STKM on SCA: A Unified Framework with Components, Workflows and Algorithmic Skeletons

Marco ALDINUCCI
University of Torino (Italy)

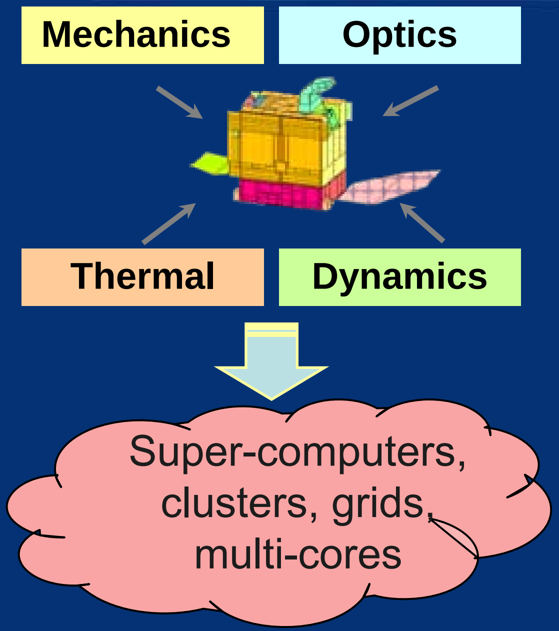
Marco DANELUTTO
UNIPI - University of Pisa (Italy)

Hinde Lilia BOUZIANE and Christian PEREZ
Projet-team GRAAL - INRIA Rhône-Alpes - ENS Lyon (France)

Outline of the talk

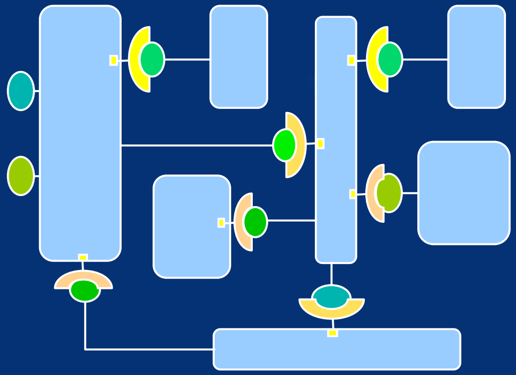
- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Context and problematic



Programming simply and independently from execution resources

Component models

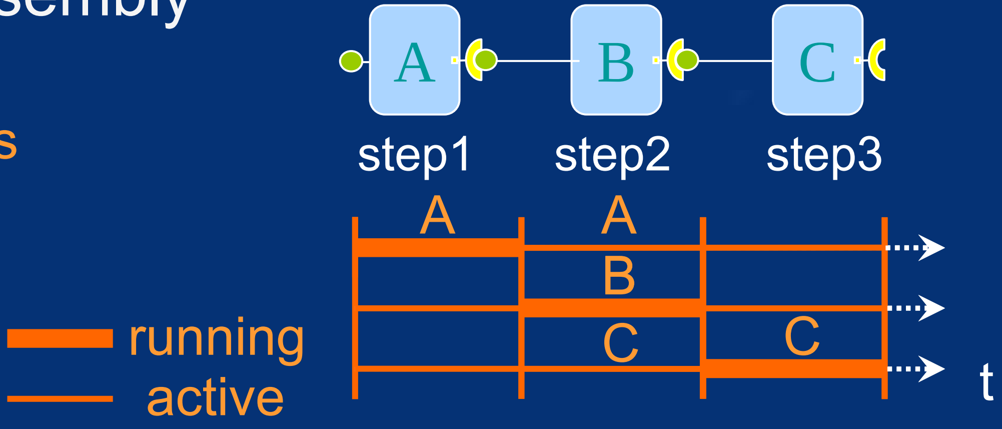


- Context: complex applications (size, heterogeneity)
- Promising approach: component models
  - Examples: CCM (OMG), GCM (NoE CoreGrid), SCA (OSOA), CCA (CCA Forum-USA), etc.

Limitations of existing component models

- Assembly models close to the computing resources
  - Behavior hidden in the assembly

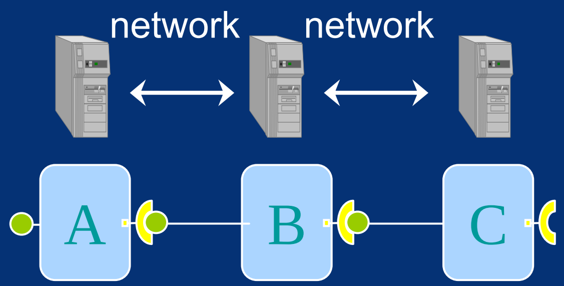
- "Over-consumption" of resources



➾ **Workflow models**

  - Simple spatial relations

- Resource dependencies
- Complex design
  - Parallel paradigms (e.g. master-worker)



