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

Aurelien Bloch¹, Endri Bezati², Marco Mattavelli¹

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

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

Problem statement
For shared memory architecture a lot of unnecessary copies are generated

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