OpenRTM-aist (C++) - バグ #3579

trunkバージョンでactivate_component()がBAD_PARAMETERで失敗する場合がある

2016/08/09 11:33 - kanehiro

ステータス:	終了	**開始日:**	2016/08/09
優先度:	通常	**期日:**	
担当者:	n-miyamoto	**進捗率:**	90%
カテゴリ:		**予定工数:**	0.00時間
対象バージョン:			

説明

r2757のtrunkとhrpsys-baseを組み合わせて使用しています。
RTC AのecにRTC
B,C,D,E,F,Gをadd_component()して、A,B,C,D,E,F,Gの順序で、hrpsys-baseのstart()というpythonの関数（https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L104）を使ってコンポーネントをアクティベートしています。
すると、B,C,Dあたりのアクティベートが結構な頻度でBAD_PARAMETERで失敗します。失敗するのはB,Cだったり B,C,Dだったりします。
start関数の中の既にアクティベートされているかをチェックする箇所（https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L108-L109）をコメントアウトすると失敗することはなくなります。

履歴

#1 - 2016/08/09 14:32 - n-ando

- 担当者 を n-ando にセット

#2 - 2016/08/09 15:04 - n-ando

これも、以前問合せがあった件と根は同じかと思います。調査します。

#3 - 2016/12/02 07:44 - kanehiro

直したはず、とのことでしたので試してみました。
うまくいく場合もあるのですが、多くの場合BでBAD_PARAMETERが生じて、C,Dも失敗したりしなかったりします。

#4 - 2017/02/21 18:25 - n-miyamoto

- 担当者 を n-ando から n-miyamoto に変更

調査した結果、activateComponent関数がBAD_PARAMETERを返す条件がExecutionContextWorkerのfindComponent関数でRTCがリストから見つけられなかった場合だけだという事は分かりました。

この問題については、#3697でECに即座にRTCが追加される修正を行っているので、以下のように全てのRTC追加後にアクティベートしてもBAD_PARAMETERを返すことはありません。hrpsys-baseで処理しても同じだと思います。
以下のように全てのRTC追加後にアクティベートしたのですが、問題を再現することはできませんでした。

```
EC_A  =  RTC_A.get_owned_contexts()[0]

EC_A.add_component(RTC_B)
EC_A.add_component(RTC_C)
EC_A.add_component(RTC_D)
EC_A.add_component(RTC_E)

EC_A.activate_component(RTC_A)
EC_A.activate_component(RTC_B)
EC_A.activate_component(RTC_C)
EC_A.activate_component(RTC_D)
EC_A.activate_component(RTC_E)
```

ただ、以下のようにRTC_Aをアクティベートした後にRTCの追加を行うと同じ問題が発生することは確認できました。

```
EC_A  =  RTC_A.get_owned_contexts()[0]

EC_A.activate_component(RTC_A)

EC_A.add_component(RTC_B)
EC_A.add_component(RTC_C)
EC_A.add_component(RTC_D)
```

EC_A.add_component(RTC_E)

EC_A.activate_component(RTC_B)
EC_A.activate_component(RTC_C)
EC_A.activate_component(RTC_D)
EC_A.activate_component(RTC_E)

RTCが全てInactive状態の場合はスレッドの実行を一時停止するようになっており、この場合はadd_componentで即座にRTCを追加することができます。
ただRTC_AがActive状態の場合、スレッドの実行を再開しているのでadd_componentで即座にRTCは追加されません。
この場合、svc関数内でinvokeWorkerPostDo関数が呼び出されるまでECにRTCは追加されません。

add_componentを実行してからactivate_componentを実行するまでの間に実行コンテキストでinvokeWorkerPostDo関数が呼び出されている必要があるので、BやCのRTCのアクティベートが失敗しやすいと思います。

