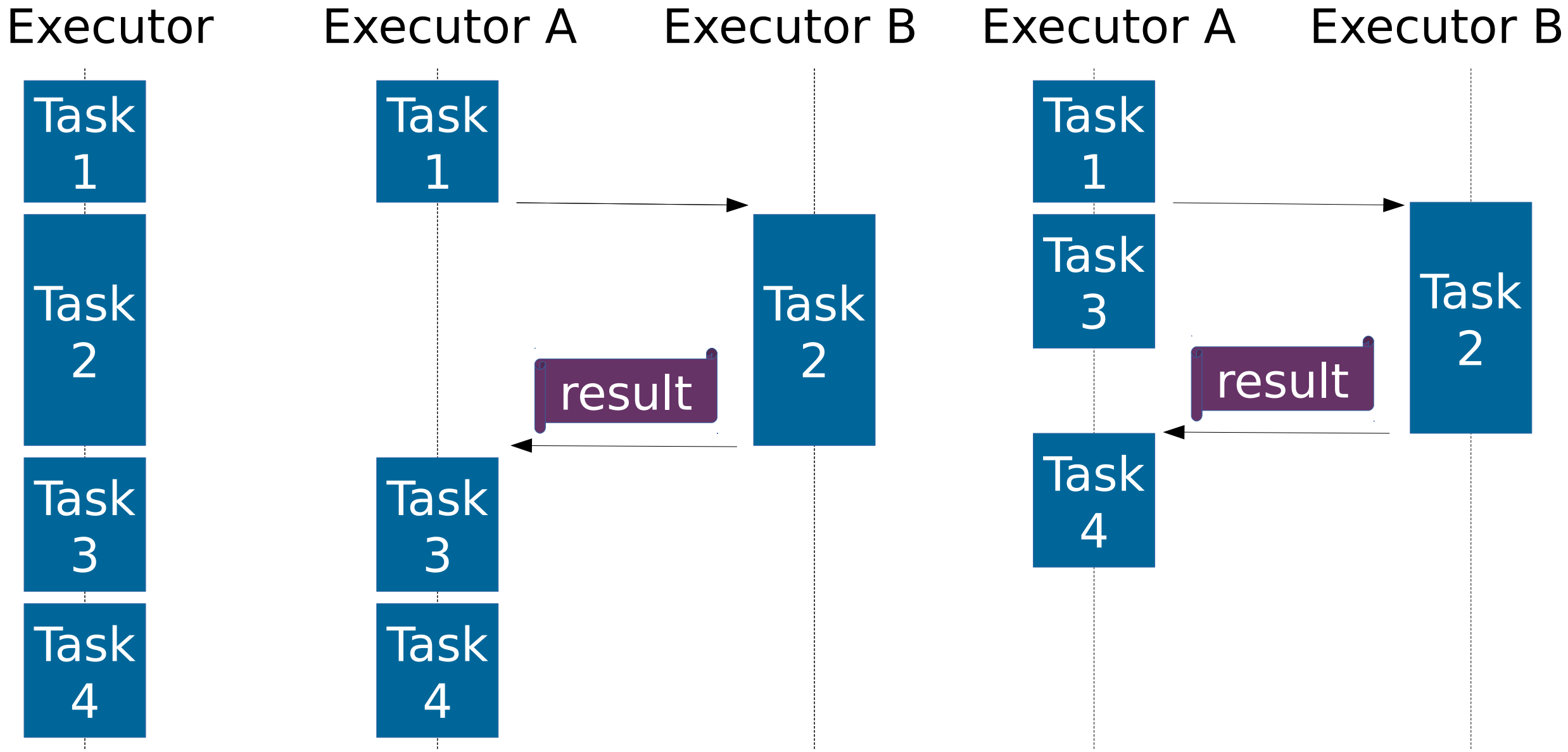
Introduction to RPC, Apache Thrift Workshop

