Unleash The Daemons

KL Ruby Brigade Nov 13

Milad Rastian

- Software Engineer
- Python/Perl/PHP/Ruby(?)
- System Administrator
- IT team leader at Sam Media
- Twitter @slashmili

DAEMON

- How to pronounce it:
    - [dee-mon]/[die-mon]/[day-mon]

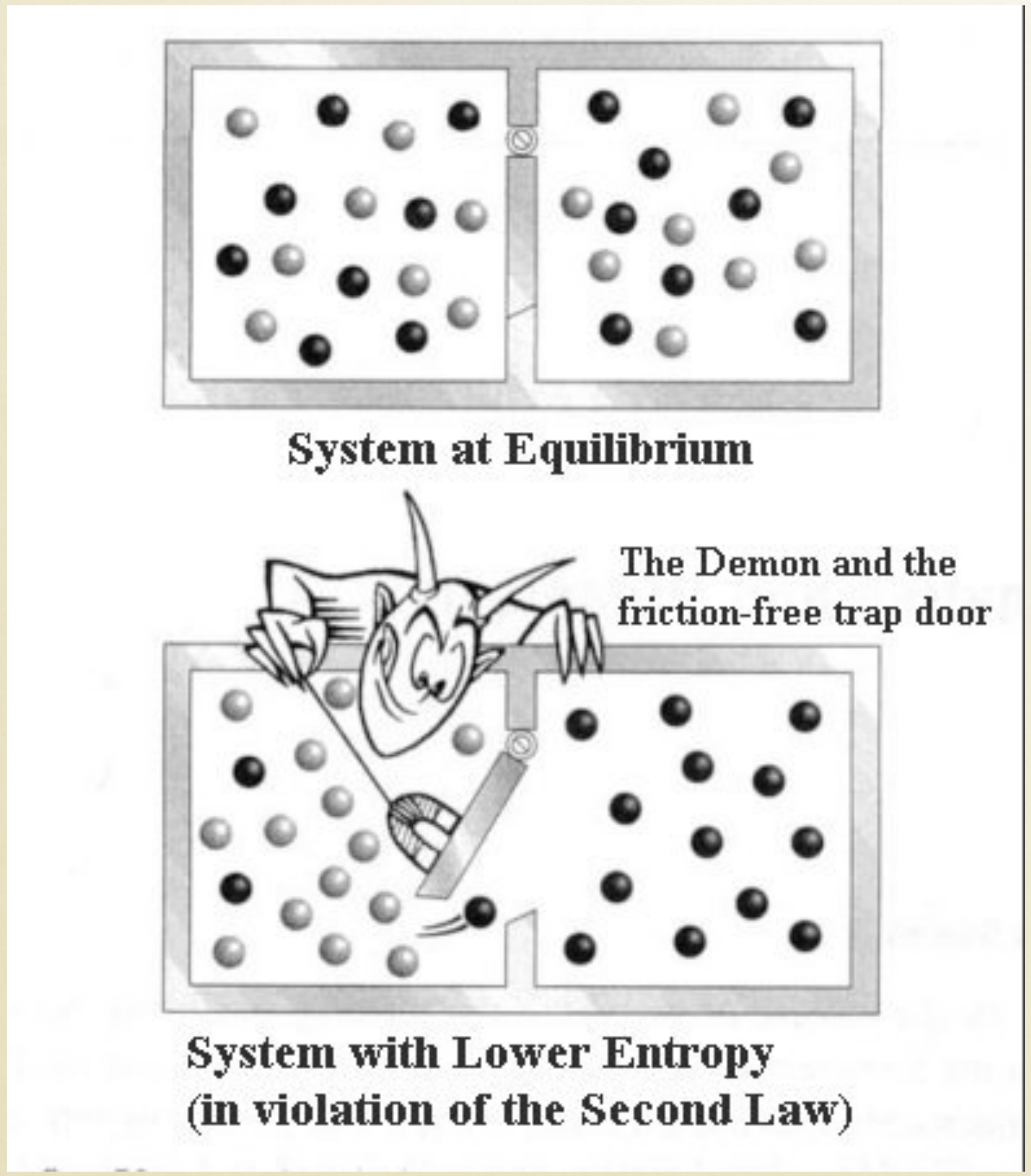
      ✓ ✓
- What is it?