#5 - 2017/02/22 08:24 - n-ando

- 進捗率 を 0 から 50 に変更

ECがrunning状態では、1周期分の実行が終わるまでは、たとえadd_componentで追加した後であっても、論理的にはコンポーネントが追加されていない、ということですね。
これはECの構造上仕方ないような気もしますが、activate_componentのリクエストをキューに入れておいて、論理的にコンポーネントが追加された直後に、active化する、という方法も取れなくはないですね。
ただ、これをやっても、その後対象のRTCに対するいろいろなオペレーションについても、同様のことが起こりえるので、RTCに対してaliveかどうかを問い合わせれば、同様に例外を返すような気がします。

#6 - 2017/02/22 12:12 - kanehiro

宮本様、安藤様、

調査ありがとうございます。
ソースコードを最新（r2953）にして再度テストしたところ、現象が変わってしまいました。
BAD_PARAMETERではなく、RTC_ERRORとなるようになりました。
問題が起きている箇所では、add_component()してから、active化しています。
今は12個のRTCを直列実行するようにして、前から順にactive化しています。
途中（6個目）がRTC_ERRORを返すとその後のRTCも全てRTC_ERRORとなるのですが、ヒントになりますでしょうか。
最初にRTC_ERRORを起こしているのはonActivate()で重い処理を行っていて、少し時間がかかるのですが、なにかタイムアウトのようなものがあるのでしょうか。

#7 - 2017/02/22 12:57 - n-miyamoto

金広様

rtc.confのexec_cxt.activation_timeoutで設定した時間内にアクティブ状態に遷移しない場合はRTC_ERRORになります。
デフォルトでは0.5秒になっています。

PeriodicExecutionContextは直列にRTCを実行するので、例えば6個目のRTCのonActivate()に時間がかかると、7個目以降のRTCのonActivate()はしばらく実行されないのでアクティブ状態に遷移するまで時間が掛かります。
6個目のRTCのonActivate終了まで7個目以降のRTCはアクティブ状態に遷移できないため、タイムアウトでRTC_ERRORを返しているのではないかと思います。

#8 - 2017/02/22 13:05 - kanehiro

宮本様、

#6のコメントの前に1度しかテストしなかったのですが、再度テストしたところやはりBAD_PARAMETERが起きます。
一つ非常に重要なことをお知らせしていなかったことに気が付きました。
実行コンテキストとして、
https://github.com/s-nakaoka/choreonoid/blob/master/src/OpenRTMPlugin/ChoreonoidExecutionContext.cpp
を使用しています。こちらの実装に問題があるということはないでしょうか。

#9 - 2017/02/22 13:44 - n-miyamoto

- 担当者 を n-miyamoto から n-ando に変更

金広様

OpenHRPExecutionContextにはECにRTCを即座に追加する機能がありません。

実行コンテキストのadd_componentを呼び出すと、内部でonAddedComponent関数が呼び出されます。
PeriodicExecutionContextやExtTrigExecutionContextでは以下のようにonAddedComponent関数内でupdateComponentList関数を呼び出してRTCのリストを更新するようにしています。
updateComponentList関数が呼び出されないとRTCはリストに追加されません。
OpenHRPExecutionContextにこの実装はないので、tick関数でRTCのリストが更新されるまでアクティベートしてもRTCが見つからずにBAD_PARA

METERになります。

```
RTC::ReturnCode_t PeriodicExecutionContext::
onAddedComponent(RTC::LightweightRTObject_ptr rtobj)
{
    Guard guard(m_workerthread.mutex_);
    if (m_workerthread.running_ == false)
      {
        m_worker.updateComponentList();
      }
    return RTC::RTC_OK;
}
```

ChoreonoidExecutionContextはOpenHRPExecutionContextを継承してあるようなので、RTCが即座に追加されずアクティベートしようとするとBAD_PARAMETERになるのではないかと思います。

