Task Scheduling using Effects in Joelle

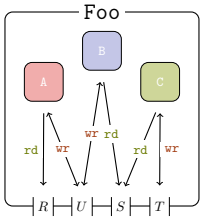
Race-Free Parallelism Through Active Objects, Ownership Types & Effects

– Stephan Brandauer, Johan Östlund, Tobias Wrigstad –

Overview

Joelle is an active object-based language relying on ownership types for isolation and effects for internal parallelism. *This project implements task scheduling for Joelle.*

http://tinyurl.com/jsched

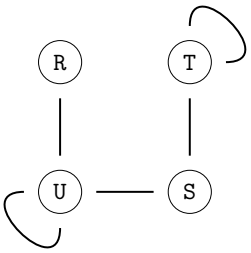


≡

```
class Foo {
region A, B, C;
Bar f in A;
...
def R() rd A {..}
def U() wr A, B {..}
def S() rd B, C {..}
def T() wr C {..}
```

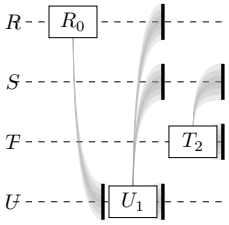
Joelle:

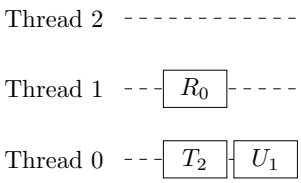
- Only asynchronous methods. Method calls as message sends, return future values where a synchronous method would return a value
- Active objects transparently run messages in parallel when safe: *"internal parallelism"*
- Most other active object implementations use only one internal thread of control. We use a work-stealing thread pool, but the illusion of one thread of control
- Objects in Joelle run in isolated memory regions which are further partitioned into *regions*
- Each field is placed in a region
- Methods are annotated with what fields they read or write
- `wr` implicitly contains `rd`



Internal Parallelism:

- Conflict graph (above) obtained from effect-annotations; *conflicting* methods are connected
- Two methods accessing a region conflict if at least one access is a write
- Conflicting tasks may not be executed at the same time





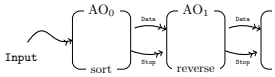
Implementation:

- Mailbox for active objects
- Each method has a separate queue
- When R is called, a corresponding task (R_0) is added to the R-queue in the receiving object's mailbox
- Furthermore, a *Barrier* (|) is added to all queues of all conflicting methods (`U`)

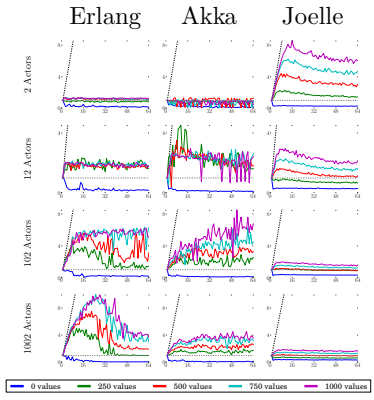
Scheduling Example:

- Tasks R_0, U_1, T_2 have been added to a mailbox in index-order
- R_0 and T_2 have no barriers in front, thus can be executed by the threadpool (above right) immediately
- Execution order of U_1 and T_2 is irrelevant for program behaviour and transparent to the program

Performance and Conclusions:



Chain (varying length N) of active objects, effectless methods. Data (varying number of values in list) is sorted and reversed



- Short chain lengths don't allow Erlang and Akka to use more than N cores at a time, thus limited speedup
- Small tasks (eg. blue lines) lead to poor scalability for all
- In our limited tests:
  - On a quad core PC: Joelle scales comparable to the competition with better absolute runtimes
  - On a 64 hardware-thread server: Joelle scales better for small Ns, worse for large Ns, with better or comparable absolute runtimes.

UPMARC
Uppsala Programming for Multicore Architectures Research Center

UPPSALA UNIVERSITET