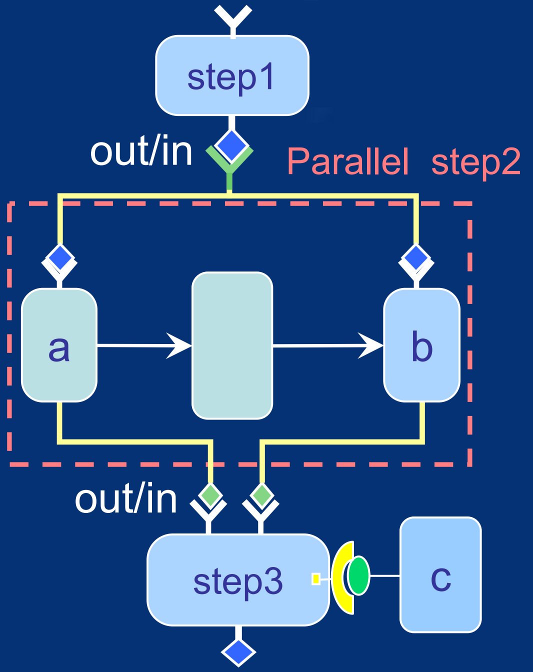
➾ **Algorithmic skeleton models**

Outline of the talk

- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Overview of STCM [EuroPar'08]

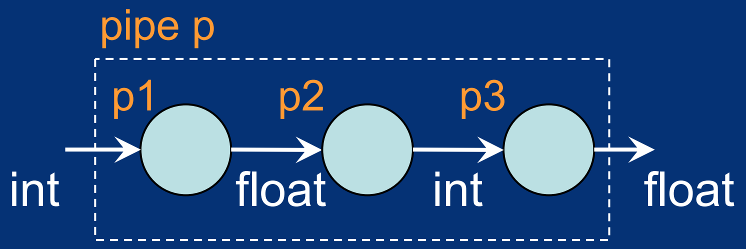
- Combination of component and workflow models
  - Spatial and temporal dimensions at the same level of assemblies
- Component-task
  - Spatial ports (classical ones)
  - Input and output ports (temporal)
  - Task
- Assembly model
  - Adaptation of a workflow language

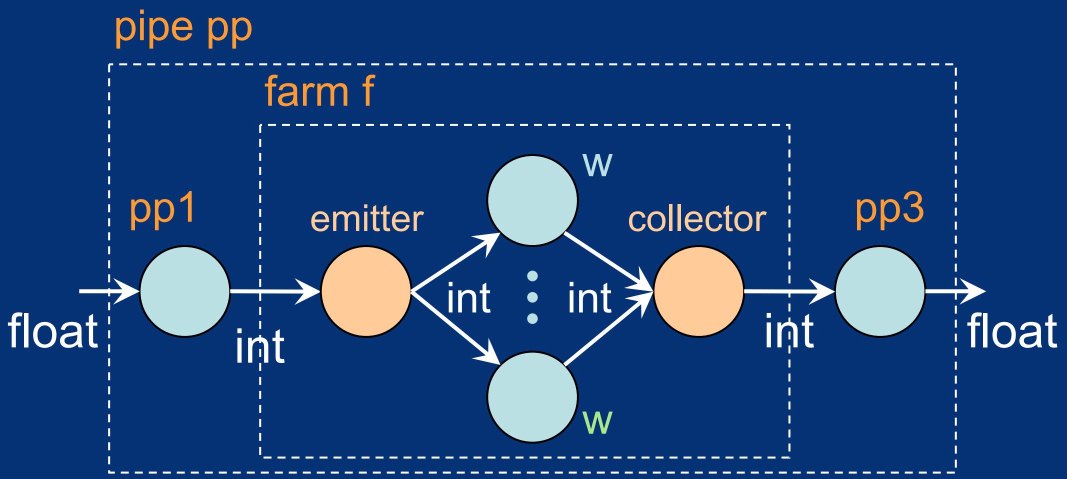


Algorithmic skeletons [M. Cole 1989]

- Structured programming (simplicity/correctness of programs)
- Hide the complexity of parallelism setup and data distribution
- Behavioral skeletons (advanced management for adaptation)

```
pipe p in (int a) out (float b)
      p1 in (a) out (float b1)
      p2 in (b1) out (int b2)
      p3 in (b2) out (b)
end pipe

farm f in (int af) out(int bf)
       w in (af) out(bf)
end farm
pipe pp in (float a) out(float b)
       pp1 in (b) out (int b1)
       f      in (b1) out (b2)
       pp3 in (b2) out (b)
end pipe
```





