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, \( \text{in: } [p1, p2], \text{ out: } [\text{distance}] \) do

\[
\text{react } (x1, y1), (x2, y2) \text{ do } \sqrt{\text{square}(x2 - x1) + \text{square}(y2 - y1)} \rightarrow \text{distance}
\]

Component GeoFilter, \( \text{in: } [\text{json}], \text{ out: } [\text{inside, outside}] \) do

\[
\text{fields area} \\
\text{init area json do} \\
\text{area \rightarrow area json} \\
\text{react json do} \\
\text{loc = ... # extract location from json} \\
r = \text{System.cmd "in_area", ["--area \# {area}", loc]} \\
\text{if } r = "inside", \text{ do: json \rightarrow inside, else: json \rightarrow outside}
\]

Component Count, \( \text{in: } [\text{any}], \text{ out: } [\text{current}] \) do

\[
\text{fields count} \\
\text{init_ do} \\
\text{count \leftarrow 0} \\
\text{react_ do} \\
\text{count \leftarrow count + 1} \\
\text{count \leftarrow current}
\]

Effects

Component Count, \( \text{in: } [\text{any}], \text{ out: } [\text{current}] \) do

\[
\text{effect state_change} \\
\text{fields count} \\
\text{init_ do} \\
\text{count \leftarrow 0} \\
\text{react_ do} \\
\text{count \leftarrow count + 1} \\
\text{count \leftarrow current}
\]

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>Property</th>
<th>Effect</th>
<th>Additional Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable state</td>
<td>state_change</td>
<td>~&lt;</td>
</tr>
<tr>
<td>Foreign process with</td>
<td>state_change</td>
<td>~&lt;</td>
</tr>
<tr>
<td>mutable state</td>
<td>state_change</td>
<td>~&lt;</td>
</tr>
<tr>
<td>I/O may occur</td>
<td>external_effect</td>
<td>after_failure</td>
</tr>
</tbody>
</table>

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