The Dynamic Pipeline Paradigm

Cristina Zoltan[1] Edelmira Pasarella[1]
Julián Aráoz[1] Maria-Esther Vidal[2]

[1]Universitat Politècnica de Catalunya
{*zoltan,edelmira*}*@cs.upc.edu*
araoz@upc.edu

[2]TIB Leibniz Information Centre for Science Technology
L3S Research Center, Hannover, Germany
maria.vidal@tib.eu

September 3, 2019

Outline

Motivation

Era of Big Data and Internet of Things (IoT):

Motivation

Era of Big Data and Internet of Things (IoT):

- Large volumes of data in motion are produced

Motivation

Era of Big Data and Internet of Things (IoT):

- Large volumes of data in motion are produced
- Data is continuously produced by diverse devices

Motivation

Era of Big Data and Internet of Things (IoT):

- Large volumes of data in motion are produced
- Data is continuously produced by diverse devices
- Heterogeneous formats, frequencies, densities, and quantities

Motivation

Era of Big Data and Internet of Things (IoT):

- Large volumes of data in motion are produced
- Data is continuously produced by diverse devices
- Heterogeneous formats, frequencies, densities, and quantities
- Most of the data must be processed at **real-time**

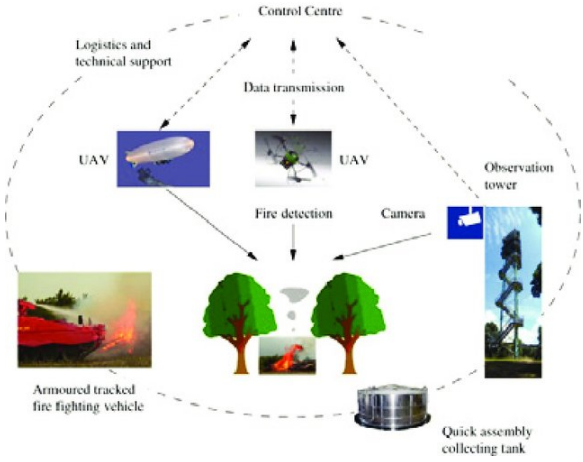
Motivation

Era of Big Data and Internet of Things (IoT):

- Large volumes of data in motion are produced
- Data is continuously produced by diverse devices
- Heterogeneous formats, frequencies, densities, and quantities
- Most of the data must be processed at **real-time**

Motivation

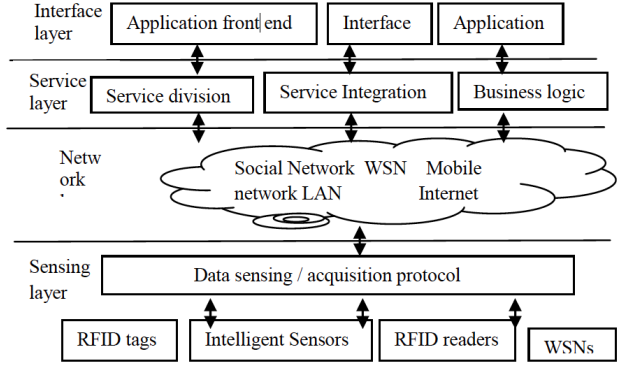
An example: Fire prevention/alarm



Schematic structure of the integrated forest fire detection and fire fighting system [Kruell, W. et al., 2012]

Motivation

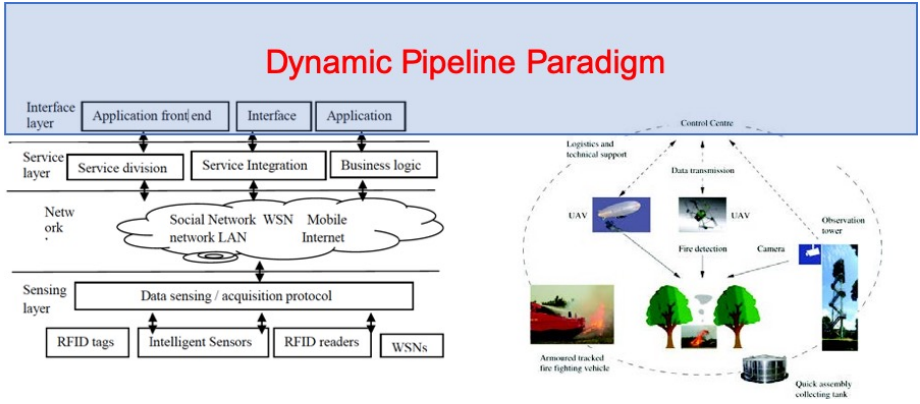
An example: Fire prevention/alarm



Service Oriented Architecture for fire IoT [Vijayalakshmi, SR. & Muruganand, S.,2017]