Outline of the talk

- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Objectives

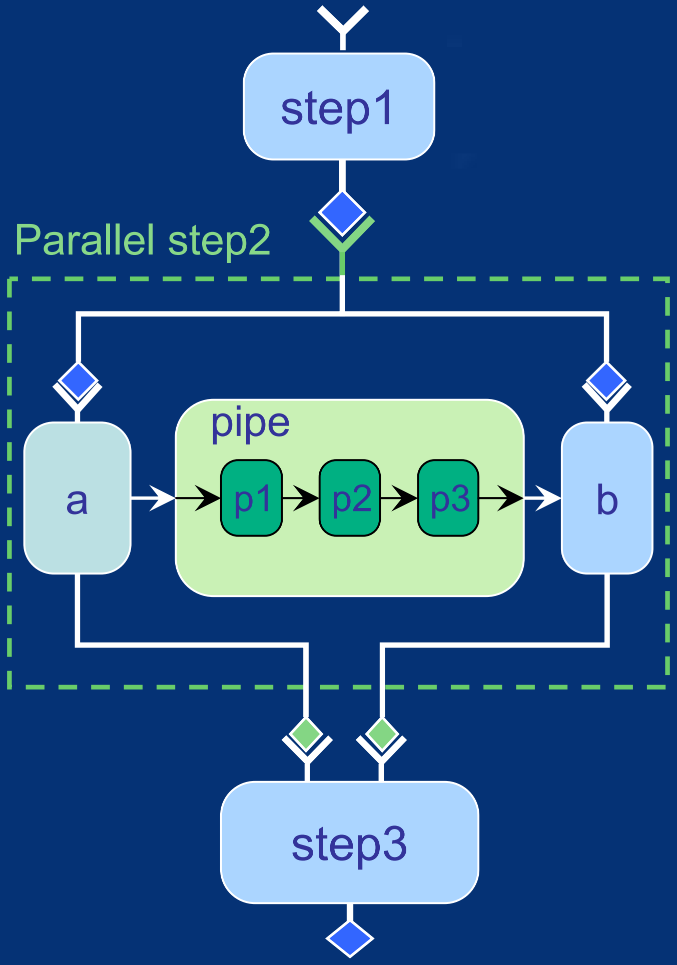
- Bringing together suited properties
  - Code reuse facility (Component models)
  - Capability of resources usage optimization (Workflows)
  - Simplicity of programming parallel parts of an application (Skeletons)

- Portability on different execution resources
  - Code reuse
  - Efficiency

Overview of STKM [CBHPC'08]

Assembly model (1/2)

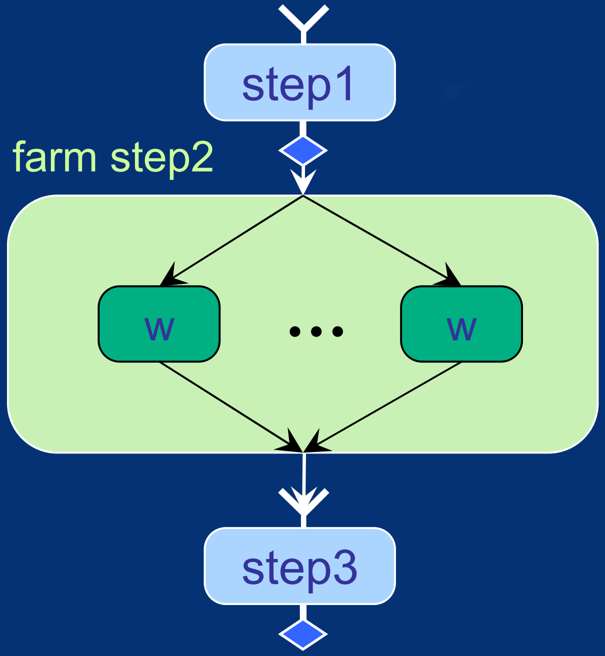
- Extension of STCM assembly language
  - skeleton constructs
- An STKM skeleton construct is a composite with a predefined behavior
- Parameters
  - Skeleton's input/output data
  - User's components
- Usage in spatial and temporal dimension



Overview of STKM [CBHPC'08]

Assembly model (2/2)

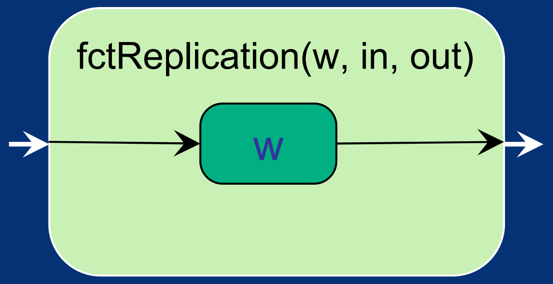
```
component Example{
   … Step1 and Step3 components…

   farm Step2{
      inputSkel double inS2;
      outputSkel string outS2;

      worker sequential w {
         inputSkel double inW;
         outputSkel string outW;
         component Worker{   streamIn double inW;
                             streamOut string outW;
         };
         connect outW to Worker.outW;
         connect Worker.inW to inW;
   };

instances:  Step1 step1; Step2 step2; Step3 step3;
 … Connexions step1 <=> step2 <=> step3 …
sequence ApplMain{
   exectask(step1); exectask(step2); exectask(step3);
};
```

