RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

cea cnrs Inria

FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

COMET : From Dynamic Data-Parallel Dataflows to Task Graphs

Jerry LACMOU ZEUTOUO
SDMA Team, MIS Laboratory, UPJV
Christian PEREZ
INRIA, Avalon, LIP

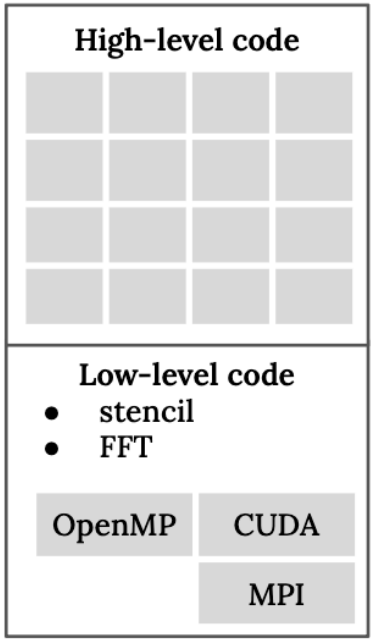
Sommaire

1. **Introduction to COMET**
   1. Managing code variability in HPC applications
   2. Component-based programming model
   3. COMET composition units
   4. COMET assembly
   5. Implementing the COMET model
2. **Handling Dynamic Data-Parallel Dataflows with COMET**
   1. CIGRA : COMET Instruction GRAph
   2. Reverse topological sorting to a CIGRA
   3. Experimental results according to operation between metatasks
   4. Kinds of dynamic data connection with COMET
   5. Code generation strategies
   6. Experimental results according to code generation strategies
3. **Conclusion**

