Kamailio SIP Server

async processing in config

www.kamailio.org

Daniel-Constantin Mierla
Co-Founder Kamailio
@miconda

www.asipto.com

What’s the issue?

avoid blocking operations

The benefit?

increase the performance

Performance?

not an issue for kamailio

thank you!

Questions?

But then …

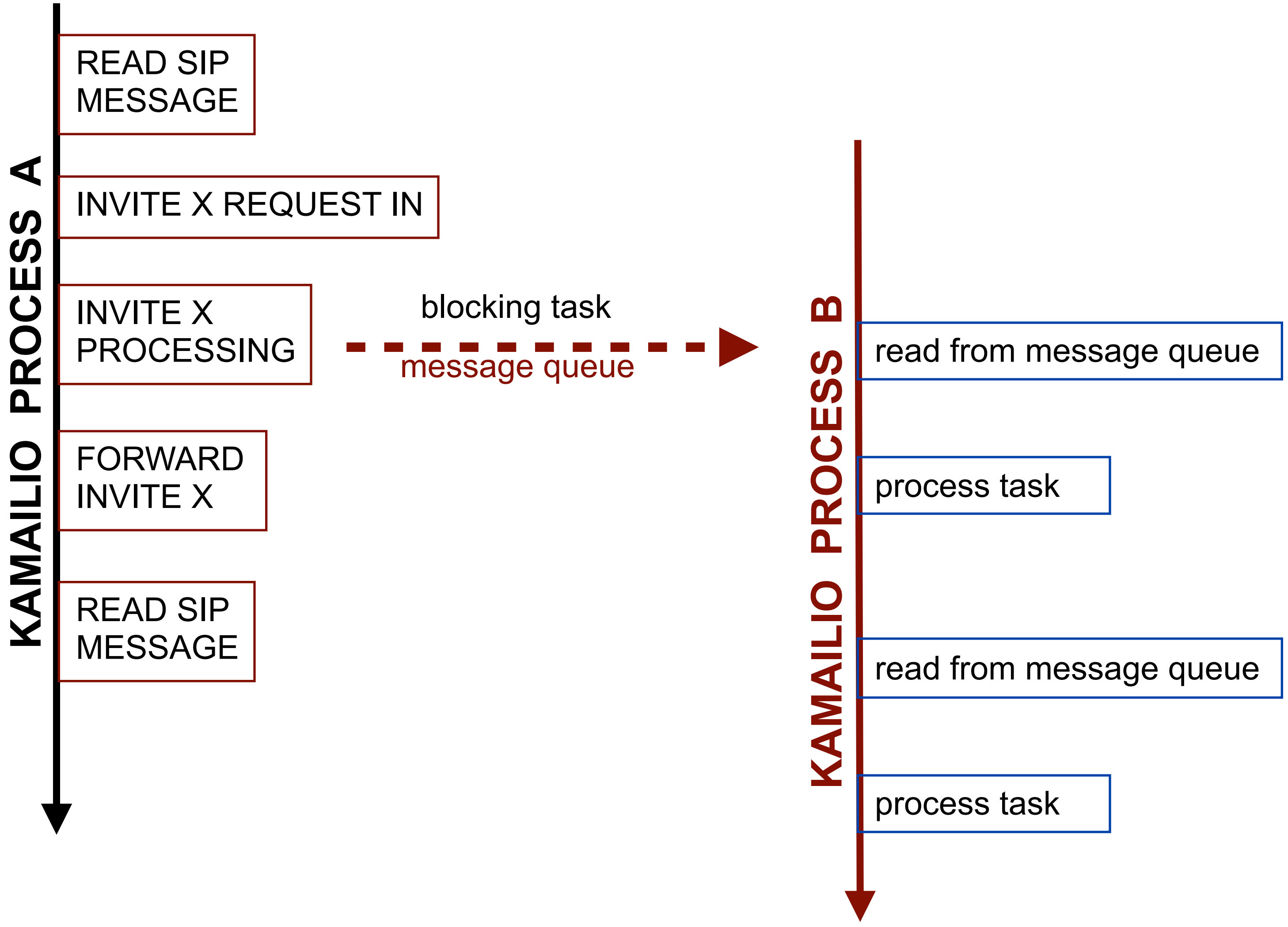
- Interaction with database
  - sometimes you must wait
  - sometimes is not that real-time critical
- DNS
  - dns is down, internet is down
- Interaction with other external systems
  - rating engine - billing systems