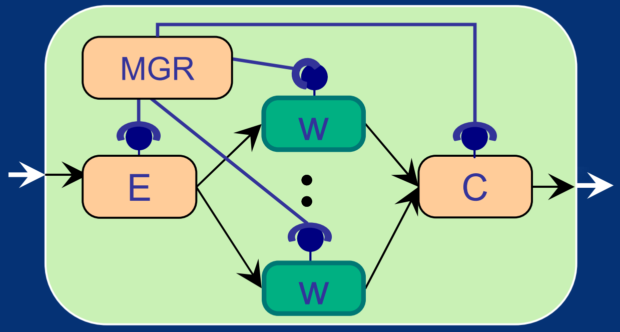


Assembly transformation (1/2)

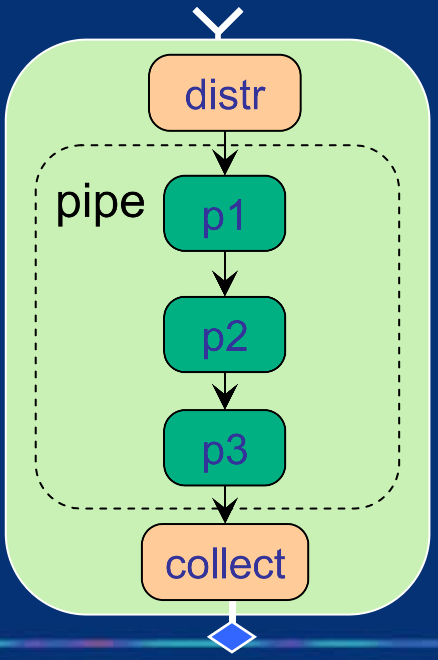
User view

At the responsibility of the framework





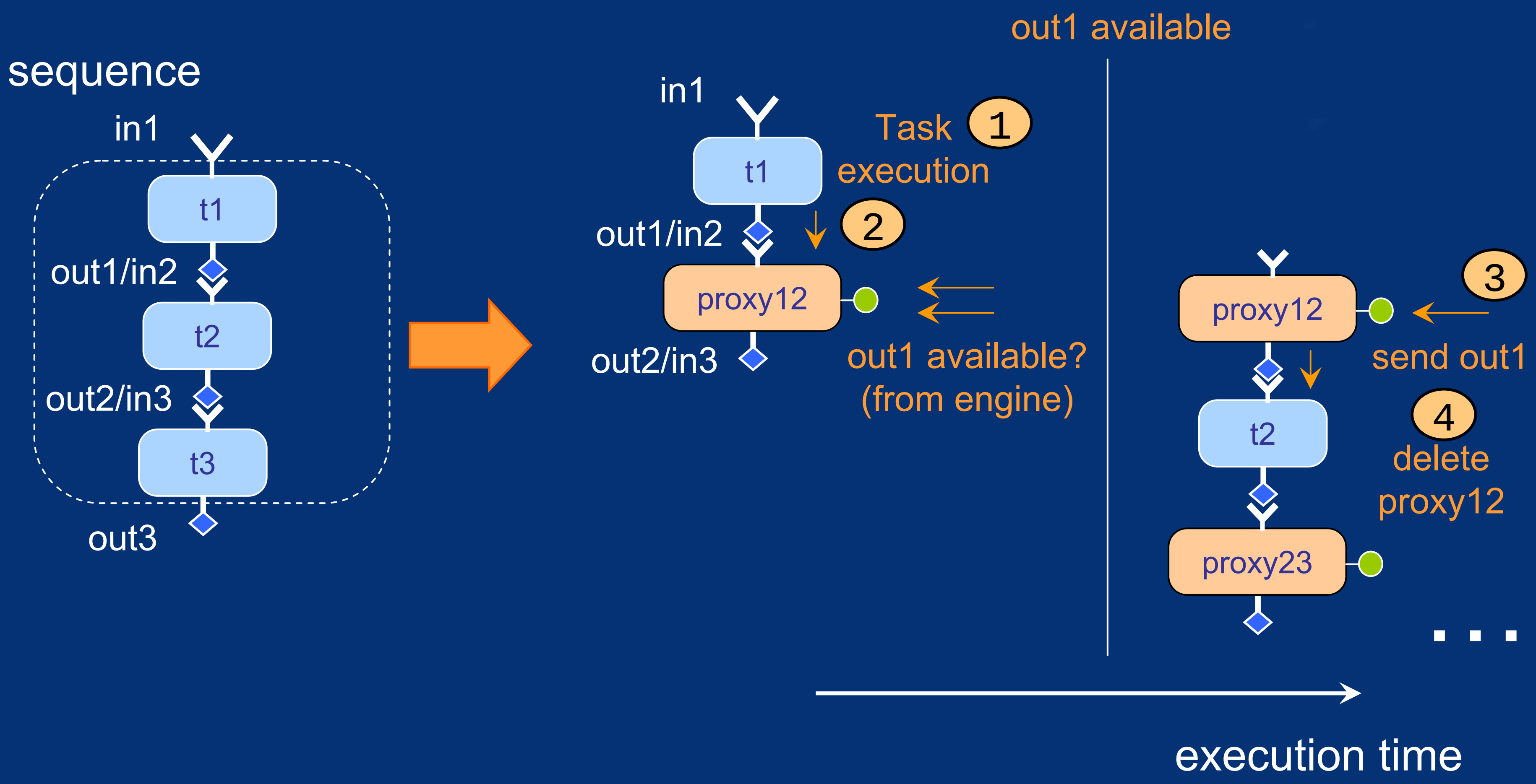
```
component ForExample{
 … T1 and T2 and T3 components…

instances:  T1 t1; T2 t2; T3 t3;
 … Connexions t1 <=> t2 <=> t3 …
for (i=1 to 10) {
   exectask(t1);
   exectask(t2);
   exectask(t3);
};
```



