Skitter: A DSL for Distributed Reactive Workflows
Saey, Mathijs; De Koster, Joeri; De Meuter, Wolfgang

Publication date: 2018

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 08. Aug. 2021
A DSL for Distributed Reactive Workflows

Mathijs Saey, Joeri De Koster, Wolfgang De Meuter

Context

We are producing a lot of data, we need software that reacts to this data instantaneously.

Scale of data forces us to execute on a cluster. Need to deal with partial failure, replication, consistency, ...

Problem Statement

We want a programming language which allows one to write scalable, reactive big data applications from a set of existing, reactive components.

Related Work

<table>
<thead>
<tr>
<th></th>
<th>Reactive Programming</th>
<th>Stream Processing</th>
<th>Scientific Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive</td>
<td>✓</td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td>Scalable</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Existing Components</td>
<td>?</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>

Write Reactive Components

```
component Distance, in: [p1, p2], out: [distance] do
  react (x1, y1), (x2, y2) do
    sqrt(square(x2 - x1) + square(y2 - y1)) ~> distance
  end
end
```

Component with mutable state

```
component GeoFilter, in: [json], out: [inside, outside] do
  fields area
  init area_json do
    area <~ area_json
  end
  react json do
    loc = ... # extract location from json
    r = System.cmd "in_area", ["--area #={area}", loc]
    if r == "inside", do: json ~> inside, else: json ~> outside
  end
end
```

As a wrapper for foreign code

```
component Count, in: [any], out: [current] do
  effect state_change
  fields count
  init _ do
    count <~ 0
  end
  react _ do
    count <~ count + 1
    count ~> current
  end
end
```

Component with mutable state

Effects

```

effect state_change
```

Property | Effect | Additional Primitives
---|---|---
Mutable state | state_change | <-
Foreign process with mutable state | state_change hidden | create_checkpoint, restore_checkpoint, clean_checkpoint
I/O may occur | external_effect | after_failure

Compose Reactive Workflows

Execute on a Cluster

Skitter passes messages between connected components and activates react and other functions

Each token entering a workflow is processed concurrently, Skitter will replicate individual components as needed

<table>
<thead>
<tr>
<th>/</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>state_change</td>
<td>1</td>
</tr>
<tr>
<td>state_change_hidden</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/</th>
<th>external_effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replay</td>
</tr>
<tr>
<td>state_change</td>
<td>restore, replay, (after_failure)</td>
</tr>
<tr>
<td>state_change_hidden</td>
<td>restore checkpoint, replay, (after_failure)</td>
</tr>
</tbody>
</table>

Skitter automatically handles replication and partial failure handling based on the effects of a component