Kamailio
application architecture

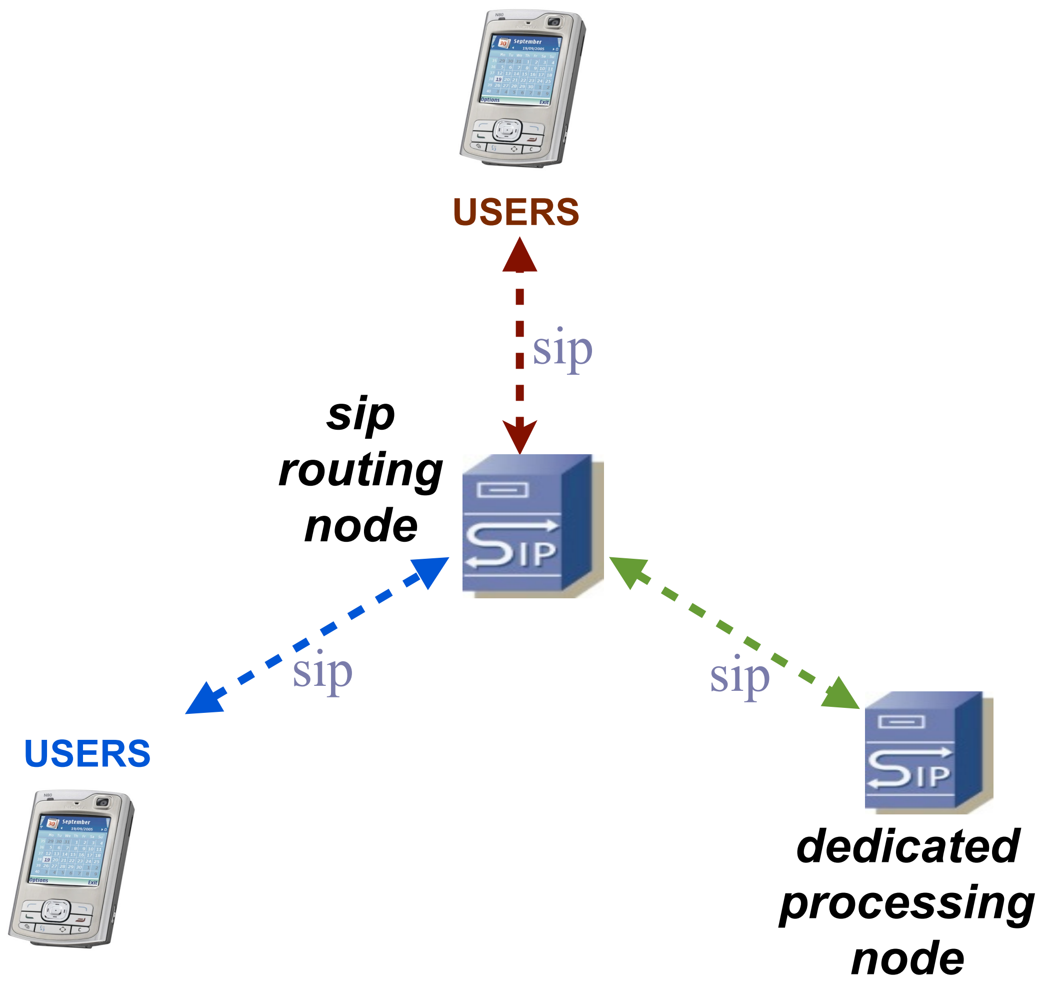
Application design

- *multi-process application*
  - not multi-threading
- *dedicated processes for*
  - main attendant
  - tcp attendant
  - sip tcp receivers
  - sip udp receivers
  - sip sctp receivers
  - control interface
    - FIFO receiver
    - DATAGRAM/UDP receiver
    - XMLRPC receiver
  - timers
    - main timers (sip retransmissions)
    - dedicated timers
    - custom timers
  - special handling
    - custom processing

Blocking task delegation to another process



Blocking task delegation to another instance



Basic Kamailio Components

Cases

Impacting SIP routing

- need to wait for async task to finish
  - suspend sip message processing
  - handle other SIP messages
  - when async task done, resume processing of suspended message

No impact on SIP routing

- unrelated event
- can be pushed out of SIP handling process and that’s a;;