Assembly transformation (2/2)

User view

At the responsibility of the framework



sequence

in1

t1

out1/in2

t2

out2/in3

t3

out3

in1

t1

Task execution 1

out1/in2

2

proxy12

out2/in3

out1 available? (from engine)

out1 available

proxy12

3

send out1

t2

4

delete proxy12

proxy23

. . .

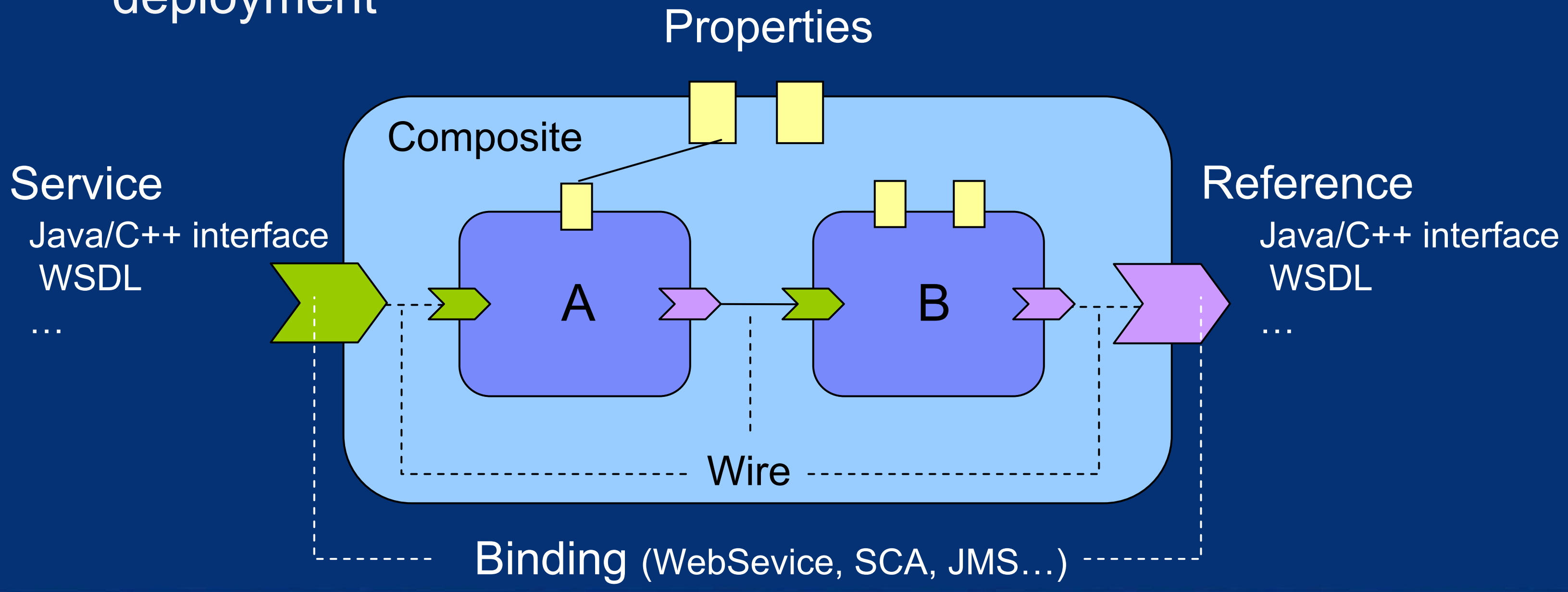
execution time

Outline of the talk

- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Service Component Architecture

