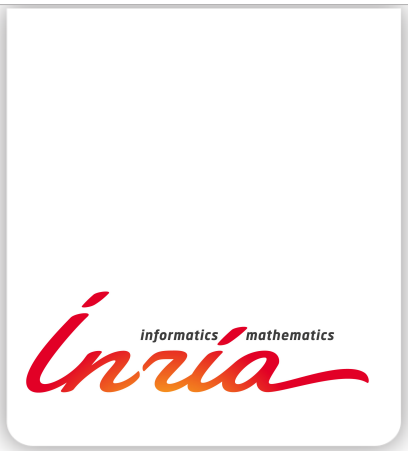
Component-based parallelization of multi-stencil programs

Héléne Coullon, Christian Perez

RESEARCH REPORT
N° 7003
April 2015
Project-Teams Avalon

ISSN 0249-6399 ISRN INRIA/RR--7003--FR+ENG



Component-based parallelization of multi-stencil programs

Héléne Coullon*†, Christian Perez‡

Project-Teams Avalon

Research Report n° 7003 — April 2015 — 7 pages

Abstract: abstract

Key-words: No keywords

* Footnote for first author
† Shared foot note
‡ Footnote for second author

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Exemple de document utilisant le style rapport de recherche Inria

Résumé : Resume en francais

Mots-clés : mots-cles

Contents

1 Introduction

2 Computational model of multi-stencil programs

To numerically solve a set of PDEs, iterative methods are frequently used to approximate the solution by a step by step phenomena. Thus, the continuous time and space domains are discretized so that a set of numerical computations are iteratively (time discretization) applied on a mesh (space discretization). In other words, the PDEs are transformed to a set of numerical computations applied at each time step on all elements of the discretized space domain. Among the numerical computations is found a set of numerical schemes, also called *stencil computations*. A formal definition of a *stencil program*, and the parallelization of such program, are presented in this section.

2.1 Definitions

A mesh $\mathcal{M}$ defines the discretization of the continuous space domain Ω of a set of PDEs and is defined as followed.

Definition 1 *A mesh is a connected undirected graph $\mathcal{M} = (V, E)$, where V is the set of vertices and E the set of edges. The set of edges E of a mesh $\mathcal{M} = (V, E)$ does not contain bridges.*

Definition 2 *D_i is a set of elements of a mesh $\mathcal{M} = (V, E)$, constructed by a function $domain_i$ which defines a precise association between V and E, $domain_i : V \times E \to D_i$.*

For example, the set of cells D_0 in a Cartesian 2D mesh could be defined by exactly four vertices and four edges connected as a cycle. But we could also define another set of elements D_1 as the simple set of vertices V etc.

A mesh can be structured (as Cartesian or curvilinear meshes), unstructured, regular or irregular (without the same topology for each element) and hybrid.

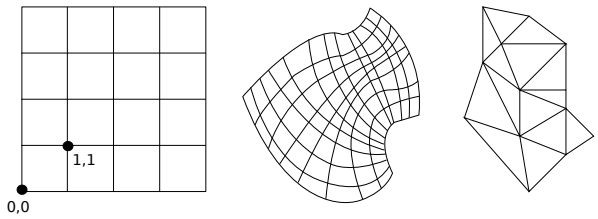


Figure 1: From left to right, Cartesian, curvilinear and unstructured meshes.

Definition 3 *The discretization of the continuous time domain $\mathcal{T}$ is denoted T such that $\forall t_i, t_{i+1} \in T, \exists \Delta t \in \mathbb{R}$, $t_{i+1} = t_i + \Delta t$. Thus, T is responsible for the iteration time steps of the numerical simulation.*

In a numerical simulation a set of data, or quantities, are applied onto the mesh and represent the set of values to compute, or to use, for computation.

Definition 4 *The set of data applied on the mesh is denoted by Δ, such that $\delta \in \Delta$ is a function which associates each element $d \in D_i$ to a value $v \in V$, $\delta : D_i \to V$.*

One can notice that in applied mathematics, the signature of δ would be $\delta : E_i \times T \to V$, however when programming a numerical simulation it is not wise to store all values of each time iteration.