- TMX
  - suspend-resume transaction
- UAC
  - send custom messages from kamailio config
- ASYNC
  - asynchronous execution of routing blocks
  - asynchronous sleep
- MQUEUE
  - internal message queue
- RTIMER
  - create custom processes that can react on timer
- JSONRPC-C
  - send json-rpc commands and process responses asynchronously
- EVAPI
  - publishing event messages via tcp and process responses asynchronously

```
evapi_relay(“{ \"event\": \"test\",\n \"data\": { \"fU\": \"$fU\" }\n}");
```

Send push notifications

- suspend invite - resume on registration -

Push notifications

Config

```
# ----- htable params -----
modparam("htable", "db_url", DBURL)
modparam("htable", "htable", "vtp=>size=10;autoexpire=120;dbtable=htable;dbmode=1")
modparam("htable", "htable", “a=>size=6;")

request_route {
  …
          # send the push
          route(PUSHAYASYNC);

          # user location service
          route(LOCATION);
}
```

Push notifications

Config

```
# Handle SIP registrations
route[REGISTRAR] {
        if (!is_method("REGISTER"))
                return;

        if(isflagset(FLT_NATS))
        {
                setbflag(FLB_NATB);
                # uncomment next line to do SIP NAT pinging
                ## setbflag(FLB_NATSIPPING);
        }
        if (!save("location"))
                sl_reply_error();

        route(PUSHJOIN);
        exit;
}
```

Push notifications

Config

```
# do the PUSH notification
route[SENDPUSH] {
        $var(luaret) = 0;

        if(lua_runstring(“do_push([[$hdr(X-VxTo)]], [[$tU]], [[$hdr(X-VxFrom)]], [[$fU]], [[$ci]])")<0)
        {
                send_reply("501", "No link to destination");
                exit;
        }
        if($var(luaret)!=1)
        {
                send_reply("501", "Unknown destination");
                exit;
        }
        send_reply("110", "Push sent");
}
```

Push notifications

Config

```
# do push in async mode
route[PUSHASYNC] {
        if (!is_method("INVITE"))
                return;

        if(registered("location"))
                return;

        route(SENDPUSH);

        if(!t_suspend())
        {
           xlog("failed suspending trasaction [$T(id_index):$T(id_label)]\n");
                send_reply("501", "Unknown destination");
                exit;
        }
        xdbg("suspended transaction [$T(id_index):$T(id_label)] $fU => $rU\n");
        $sht(vtp=>join::$rU) = "" + $T(id_index) + ":" + $T(id_label);
        xdbg("htale key value [$sht(vtp=>join::$rU)]\n");
        exit;
}
```

