Automatic pipeline parallelism for event-loops.

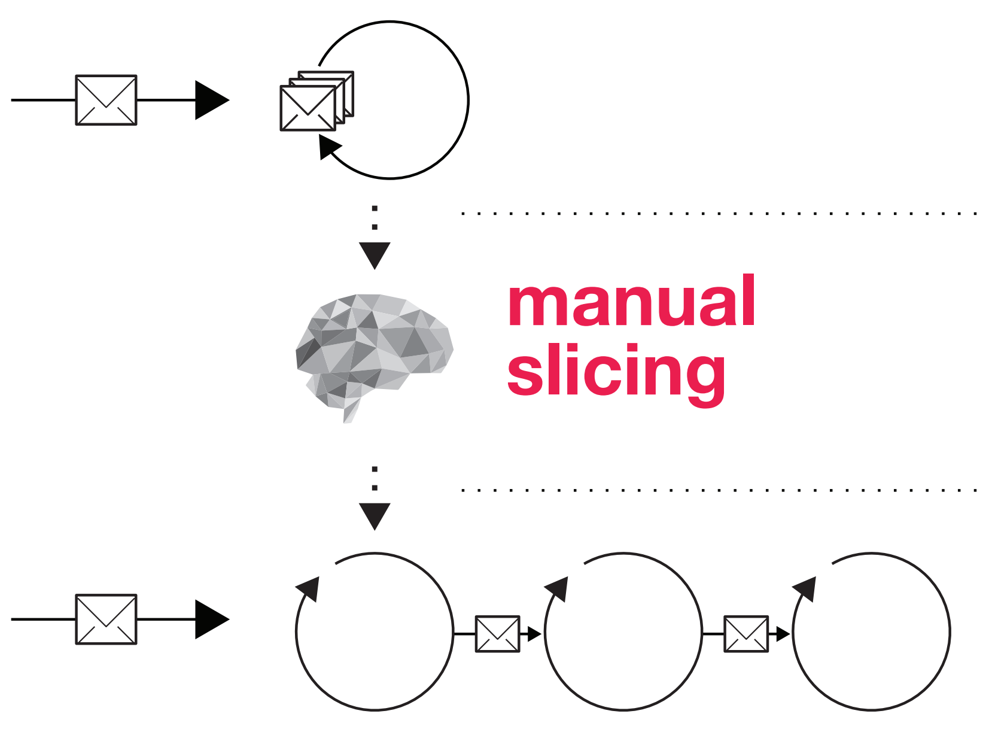
Etienne Brodu
etienne.brodu@insa-lyon.fr

Stéphane Frénot
stephane.frenot@insa-lyon.fr

Frédéric Oblé
frederic.oble@worldline.com

An event-loop is a pipeline on a single core.

event-loop



+ linearly scalable
+ no synchronization
- **limited to a single core**

need to balance stages
at each code modification

+ linearly scalable
- **costly evolution**

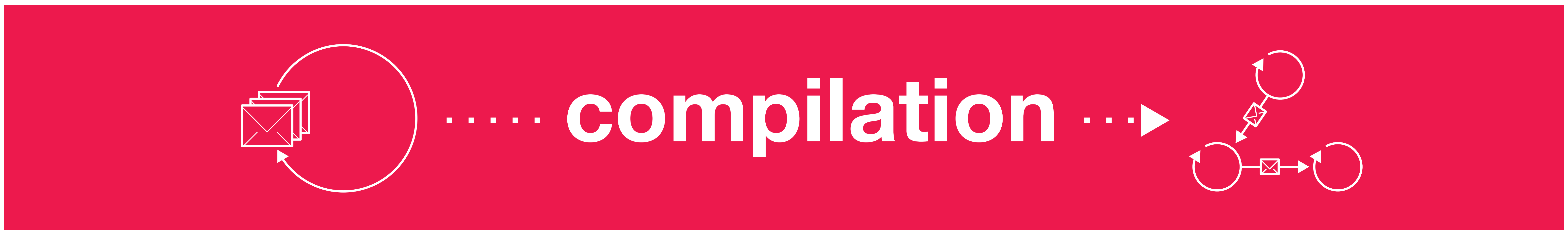
pipeline parallelism

problematic

an event-loop is limited, and manually switching to pipeline parallelism is costly.

proposition

We propose to automate the slicing process at compile time.