Tomasz Powchowicz

It is all about effective communication

- Synchronous, asynchronous execution



• Thread vs Process

	Thread	Process
Address space	Shares address space	Have own address space
Communication	Mutex, Condition variable, Critical section, Spin lock, Atomics	Inter-process communication (IPC): File, Signal, Socket, Unix Domain Socket, Message queue, Pipe, Named pipe, Semaphore, Shared memory, Message passing (e.g. RPC, RMI), Memory mapped file
Error resiliency	Data on stack and on heap could be damaged.	Address space is protected.
Spawning	Cheap	Expensive
Security	Secure data could be accessed from other thread.	Memory is secured, other resources could be separated by introducing an extra layer such as LXC or KVM.
Scaling	Limited to the number of cores in CPU.	Could be scaled to many CPUs.
Context switching	Fast - shared virtual memory maps.	Slow - A translation lookaside buffer (TLB) flush is necessary.

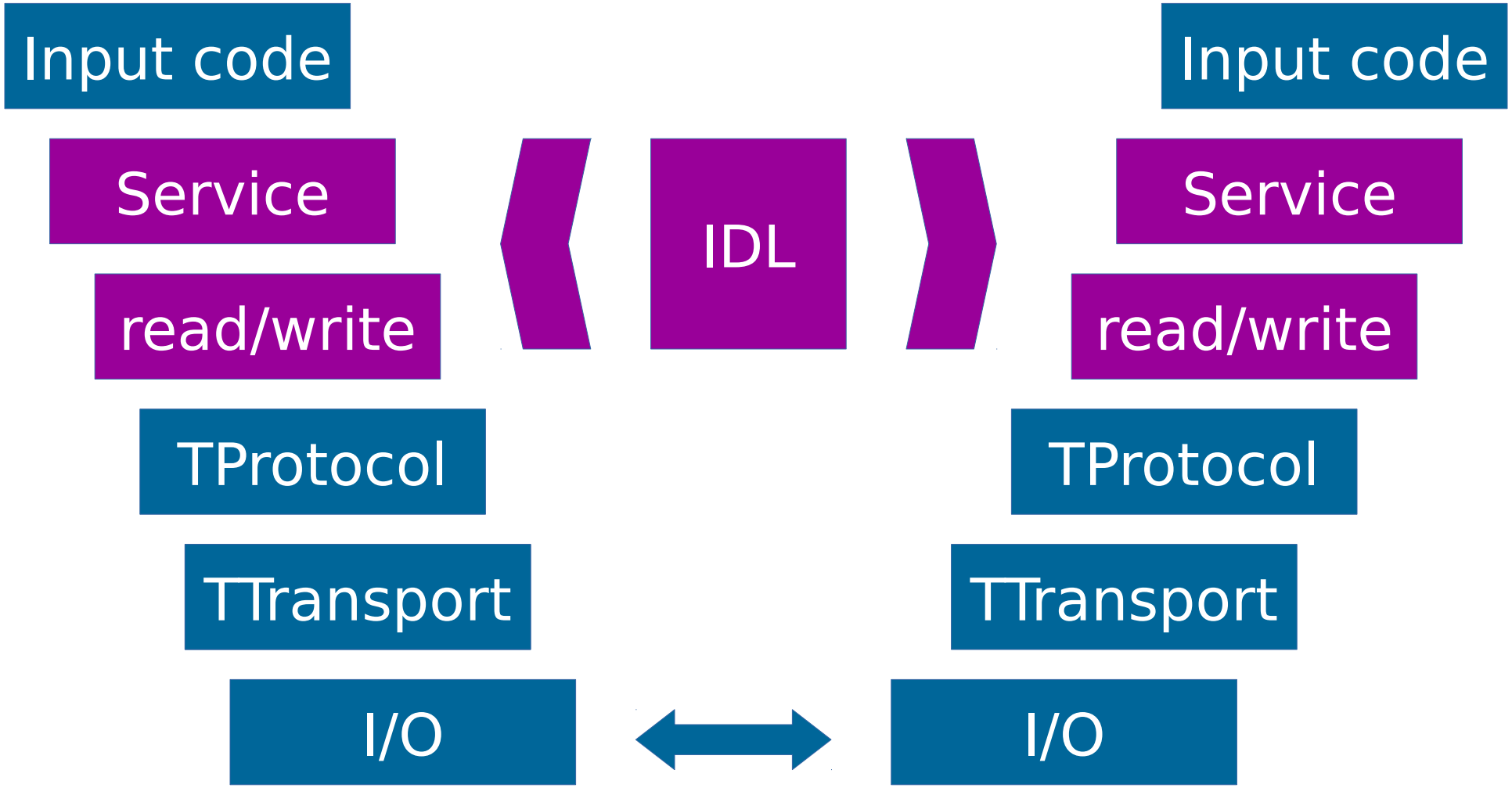
- RPC definition

“In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote.”
source wikipedia: Remote procedure call

Apache Thrift here comes the hero

- Apache Thrift
  - Developed at Facebook, released as a open source project led by Apache Software Foundation
  - Uses Thrift IDL (Inferface definition language) to provide cross-platforms services
  - Supports a set of programming languages: ActionScript, C, C++, C#, Cappuccino, Cocoa, Delphi, Erlang, Go, Haskell, Java, Node.js, Objective-C, OCaml, Perl, PHP, Python, Ruby and Smalltalk
  - Client and server source code is generated from Thrift IDL.
  - The language binds are natural e.g. Java: ArrayList<String>, C++ std::vector<std::string>.
  - Supports various transports (Sockets, Unix Domain Sockets, Framed, File, Memory, Zlib) and protocols (binary, compact, dense, JSON, Debug)
  - License: Apache License 2.0

- Apache Thrift stack



- Apache Thrift IDL (Interface definition language)
  - base types:
    bool, byte: (8-bit signed),
    i16 (16-bit signed), i32 (32-bit signed), i64 (64-bit signed),
    double (64-bit floating point),