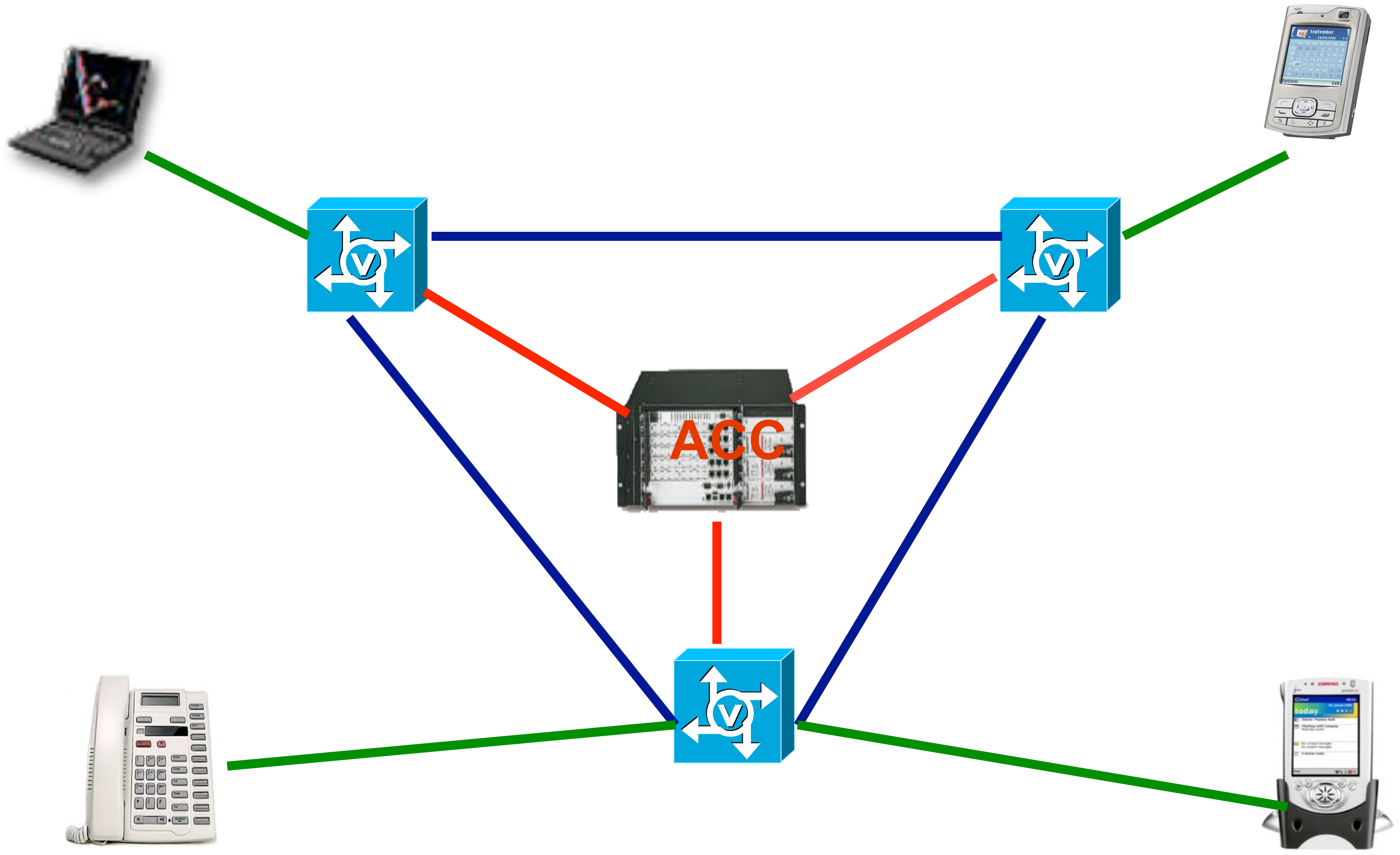
Push notifications

Config

```
# join pending INVITE with the incoming REGISTER
route[PUSHJOIN] {
	if (!is_method("REGISTER"))
		return;
	$var(hjoin) = 0;
	lock("$tU");
	$var(hjoin) = $sht(vtp=>join::$tU);
	$sht(vtp=>join::$tU) = $null;
	unlock("$tU");
	if($var(hjoin)==0)
		return;
	$var(id_index) = $(var(hjoin){s.select,0,:}{s.int});
	$var(id_label) = $(var(hjoin){s.select,1,:}{s.int});
	xdbg("resuming trasaction [$var(id_index):$var(id_label)] $tU ($var(hjoin))\n");
	t_continue("$var(id_index)", "$var(id_label)", "LOCATION");
}
```

Central Accounting System

Accounting Server



Proxy Server Config

```
onreply_route[ACC] {
 if(status!="200")
    return;
  $uac_req(method)="ACCOUNTING";
  $uac_req(ruri)="sip:store@accounting.kamailio.org;transport=sctp";
  $uac_req(furi)="sip:server@server1.kamailio.org";
  $uac_req(hdrs)="Content-Type: text/accounting-csv\r\n";
  pv_printf(“$uac_req(body)”, "$TS,$ci,$ft,$tt,$T_req($fu),$T_req($ru)");
  uac_send_req();
}
```

Acc Server Config

```
request_route {
 if(method=="ACCOUNTING" && $rU="store")
 {
     sql_query("ca",
           "insert into accounting
                  (timeval,callid,ftag,ttag,src,dst)
                   values ('$(rb{s.select,0,,})',
                   '$(rb{s.select,1,,})',
                   '$(rb{s.select,2,,})',
                   '$(rb{s.select,3,,})',
                   '$(rb{s.select,4,,})',
                   '$(rb{s.select,5,,})' )",
           "ra");
    send_reply("200", "Stored");
 }
}
```

SIPWEET

Design

- Lua twitter library
- Twitter operation is an HTTP request
  - can take some time to be processed
  - we cannot afford that when processing SIP signaling
  - solution: use asynchronous processing
    - config file message queue
    - dedicated process for twitter operations
- Kamailio modules
  - app_lua
  - mqueue
  - rtimer
  - sqlops