1. Introduction to COMET

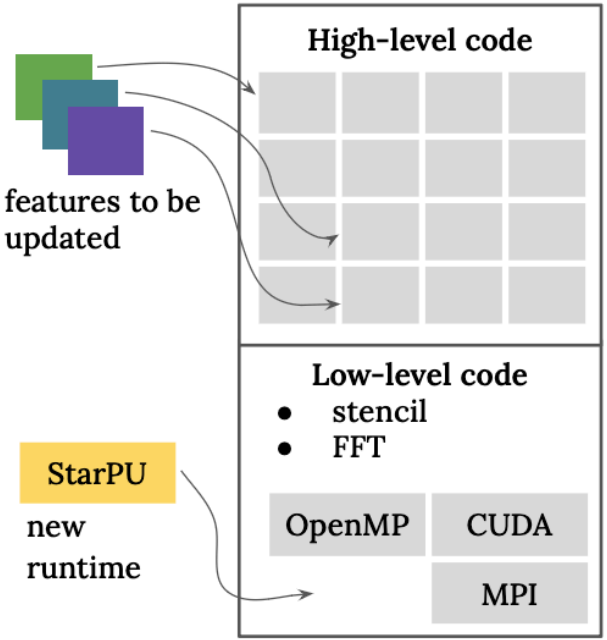
1. Introduction to COMET

Managing code variability in HPC applications



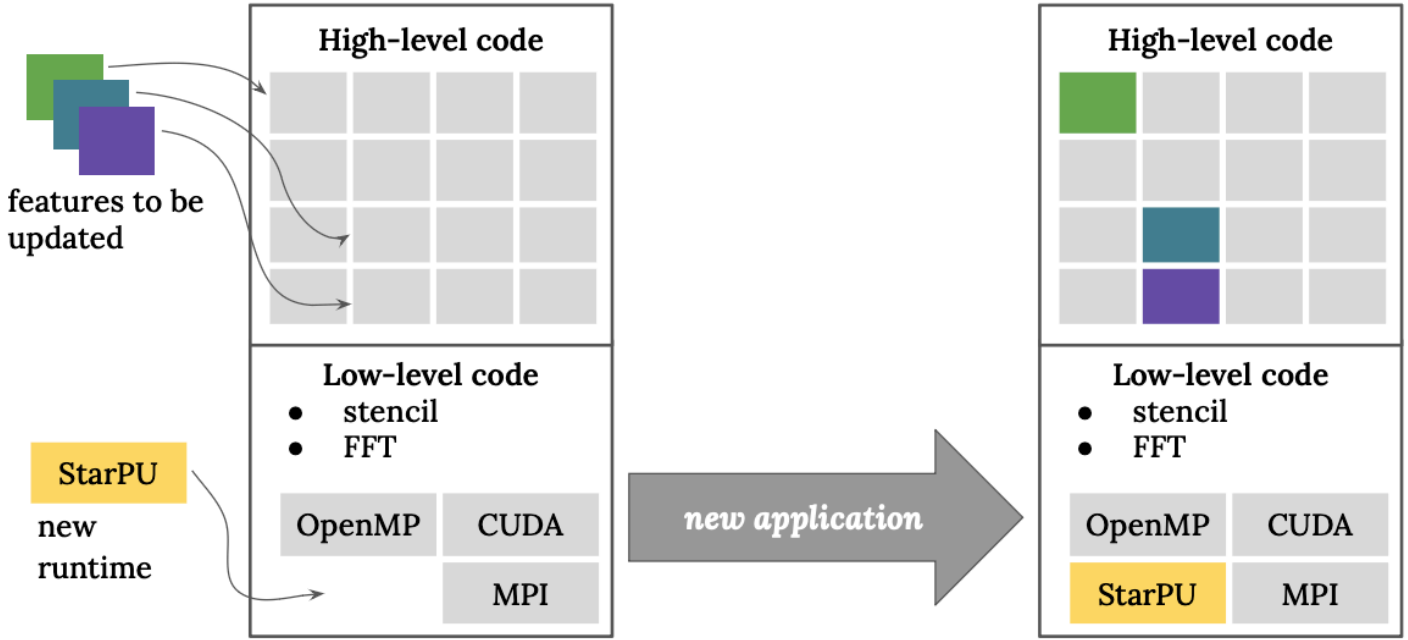
1. Introduction to COMET

Managing code variability in HPC applications



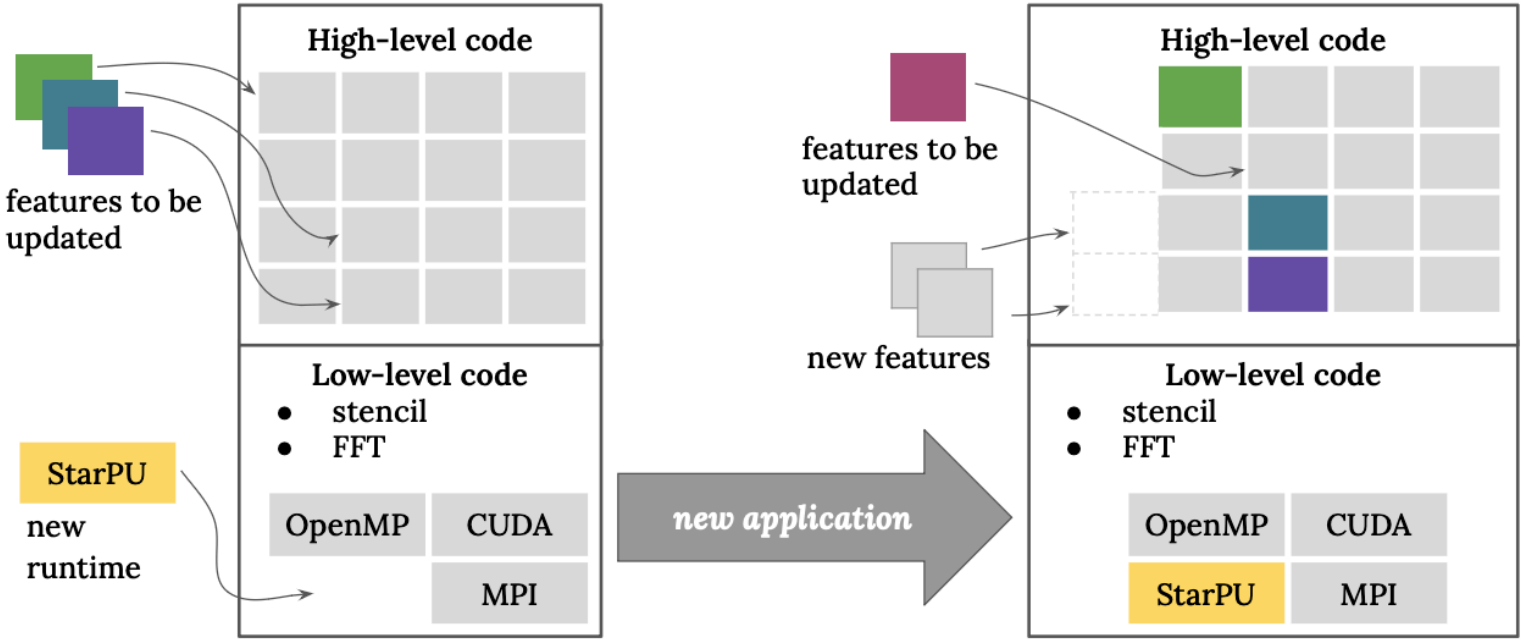
1. Introduction to COMET

Managing code variability in HPC applications



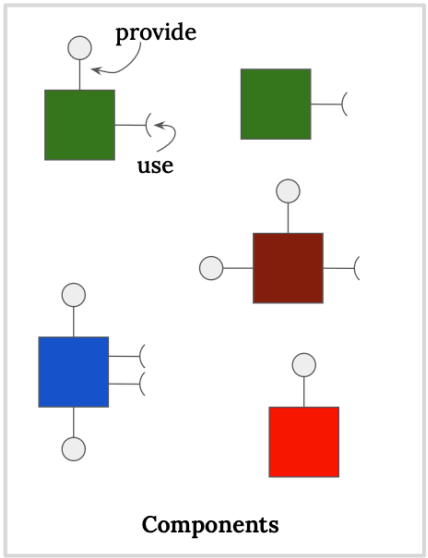
1. Introduction to COMET

Managing code variability in HPC applications



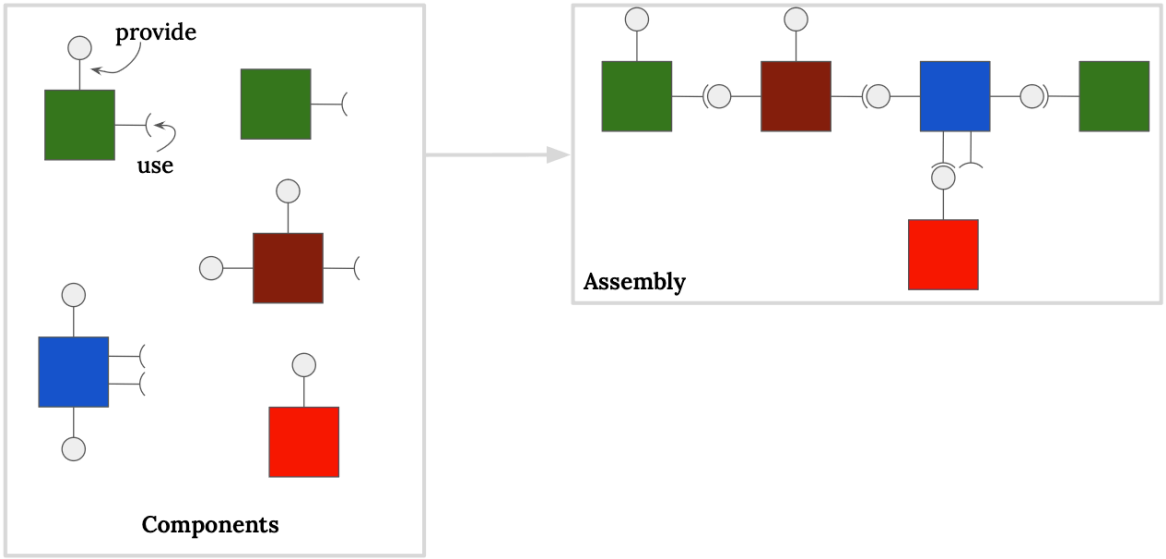
1. Introduction to COMET

Component-based programming model



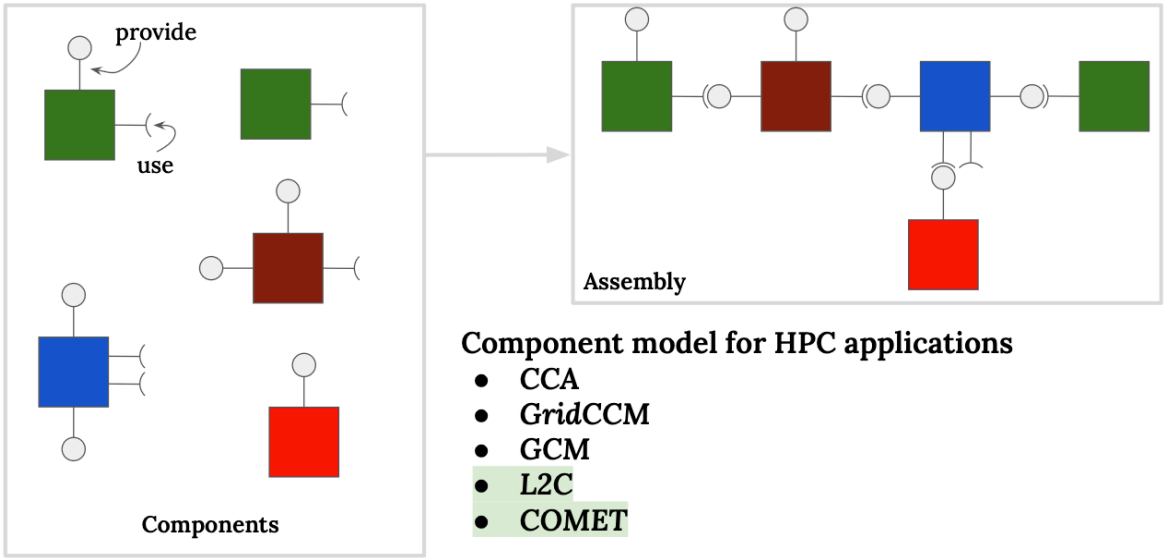
1. Introduction to COMET

Component-based programming model



1. Introduction to COMET

Component-based programming model



Component model for HPC applications

- CCA
- *GridCCM*
- GCM
- L2C
- COMET

1. Introduction to COMET

L2C and COMET

L2C
- *Ports : Use + Provide*
- *Ports: Corba + MPI*
- *Abstract C++ classes to define services*
- *Annotated C++ classes to define components*
- *XML assembly description file*

1. Introduction to COMET

L2C and COMET

COMET

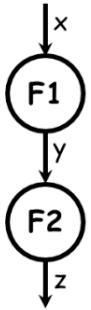
L2C

- ***Ports : Use + Provide***
- ***Ports: Corba + MPI***
- ***Abstract C++ classes to define services***
- ***Annotated C++ classes to define components***
- ***XML assembly description file***

- *Ports: Data + Partitioned Data*
- *Task graph-based executive scheduling support*

