mruby extension module for monitoring system

Takanori Suzuki
Agenda

- **Introduction** of MIRACLE ZBX
- **What** is mruby extension module
- **Why** mruby module is needed
- **Structure** of mruby extension module
- **How to use** mruby extension module
Introduction of MIRACLE ZBX (1/2)

- Monitoring system developed in MIRACLE LINUX
- Forked from OSS Zabbix under GPL2
  - Support many monitoring types including agent, ssh, SNMP, IPMI, etc...
- Add some special features that we need
  - Additional event log filter in Web interface
  - Additional runtime configuration
  - Customized Windows eventlog key
  - etc...

* Zabbix is a registered trademark of Zabbix LLC
Introduction of MIRACLE ZBX (2/2)

- Customizable monitoring features
  - “UserParameter” feature
    - Execute any command from monitoring agent process and get the result

- “External check” feature
  - Execute any command from monitoring server process and get the result

- “Loadable module” feature in C
  - Execute C module function and get the result
- Customizable monitoring features
  - “UserParameter” feature
    - Execute any command from monitoring agent process and get the result

- “External check” feature
  - Execute any command from monitoring server process and get the result

- “Loadable module” feature in C
  - Execute C module function and get the result

I made mruby extension module by this feature
What is mruby extension module

- Execute mruby function in mruby file and return the result to monitoring server
  - Similar as mruby version of “Loadable module”

  key: `mruby.module[sample.rb, args]`

- Execute mruby code string and return the result to monitoring server

  key: `mruby.eval[p "hello world"]`
Why mruby module is needed

- **Easier** than C

- **Faster** than scripting language

- Embed to existing process, **no fork**
Structure of mruby extension module
Overview of mruby extension module

- Work as a C loadable module
- Translate C function call into mruby function call
When the mruby code is called

- `zbx_module_init()` in "MonitoringModule" class
  - When the agent process start

- `zbx_module_run(args)` in "MonitoringModule" class
  - When the monitoring key is monitored
  - The function can have Arguments.
    
    key: `mruby.module[sample.rb, arg1, arg2,...]`
    
    `mruby function: zbx_module_run(arg1, arg2,...)`

- `zbx_module_uninit()` in "MonitoringModule" class
  - When the agent process stop
Problems with multiprocess

Monitoring agent works in multiprocess

- For sharing data, **shared memory** is needed.

- For locking, **semaphore lock** is needed.
Shared memory

mruby-cache

https://github.com/charlescui/mruby-cache

- Made by CharlesCui
- “Mruby Inter Process Share Memory. Exchange memory space with mmap for multi mruby process.”

```ruby
cache = Cache.new :namespace => "foo"
cache["key"] = "value"
cache["key"] # => "value"
```
Semaphore lock

mruby-semlock

https://github.com/tszki/mruby-semlock

- Made for mruby extension module
- Implementation to use semaphore lock

```ruby
sem = Semlock.new "./sample.rb", 0, 1, 0600
# keyfile, prj_num, sem_num, permission
sem.lock(0)  # lock with waiting
sem.unlock(0)  # unlock
status = sem.trylock(0)  # lock without waiting
sem.unlock(0) if status  # unlock
sem.remove  # remove semaphore
```
Sample code in multiprocess

Updating shared variable safely

cache = Cache.new :namespace=>"foo"
sem = Semlock.new "./sample.rb", 0, 1, 0600

cache["key"] = "0"
sem.lock(0)  # lock

cache["key"] = (cache["key"].to_i + 1).to_s
# updating safely

sem.unlock(0)  # unlock
sem.remove
Other included mrbgems

- mruby-base64
- mruby-cache
- mruby-curl
- mruby-digest
- mruby-dir
- mruby-http
- mruby-httprequest
- mruby-io
- mruby-json
- mruby-marshall
- mruby-mtest
- mruby-mutex
- mruby-oauth
- mruby-pack
- mruby-polarssl

- mruby-process
- mruby-semlock
- mruby-simplehttp
- mruby-sleep
- mruby-socket
- mruby-userdata
How to use mruby extension module
Installation

- MIRACLE ZBX 3.0.0alpha2-1
  http://ftp.miraclelinux.com/zbx/preview/miracle-zbx-3.0.0alpha2-pkgs.tar.gz

  $ tar xzvf miracle-zbx-3.0.0alpha2-pkgs.tar.gz
  $ cd miracle-zbx-3.0.0alpha2-pkgs/x86_64
  $ sudo rpm -ihv miracle-zbx-3.0.0alpha2-1.ML7.x86_64.rpm
  miracle-zbx-agent-3.0.0alpha2-1.ML7.x86_64.rpm

- mruby extension module
  Project page:
  https://github.com/tszki/mruby-extension-module-for-miracle-zbx/
  mruby extension module rpm:
  https://github.com/tszki/mruby-extension-module-for-miracle-zbx/raw/master/rpm/mruby-extension-module-for-miracle-zbx-1.0.0-1.ML7.x86_64.rpm

  $ sudo rpm -ihv \
  mruby-extension-module-for-miracle-zbx-1.0.0-1.ML7.x86_64.rpm
Setting

- “loadable module” setting is set by installed conf file.

<table>
<thead>
<tr>
<th>/etc/zabbix/zabbix_agentd.d/mruby_extension_module.conf</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadModulePath=/usr/lib64/zabbix/modules</td>
</tr>
<tr>
<td>LoadModule=mruby_extension_module.so</td>
</tr>
</tbody>
</table>

- Copy sample mruby files

```
# cp -p /usr/lib64/zabbix/modules/mruby_module/sample.rb.sample /usr/lib64/zabbix/modules/mruby_module/sample.rb
```

- Start agent service

```
# systemctl start zabbix-agent.service
```
Checking

- **mruby.module[]**

```bash
$ zabbix_get -s 127.0.0.1 -k 'mruby.module[sample.rb]'  
Hello world. I drunk 1 cups of water

$ zabbix_get -s 127.0.0.1 -k 'mruby.module[sample.rb, Hi.]'  
Hi. I drunk 2 cups of water
```

- **mruby.eval[]**

```bash
$ zabbix_get -s 127.0.0.1 -k 'mruby.eval[p "hello world"]'  
hello world
```
Demo