Motivation

An example: Fire prevention/alarm



Dynamic Pipeline Paradigm: An alternative for developing systems based on IoT

Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages

Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages
- Flexible and adaptable to the data received as input

Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages
- Flexible and adaptable to the data received as input
  - Programs stretch and shrink according to the spawning and duration of the stages of a program

Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages
- Flexible and adaptable to the data received as input
  - Programs stretch and shrink according to the spawning and duration of the stages of a program
  - Development of non-blocking solutions that enable the generation of results incrementally

Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages
- Flexible and adaptable to the data received as input
  - Programs stretch and shrink according to the spawning and duration of the stages of a program
  - Development of non-blocking solutions that enable the generation of results incrementally
- Parallelism
  - A given value has a single owner at each instant of time

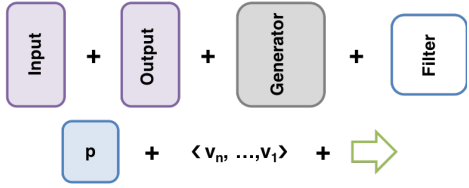
Dynamic Pipeline Computational Model

- Data-driven algorithms are specified as one-dimensional and unidirectional pipe of stages
- Flexible and adaptable to the data received as input
  - Programs stretch and shrink according to the spawning and duration of the stages of a program
  - Development of non-blocking solutions that enable the generation of results incrementally
- Parallelism
  - A given value has a single owner at each instant of time
  - Linear pipe $\Rightarrow$ neither forks nor joins

Dynamic Pipeline Computational Model

Dynamic Pipeline Framework

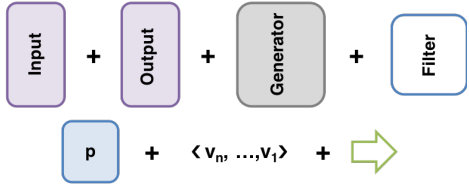
Components



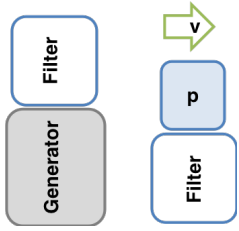
Dynamic Pipeline Computational Model

Dynamic Pipeline Framework

Components



Component composition



Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Stream Incoming data

Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Stream Incoming data
- Input stage receives a data streams in their original formats and transforms them according to the application requeriments. This stage is in charge of handling the heterogeneity of the input data

Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Input stage receives a data streams in their original formats and transforms them according to the application requeriments. This stage is in charge of handling the heterogeneity of the input data
- Output stage outputs the results in a format according to the application requirements

Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Parameter corresponds to the notion of a parameter in programming languages

Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Parameter corresponds to the notion of a parameter in programming languages
- Filter corresponds to a schema of behavior given as a parameterized script

Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Parameter corresponds to the notion of a parameter in programming languages
- Filter corresponds to a schema of behavior given as a parameterized script
- Generator This stage is parameterized by a filter and it is in charge of spawning new instances of filters stages

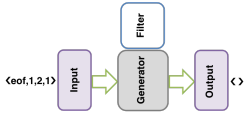
Dynamic Pipeline Computational Model

(Informal) Description of the Dynamic Pipeline Framework Components

- Stream Incoming data
- Input stage receives a data streams in their original formats and transforms them according to the application requeriments. This stage is in charge of handling the heterogeneity of the input data
- Output stage outputs the results in a format according to the application requirements
- Parameter corresponds to the notion of a parameter in programming languages
- Filter corresponds to a schema of behavior given as a parameterized script
- Generator This stage is parameterized by a filter and it is in charge of spawning new instances of filters stages
- Channel is the mechanism used to connect the different stages of a DP while transporting data through them

