Orchestration and beyond

Pejman Attar
(joint work with F. Boussinot)

MSR'11 - Lille

16 november 2011

INRIA

Context : Service orchestration

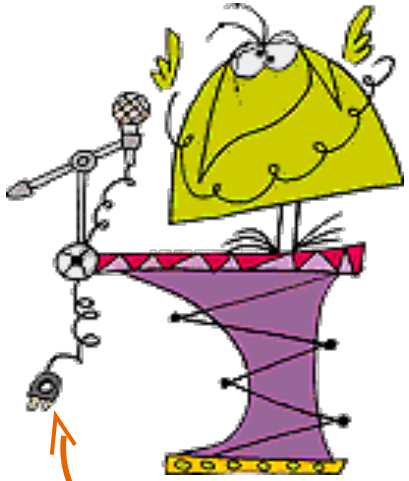
- **Orchestration :** Combining services (Web, Databases)
- **Orchestration Languages :** Orc, Jolie
- **Synchronous Languages :** Esterel, Lustre, Signal
- **Reactive Languages :** SugarCubes, FunLoft
- **GALS systems**

Our objectives

- Kernel orchestration language with simple (abstract) semantics
- No memory No problem
- Synchronous/asynchronous parallelism
- Absent resource management: timeout / watching
- Multi-core architectures

Dynamic Synchronous Language for Orchestration (DSL)

Orchestration paradigm (one site)



Orchestration

Function call

result

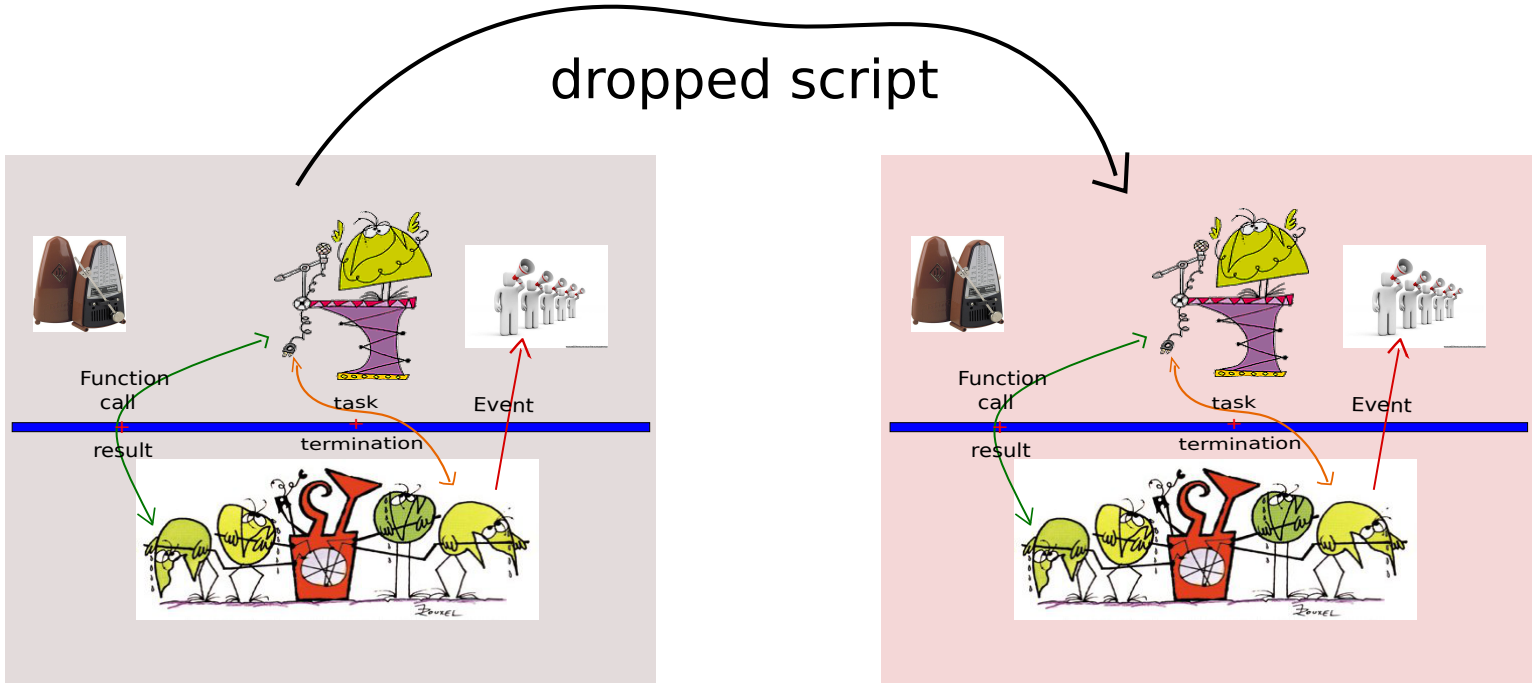
task

termination

Event

Host

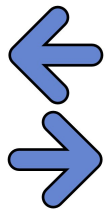
Orchestration paradigm (multiple sites)



Events are not shared between the sites

Example : *Simon*

```
launch sleep(1000000);
init_windows();
launch new_frame();
loop
 do
  launch show_frame(300000);
   (
    launch observer ()
     ||
    do
     repeat get_length() do          await NOK
      await OK;                      launch failure ();
      cooperate                 ||   launch new_frame ();
     end                             generate error;
    watching done;
    launch success ();
    launch increment_frame ();
   )
 watching error;
 launch sleep(100000);
end
```



- init_windows
- get_length
- sleep
- new_frame
- show_frame
- observer
- failure
- success
- increment_frame

$s \in \mathit{Script} ::=$

$$
\begin{array}{l}
\mid \quad s ; s \\
\mid \quad s \,||\, s \\
\mid \quad \texttt{if}\ b\ \texttt{then}\ s\ \texttt{else}\ s\ \texttt{end} \\
\mid \quad \texttt{loop}\ s\ \texttt{end} \\
\mid \quad \texttt{repeat}\ n\ \texttt{do}\ s\ \texttt{end} \\
\mid \quad \texttt{cooperate} \\
\mid \quad f\ (\ldots) \\
\mid \quad \texttt{launch}\ t\ (\ldots) \\
\mid \quad \texttt{generate}\ \mathit{ev} \\
\mid \quad \texttt{await}\ \mathit{ev} \\
\mid \quad \texttt{do}\ s\ \texttt{watching}\ \mathit{ev} \\
\mid \quad \texttt{drop}\ s\ \texttt{in}\ \mathit{site}
\end{array}
$$

Assumed properties

- Instantaneous termination of functions
- Data-race freedom for fnctions and tasks
- Existence of site for Drop