History

- The term was coined by programmers in MIT.
- They took the name from Maxwell's demon.
- Daemon doesn' t have particular bias towards good or evil.

History(continue)

More like this

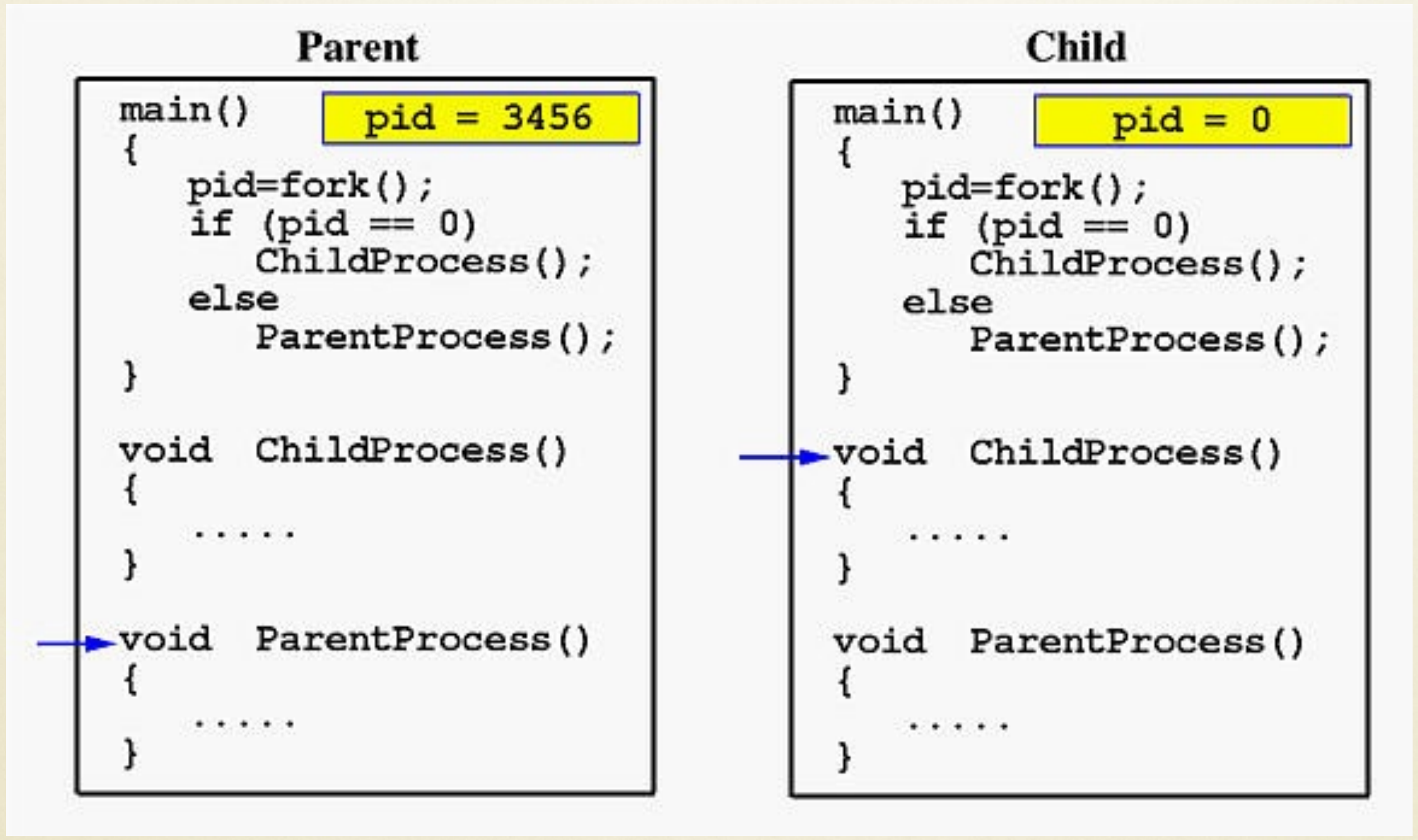

System at Equilibrium
The Demon and the friction-free trap door
System with Lower Entropy (in violation of the Second Law)