1. Introduction to COMET

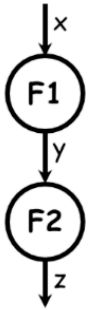
COMET composition units



Example of dataflow

1. Introduction to COMET

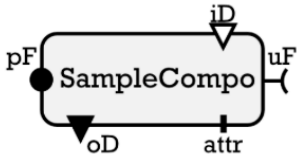
COMET composition units



Example of dataflow

Component *SampleCompo*

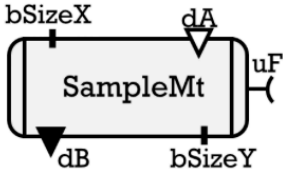
provide	ProvideFunc	*pF*
use	UseFunc	*uF*
indata	Array2D	*iD*
outdata	Array2D	*oD*
attribute	Integer	*attr*



Component description

Metatask *SampleMt*

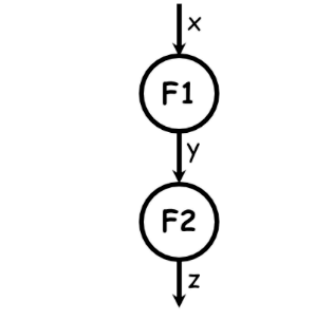
use	UseFunc	*uF*
indata	BlockArray2D	*dA*
outdata	BlockArray2D	*dB*
attribute	Integer	*bSizeX*
attribute	Integer	*bSizeY*
relation	Align	5p-stencil



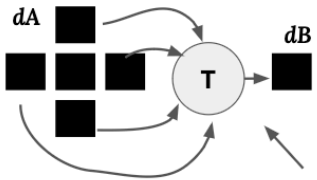
Metatask description

1. Introduction to COMET

COMET composition units

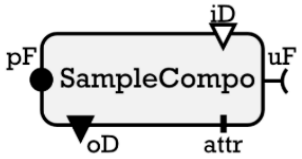


Example of dataflow



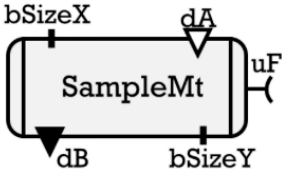
$dB(i,j) \leftarrow dA(i-1,j), dA(i+1,j), dA(i,j), dA(i,j-1), dA(i,j+1)$

Component *SampleCompo*		
provide	ProvideFunc	*pF*
use	UseFunc	*uF*
indata	Array2D	*iD*
outdata	Array2D	*oD*
attribute	Integer	*attr*



Component description

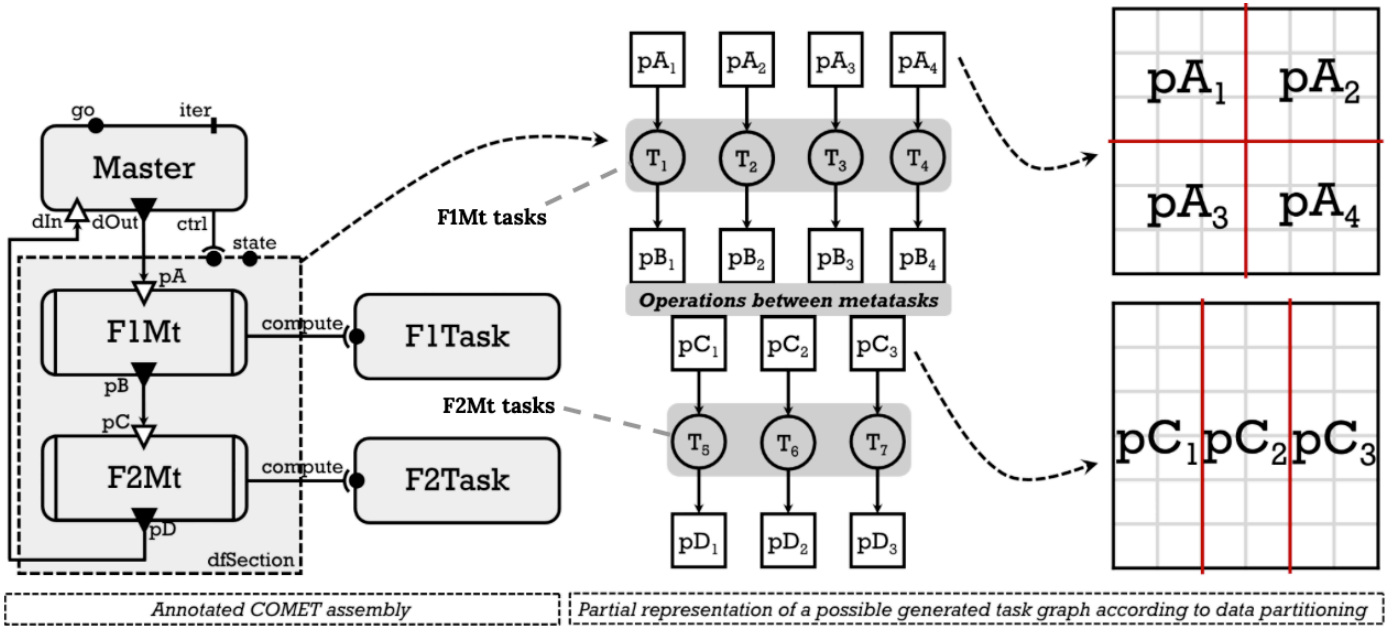
Metatask *SampleMt*		
use	UseFunc	*uF*
indata	BlockArray2D	*dA*
outdata	BlockArray2D	*dB*
attribute	Integer	*bSizeX*
attribute	Integer	*bSizeY*
relation	Align	5p-stencil