    string (UTF-8 encoding).
  - structs - for aggregate data.
  - containers:
    list (STL vector, Java ArrayList),
    set (STL set, Java HashSet, set in Python),
    map (STL map, Java HashMap, Python/Ruby dictionary).
  - enums - for definition of enumeration types.
  - services - for defining interfaces (API).
  - exceptions - uses native exception mechanism appropriate in a target programming language.

Heartbeat monitors the availablity of a resource

- Heartbeat (ping pong)

1) Install node.js & thrift compiler

```
sudo apt-get install nodejs
sudo apt-get install thrift-compiler
```

2) Create ./thrift/Heartbeat.thrift

```
service Heartbeat {
  string ping();
}
```

3) Generate Heartbeat.thrift to gen-nodejs directory

```
thrift --gen js:node ./thrift/Heartbeat.thrift
```

4) Install npm thrift module (locally or globaly -g)

```
npm install thrift
```

5) Create files ./src_nodejs/server.js and ./src_nodejs/client.js

6) Run the server and the client.

```
node ./src_nodejs/server.js
node ./src_nodejs/client.js
```

• Heartbeat (ping pong) sources

```
‘use strict’;
var thrift = require('thrift');
var Heartbeat = require('../gen-nodejs/Heartbeat.js');
var server = thrift.createServer(Heartbeat, {
  ping: function (result) {
        console.log(‘Ping!’);
        result(null, ‘Pong’);
  },
});
server.listen(9090);
```

```
‘use strict’;
var thrift = require('thrift');
var Heartbeat = require('../gen-nodejs/Heartbeat.js');
var connection = thrift.createConnection(‘localhost’, 9090, {
  transport: thrift.TBufferedTransport,
  protocol: thrift.TbinaryProtocol });
var client = thrift.createClient(Heartbeat, connection);
function onHeartbeatTimeout() {
        console.log(‘Heartbeat time-out.’); }
function ping() {
        var exitTimer = setTimeout(onHeartbeatTimeout, 1000);
        client.ping(function (err, response) {
                console.log(response);
                clearTimeout(exitTimer);
                setTimeout(ping, 1000);
        });
}
connection.on('error', function (err) {
        console.log(‘Connection error.’); });
ping();
```