Than this

Daily usage

• Unicorn	unicorn_rails -c config/unicorn.rb -D
• Sidekiq	rake sidekiq:start
• New Relic agent	
• Nginx	service nginx start

Fork system call



```
main()            pid = 3456
{
   pid=fork();
   if (pid == 0)
      ChildProcess();
   else
      ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```

```
main()            pid = 0
{
   pid=fork();
   if (pid == 0)
      ChildProcess();
   else
      ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```

DAEMONIZING

- fork
- setsid
- fork
- chdir
- umask
- close stdin, stdout stderro
- pid

```
class MySuperDuperApp
  def start
    fork do
      new_session
      fork do
        change_dir
        change_user_and_group
        change_umask
        save_pid
        close_fds
        main
      end
    end
  end
(MySuperDuperApp.new).start
```

```
def new_session
  Process.setsid
end
 def change_dir
   Dir.chdir '/var/
www/ '
 end
def change_user_and_group
  Process.euid = 1000
  Process.egid = 1000
end

def change_umask
  File.umask(0002)
end

def save_pid
 File.open('/var/run/mysuperduperapp.pid', 'w') do|f|
   f.write(Process.pid)
 end
end

def close_fds
    [STDOUT, STDERR].each do |fd|
      fd.reopen("/dev/null", "w")
    end
   STDIN.reopen("/dev/null")
end
```

WEBrick Daemon: https://github.com/ruby/ruby/blob/trunk/lib/webrick/server.rb#L45

Cronjob Vs. Daemon

- Cron runs processes periodically.
- Daemon runs forever!