Metatask description

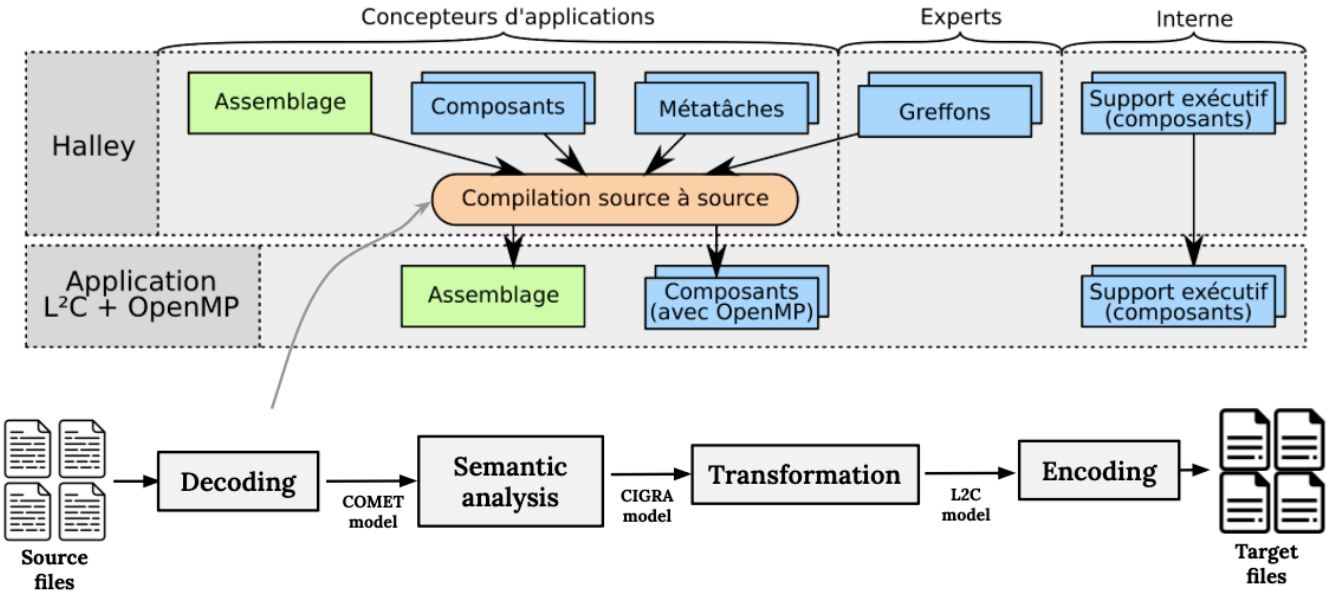
1. Introduction to COMET

COMET assembly and a possible generated task graph



1. Introduction to COMET

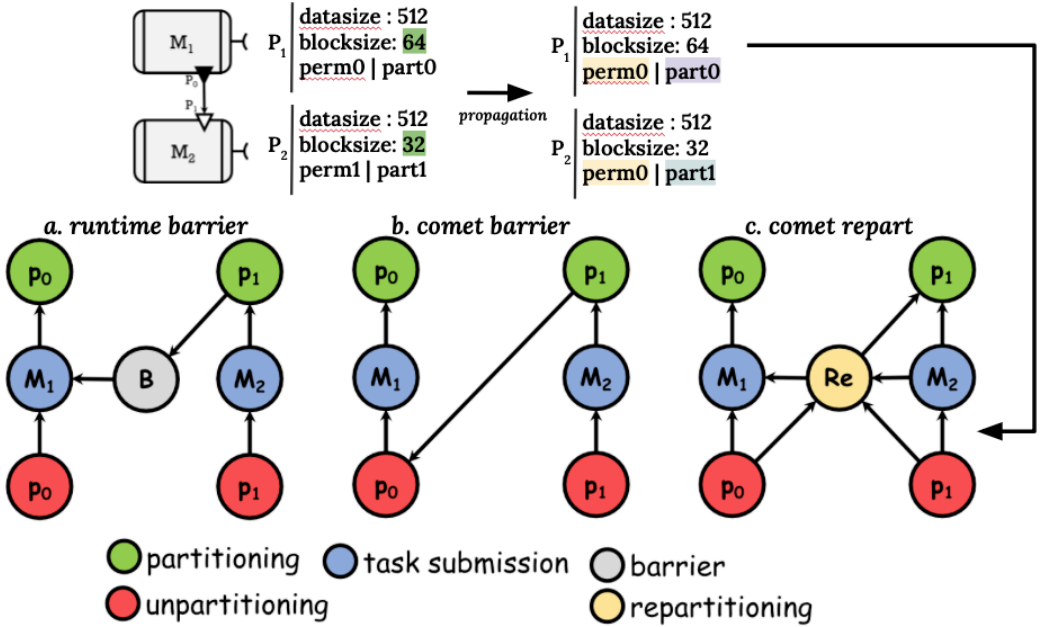
Implementing the COMET model



2. Handling Dynamic Data-Parallel Dataflows with COMET

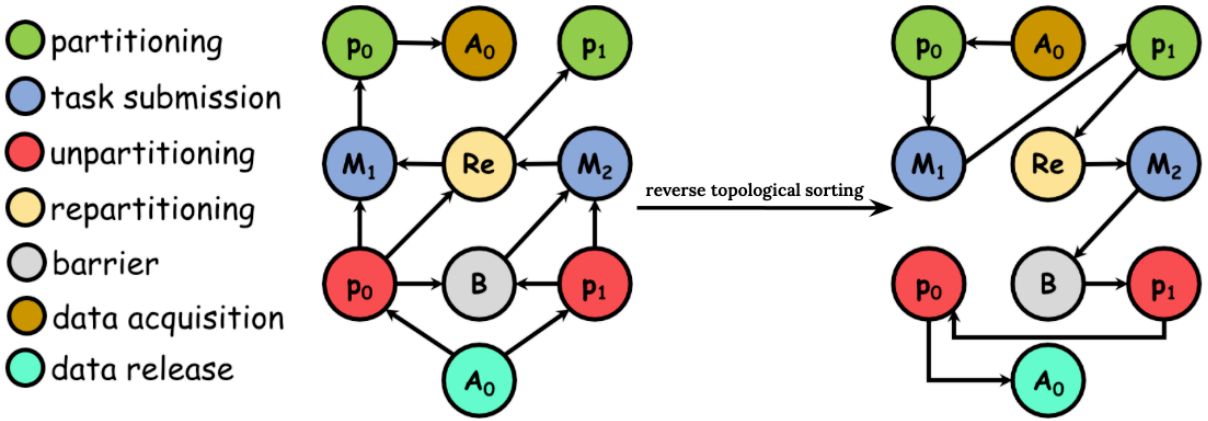
2. Handling Dynamic Data-Parallel Dataflows with COMET

CIGRA : COMET Instruction GRAph



2. Handling Dynamic Data-Parallel Dataflows with COMET

Reverse topological sorting to a CIGRA



2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192

Blocksizes :

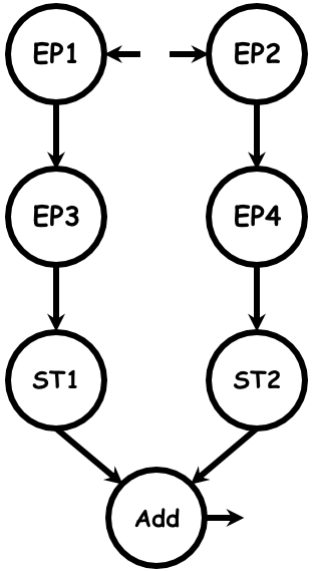
1024x256

EP1, EP2, ST1, ST2

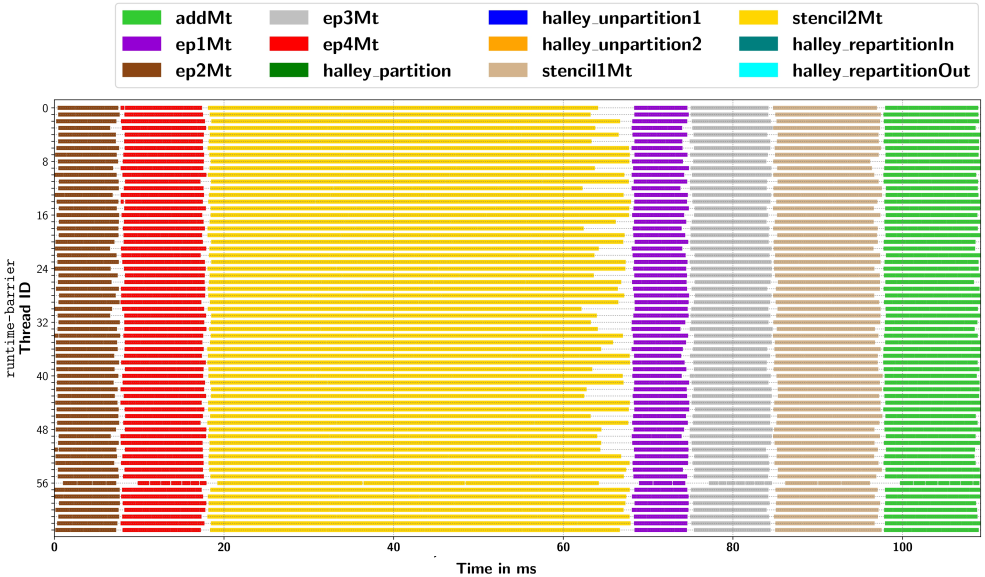
1024x128

EP3, EP4, Add

Runtime : mpc-openmp



The bench



a. runtime barrier

2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192

Blocksizes :