- Script Big-step Semantics

$$P \vdash s \xrightarrow{b} s', G, D$$

$$P \vdash \texttt{drop}\ s\ \texttt{in}\ site \xrightarrow{tt} \texttt{nothing}, \emptyset, \{site \downarrow s\}$$

$$f_s(P) = G \quad where \quad P \vdash s \xrightarrow{b} s', G, D$$

- Site Semantics

$$\{..., \{site : s\}, \{site \downarrow s'\}, ...\} \mapsto \{..., \{site : s \,||\, s'\}, ...\}$$

$$\frac{Lfp_s \vdash s \xrightarrow{b} s', Lfp_s, D}{\{..., \{site : s\}, ...\} \mapsto \{..., \{site : s'\}, ...\}, D}$$

Generate

$$P \vdash \texttt{generate}\ ev \xrightarrow{tt} \texttt{nothing}, \{ev\}, \emptyset$$

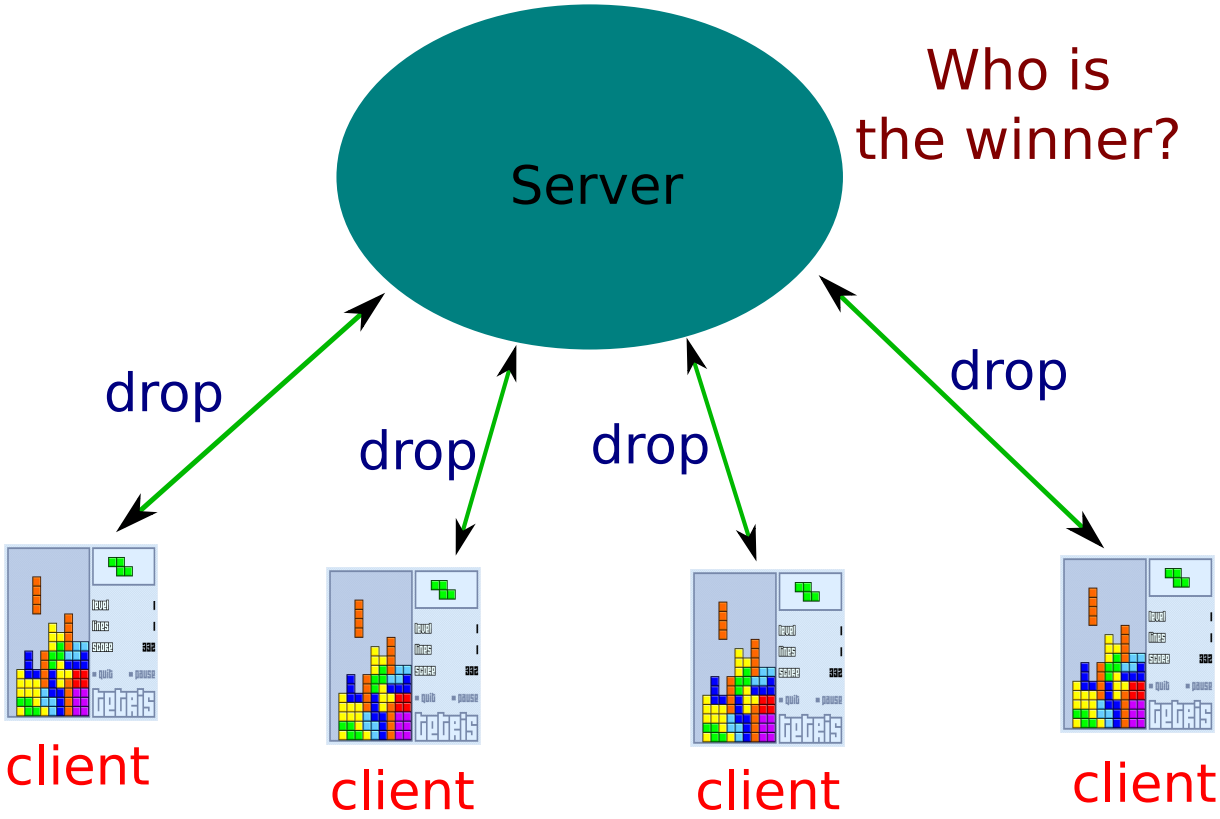
Await

$$\frac{ev \in P}{P \vdash \texttt{await}\ ev \xrightarrow{tt} \texttt{nothing}, \emptyset, \emptyset}$$

$$\frac{ev \notin P}{P \vdash \texttt{await}\ ev \xrightarrow{ff} \texttt{await}\ ev, \emptyset, \emptyset}$$

