Multi-Scheduling in FunLoft

Pejman Attar

INRIA

29 juin 2010
We have 3 different levels in DSLM:

- Scripts
- Agents
- Sites
Script are composed of expression and instruction where expression should finished in one instant but the instruction can take several instant to finished.
Syntax:
Expression

\[ e \in \text{Exp}^\sim ::= ~ \]
\[ x \mid v \mid f(\bar{e}) \]
\[ \mid !e \mid e := e \]
$s \in \text{Script} \quad ::= \quad$

$\text{Exp}$

$s; s$

$\text{let } x = e \text{ in } s \text{ end}$

$\text{if } e \text{ then } s \text{ else } s \text{ end}$

$\text{loop } s \text{ end} | \text{repeat } e \text{ do } s \text{ end}$

$\text{cooperate}$

$\text{launch } m(\vec{e})$

$\text{generate } ev \text{ with } v | \text{await } ev \text{ with } l | \text{do } s \text{ watching } e$

$\text{drop } s \text{ in } site : Ag$

$\text{letagent } Ag = s \text{ in } site \text{ end}$

$\text{migrate } Ag \text{ to } site$
Agents

Each agent contains a script (parallel scripts). Each agent has its own memory, and a script that is not belonging to the agent cannot access to its memory.
Sites is contains a set of Agents which are executed in the same peace, and a set of events which are produce during the courant instant on the site.
Example: Simulation of gas molecules (1 room)
Example: Simulation of gas molecules (2 room)
Example: Simulation of gas molecules (2 room with a connection)
Example: Compute on a remote site

Site 1:

- Events
- Memory

\[ \text{let } x = f(0) \text{ in}
\quad \text{drop}
\quad \text{generate ev with } x
\quad \text{in site 2}
\quad \text{end} \]

Site 2:

- Events
- Memory

\[ \text{let } x = \text{ref 0 in}
\quad \text{await ev with } x;
\quad \text{print(!x)}
\quad \text{end} \]
Example: Compute on a remote site

```
let x = f(0) in
  drop
  generate ev with x in site2
end
```

```
let x = ref 0 in
  await ev with x;
  print(!x)
end
```
Example: Compute on a remote site

Site1

Events

Memory

let x = f(0) in
drop
  generate ev with x
in site2
  end

S1

Site2

Events

Memory

(l <= 0)

let x = ref 0 in
  await ev with x;
  print(!x)
  end
Example: Compute on a remote site

Site 1

Events

Memory

let x = f(0) in
  drop
  generate ev with x
  in site 2
end

S1

Site 2

Events

(ev, x)

Memory

I <= x

let x = ref 0 in
  await ev with x;
  print(!x)
end

generate ev with x

S1
Dynamique scheduling

We want to use all the disponible resouces (cores). To resoulve this problem we propose:

- Expansion and contraction of a site
- Auto-Migration of Agents
Schedulers are the native thread which are mapped to each core. Each scheduler can contain one or several agents.
A site is a set of synchronized schedulers which means all the schedulers belonging to the same site are executed in a same pace.
Expansion and Contraction of a site

- **Expansion of a site**

  \[ \ldots, [\text{sched} : \{Ag, Ag', \ldots \}, \ldots], \ldots \rightarrow \]

  \[ \ldots, [\text{sched} : \{Ag, \ldots \}, \text{sched}' : \{Ag'\}, \ldots ], \ldots \]

- **Contraction of a site**

  \[ \ldots, [\text{sched} : \{Ag, \ldots \}, \text{sched}' : \{Ag'\}, \ldots ], \ldots \rightarrow \]

  \[ \ldots, [\text{sched} : \{Ag, Ag', \ldots \}, \ldots ], \ldots \]
Expansion and Contraction of a site
Expansion and Contraction of a site

Pejman Attar

Multischeduling in FunLoft
Expansion and Contraction of a site

Pejman Attar

Multischeduling in FunLoft
Expansion and Contraction of a site

Multischeduling in FunLoft
Auto migration of agents

Automatic migration rule

..., [sched : \{Ag, ...\}, sched' : \{Ag', ...\}, ...], ...
→ ..., [sched : \{Ag, Ag', ...\}, sched' : \{...\}, ...], ...
Multischeduling in FunLoft
Next Step

- Soundness of Typing
- Fixpoint
- Implementation