安藤様

この件はどう対応しましょうか？

#10 - 2017/02/22 17:13 - kanehiro

tick関数でRTCのリストが更新されるまで、というのはECが一回回るまで、という理解でよろしいでしょうか。
こちらでの使い方では、A,B,C,Dとある時に、AのecにB,C,Dをadd_component()して、A,B,C,Dの順序でアクティベートしています。
アクティベートの際には、アクティブ状態に遷移したことを確認しているのですが、遷移した＝ECが1回回った、ということではないのでしょうか。
回ったとすればB,C,DがBAD_PARAMETERを返さないような気がするのですが。

#11 - 2017/02/22 19:09 - n-miyamoto

- 担当者 を n-ando から n-miyamoto に変更

#4のコメントにも書いたのですが、BAD_PARAMETERを返すのはRTCがリストから見つからなかった場合だけです。
RTCがアクティブ状態に遷移しているのにBAD_PARAMETERを返すことはソースコードを見た限りでは無いように見えますし、依然こちらの環境では問題を再現できていません。
お手数ですが、ログファイルを見せて頂いてもよろしいでしょうか？

```
RTC::ReturnCode_t
ExecutionContextWorker::activateComponent(RTC::LightweightRTObject_ptr comp,
                                          RTObjectStateMachine*& rtobj)
{
    RTC_TRACE(("activateComponent()"));
    RTObjectStateMachine* obj = findComponent(comp);
    if (obj == NULL)
      {
        RTC_ERROR(("Given RTC is not participant of this EC."));
        return RTC::BAD_PARAMETER;                    //RTCがリストに登録されていない場合、BAD_PARAMETERを返す
      }
    RTC_DEBUG(("Component found in the EC."));
    if (!(obj->isCurrentState(RTC::INACTIVE_STATE)))
      {
        RTC_ERROR(("State of the RTC is not INACTIVE_STATE."));
        return RTC::PRECONDITION_NOT_MET;
      }
    RTC_DEBUG(("Component is in INACTIVE state. Going to ACTIVE state."));
    obj->goTo(RTC::ACTIVE_STATE);                     //BAD_PARAMETERを返した場合、遷移先にActiveがセットされない
    rtobj = obj;
    RTC_DEBUG(("activateComponent() done."));
    return RTC::RTC_OK;
```

#12 - 2017/02/23 08:19 - kanehiro

- ファイル rtcd.log を追加

ログファイルを添付しますのでよろしくお願いします。
（本題とはずれるのですが、ログの出力にエスケープシーケンスが使われるようになって、標準出力に出しているときには見やすくてよいのですが、ファイルに落としてエディタで開くと見づらくなります。よいビューア等ありますでしょうか？）

#13 - 2017/02/23 10:47 - n-miyamoto

ログファイルを添付していただいてありがとうございます。

おそらくですが、add_componentが11回、activate_componentが14回、get_component_stateが17回呼び出されていると思います。

- add_component
  - 1回目 RangeNoiseMixer0
  - 2回目 SequencePlayer0
  - 3回目 Standup0
  - 4回目 KalmanFilter0
  - 5回目 HumanoidMotionControl20
  - 6回目 PanTilt0
  - 7回目 Swinger0
  - 8回目 StateHolder0
  - 9回目 RobotGateway0
  - 10回目 AccelerationChecker0
  - 11回目 DataLogger0

- activate_component
  - 1回目 RTC_ERROR(タイムアウト) PDcontroller0(owned_context)
  - 2回目 RTC_ERROR(タイムアウト) VirtualRobot0(owned_context)
  - 3回目 BAD_PARAMETER(リストにRTCが無い) RangeNoiseMixer0(participating_context)
  - 4回目 BAD_PARAMETER(リストにRTCが無い) SequencePlayer0(participating_context)
  - 5回目 BAD_PARAMETER(リストにRTCが無い) Standup0(participating_context)
  - 6回目 BAD_PARAMETER(リストにRTCが無い) KalmanFilter0(participating_context)
  - 7回目 BAD_PARAMETER(リストにRTCが無い) HumanoidMotionControl20(participating_context)
  - 8回目 BAD_PARAMETER(リストにRTCが無い) PanTilt0(participating_context)
  - 9回目 BAD_PARAMETER(リストにRTCが無い) Swinger0(participating_context)
  - 10回目 RTC_OK StateHolder0(participating_context)
  - 11回目 RTC_OK RobotGateway0(participating_context)
  - 12回目 RTC_OK AccelerationChecker0(participating_context)
  - 13回目 RTC_OK DataLogger0(participating_context)
  - 14回目 RTC_OK OpenHRPInterpreterService0(owned_context)