Extensions of DSL

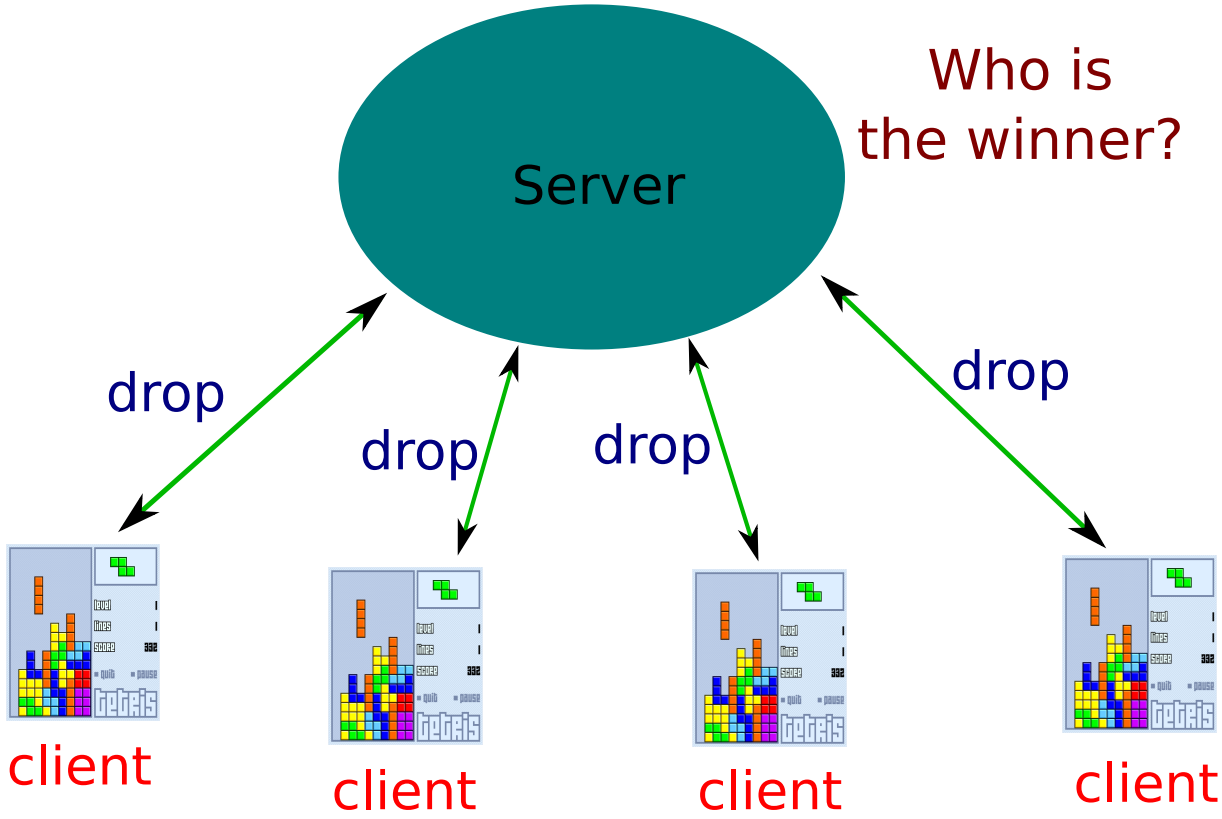
Need of memory at orchestration level



The last loser is the winner

Need of memory at orchestration level

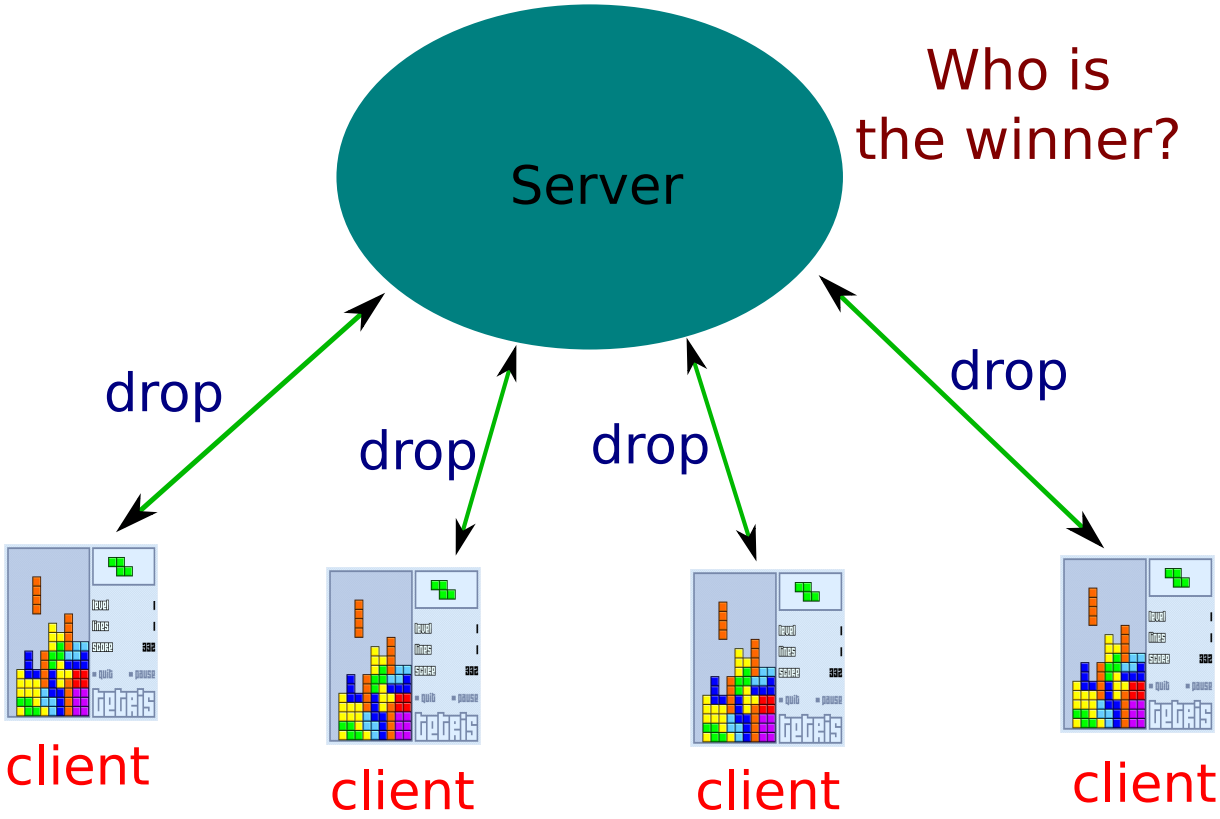
- Excessively Static



The last loser is the winner

Need of memory at orchestration level

- Excessively Static
- Memory will be helpfull



The last loser is the winner

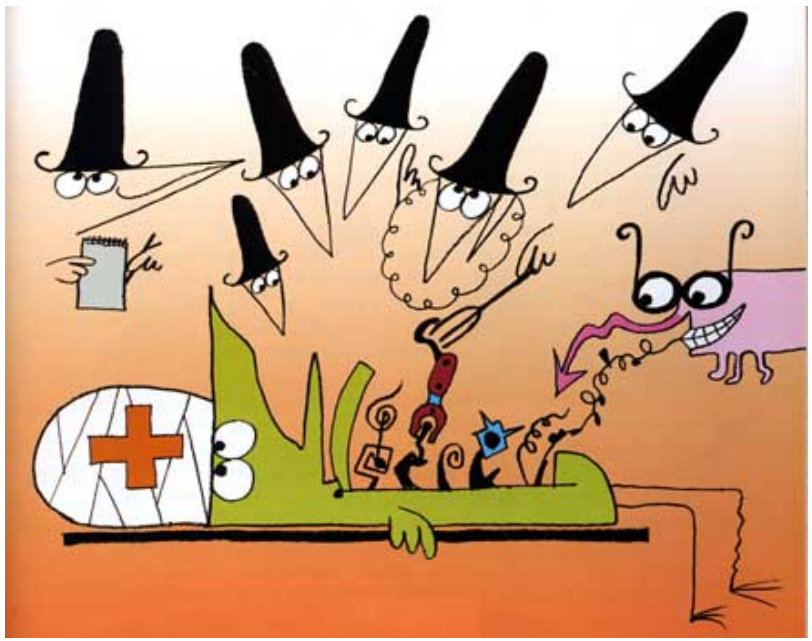
DSL + Memory = DSLM
Dynamic Script Language with Memory

1- Memory protection against concurrent access (Avoid data-races)
2- Maximal usage of cores