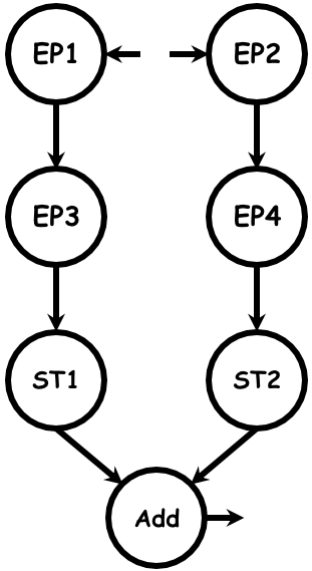
1024x256

EP1, EP2, ST1, ST2

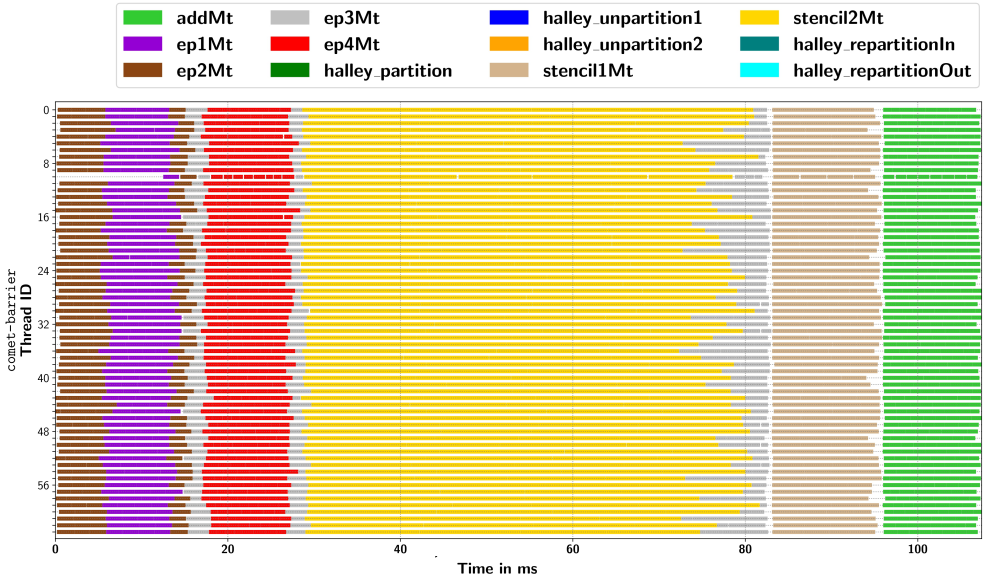
1024x128

EP3, EP4, Add

Runtime : mpc-openmp



The bench



b. comet barrier

2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192

Blocksizes :

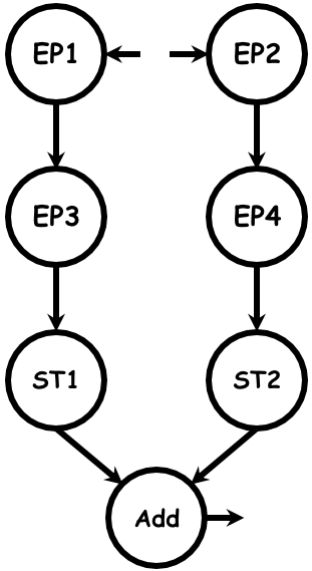
1024x256

EP1, EP2, ST1, ST2

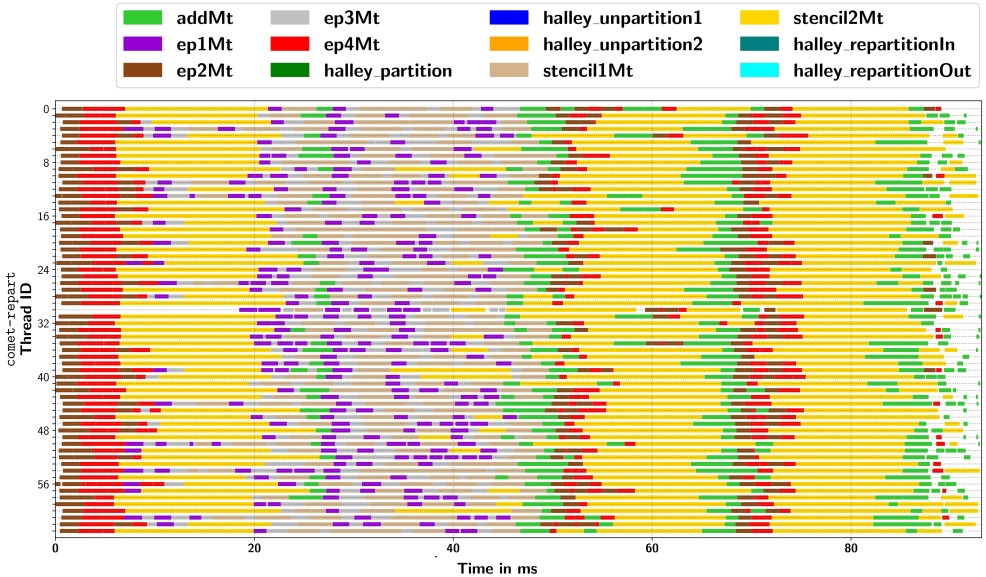
1024x128

EP3, EP4, Add

Runtime : mpc-openmp



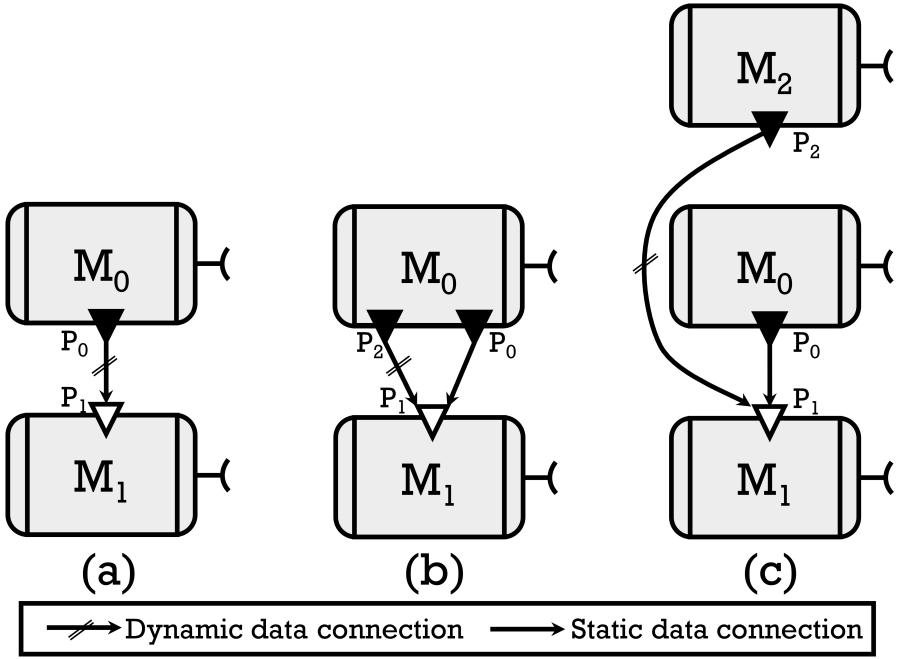
The bench



c. comet repart

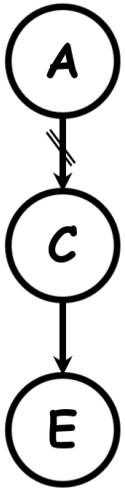
2. Handling Dynamic Data-Parallel Dataflows with COMET