Decoding a message

leveraging the benefits offered by other languages

git clone

https://github.com/tpowchowicz/hubart.git

• Decoding a message: Worker (Thrift IDL)

```
struct WokerRange
{
  1: i64 first,
  2: i64 last
}
service Worker {
  /**
   * Logins to the worker.
   * @return a new worker identifier
   */
   string login();
   /**
    * Logouts from a worker.
    * @param worker_id the worker identifier
    */
   void logout(1: string worker_id);
   /*
    * Reserves a range.
    * @return a reserved range
    */
   WorkerRange reserve();
   /**
    * Reports results.
    * @param range the requested range
    * @param results the list of deccrypted
    *                messages.
    */
   void report(1: WorkerRange range,
               2: list<string> results);
}
```

• Decoding a message: a server

```
‘use strict’;
var thrift = require('thrift');
var Worker = require('../gen-nodejs/Worker.js'),
    ttypes = require('../gen-nodejs/Worker_types.js');
...
var server = thrift.createServer(Worker, {
  login: function(result) {
    ...
    result(null, ‘client_’ + nextWorkerId);
  },
  logout: function(id, result) {
     ...
    result(null);
  },
  reserve: function(result) {
    ...
    result(null, range);
  },
  report: function(workerRange, list, result) {
    ...
    result(null);
  },
});
server.listen(9090);
```

```
var http = require('http');
http.createServer(function(req, res) {
    res.writeHead(200, {'content-type': 'text/plain'});
    var value = ‘Progress: ‘ + (lastRange.first * 100 /
MAX).toFixed(3) + ‘ % of combinations.\n’;
        value += ‘Last range: ‘ + lastRange.last + ‘\n’;
        value += ‘Workers: ‘ + workers + ‘\n’;
        value += ‘Secret: \n’;
        for (let item of foundedList) {
                value += item + ‘\n’;
        }
    res.end(value);
}).listen(8000, '0.0.0.0');
```

• Decoding a message: a client (JS and C++)

```javascript
'use strict';
var thrift = require('thrift');
var Worker = require('../gen-nodejs/Worker.js');
var ttypes = require('../gen-nodejs/Worker_types.js');
var connection = thrift.createConnection("localhost", 9090, {
  transport : thrift.TBufferedTransport,
  protocol : thrift.TBinaryProtocol
});
var client = thrift.createClient(Worker, connection);
function calc(workerId) {
  client.reserve().then(range => {
      if (range.last == 0) {
        client.logout(workerId);
      } else {
        client.report(range, decrypt(range)).then(
          function() {
            setImmediate(function() {
              calc(workerId);
          });
        });
      }
  });
}
client.login().then(workerId => { calc(workerId); });
//process.stdout._handle.setBlocking(true);
```

```cpp
std::shared_ptr<TTransport> socket(new TSocket("localhost", 9090));
std::shared_ptr<TTransport> transport(new TBufferedTransport(socket));
std::shared_ptr<TProtocol> protocol(new TBinaryProtocol(transport));
WorkerClient client(protocol);
try {
  transport->open();
  std::string worker_id; client.login(worker_id);
  bool running = true;
  while (running) {
    WorkerRange range; client.reserve(range);
    if (range.last == 0) {
      break;
    } else {
      std::vector<std::string> fibs_vector;
      decrypt(range, fib_vector);
      client.report(range, fibs_vector);
    }
  }
  client.logout(worker_id);
  transport->close();
} catch (TException& tx) {
  std::cout << "ERROR: " << tx.what() << std::endl;
}
```

- Thrift library building from source (C++ compilation)
  - Download sources from https://thrift.apache.org/download
  - Requirements for building from source GNU build tools: http://thrift.apache.org/docs/install

    e.g. sudo apt-get install libssl-dev
  - Installation (described in README.md)

    ./bootstrap.sh

    ./configure

    make

    sudo make install

- Resources
  - Thrift IDL (interface description language)
    https://thrift.apache.org/docs/idl
  - Thrift tutorial

    http://thrift-tutorial.readthedocs.io/en/latest/index.html
  - Thrift installation

    http://thrift.apache.org/docs/install
  - Thrift download

    https://thrift.apache.org/download

Thank you

adbglobal.com