1- Memory

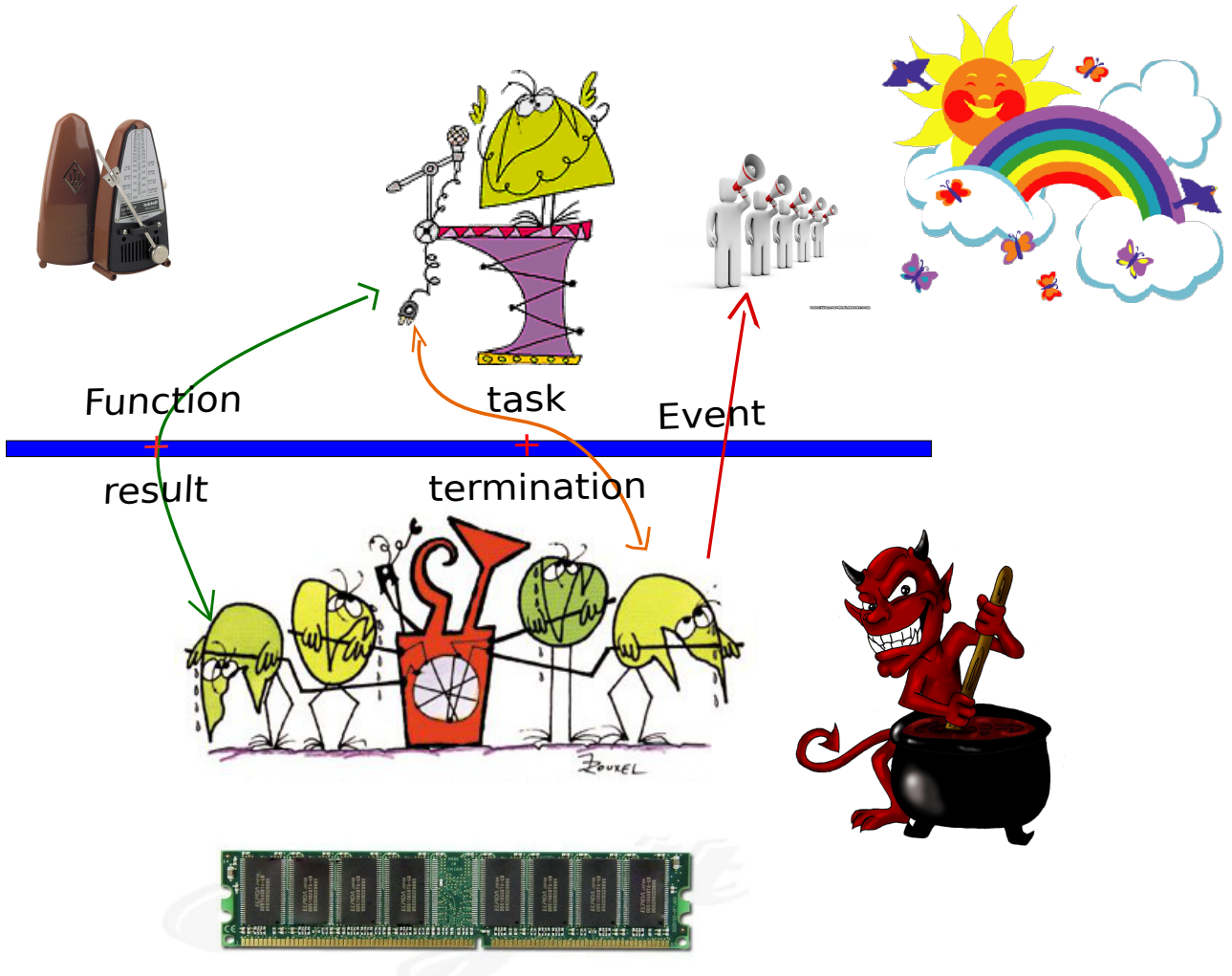
Creation	let x = ref exp in s end
Assignment	x := exp;
Access	!x

Memory sharing?

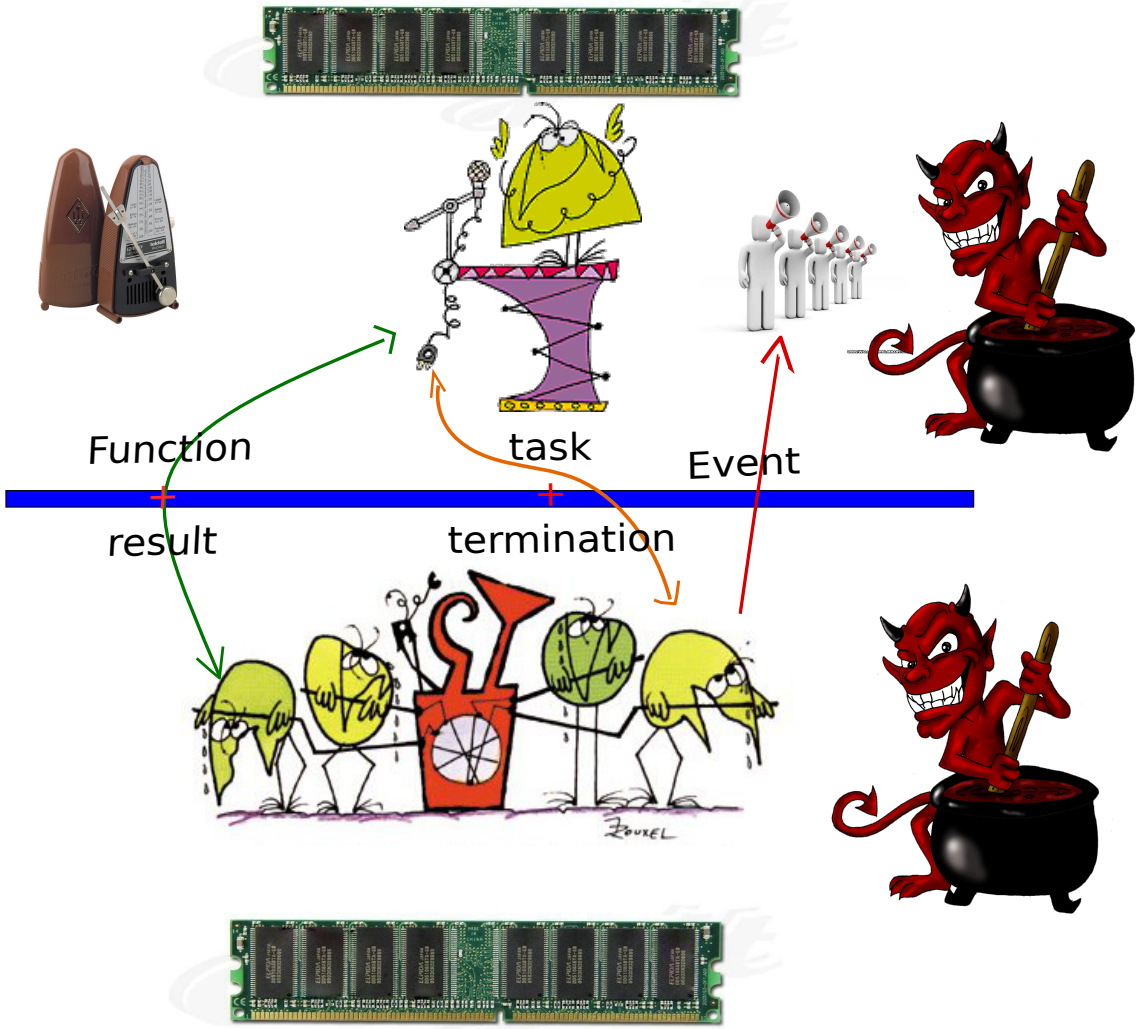


Bad idea!