event-loop assures

execution independence

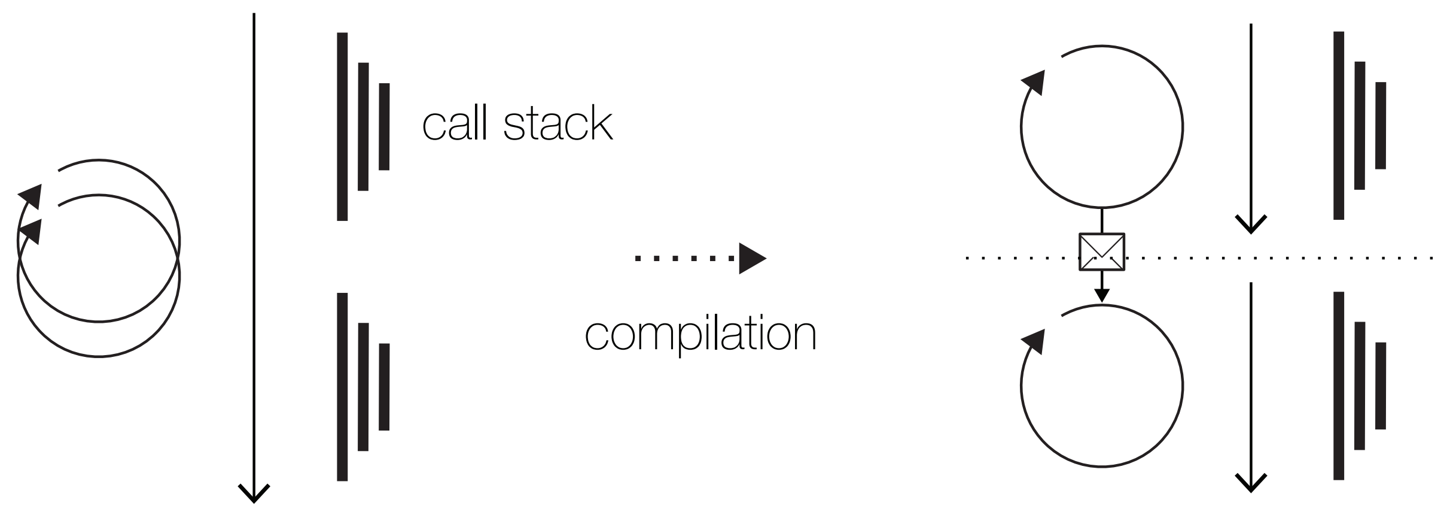
&

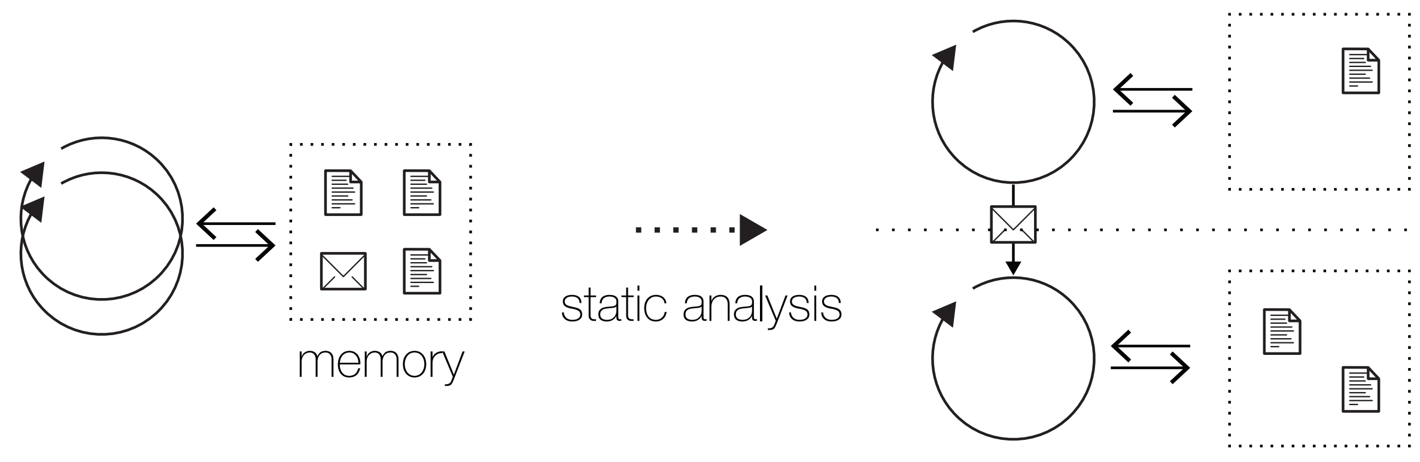
static analysis provides

memory independence

event handler **1**

event handler **2**





compilation example

source[5]

```
var app = require('express')(),
    fs = require('fs'),
    count = 0;

app.get('/', function handler(req, res){
  fs.readFile(__filename, function reply(err, data){
    count += 1;
    var code = (''+ data)
                     .replace(/\n/g, '<br>')
                     .replace(/ /g, '&nbsp');
    res.send(err
          || 'downloaded ' + count +
           ' times<br><br><code>' +
           code + '</code>');
  });
});

app.listen(8080);
console.log('>> listening 8080');
```

target[5]

```
flx source.js {}
>> handler-1000 [res]
  var app = require('express')(),
       fs = require('fs'),
       count = 0;
  app.get('/', >> handler-1000);
  app.listen(8080);
  console.log('>> listening 8080');
```

req, res

```
flx handler-1000 {fs}
-> reply-1001 [res]
  function handler(req, res) {
    fs.readFile(__filename, -> reply-1001);
  }
```

res, err, data

```
flx reply-1001 {count}
-> null
  function reply(err, data) {
    count += 1;
    var code = ('' + data)
                   .replace(/\n/g, '<br>')
                   .replace(/ /g, '&nbsp');
    res.send(err
          || 'downloaded ' + count +
             ' times<br><br><code>' +
             code + '</code>');
  }
```

[1] A Adya, J Howell and M Theimer. "Cooperative Task Management Without Manual Stack Management."
In : USENIX Annual Technical Conference (2002).

[2] JR von Behren, J Condit and EA Brewer. "Why Events Are a Bad Idea (for High-Concurrency Servers)."
In : HotOS (2003).

[3] J Ousterhout. "Why threads are a bad idea (for most purposes)".
Presentation given at the 1996 Usenix Annual Technical Conference (1996).

[4] M. Welsh, S. Gribble, E. Brewer, and D. Culler. A Design Framework for Highly Concurrent Systems.
CS Technical Report UCB/CSD-00-1108, University of California, Berkeley, October 2000.

[5] flx-example: https://github.com/etnbrd/flx-example/tree/1.0. Accessed: 2014-08-22.