- OSOA collaboration 2007
- A component model for services composition
  - Independent from any technology
- Models specifications for
  - Assembly, client, component implementation, packaging and deployment



Properties

Composite

Service

Java/C++ interface

WSDL

…

A

B

Reference

Java/C++ interface

WSDL

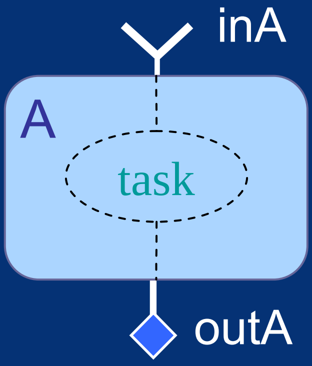
…

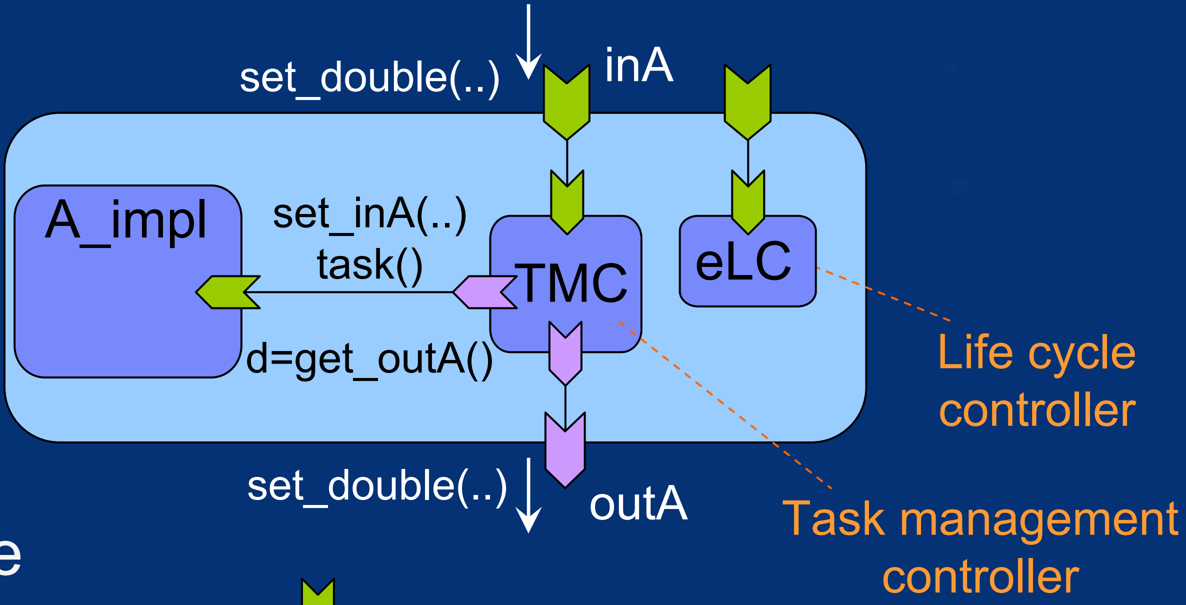
Wire

Binding (WebSevice, SCA, JMS…)

Mapping STKM concepts on SCA

- A component

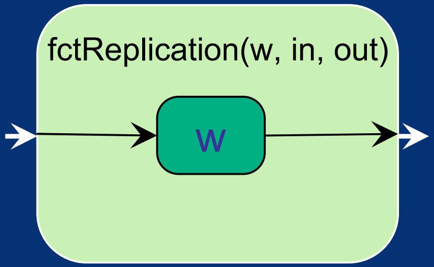


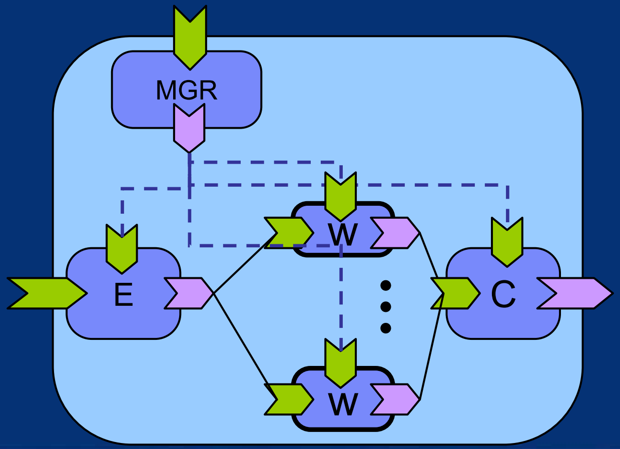


Life cycle controller

Task management controller

- A skeleton example





Without control part

Outline of the talk

- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Evaluation

- SCA implementation
  - Tuscany Java SCA version 1.2.1
  - Adaptation for dynamicity requirements
- Components’ implementation
  - Java 1.5
- Benchmark
  - Sequence, loop, pipeline and nested composition of pipeline and functional replication behavioral skeleton
  - Parametric
    - Different types and sizes of tasks (time, data)
- Resources
  - Cluster of 24 Intel Pentium 3, 800MHz, 1GB RAM, 100MBit/s Ethernet
- Static decisions
  - Transformation and components’ placement

