



HAL
open science

A compiler providing incremental scalability for web applications

Etienne Brodu, Stéphane Frénot, Fabien Cellier, Frédéric Oblé

► To cite this version:

Etienne Brodu, Stéphane Frénot, Fabien Cellier, Frédéric Oblé. A compiler providing incremental scalability for web applications. Middleware Posters and Demos '14, Dec 2014, Bordeaux, France. , ACM, 2014, Proceedings of the Posters & Demos Session. 10.1145/2678508.2678526 . hal-01076857

HAL Id: hal-01076857

<https://hal.science/hal-01076857>

Submitted on 9 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A compiler providing incremental scalability for web application

Etienne Brodu
etienne.brodu@insa-lyon.fr

Stéphane Frénot
stephane.frenot@insa-lyon.fr

Fabien Cellier
fabien.cellier@worldline.com

Frédéric Oblé
frederic.oble@worldline.com

To develop a **web application**, one have to **choose**

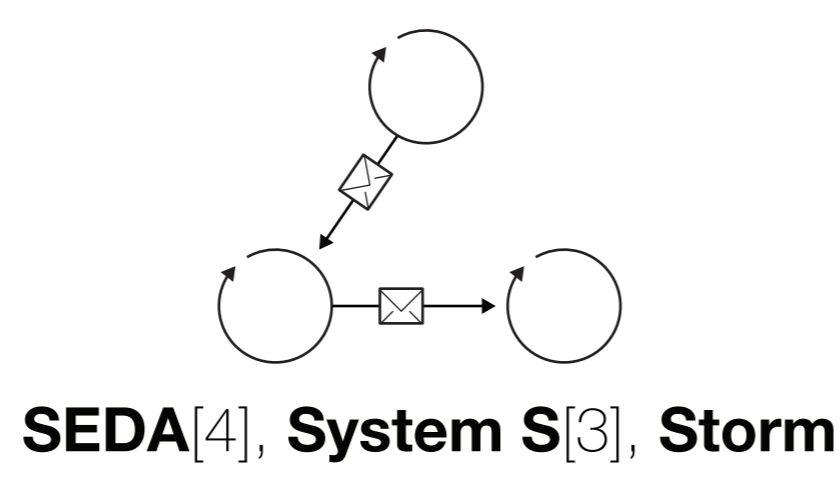
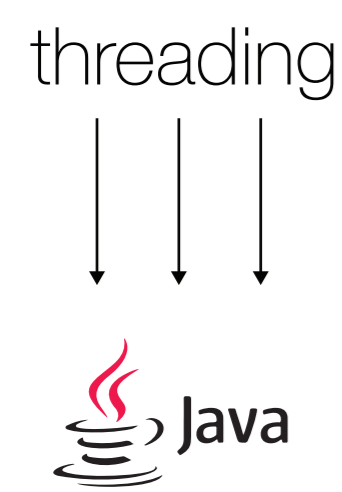
features
scalability

performance
scalability

Monolithic models

Or

Distributed models



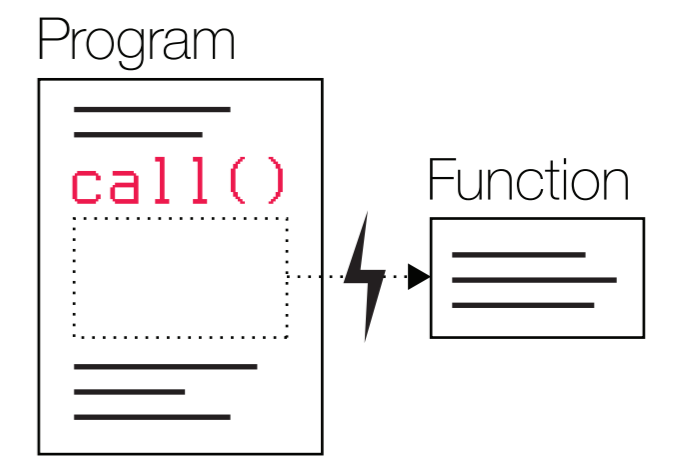
We want to **compile** into

The compiler extracts **autonomous parts** by searching for **rupture points** marking them out

A **Rupture point** is a call of a loosely coupled function

In **node**, an **asynchronous call** is a **rupture point**.

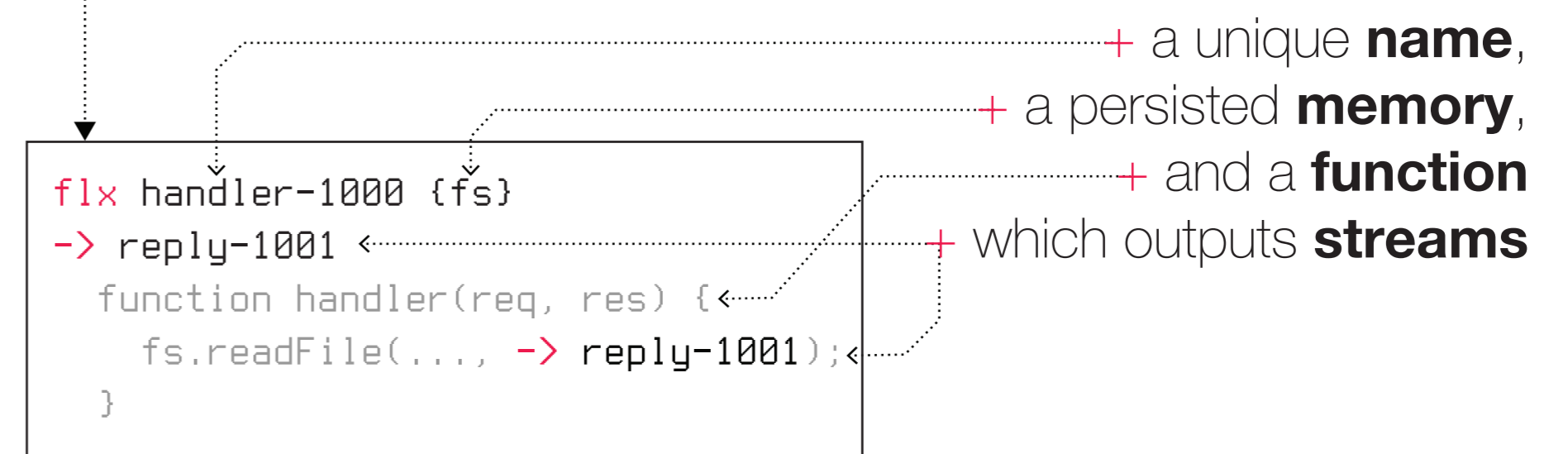
I/O Operation like **fs** and **express**.



```
var app = require('express')();
app.get('/', function handler(req, res){
  fs.readFile(_filename, reply);
});
```

Application parts are enveloped into **fluxions**.

A **fluxion** is an autonomous with :



..... compilation>

source[1]

target[1]

```
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');
```

```
f1x 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');
```

```
f1x handler-1000 {fs}
-> reply-1001 [res]
function handler(req, res) {
  fs.readFile(_filename, -> reply-1001);
}
```

```
f1x 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] fx-example: <https://github.com/etnbrd/fx-example/tree/1.0>. Accessed: 2014-08-22.

[2] Fox, A. et al. 1997. Cluster-based scalable network services.

[3] Jain, N. et al. 2006. Design, implementation, and evaluation of the linear road benchmark on the stream processing core. SIGMOD '06 Proceedings of the 2006 ACM SIGMOD international conference on Management of data.

[4] Welsh, M. et al. 2000. A design framework for highly concurrent systems.