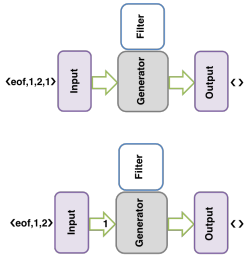
Dynamic Pipeline Computational Model

An example: Evolution of a Dynamic Pipeline



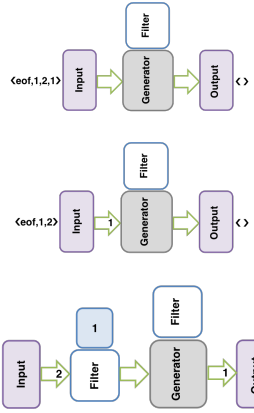
Dynamic Pipeline Computational Model

An example: Evolution of a Dynamic Pipeline



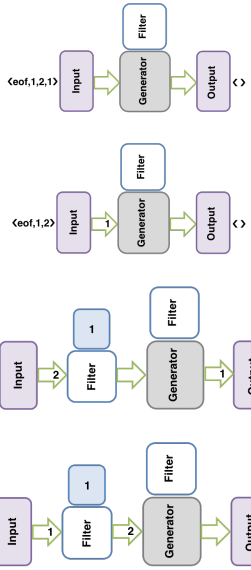
Dynamic Pipeline Computational Model

An example: Evolution of a Dynamic Pipeline



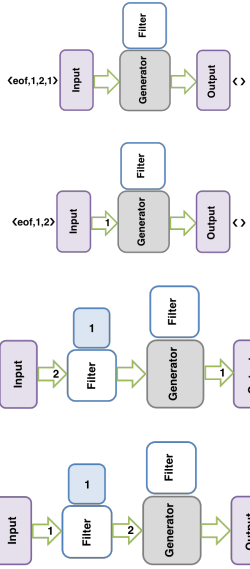
Dynamic Pipeline Computational Model

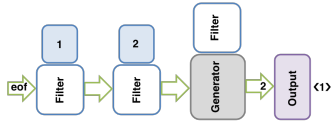
An example: Evolution of a Dynamic Pipeline



Dynamic Pipeline Computational Model

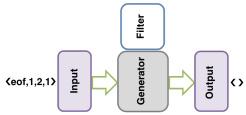
An example: Evolution of a Dynamic Pipeline

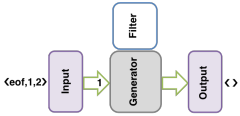


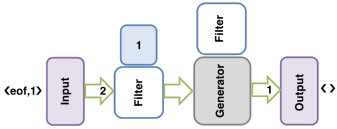


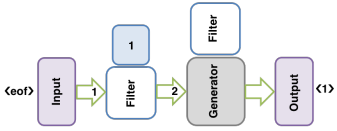
Dynamic Pipeline Computational Model

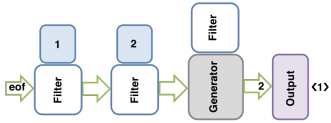
An example: Evolution of a Dynamic Pipeline

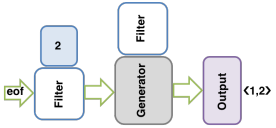






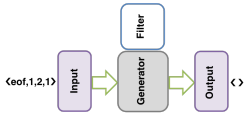


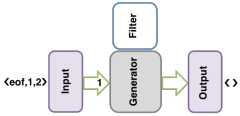


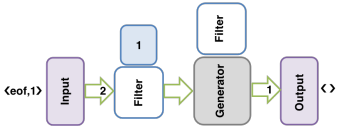


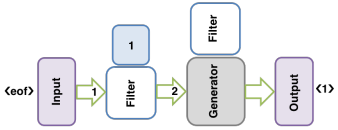
Dynamic Pipeline Computational Model

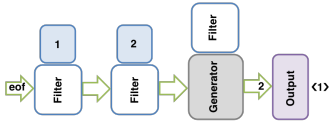
An example: Evolution of a Dynamic Pipeline