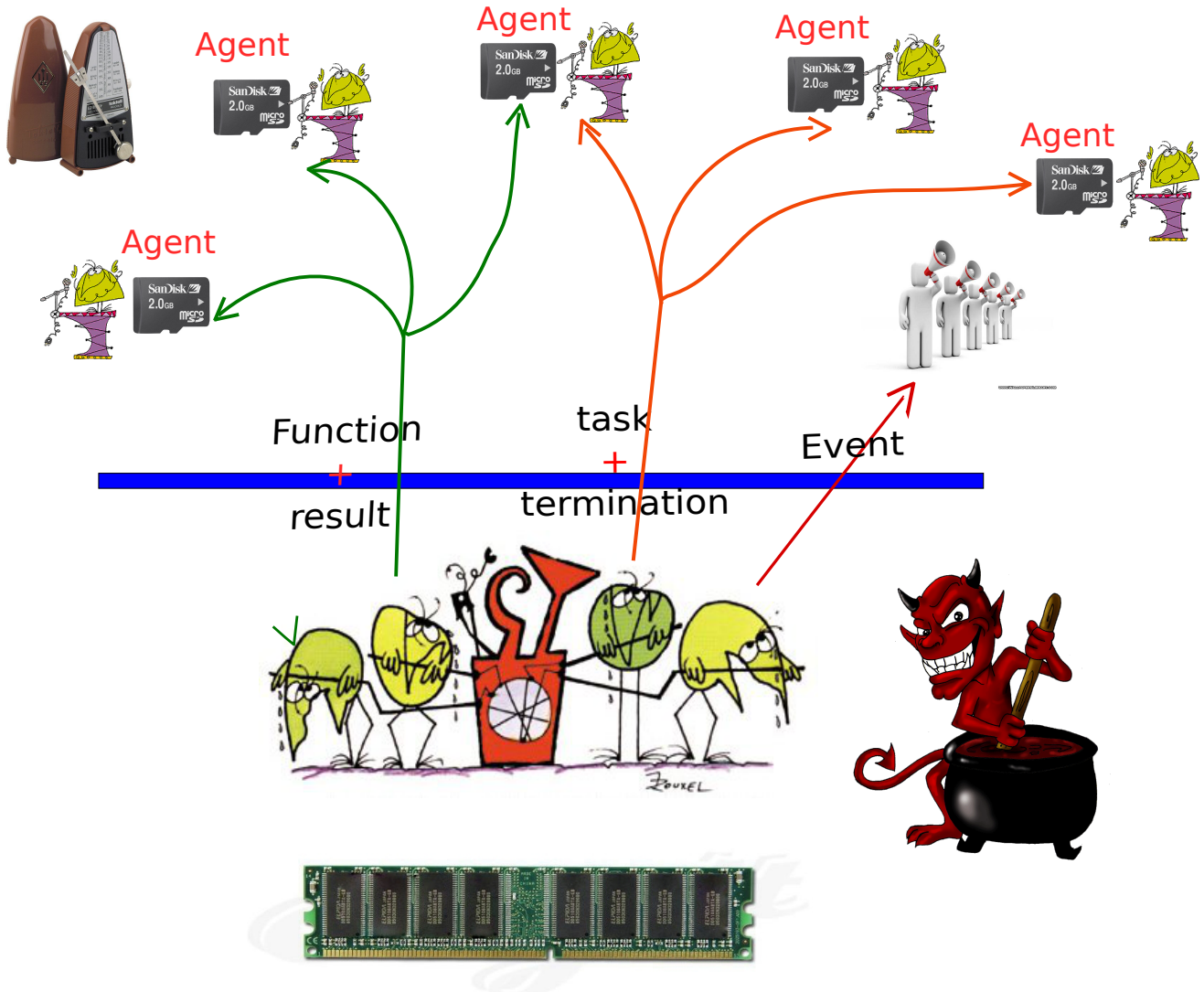
DSL model



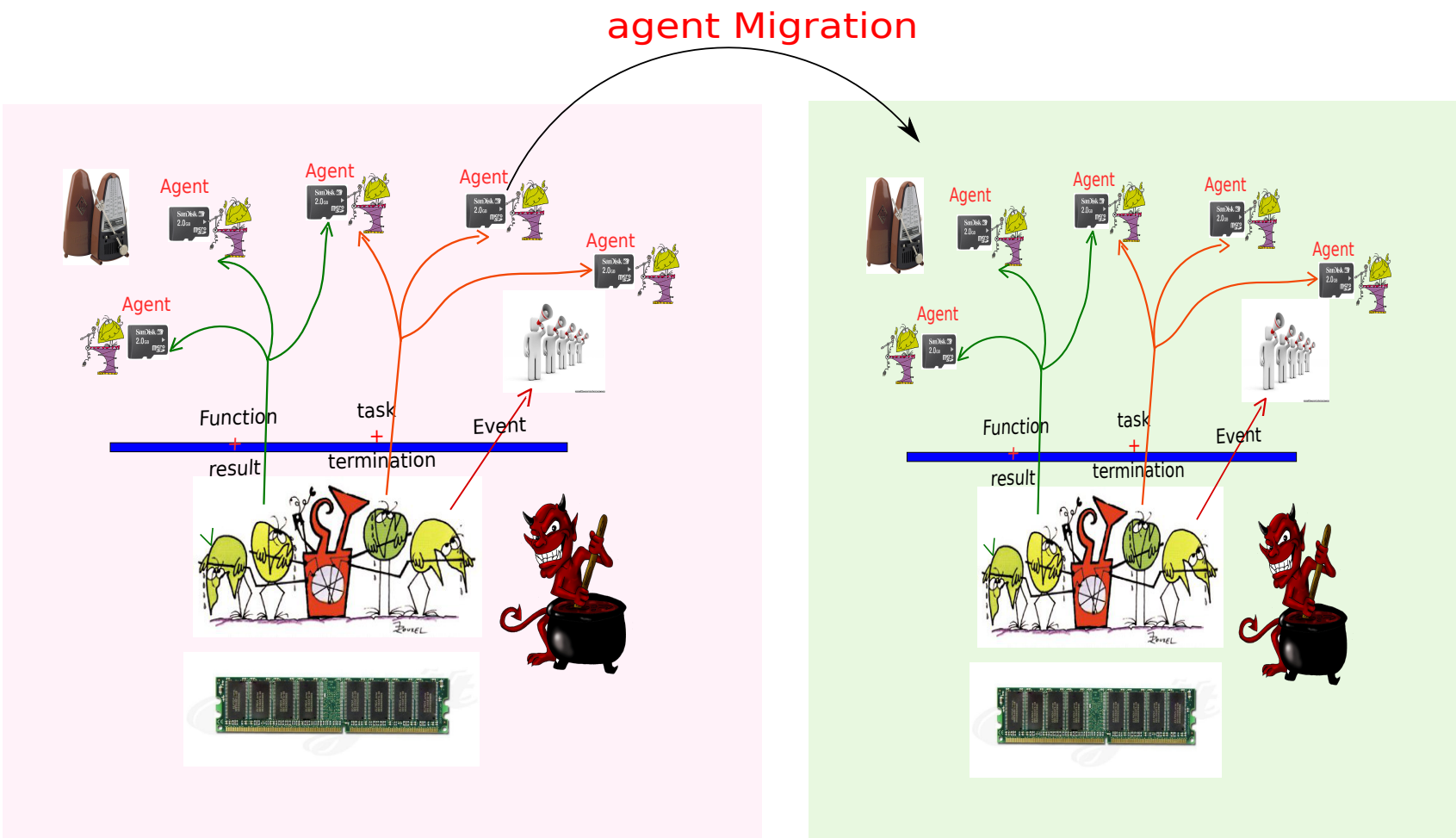
Evil model



DSLM model (one site)

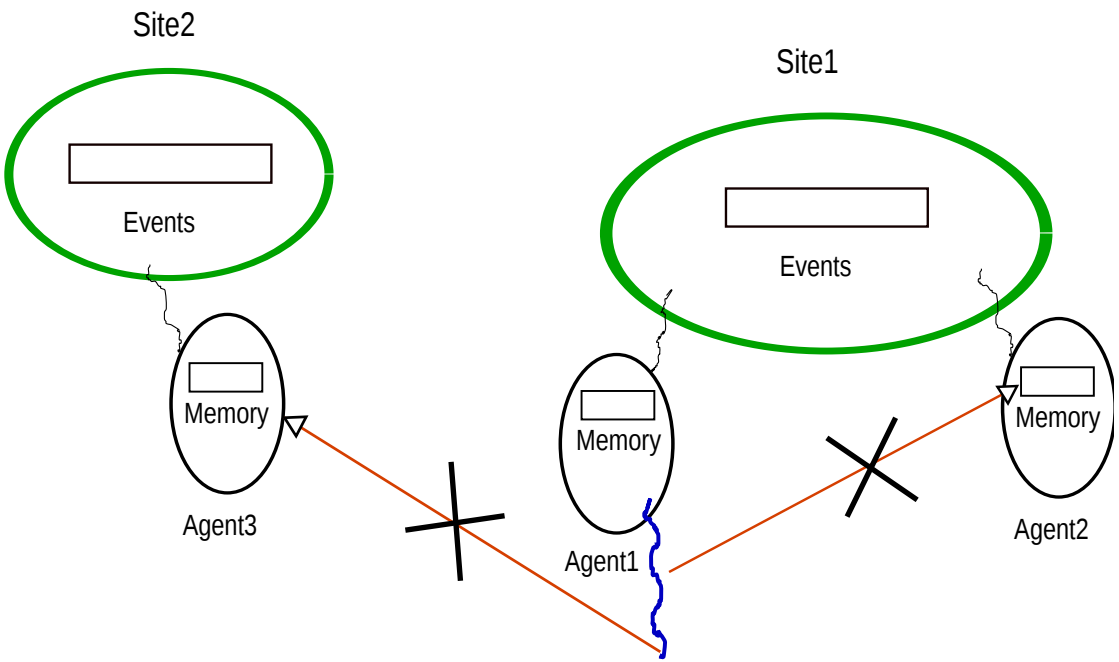


DSLM model (multiple sites)



Agent

