Sphactor

actor model concurrency for creatives

HKU expertise centre creative technology
Background

• >3900 students
• one of the largest culture-oriented institutes in Europe
• Expertise Centre Creative Technology
Context: Motion Capture
## Programming Didactics

<table>
<thead>
<tr>
<th>User</th>
<th>Operator</th>
<th>Scripter</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consuming technology</td>
<td>Combining technologies</td>
<td>Lego-ing with technologies</td>
<td>Final boss</td>
</tr>
</tbody>
</table>
Multi Core?

42 Years of Microprocessor Trend Data

- Transistors (thousands)
- Single-Thread Performance (SpecINT x 10^3)
- Frequency (MHz)
- Typical Power (Watts)
- Number of Logical Cores

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten.

New plot and data collected for 2010-2017 by K. Rupp.
Multi Core?

42 Years of Microprocessor Trend Data

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp
Actor Model

- message passing
- defined 1973 Hewitt
- 80's -> erlang -> whatsapp
- actor == sequential program sending and receiving
- actors are simple
Actor Model - sphactor
def do(msg):
    img = decode(msg)
    blob = detectFace(img)
    blob.traceContour()
    blob.validate()
    return blob
Actor Model - sphactor
Actor states

- INIT
- STOP
- DESTROY
- IDLE
- SOCK
- TIME
gazebosc
Gazebosc demo
import sph
# pip install python-osc
from pythonosc import osc_message_builder

class tester(object):
    def handleMsg(self, msg, type, name, uuid, *args, **kwargs):
        # just pop the first string and return the rest
        t = msg.popstr()
        print("Message received: {}".format(t))
        msg = osc_message_builder.OscMessageBuilder(address="/Hello")
        msg.add_arg("hello from python")
        osc = msg.build()
        return osc.dgram
Gazebo C++ Actor

```cpp
#include "libsphactor.h"

class Test {
public:
    zmsg_t *
    handleMsg( sphactor_event *ev ) {
        char *cmd = zmsg_popstr(event->msg);
        zsys_info("Cpp actor %s says: %s", event->name, cmd);
        // if there are strings left publish them
        if ( zmsg_size(event->msg) > 0 ) {
            return event->msg;
        } else {
            zmsg_destroy(&event->msg);
        }
        return nullptr;
    }
};
```
#include "libsphactor.h"

int main() {
    Test a = Test();
    sphactor_t *actora = sphactor_new(a, "hello-a", nullptr);
    // actora is running, request its name
    const char *name = sphactor_ask_name(actora);
    assert( streq(name, "hello-a"));

    ...
    // connect it another actor
    sphactor_ask_connect(actora, sphactor_ask_endpoint(actorb));
    ...
    // cleanup
    sphactor_destroy(&actora);
    return 0;
}
API

**sphactor API (main thread)**

`sphactor_new`  
(handler, args, name, uuid);

`sphactor_destroy`(self);

`sphactor_ask_endpoint`(self);

`sphactor_ask_connect`  
(self, endpoint);

`sphactor_ask_disconnect`  
(self, endpoint);

`sphactor_ask_set_timeout`  
(self, timeout);

**sphactor_actor API (actor thread)**

`sphactor_actor_poller_add`  
(self, fd, handler);
Under the hood

ZeroMQ / czmq

Dear IMGUI
SDL
Liblo
Embeds Python
Wrapping up

- Actor Model Framework aimed at simplicity
- Early stage so all the cliche todo's
- Try it, help us out. Especially if:
  - want a tool so you can play with technology
  - familiar with file descriptors/reactor pattern
Sphactor
actor model concurrency for creatives

https://github.com/hku-ect/libsphactor

https://github.com/hku-ect/gazebosc

Background paper: see FOSDEM event link

HKU
expertise centre creative technology