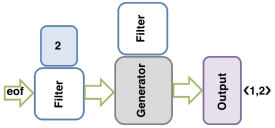


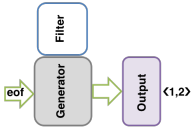






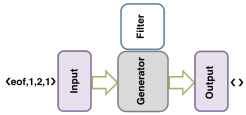


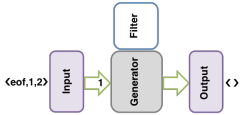


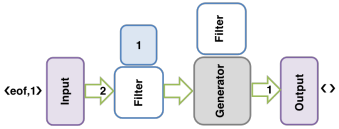


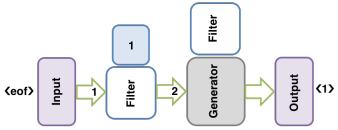
Dynamic Pipeline Computational Model

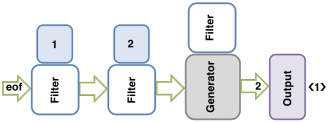
An example: Evolution of a Dynamic Pipeline

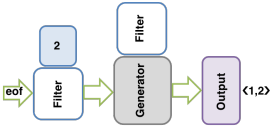


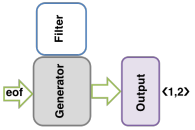


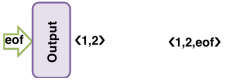








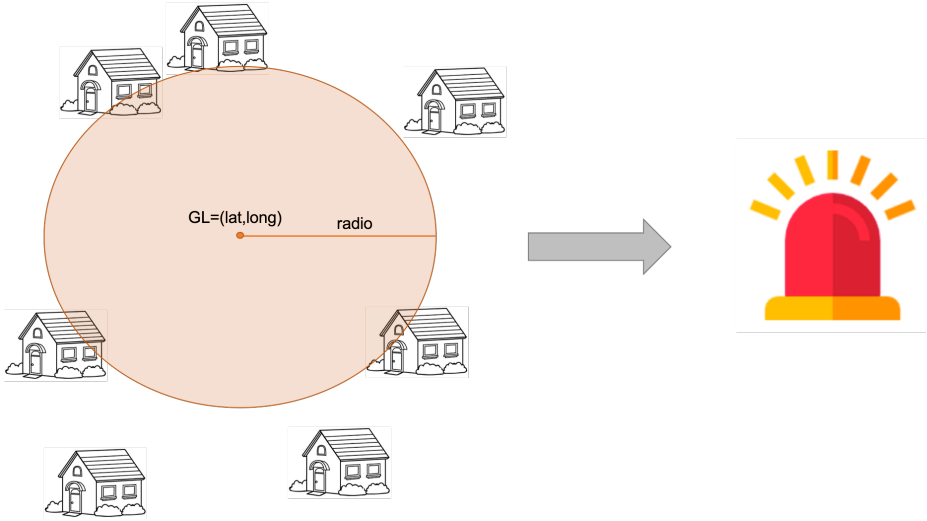




Specifying a DP-solution to a Problem

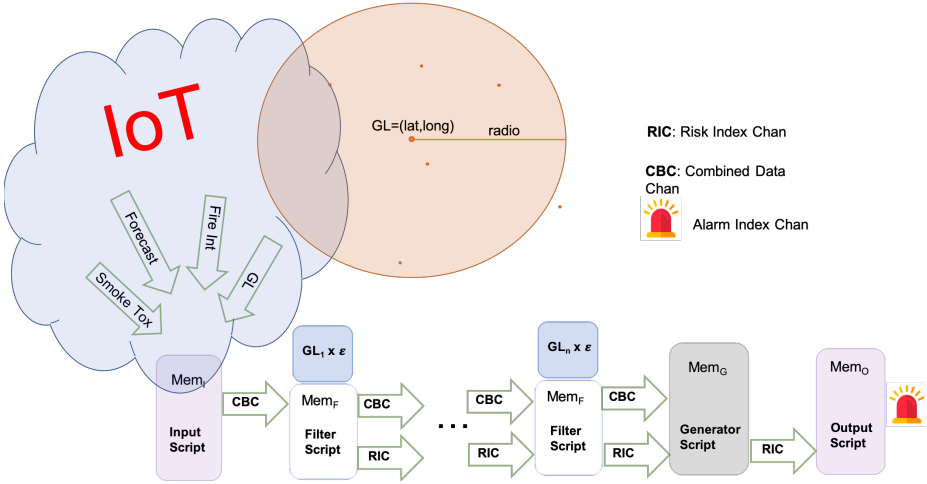
An example: A DP-solution for an Alarm System to Evacuate Homes by Fire