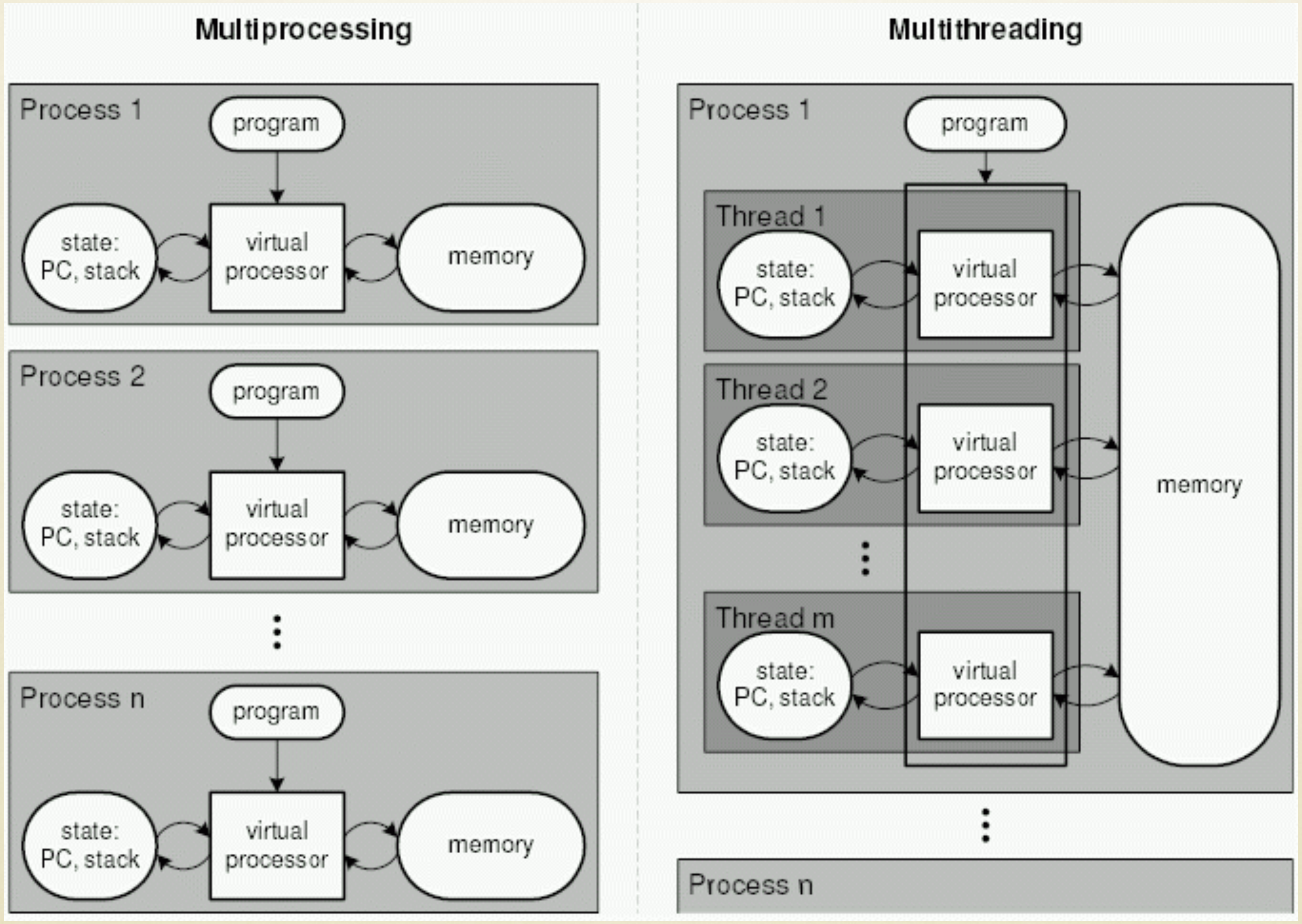
Daemon in Jruby

```
% puma -d
Puma 2.0.0.b7 starting...
* Min threads: 0, max threads: 16
* Environment: development
* Listening on tcp://0.0.0.0:9292
NotImplementedError: fork is not available on this platform
        fork at org/jruby/RubyProcess.java:1078
      daemon at /usr/local/rvm/gems/jruby-1.7.3@test/gems/puma-2.0.0.b7-java/lib/puma/
daemon_ext.rb:3
  run_single at /usr/local/rvm/gems/jruby-1.7.3@test/gems/puma-2.0.0.b7-java/lib/puma/cli.rb:417
         run at /usr/local/rvm/gems/jruby-1.7.3@test/gems/puma-2.0.0.b7-java/lib/puma/cli.rb:402
      (root) at /usr/local/rvm/gems/jruby-1.7.3@test/gems/puma-2.0.0.b7-java/bin/puma:10
        load at org/jruby/RubyKernel.java:1046
      (root) at /usr/local/rvm/gems/jruby-1.7.3@test/bin/puma:1
        eval at org/jruby/RubyKernel.java:1066
      (root) at /usr/local/rvm/gems/jruby-1.7.3@test/bin/ruby_noexec_wrapper:14
```

Efficient code in Daemons

- Forking v.s. threading in the main method
- Processes talking to each other
- Sending signal to a processes

Forking Vs. threading in the Main method



Thread Concurrency

```
x = Mysql2::Client.new
y = Mysql2::Client.new

Benchmark.bm do |b|
  b.report('w/o') do
    x.query("SELECT SLEEP(1)")
    y.query("SELECT SLEEP(1)")
  end

  b.report('with') do
    a = Thread.new{ x.query("SELECT SLEEP(1)") }
    b = Thread.new{ y.query("SELECT SLEEP(1)") }
    a.join
    b.join
  end
end
```

Processes talking to each other

- DRb
- AMQP
- Jabber
- Socket
- ØMQ

Handling Signal

- Register signal handler

```
Signal.trap("TERM") do
  puts "USR1 caught"
end
```

- Send the signal by kill command

```
kill -SIGTERM <pid>
```

- All the available signals : `kill -l`

Managing Daemons in Ruby

- God
- Forman
- bluepill
- Monit
- Daemon-kit

Questions?