Metrics

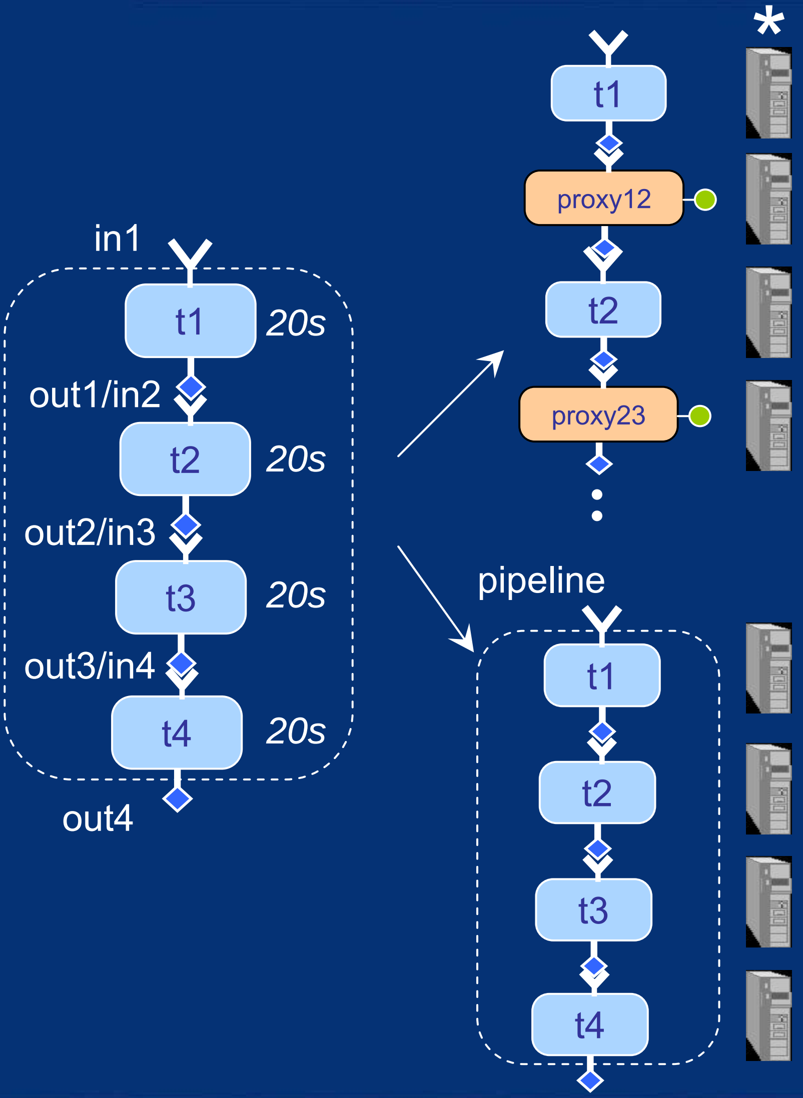
- (Dynamic) deployment overheads

	Time in *s*
Remote node launching (with ssh + common daemon library)	45.56
Programmed port connection (ad_hoc API)	3.20

- Round Trip Time (*ms*) depending on components' placement

	Intra-node Inter-component	Inter-node Intra-host	Inter-node Inter-host
Default SCA protocol	0.076	20.35	20.17
Web Service protocol	22.66	24.23	24.11