The Problem



Specifying a DP-solution to a Problem

A DP & IoT Solution to the Fire Alarm System



Specifying a DP-solution to a Problem

Dynamic Pipeline Template

$$DP_T = (Input_T, Filter_T, Generator_T, Output_T)$$

- $Input_T = (DC, script_{In}, M_{In}, IC)$
- $Output_T = (OC, script_{Out}, M_{Out}, RC)$
- DC, IC, OC and RC are sequences of (typed) channels.

Specifying a DP-solution to a Problem

Dynamic Pipeline Template

$$DP_T = (Input_T, Filter_T, Generator_T, Output_T)$$

- $Input_T = (DC, script_{In}, M_{In}, IC)$
- $Output_T = (OC, script_{Out}, M_{Out}, RC)$
- DC, IC, OC and RC are sequences of (typed) channels.
- $Generator = (script_{Gen}, M_{Gen}, Filter_T, IC, OC)$

Specifying a DP-solution to a Problem

Dynamic Pipeline Template

$$DP_T = (Input_T, Filter_T, Generator_T, Output_T)$$

- $Input_T = (DC, script_{In}, M_{In}, IC)$
- $Output_T = (OC, script_{Out}, M_{Out}, RC)$
- DC, IC, OC and RC are sequences of (typed) channels.
- $Generator = (script_{Gen}, M_{Gen}, Filter_T, IC, OC)$
- $Filter_T = (IC, p, \langle \mathrm{A}_1, \ldots, \mathrm{A}_m \rangle, M_{Filter})$

Specifying a DP-solution to a Problem

Dynamic Pipeline Template

$$DP_T = (Input_T, Filter_T, Generator_T, Output_T)$$

- $Input_T = (DC, script_{In}, M_{In}, IC)$
- $Output_T = (OC, script_{Out}, M_{Out}, RC)$
- DC, IC, OC and RC are sequences of (typed) channels.
- $Generator = (script_{Gen}, M_{Gen}, Filter_T, IC, OC)$
- $Filter_T = (IC, p, \langle \mathrm{A}_1, \ldots, \mathrm{A}_m \rangle, M_{Filter})$
  - p is the filter parameter, if any
  - $\langle \mathrm{A}_1, \ldots, \mathrm{A}_m \rangle$ is a stack of scripts
  - M_{Filter} stands for the memory of F

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Input_{cc} = (DC, script_{In}, M_{In}, IC)$
  - $M_{In} = [e : Edge]$
  - $DC = \langle C_{data} \rangle$
  - $IC = \langle C_{edges}, C_{set\_of\_nodes} \rangle$
  - $script_{In}$: To output in C_{edges} the result of applying the identity transformation to the edge in C_{data}. When receiving eof on C_{data} to output *eof* on both channels C_{edges} and $C_{set\_of\_nodes}$, and to die.

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Input_{cc} = (DC, script_{In}, M_{In}, IC)$
  - $M_{In} = [e : Edge]$
  - $DC = \langle C_{data} \rangle$
  - $IC = \langle C_{edges}, C_{set\_of\_nodes} \rangle$
  - $script_{In}$: To output in C_{edges} the result of applying the identity transformation to the edge in C_{data}. When receiving eof on C_{data} to output *eof* on both channels C_{edges} and $C_{set\_of\_nodes}$, and to die.
- $Output_T = (OC, script_{Out}, M_{Out}, RC)$
  - $M_{Out} = [s : Set\_of\_Nodes]$
  - $OC = \langle C_{set\_of\_nodes} \rangle$
  - $RC = \langle C_{connected\_components} \rangle$
  - $script_{Out}$: To output in $C_{connected\_components}$ the result of applying the identity transformation to the set of nodes in $C_{set\_of\_nodes}$. To die when receiving an *eof* on the $C_{set\_of\_nodes}$

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Generator_{cc} = (script_{Gen}, Filter_T, IC, OC)$

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Generator_{cc} = (script_{Gen}, Filter_T, IC, OC)$
  - $script_{Gen}$: If an edge e arrives on C_{edges}, to spawn a new instance of the filter. To die when an *eof* arrives on C_{edges}. No special parameter is required by the filters but M_{Filter} must be initialized with the set of nodes in the edge e.

