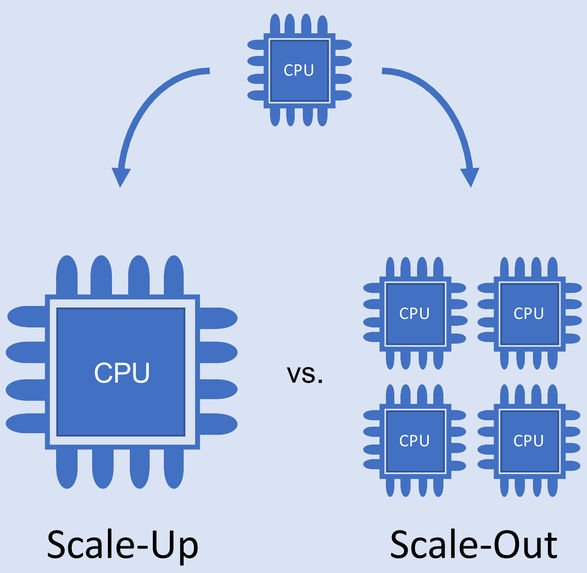
Composite data types in dynamic dataflow languages as copyless memory sharing mechanism

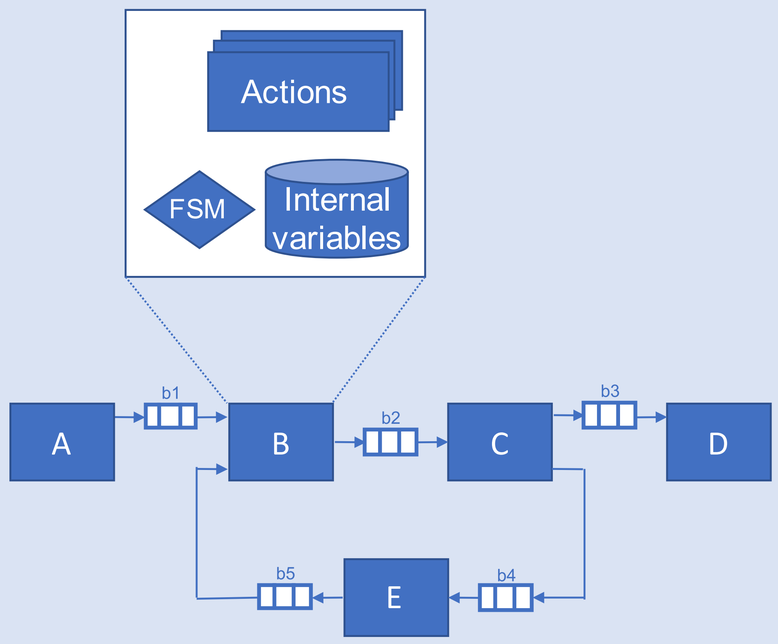
Aurelien Bloch[1], Endri Bezati[2], Marco Mattavelli[1]
[1] EPFL SCI-STI-MM, [2] EPFL VLSC
École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
{aurelien.bloch,endri.bezati,marco.mattavelli}@epfl.ch

Introduction



New challenges in software design
- Portability of applications
- Abstracting massive parallelism

Dataflow model of computation



Tool: ORCC

(a) An example with three actors (*Producer*, *Increment* and *Consumer*) and two buffers (b_1 and b_2).

```
actor Consumer () int In ==> :
 consume: action In:[a] repeat 128 ==> end
end
```

(b) Consumer.cal

```
actor Producer (int count=10) ==> int Out:
 int i := 0;

 prod : action ==> Out:[tbl] repeat 512
 guard i < count
 var List(type:int, size=512) tbl
 do
  tbl := [ j : for int j in 1 .. 512 ];
  i := i + 1;
 end
end
```

(c) Producer.cal

```
actor Inc () int In ==> int Out:

 exec : action
  In :[a] repeat 512 ==>
  Out:[a] repeat 512
 do
  foreach int i in 32 .. 63 do
   a[i] := a[i] + 1;
  end
 end
end
```

(d) Increment.cal

Fig. 1: RVC-CAL program example: dataflow network topology and actors source code.

Problem statement

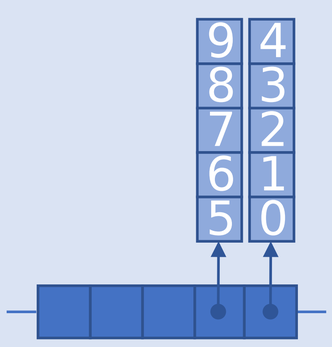
For shared memory architecture a lot of unnecessary copies are generated

Design proposition

Composite data types

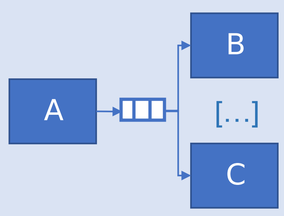
Introduce composite data types such as *list* to represent actions firing
- e.g. two firings of five integers

Using primitive type (Integer)

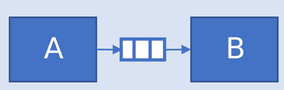


Using list

Buffer identification



Multiple fan-out



One stage communication

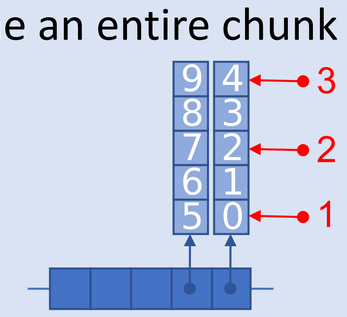
Chain of actors

Implementations

Fully dynamic solution
- Consume data at any rate

Semi dynamic solution
- Consume data of a size dividing an entire chunk

Static solution
- Always consume an entire chunk



Conclusions

- Tradeoff between memory copy and memory allocation
- Not beneficial for all applications

Future work

- Automatic selection of the appropriate implementation
- Integration in TURNUS framework