_main();
    }
    void run_main() {
        CkWaitQD();
        CkExit();
    }
}
#include "simplering.def.h"
```

Add [threaded] to the declaration of run_main(), and it exits cleanly.

Without threaded instrumented output:
$ ./simplering
Charm++: standalone mode (not using charmrun)
Charm++> Running in non-SMP mode: numPes 1
Converse/Charm++ Commit ID: v6.8.0-beta1-74-g74f94c4
Charm++> scheduler running in netpoll mode.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 unique compute nodes (8-way SMP).
Charm++> cpu topology info is gathered in 0.000 seconds.
QD created 1, created 1, processed 0
QD processed 1, created 1, processed 1
QD created 1, created 2, processed 1
QD processed 1, created 2, processed 2
^C

And with [threaded]

$ ./simplering
Charm++: standalone mode (not using charmrun)
Charm++> Running in non-SMP mode: numPes 1
Converse/Charm++ Commit ID: v6.8.0-beta1-74-g74f94c4
Charm++> scheduler running in netpoll mode.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 unique compute nodes (8-way SMP).
Charm++> cpu topology info is gathered in 0.000 seconds.
QD created 1, created 1, processed 0
QD created 0, created 1, processed 0
QD processed 1, created 1, processed 1
QD created 1, created 2, processed 1
QD created 1, created 3, processed 2
QD processed 1, created 3, processed 3
QD created 1, created 4, processed 3
QD processed 1, created 4, processed 4
[Partition 0][Node 0] End of program
A little more detailed instrumentation:

$ ./simplering
Charm++: standalone mode (not using charmrun)
Charm++> Running in non-SMP mode: numPes 1
Converse/Charm++ Commit ID: v6.8.0-beta1-74-g74f94c4
Charm++> scheduler running in netpoll mode.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 unique compute nodes (8-way SMP).
Charm++> cpu topology info is gathered in 0.000 seconds.
QD created 1, created 1, processed 0
QD created 0, created 1, processed 0
QD processed 1, created 1, processed 1
In run_main Before waitQD call
In CkWaitQD pre
QD created 1, created 2, processed 1
QD processed 1, created 2, processed 2
In chare
In chare else
In chare calling suspend

So, it looks like adjusting the 'standin' scheduler thread code to call CsdScheduleForever instead of CsdSchedulePoll solves the problem - CsdSchedulePoll is written such that SCHEDULE_IDLE never happens. Because of that, QD never makes progress, while the scheduler is sitting around with no messages in the queue. Miniscule change, but it's in quite old code that may have knock-on consequences. This will need cautious review.

- Status changed from New to Implemented

https://charm.cs.illinois.edu/gerrit/2453

Decision in Core was to go ahead with fixing this for now. After the release, we'll re-assess the 'cute trick' giving the ability to suspend the main thread running the scheduler. This re-assessment will start with replacing the trick with a CkAbort call, and seeing what if anything breaks.
- Related to Support #1548: Reassess whether the primary scheduler thread should support CthSuspend added

#10 - 05/03/2017 11:45 AM - Phil Miller
- Status changed from Implemented to Merged

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>simplering.ci</td>
<td>341 Bytes</td>
<td>04/23/2017</td>
<td>Phil Miller</td>
</tr>
<tr>
<td>simplering.cc</td>
<td>1.85 KB</td>
<td>04/23/2017</td>
<td>Phil Miller</td>
</tr>
</tbody>
</table>