- get_component_state
  - 1回目 ACTIVE_STATE
  - 2回目 CREATED_STATE(リストにRTCが無い)
  - 3回目 CREATED_STATE(リストにRTCが無い)
  - 4回目 CREATED_STATE(リストにRTCが無い)
  - 5回目 CREATED_STATE(リストにRTCが無い)
  - 6回目 CREATED_STATE(リストにRTCが無い)
  - 7回目 CREATED_STATE(リストにRTCが無い)
  - 8回目 CREATED_STATE(リストにRTCが無い)
  - 9回目 INACTIVE_STATE
  - 10回目 ACTIVE_STATE
  - 11回目 INACTIVE_STATE
  - 12回目 ACTIVE_STATE
  - 13回目 INACTIVE_STATE
  - 14回目 INACTIVE_STATE
  - 15回目 ACTIVE_STATE
  - 16回目 INACTIVE_STATE
  - 17回目 ACTIVE_STATE

ログファイルを見た限りでは、以下のRTCはアクティブ状態に遷移していません。

- RangeNoiseMixer0
- SequencePlayer0
- Standup0
- KalmanFilter0
- HumanoidMotionControl20
- PanTilt0
- Swinger0

activate_componentの10～14回目ですが、"onActivated() done."が表示されているのでRTC_OK以外を返すことはありません。
3～9回目がBAD_PARAMETERを返しているのはtickが呼び出されておらずRTCのリストが更新されていないためだと思います。

以下のことは間違いないので、何か問題を再現するのに足りない情報があるのではないかと思います。

- activate_componentでBAD_PARAMETERを返したRTCはアクティブ状態に遷移していない
- アクティブ状態に遷移したRTCはactivate_componentでRTC_OKかRTC_ERRORを返している。

「アクティブ状態に遷移したことを確認した」とのことですが、どのように確認したのでしょうか？

#14 - 2017/02/23 11:22 - kanehiro

コンポーネントをアクティベートするのに、次のPythonの関数を使用しています。
https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L98-L123
この関数の中で、isActive()という関数を呼んでおり、これでアクティブ状態に遷移したことを確認しています。

https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L165-L171

#10のコメントですが、よく考えるとすみません、間違っていました。
A,B,C,Dの内、Aはシミュレータで自動的にアクティブ化されるのを忘れていました。
したがって、Aが既にアクティブな状態でB,C,Dを追加しており、Aがアクティブであることを確認してもB,C,Dはまたリストには追加されていないのでBAD_PARAMETERが起きる、ということのようです。

宮本さん、安藤さんの想像されていることが起きているようなのですが、1.1ではこういう問題は起きていないので、1.2ではどう対処するのがよいのか教えて頂ければと思います。

#15 - 2017/02/23 12:37 - n-miyamoto

- 担当者 を n-miyamoto から n-ando に変更

金広様

一つ確認したいことがあるのですが、#10のコメントでアクティブ状態に遷移したことを確認したのはAだけで、B、C、Dについては確認していないという事でしょうか？

1.2での対処方法ですが、PeriodicExecutionContextとExtTrigExecutionContextについてはECをstopすれば即座にRTCをリストに追加できます。
OpenHRPExecutionContextについては、add_componentとactivate_componentの間にtickを呼び出せばリストにRTCが追加されるのでアクティベートできると思います。