Adaptation capability: a sequence use case



in1

t1 20s

out1/in2

t2 20s

out2/in3

t3 20s

out3/in4

t4 20s

out4

*

t1

proxy12

t2

proxy23

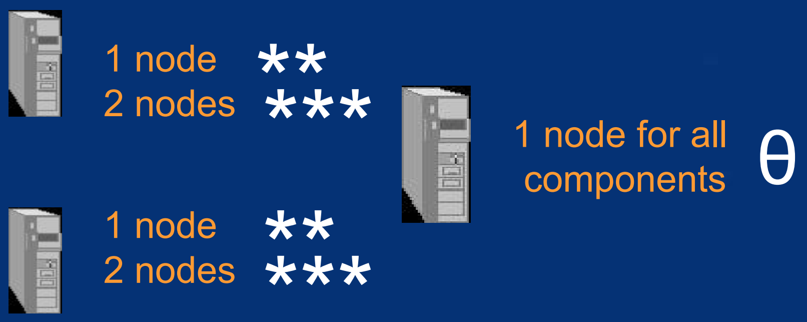
pipeline

t1

t2

t3

t4



1 node **
2 nodes ***

1 node **
2 nodes ***

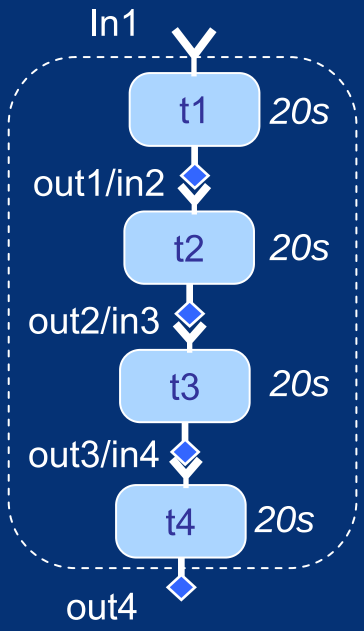
1 node for all components θ

	Efficienty (%)
*	27.03
**	20.13
***	28.38
θ	28.50
Pipeline	51.14

Form recognition

- A *for* loop
  - Input/output: arrays of data (doubles) with a same size
  - Body = a sequence with stateless components

for (i=1; i < inLoop.size, i++)



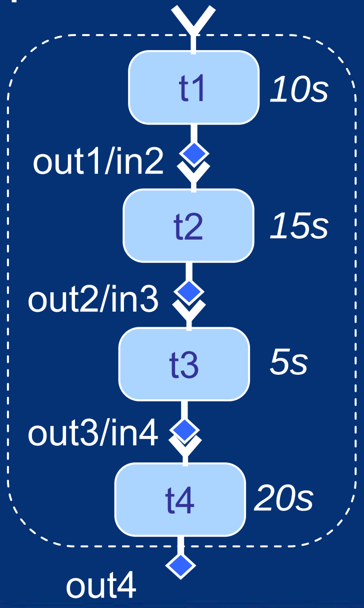
	Efficienty (%)
Array size: 10	
Body as sequence	28.40
t1 to t3 created once	56.16
Pipelined body	76.96
Array size: 100	
Body as sequence	28.40
t1 to t3 created once	90.68
Pipelined body	95.90

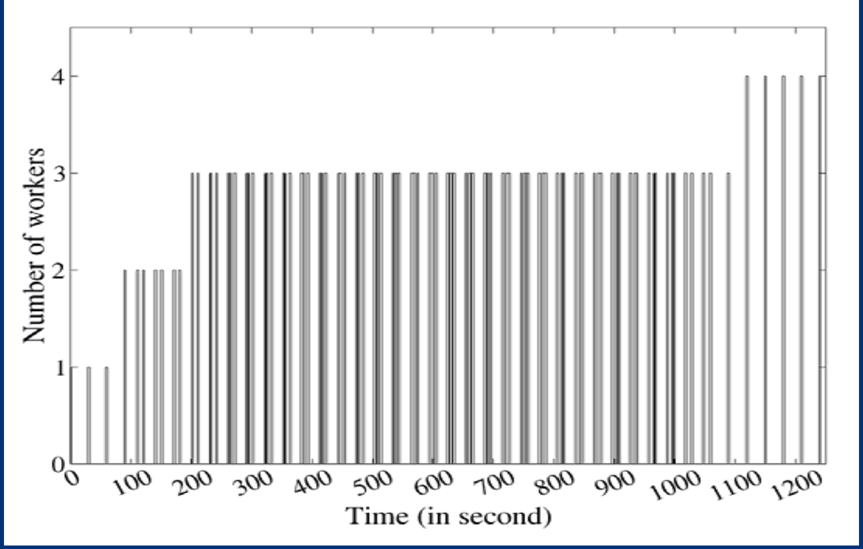
Behavioral skeleton management

- Pipeline skeleton
- Criteria at transformation
  - Performance
  - Resource usage

	Exec. time in *s*
Pipeline	3105
Farm: 3 workers for t3	1182
Functional replication with dynamicity for t3	1403

pipeline





Outline of the talk

- Introduction
  - Software component models
- Existing models
  - STCM: a spatio-temporal component model
  - Skeletons based models for parallel programming
- STKM: a proposal of skeletons introduction in STCM
- An SCA based implementation of STKM
  - Overview of SCA
  - A projection of STKM on SCA
- Experiments
- Conclusions and future works

Conclusions

- A combination of component models, workflows and skeletons
- Previous works
  - STCM: merging component models with workflows
  - Skeleton models
  - STKM proposal (theoretical study)
- Contributions
  - STKM prototype on SCA
  - Performance evaluation
- Simplicity of design and adaptation capabilities

Perspectives

- Generic skeletons constructs for easy extension with new skeletons
  - Ongoing work (PhD in the GRAAL project-team)
- Framework implementation for automatic generation of assemblies at execution
  - Model driving engineering
  - Choice and decisions techniques
- Applications implementation
  - Using more recent and advanced frameworks

Questions ?