$$\text{Agent} = \text{script } (s_1||s_2||...) + \text{private memory}$$



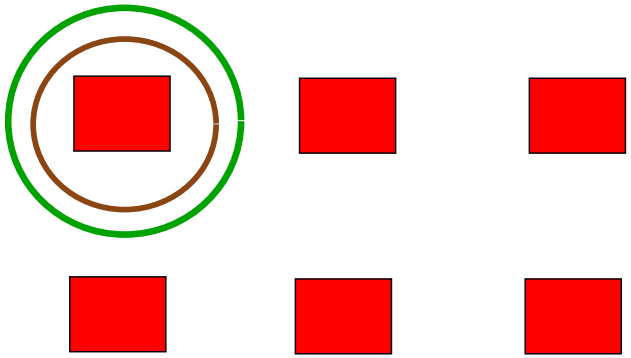
Requirement :

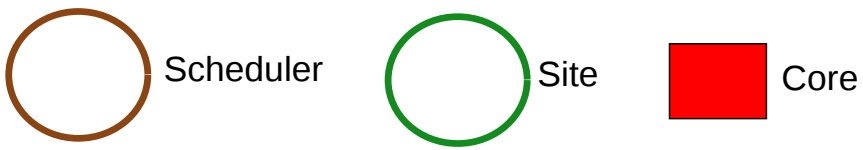
A script can only access the memory of the agent it belongs to.