安藤様

OpenHRPExecutionContextについてですが、PeriodicExecutionContextと同様のonAddedComponent関数を実装したらtick関数を呼び出さなくてもRTCはリストに追加できると思うのですが、どう対応しましょうか？

#16 - 2017/02/23 13:04 - kanehiro

いずれも同じstart()関数を使っていますので、activate_component()がRTC_OKを返せばその後にチェックするのですが、BAD_PARAMETERを返しているのでチェックしていないです。

対処法について、tick()はシミュレータがシミュレーション世界の時刻に同期して呼び出しているので、シミュレータでないところから呼ぶのは避けたいところです。

ユーザの視点から考えると、add_component()を呼んだら追加の待機リストに入れてすぐに帰ってくるのではなく、実際に追加されてから返ってきてくれた方がありがたいのですが、そうはできないでしょうか。

また、1.1ではできていたものが1.2ではできなくなったのはなぜでしょうか？

#17 - 2017/02/23 14:05 - n-ando

金広さん

1.2(=trunk)のECはこれまでの1.1のECとクラス構成がだいぶ変わっています。これは、拡張のしやすさと動的なRTCの入れ替えと排他制御をちゃんとやるための変更です。

1.1.2
においても、ECへのRTCのadd/removeは原則としてECがstopping状態、もしくはinactive状態のときのみのを想定しており、ECが持つRTCのリストは排他制御をしていませんでした。これは、ECがリアルタイム実行時にクリティカルセクションで排他制御を極力行わない、特にネットワーク経由からの呼び出しに絡む部分にとロックを取り合わないように配慮しての実装です。

したがって、ECのスレッドが動いている最中にadd/removeをしてしまうと、ECが持つRTCリストには排他制御されていないので破壊される恐れもあります。

1.2では、ECへのRTCのadd/removeは、クリティカルセクションとは無関係のRTCリストにいったんキューイングされ、実行周期のスリープ前にまとめてECのRTCリストを更新しています。これにより、クリティカルセクション中にRTCのリストをネットワークからの呼び出し側と取り合う事態は避けられ、リアルタイム実行中でも一応RTCの出し入れが可能になると考えています。

add/remove
の呼び出しを同期型にした場合、ECの周期が非常に長いケースでは、その分add/removeの呼び出しで待たされてしまう恐れがあります。ただ、今回のようなケースもあるので、オプションでadd/removeを同期・非同期を切り替えられるようにすることも考慮したいと思います。

#18 - 2017/02/23 14:20 - kanehiro

なんと、そうだったのですか。それは知りませんでした。

同じような話がコンポーネントのactivate/deactivateでもありましたね。1.2から同期型も選べるようになるのでしたでしょうか。start()関数ではactivate_component()した後にアクティブ状態に移行したかをポーリングするというちょっと気持ち悪いことをしています。

#19 - 2017/02/24 16:47 - n-miyamoto

- 担当者 を n-ando から n-miyamoto に変更

- 進捗率 を 50 から 90 に変更

OpenHRPExecutionContextを修正しました。
add_componentを実行すると以下のonAddedComponent関数が呼び出されてRTCリストが更新されます。

```
    RTC::ReturnCode_t    OpenHRPExecutionContext::
                onAddedComponent(RTC::LightweightRTObject_ptr    rtobj)
    {
                Guard    guard(m_tickmutex);

                ExecutionContextBase::m_worker.updateComponentList();

                return    RTC::RTC_OK;
    }
```

tick関数はm_tickmutexのミューテックスをロックするようになっているので、

- tick関数実行中は終了までRTCリスト更新を待つ
- RTCリスト更新中はtick関数の実行を待つ

となります。

問題が解決したかどうか、ご確認をお願いします。

#20 - 2017/02/27 09:20 - kanehiro

- ステータス を 新規 から 終了 に変更

宮本様、

問題が解決されました。ありがとうございます。

ファイル

rtcd.log	13.3 MB	2017/02/23	kanehiro