Kinds of dynamic data connection with COMET



2. Handling Dynamic Data-Parallel Dataflows with COMET

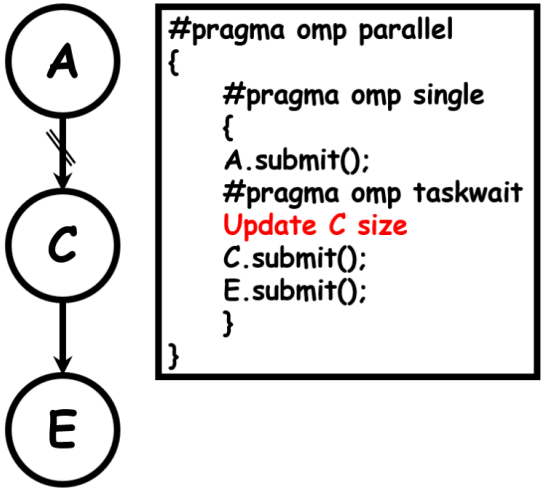
Flat code generation strategy



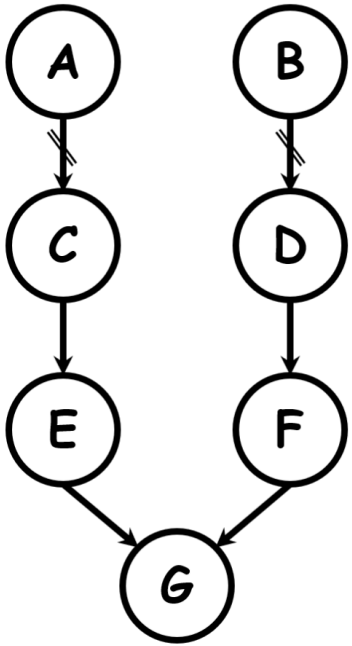

```
#pragma omp parallel
{
    #pragma omp single
    {
    A.submit();
    #pragma omp taskwait
    Update C size
    C.submit();
    E.submit();
    }
}
```

2. Handling Dynamic Data-Parallel Dataflows with COMET

Flat code generation strategy



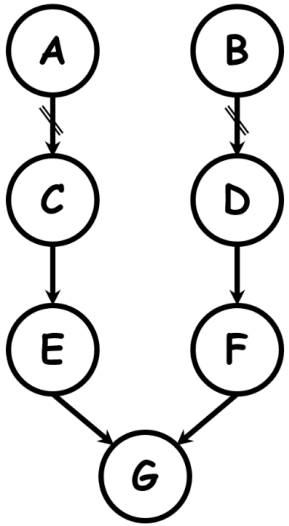

```
#pragma omp parallel
{
    #pragma omp single
    {
    A.submit();
    #pragma omp taskwait
    Update C size
    C.submit();
    E.submit();
    }
}
```




```
#pragma omp parallel
{
    #pragma omp single
    {
    A.submit();
    #pragma omp taskwait
    Update C size
    C.submit();
    E.submit();
    B.submit();
    #pragma omp taskwait
    Update D size
    D.submit();
    F.submit();
    G.submit();
    }
}
```

2. Handling Dynamic Data-Parallel Dataflows with COMET

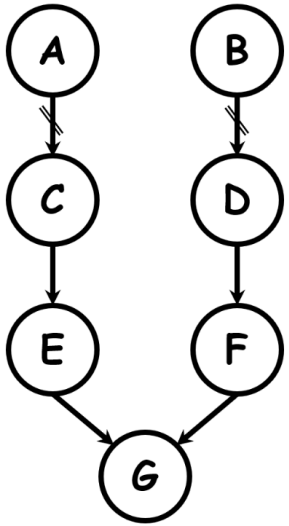
Nested code generation strategy




```
#pragma omp parallel
{
    #pragma omp single
    {
    #pragma omp task o(A)
        A.submit();
        #pragma omp taskwait
        Update C size
    #pragma omp task io(C) i(A)
        C.submit();
        #pragma omp taskwait
    #pragma omp task io(E) i(C)
        E.submit();
        #pragma omp taskwait
    #pragma omp task o(B)
        B.submit();
        #pragma omp taskwait
        Update B size
    }
    …
    …
}
```

2. Handling Dynamic Data-Parallel Dataflows with COMET

Weak-nested code generation strategy




```
#pragma omp parallel
{
    #pragma omp single
    {
    #pragma omp task o(A) weak
        A.submit();
        #pragma omp taskwait
        Update C size
    #pragma omp task io(C) i(A) weak
        C.submit();
    #pragma omp task io(E) i(C) weak
        E.submit();
    #pragma omp task o(B) weak
        B.submit();
        #pragma omp taskwait
        Update B size
    }
    …
    …
}
```