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Generator_{cc} = (script_{Gen}, Filter_T, IC, OC)$
  - $script_{Gen}$: If an edge e arrives on C_{edges}, to spawn a new instance of the filter. To die when an *eof* arrives on C_{edges}. No special parameter is required by the filters but M_{Filter} must be initialized with the set of nodes in the edge e.
- $Filter_{cc} = (IC, \emptyset, \langle A_1, A_2 \rangle, M_{Filter})$

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Generator_{cc} = (script_{Gen}, Filter_T, IC, OC)$
  - $script_{Gen}$: If an edge e arrives on C_{edges}, to spawn a new instance of the filter. To die when an *eof* arrives on C_{edges}. No special parameter is required by the filters but M_{Filter} must be initialized with the set of nodes in the edge e.

- $Filter_{cc} = (IC, \emptyset, \langle A_1, A_2 \rangle, M_{Filter})$
  - Find connected components $script_{A_1}$
    If the edge in C_{edges} is incident to any node present in the memory, add the other node (if not already) to the memory M_{Filter}. Otherwise, passes this edge to the neighbor through C_{edges}. When receiving an *eof* on C_{edges}, to pass this mark to the neighbor through C_{edges} and to die. Now the second actor A_2 is on the top of the stack of actors

Specifying a DP-solution to a Problem

Dynamic Pipeline Template: Connected Components of an Undirected Graph

$$DP_{cc} = (Input_{cc}, Filter_{cc}, Generator_{cc}, Output_{cc})$$

- $Generator_{cc} = (script_{Gen}, Filter_T, IC, OC)$
  - $script_{Gen}$: If an edge e arrives on C_{edges}, to spawn a new instance of the filter. To die when an *eof* arrives on C_{edges}. No special parameter is required by the filters but M_{Filter} must be initialized with the set of nodes in the edge e.

- $Filter_{cc} = (IC, \emptyset, \langle A_1, A_2 \rangle, M_{Filter})$
  - Find connected components $script_{A_1}$
    If the edge in C_{edges} is incident to any node present in the memory, add the other node (if not already) to the memory M_{Filter}. Otherwise, passes this edge to the neighbor through C_{edges}. When receiving an *eof* on C_{edges}, to pass this mark to the neighbor through C_{edges} and to die. Now the second actor A_2 is on the top of the stack of actors
  - Enlarge connected components $script_{A_2}$
    If the set of nodes in $C_{set\_of\_nodes}$ intersects the set in M_{Filter}, to update the memory M_{Filter} with the union of these two sets of nodes and no output is produced. Otherwise, to pass this set of nodes to the neighbor. When receiving the *eof*, to output this mark in $C_{set\_of\_nodes}$ and to die. Hence the stack of scripts becomes empty and the filter dies.

Conclusions and Future Work

- Dynamic pipeline paradigm
  - Flexible non-blocking parallel computational model
  - Suitable to real time stream processing
  - Maximize and not to fix in advance the number of processor is key to take advantages of parallelism

Conclusions and Future Work

- Dynamic pipeline paradigm
  - Flexible non-blocking parallel computational model
  - Suitable to real time stream processing
  - Maximize and not to fix in advance the number of processor is key to take advantages of parallelism
- Future Work
  - To implement the Dynamic Pipeline Framewok
    Go language: channels + goroutines mechanisms
  - To solve some well known and challenging classes of problems using the DP-paradigm and benchmarking these solutions
    Graphs problems, classification (machine learning) algorithms, etc.
  - To give a formal definition of the Dynamic Pipeline Computational Model
    CCS, Maude, Promela ? ...
  - Fault tolerance issues ...

Thank you

References

Krüll, Wolfgang and Tobera, Robert and Willms, Ingolf and Essen, Helmut and von Wahl, Nora.
Early forest fire detection and verification using optical smoke, gas and microwave sensors.
Procedia Engineering, 45, 584–594 2012.
Elsevier

Vijayalakshmi, SR. and Muruganand, S.
A survey of Internet of Things in fire detection and fire industries.
Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC), 703–707 2017.

IEEE