Sample implementation

- **notification of a missed call**
- **use of Twitter direct message**

Config

```
loadmodule "app_lua.so"
loadmodule "rtimer.so"
loadmodule "sqlops.so"
loadmodule “mqueue.so”

# ----- app_lua -----
modparam("app_lua", "load",
        "/usr/local/etc/kamailio/lua/sipweet.lua")

# ----- rtimer -----
modparam("rtimer", "timer",
        "name=sipweet;interval=10;mode=1;")
modparam("rtimer", "exec",
        "timer=sipweet;route=SIPWEET;")

# ----- sqlops -----
modparam("sqlops","sqlcon",
        "ca=>mysql://openser:openserrw@localhost/openser")

# ----- mqueue -----
modparam("mqueue", "mqueue", "name=sipweet")
```

SIPWEET

Config

```
# Twitter routing
route[SIPWEET] {
        # consume tweeties
        while(mq_fetch("sipweet"))
        {
                xlog("Tweeting to $mqk(sipweet) [[$mqv(sipweet)]]\n");

                # get twitter user
                sql_query("ca",
                  "select twuser from sipweetusers where sipuser='$mqk(sipweet)'",
                  "ra");
                if($dbr(ra=>rows)>0)
                {
                        $var(twuser) = $dbr(ra=>[0,0]);
                        $var(twmsg) = $mqv(sipweet);
                        if(!lua_runstring("sipweetdm([[$var(twuser)]], [[$var(twmsg)]])"))
                        {
                                xdbg("failed to send dm to: $mqk(sipweet) - $var(twuser)!\n");
                        }
                }
        }
}
```

Config

```
# Twitees queuing
route[TWQUEUE] {
        if(!is_method("INVITE"))
                return;
        mq_add("sipweet", "$rU", "Missed call from $fU ($Tf)");
}

route {
        ...
        if(!lookup("location")) {
                route(TWQUEUE);
            t_newtran();
            t_reply("404", "Not Found");
            exit;
        }
        ...
}
```

Database table

```
CREATE TABLE `sipweetusers` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `twuser` varchar(64) NOT NULL,
  `sipuser` varchar(64) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`twuser`),
  UNIQUE KEY (`sipuser`)
);
```

```
mysql> select * from sipweetusers;
+----+---------+---------+
| id | twuser  | sipuser |
+----+---------+---------+
|  1 | miconda | 101     |
+----+---------+---------+
```

SIPWEET

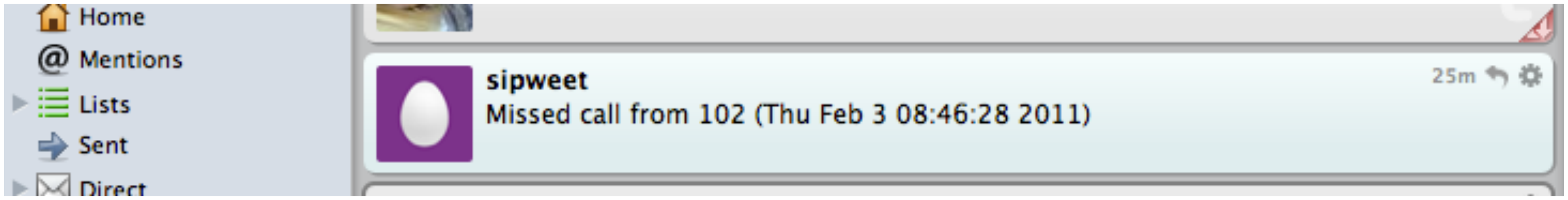
Lua script

```
-- SIPweet

-- loading module
require("twitter")

local initialized = 0
local mytw

function sipweetinit()
    if initialized == 0 then
                    mytw = twitter.client("Consumer Key",
                                           "Consumer Secret",
                                           "OAuth Token",
                                           "OAuth Token Secret");
        initialized = 1
    end
end

function sipweetdm(userid, message)
    sipweetinit()
            mytw:sendMessage{ user = userid, text = message }
end
```

SIPWEET

And the messages goes …

- nice and quick to the twitter client



sipweet
Missed call from 102 (Thu Feb 3 08:46:28 2011)

Fraunhofer FOKUS

sip:wise

asipto

Fraunhofer Forum Berlin

sipgate

Simwood

KAMAILIO WORLD

CONFERENCE & EXHIBITION

itcenter

ngvoice

digium

core net dynamics

Build your own Professional Networks

OBIHAI technology, inc.

vuc

sponsors and exhibitors

??? questions ???

www.asipto.com