OpenRTM-aist (Python) - 機能 #3413

⑭　スレーブマネージャの名前によるグルーピング

2015/12/22 09:46 - n-ando

ステータス:	終了	**開始日:**	2015/12/22
優先度:	通常	**期日:**	2016/03/25
担当者:	miyamoto	**進捗率:**	100%
カテゴリ:		**予定工数:**	30.00時間
対象バージョン:	RELEASE_1_2_0		

説明

コンポーネントを起動する際に、マスターマネージャにコンポーネント起動を依頼し、マスターは指定された名前やポリシーに従って、すでに起動済みのスレーブ上にRTCを起動したり、新たなプロセスを起動してそこにRTCをインスタンス化できる機能を実装すること。

関係しているリビジョン

リビジョン 684 - 2016/03/06 23:34 - miyamoto

[incompat,new_func,->RELENG_1_2] add findManager_by_name() and create_component_by_mgrname() to ManagerServant. refs #3413

リビジョン 685 - 2016/03/07 01:56 - miyamoto

[compat,->RELENG_1_2]The processing in ManagerServant.create_component() method has been changed. refs #3413

リビジョン 686 - 2016/03/07 02:13 - miyamoto

[compat,->RELENG_1_2]Fixed comments. refs #3413

履歴

#1 - 2016/01/14 16:23 - miyamoto

- 期日 を 2016/03/25 にセット

- 担当者 を miyamoto にセット

- 対象バージョン を RELEASE_1_2_0 にセット

- 予定工数 を 30.00時間 にセット

#2 - 2016/02/18 15:41 - n-ando

まずは、マネージャに名前を付ける方法を実装する。

- スレーブマネージャ
  - 言語による区別
  - VCのバージョンによる区別
  - 多言語のローダブルモジュール
- マニフェストファイルの導入

#3 - 2016/03/06 00:46 - miyamoto

マネージャの名前付けの方法について

例えば以下のように名前を付ける

manager_cpp_vc2008_release
manager_cpp_vc2008_debug
manager_cpp_vc2010_release
manager_cpp_vc2010_debug
manager_cpp_vc2012_release
manager_cpp_vc2012_debug
manager_cpp_vc2013_release
manager_cpp_vc2013_debug
manager_cpp_vc2015_release
manager_cpp_vc2015_debug
manager_python
manager_java

上記の名前をマネージャのインスタンス名で保持する。

```
rtcd  -o  manager.instance_name:manager_python
```

ロード可能なモジュールのプロファイルの取得(get_loadable_modules)に時間がかかる問題

Python版rtcprofの動作が終了しない不具合

Timerスレッドが終了するまでrtcprof.pyの処理が終了しないのが原因のため、Timerスレッドを起動しないようにした。

```
opts.append("-o")
opts.append("timer.enable:NO")
```

をrtcprof.pyに追加

C++版のget_loadable_modulesオペレーション呼び出し時に同じモジュールのプロファイルを2回取得してしまう現象

これはmanager.modules.load_pathとmanager.modules.C++.load_pathsに同じパスが存在した場合に2度同じ処理を繰り返してしまうのが原因
以下のコードで重複を削除すると多少操作が快適になる。

```
std::sort(paths.begin(),  paths.end());
paths.erase(std::unique(paths.begin(),  paths.end()),  paths.end());
```

大量にモジュールをロードした場合に時間がかかる問題

マスターマネージャに上記のスレーブマネージャを全て関連付けると大量にget_loadable_modulesを呼び出し、さらにDLL等を大量にロードするため操作性を著しく害する。

案1 言語、VC++のバージョンごとにマスターマネージャ起動
案2 ロードしたモジュールの情報をファイルに保存しておき、再度マネージャを起動した時に読み込む

案2について

例えば取得したRTCのプロファイルのリストを以下のようにXMLで保存する。

```
<components>
	<component  name="test">
		<implementation_id>test</implementation_id>
		<type_name>test</type_name>
		<description>test  description</description>
		<version>1.0,0</version>
		<vendor>vendor</vendor>
		<category>sample</category>
		<activity_type>STATIC</activity_type>
		<max_instance>1</max_instance>
		<language>Python</language>
		<lang_type>SCRIPT</lang_type>
		<module_file_name>test.py</module_file_name>
		<module_file_path>./test.py</module_file_path>
		<last_updated>2016-03-05  00:47:54</last_updated>
	</component>
・・・・
</components>
```