2- Improve usage of cores

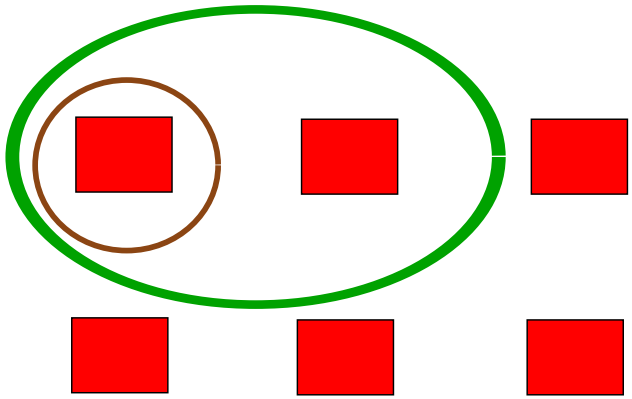
- Expansion and contraction of sites (Synchronized schedulers)
- Automatic balancing of resources among agent in the same site

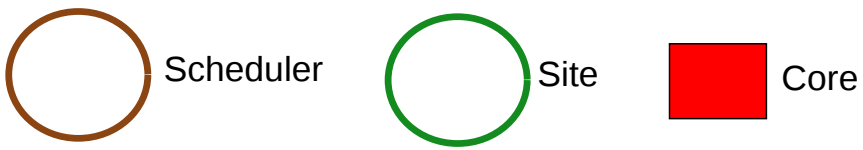
Site expansion and contraction



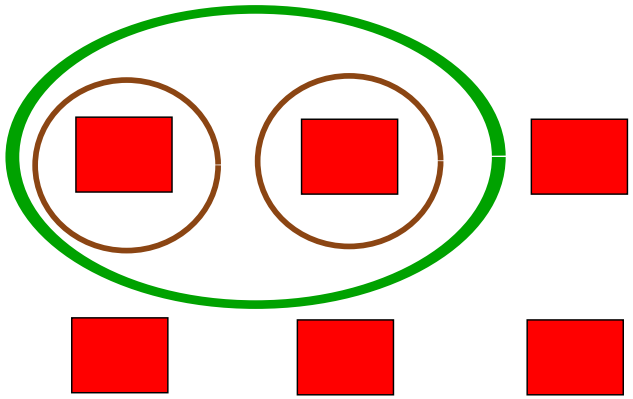


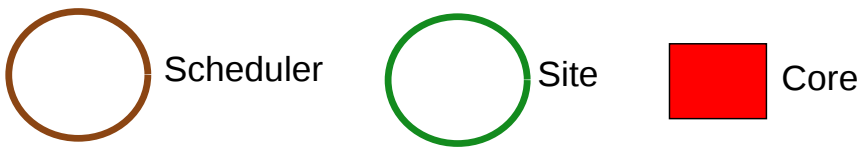
Site expansion and contraction



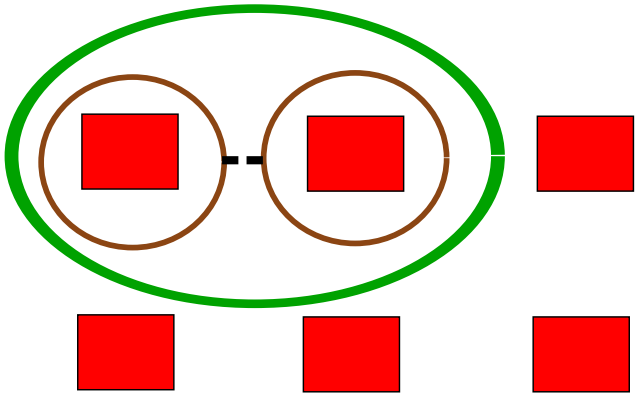