2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192
Blocksizes :
1024x128

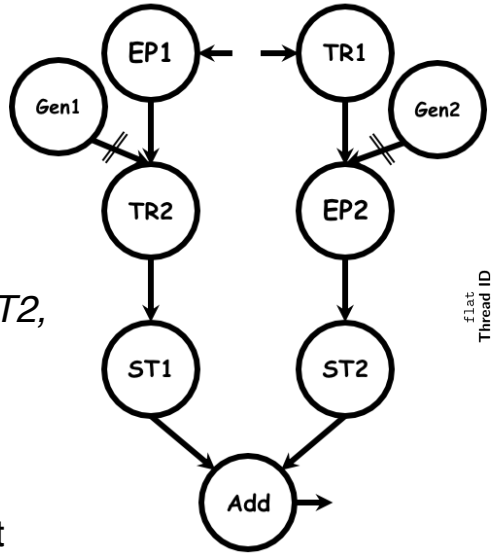
EP1, EP2, ST1, ST2,

Add

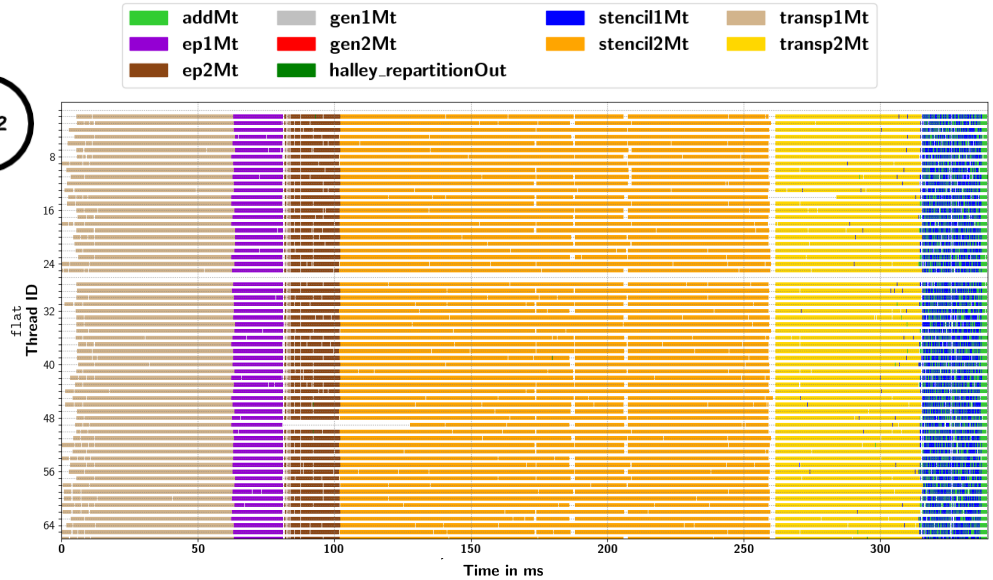
128x128

TR1, TR2

Op. bet. meta. : repart
Runtime : llvm-ompss2



The bench

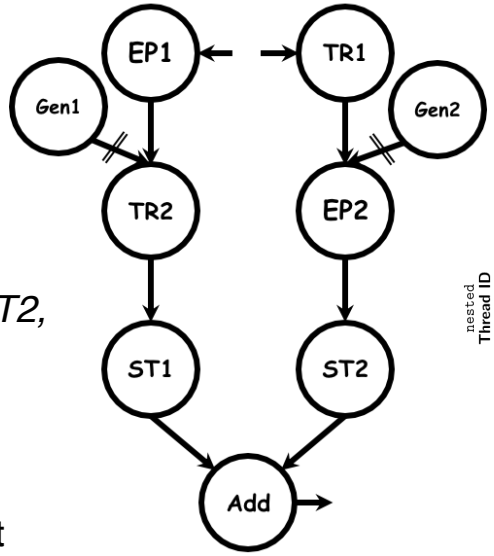


a. flat

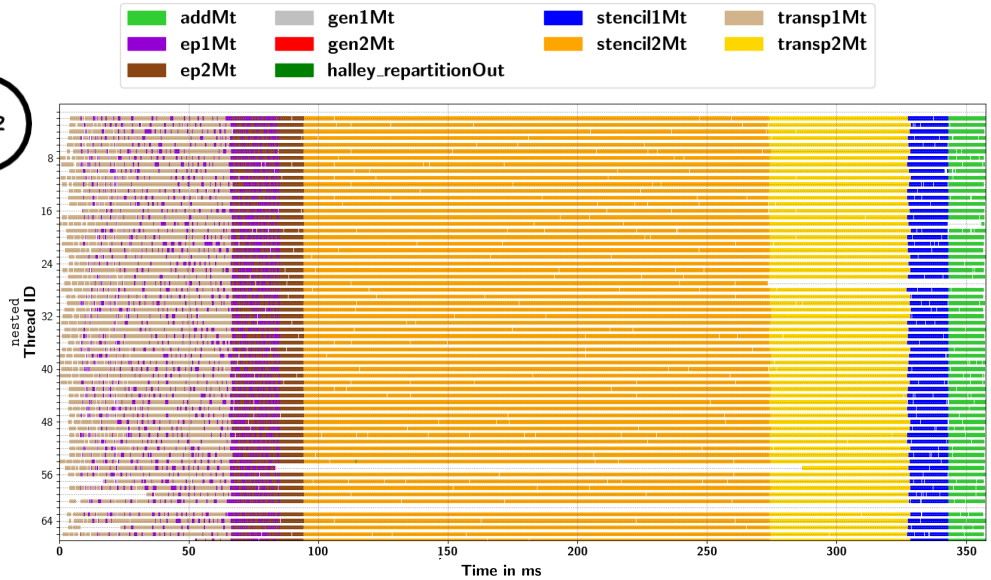
2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192
Blocksizes :
1024x128
EP1, EP2, ST1, ST2, Add
128x128
TR1, TR2
Op. bet. meta. : repart
Runtime : llvm-ompss2



The bench

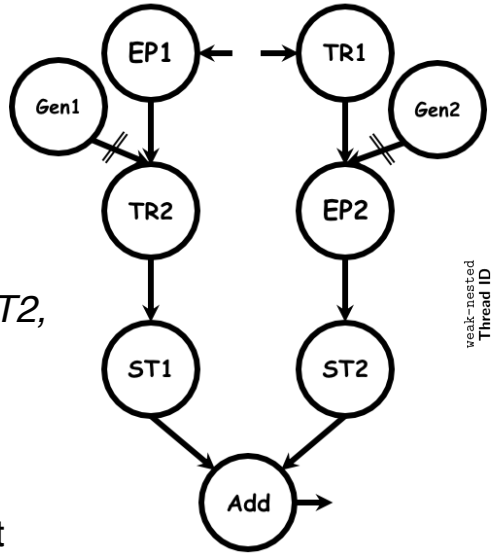


b. nested

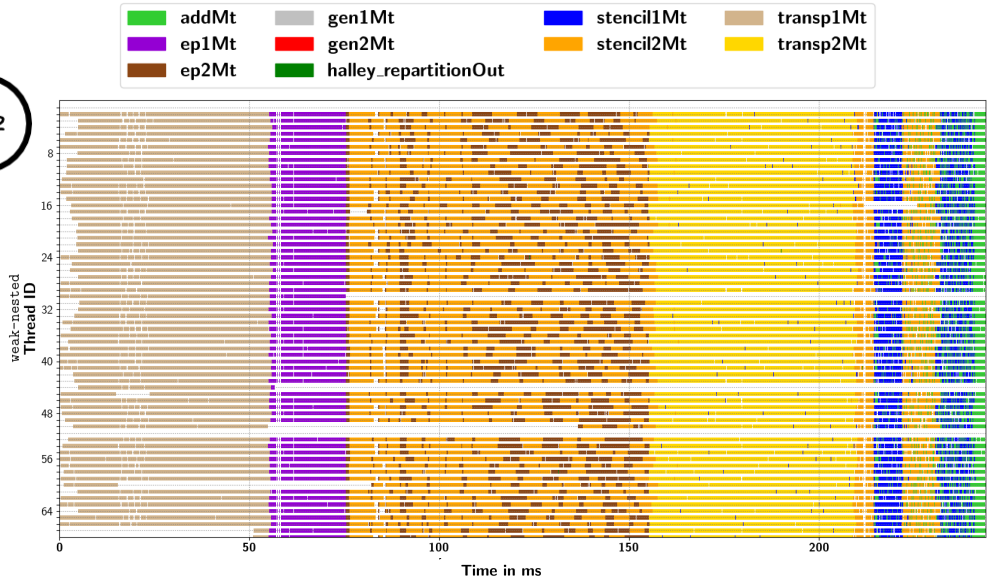
2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192
Blocksizes :
1024x128
EP1, EP2, ST1, ST2, Add
128x128
TR1, TR2
Op. bet. meta. : repart
Runtime : llvm-ompss2



The bench



c. weak-nested

2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Datasize : 8192x8192

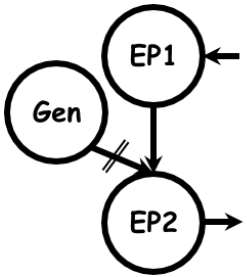
Blocksizes :

1024x1024, 1024x512, 1024x256, 1024x128

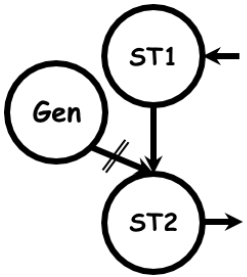
Runtimes :