再度マネージャを起動した時に保存したファイルを読み込み以下の作業を行う。

- XMLファイルにプロファイルが保存されているモジュールの内、ファイルが更新されていないものはXMLファイルに保存した情報を利用する
- XMLファイルにプロファイルが保存されているモジュールの内、ファイルが更新されているものは再度rtc_profileでプロファイルを取得する
- XMLファイルにプロファイルが保存されていないモジュールの場合は、通常通りrtc_profileでプロファイルを取得する

起動するコンポーネントの指定方法

従来の機能では、

```
create_component("comp&manager=localhost:2810")
```

のようにRTCを起動するマネージャのホストアドレスとポート番号を指定すると、指定のマネージャでRTCを起動する。

指定のスレーブマネージャに起動する場合と区別するために、ホストアドレスを指定する場合はmanager_address、マネージャ名を指定する場合はmanager_nameの項目に記述する。

```
comp&manager_address=localhost:2810
comp&manager_name=slave_manager
comp&manager_name=new_manager
```

コンポーネント名以外を指定しなかった場合、スレーブマネージャのロード済みモジュール、ロード可能モジュールに指定コンポーネントが存在する場合はスレーブマネージャで起動する。

create_component("comp")

マニフェストファイルについて

考え中

#4 - 2016/03/07 00:12 - miyamoto

- 進捗率 を 0 から 20 に変更

ManagerServantクラスのcreate_component関数の処理を変更した。

1 指定名の中に&manager_addressが存在する
→ 指定のホスト名、ポート番号のマネージャでRTCを起動
→ マネージャが存在しない場合は新たにプロセスを起動する

2 指定名の中に&manager_nameが存在する
→ 指定のインスタンス名のマネージャでRTCを起動
→ マネージャが存在しない場合は新たにプロセスを起動

3 プロセス内のマネージャがマスターマネージャである
→ 登録されているスレーブマネージャでRTCを起動

4 1、2、3でRTCを起動できなかった
→ プロセス内のマネージャでRTCを起動

まず指定したホスト名、ポート番号のマネージャにRTCを起動する従来の機能はcreate_component_by_address関数に移動してcreate_component関数内で呼び出すようにした。

そして新機能として指定した名前のマネージャでRTCを起動する機能をcreate_component_by_mgrname関数に追加した。

以下のようにRTC名&manager_name=マネージャ名で指定する。

comp&manager_name=slave_manager
comp&manager_name=new_manager

マネージャを検索するfindManager_by_name関数を追加した。
プロセス内で起動したマネージャがマスターの場合は関連付けたスレーブマネージャから検索する。
スレーブの場合は登録されているマスターマネージャからスレーブマネージャを検索する。
get_configurationでプロファイルを取得し、manager.instance_nameの項目と指定のマネージャ名が一致するかを確認する。

名前が一致するマネージャが存在しない場合、新たに起動したプロセス上のマネージャでRTCを起動する。
※現在のところPython版rtcdでしか起動できないため、今後は言語を指定することでmanager.modules.<lang>.manager_cmdに設定したrtcdで起動できるようにする。

スレーブマネージャとして起動し、findManager_by_name関数でマスターマネージャへの登録が確認できたらRTCの起動を試みる。

create_component関数内ではcreate_component_by_address関数、create_component_by_mgrname関数を呼び出す。
指定名の中にp&manager_address、&manager_nameが存在しない場合はスレーブマネージャでの起動を試みる。
プロセス内のマネージャがマスターの場合は、登録されているスレーブマネージャのcreate_component関数を呼び出して起動できるかを確認する。
戻り値が全てnilの場合はプロセス内のマネージャでの起動を試みる。

#5 - 2018/01/19 18:33 - n-miyamoto

- ステータス を 新規 から 解決 に変更

- 進捗率 を 20 から 100 に変更

#6 - 2018/09/13 09:22 - n-miyamoto

- ステータス を 解決 から 終了 に変更