Site expansion and contraction

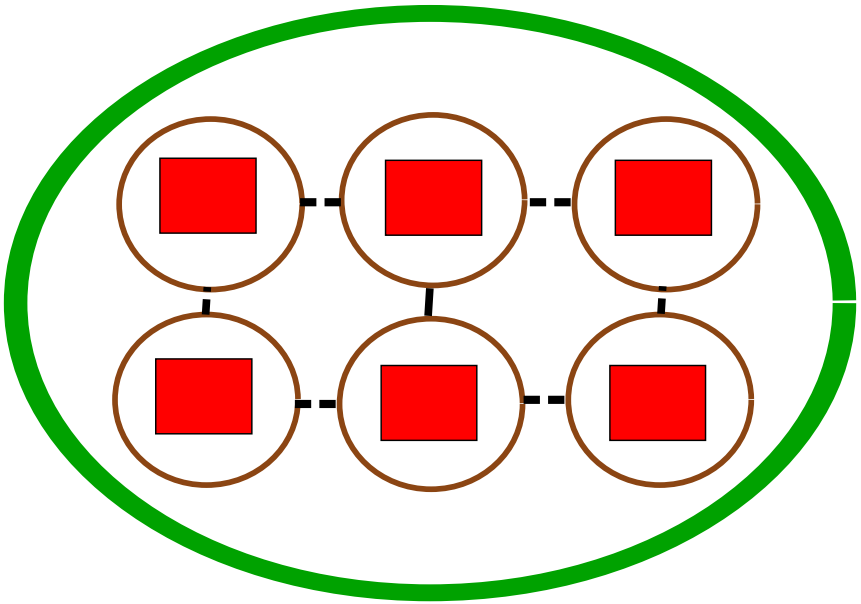


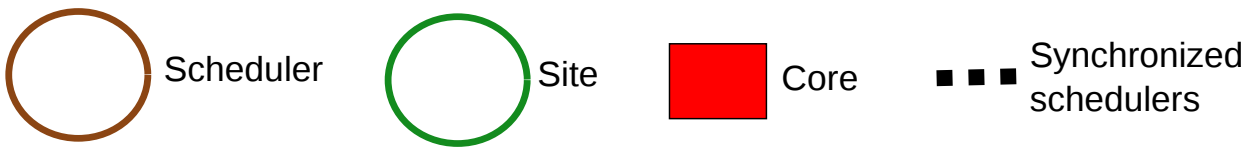


Site expansion and contraction

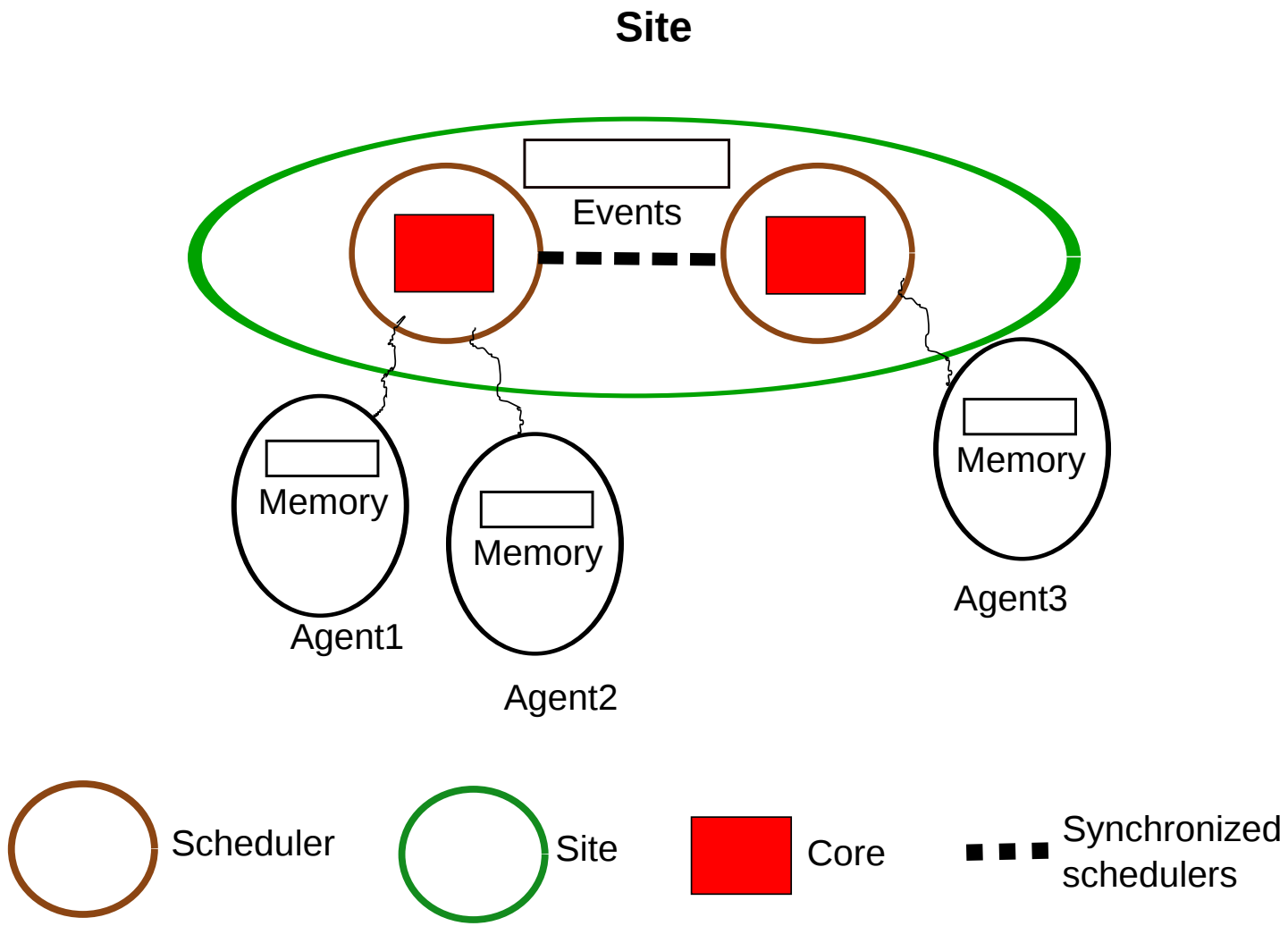


Site expansion and contraction

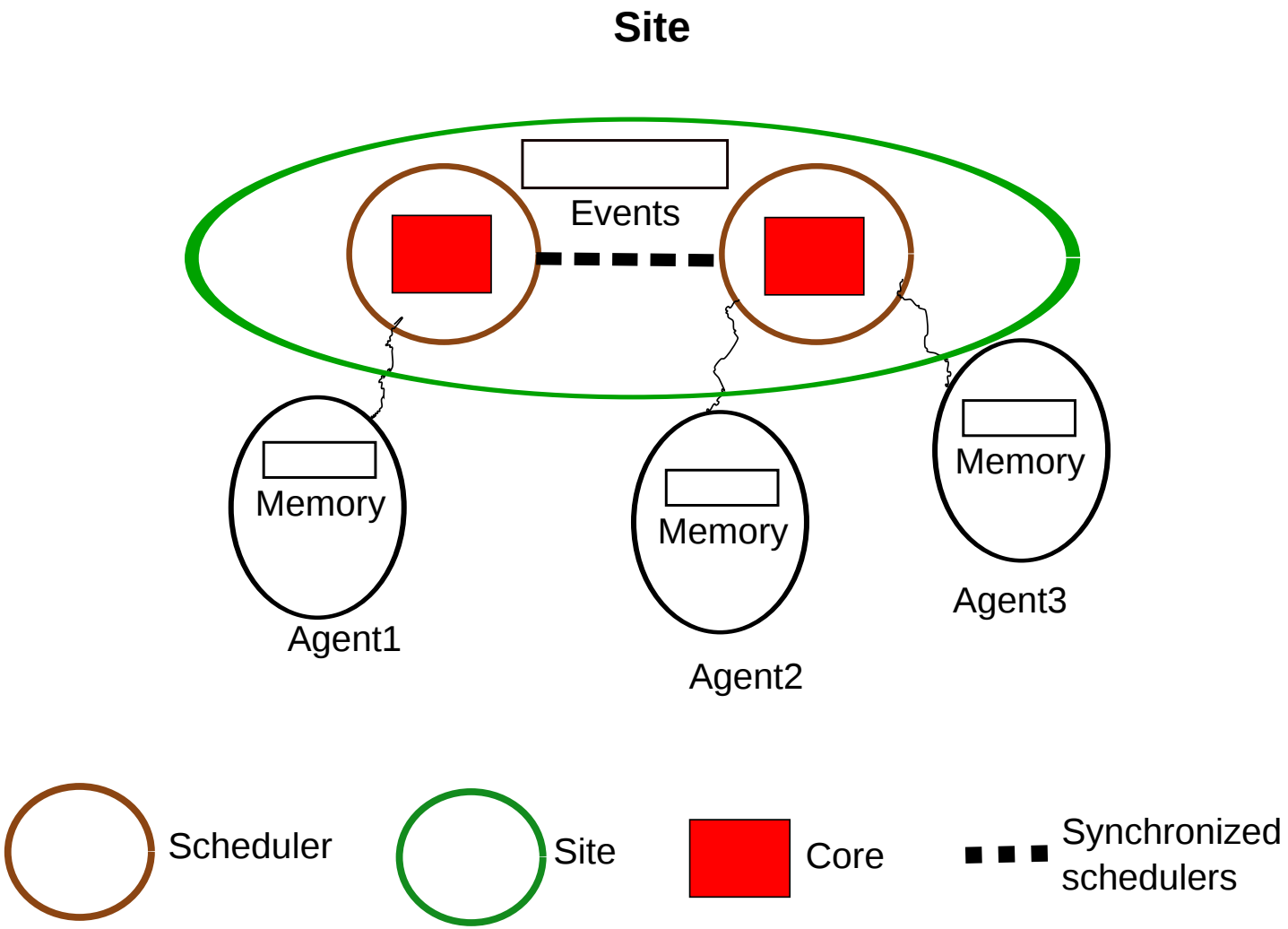




Automatic balancing



Automatic balancing



Conclusion and Future work

- Implementation of DSLM
- Introduction of security in DSL and DSLM
- Find a better name for DSLM ...

Questions?