llvm-openmp, mpc-openmp, llvm-ompss2

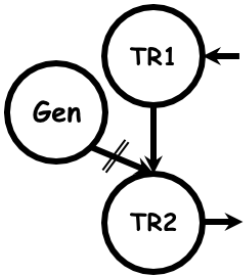
Op. bet. meta. : repart



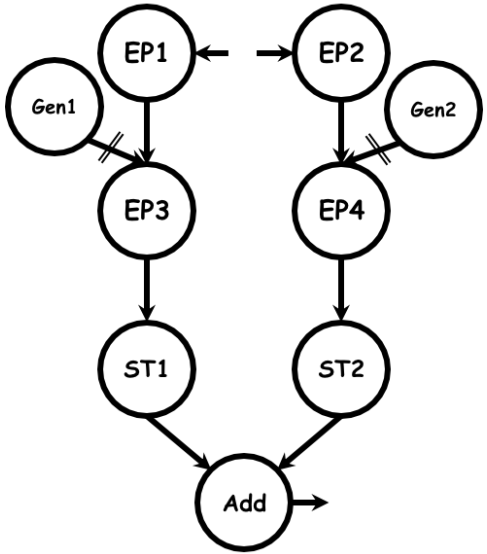
EP²



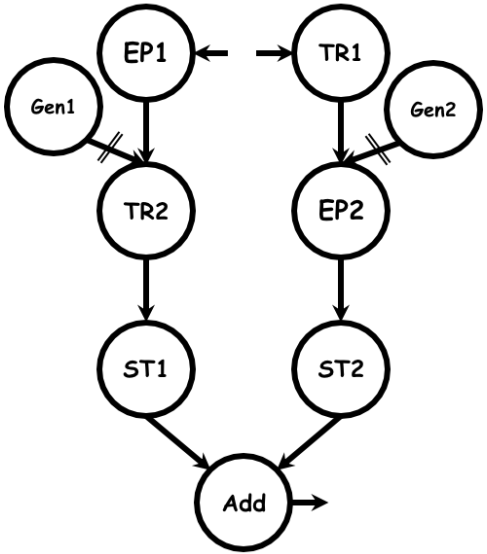
Stencil (ST²)



Transpose (TR²)



macrobench 1 (mB1)



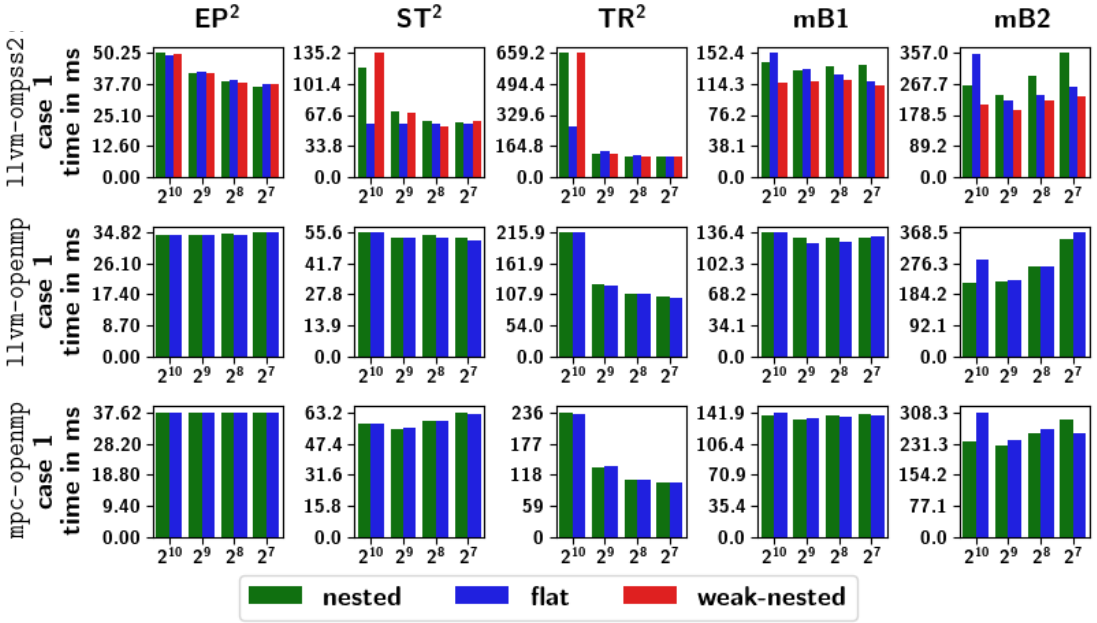
macrobench 2 (mB2)

2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Observations :

- In microbenchs, flat is better than nested and weak-nested whatever the runtime.
- In mB1 and mB2, weak-nested is better than flat and nested on the llvm-ompss2 runtime.
- In mB1 and mB2, nested is than flat and nested on llvm-ompss2 runtime.
- There are some anomalies…

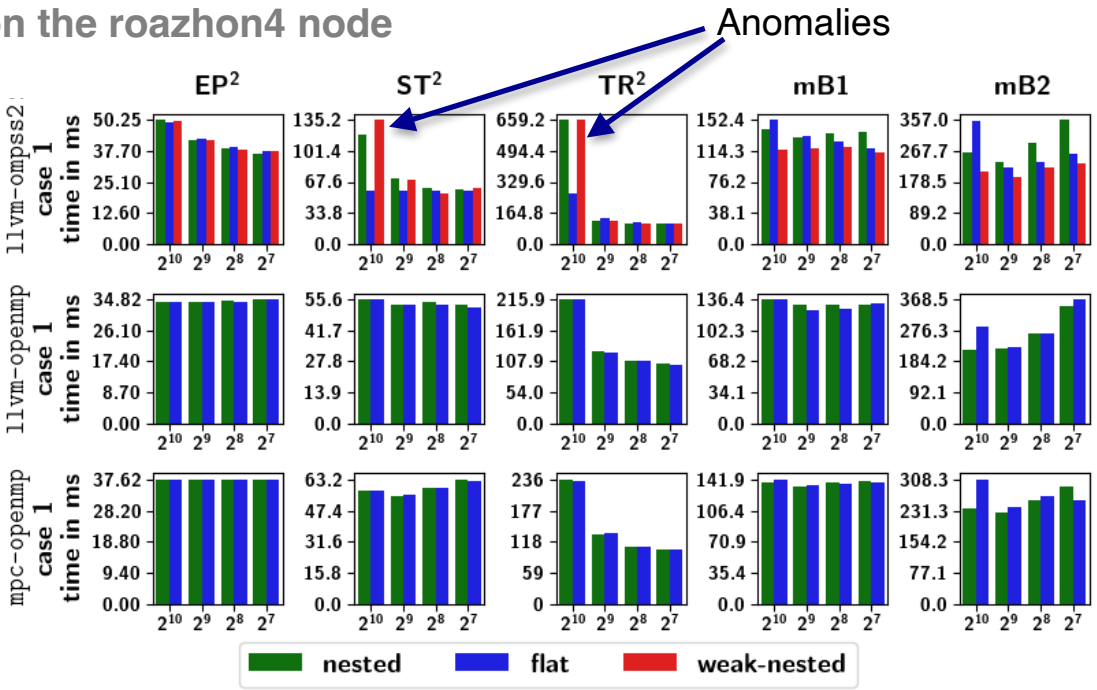


2. Handling Dynamic Data-Parallel Dataflows with COMET

Experimental results on Grid'5000 on the roazhon4 node

Observations :

- In microbenchs, flat is better than nested and weak-nested whatever the runtime.
- In mB1 and mB2, weak-nested is better than flat and nested on the llvm-ompss2 runtime.
- In mB1 and mB2, nested is than flat and nested on llvm-ompss2 runtime.
- There are some anomalies…



3. Conclusion

3. Conclusion

Wrap-up :

- COMET is a component-based programming model designed for creating HPC applications through fine-grained, dataflow-like compositions.
- We have extended it to manage applications designed by dynamic data-parallel dataflows.
- Experiments on several benchmarks show that the execution times of our different code generation approaches depend on the runtime used.
- OpenMP needs to be upgraded to incorporate the concept of weak dependencies.

What's next ?

- Understand and correct any anomalies observed.
- We have done a micro-aevol version with COMET but the performance is not good. We need to do something about that.
- We have started integrating StarPU to COMET. We are working on the management of partitioned data between both.

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

cea

cnrs

Inria

FRANCE 2030

PROGRAMME DE RECHERCHE

NUMÉRIQUE POUR L'EXASCALE

Retrouvez toutes nos actualités

NumPEx