Definition 5 *A numerical expression exp is a function which represents how to compute, for an element $d \in D_i$, a data $w \in \Delta$ (written data) with a set $R \subset \Delta$ of input data (read data), $exp : R \times D_i \to w \times D_i$.*

Definition 6 *A computation c of a numerical simulation is defined as $c(R, w, D, exp)$, where $R \subset \Delta, w \in \Delta$ and exp a numerical expression $exp : R \to w$. D is one of the subsets $D_i \subset \mathcal{M}$, such that $w : D \to V$.*

It has to be noticed that at each time iteration, all the elements of a mesh are computed. However, it happens that the computation of the mesh elements is splitted in different computations (for example the computation of the physical border). In this case additional D_i can be specified for the mesh $\mathcal{M}$.

Definition 7 *The set of n ordered computations of a numerical simulation is denoted $\Gamma = [c_i]_{0 \leq i \leq n-1}$, such that $\forall c_i, c_j$ with $i \leq j$, c_i is computed before c_j, and c_j can be computed only when c_i is finished.*

Definition 8 *Finally, a multi-stencil program is defined by the quadruplet $\mathcal{MSP}(T, \mathcal{M}, \Delta, \Gamma)$.*

Definition 9 *The neighborhood $\mathcal{N}$ of an element $d \in D_i$ is a function to define a set of elements in any $D_k \subset \mathcal{M}$, $\mathcal{N} : D_i \to D_k \times D_k \times \dots$.*

The function $\mathcal{N}$ is also sometimes called the *stencil shape*, or the *stencil* in applied mathematics.

A computation $c \in \Gamma$ can be of two different types. The first type is called a *stencil computation*.

Definition 10 *A stencil computation is defined as a quintuplet $s(R, w, D, exp, \mathcal{N})$, where $R \subset \Delta, w \in \Delta$, D is one the subsets D_i and $w : D \to V$.*

In a stencil computation s, $\forall d \in D$, the stencil numerical expression exp is applied such that $w(d) = \exp(R(d), R(\mathcal{N}(d))$. In this work, a stencil computation $s(R, w, D, \exp, \mathcal{N})$ always verifies $R \cap w = \emptyset$, otherwize an implicit numerical scheme has to be solve which is over the scope of this paper.

Figure **??** gives an example of a stencil computation $s(R, w, D, \exp, \mathcal{N})$, where $\mathcal{M}(V, E)$ is a two dimensional Cartesian mesh. A single domain D is defined in this example and is composed of cells formed by a cycle of four vertices $v \in V$ and four edges $e \in E$. Furthermore, in this example $R = \{A\}$, $w = B$, and for $(x, y) \in D$ the neighborhood function is

$$\mathcal{N} : (x, y) \rightarrow \{(x, y + 1), (x, y - 1), (x + 1, y), (x - 1, y)\}.$$

Finally, the numerical expression of this example is

$$\exp(A(x, y), A(\mathcal{N}(x, y)) = B(x, y) = A(x, y)+(A(x, y+1)+A(x, y-1)+A(x+1, y)+A(x-1, y))/4.$$

Finally, the second type of numerical computation is a local computation.

Definition 11 *A local computation is a quadruplet $l(R, w, D, exp)$, where e does not involve a neighborhood function $\mathcal{N}$.*

2.2 Parallelization

Multi-stencil mesh-based numerical simulation can be parallelized in various ways and is an interesting kind of application to take advantage of modern heterogeneous HPC architectures, mixing clusters, multi-cores CPUs, vectorization units, GPGPU and many-core accelerators.

Coarse-grain data parallelism. In a data parallelization technique, the idea is to split the data on which the program is computed in balanced sub-parts, one for each available resource. The same sequential program can afterwards be applied on each sub-part simultaneously, with some additioinal synchronizations between resources to update the data not computed locally and thus to guarantee a correct result.

More formally, the data parallelization of a multi-stencil program $\mathcal{MSP}(T, \mathcal{M}, \Delta, \Gamma)$ consists in, first, a partitioning of the mesh $\mathcal{M}$ in p balanced sub-meshes (for p resources) $\{\mathcal{M}_0, \ldots, \mathcal{M}_{n-1}\}$. This step can be performed by an external graph partitionner [] and is not adressed by this paper. As a data is mapped onto the mesh, the set of data Δ is partitionned the same way than the mesh in $\{\Delta_0, \ldots, \Delta_{n-1}\}$. The second step of the parallelization is to identify in Γ the needed synchronizations between resources to update data, and thus to build a new ordered list of computations Γ'.

Definition 12 *For n the number of computations in Γ, and $i < j < n$, a synchronisation is needed between c_i and c_j, denoted $c_i \ll c_j$, if $c_j = s_j(R_j, w_j, D_j, exp_j)$, and $w_i \cap R_j \neq \emptyset$. Moreover, the data to update is $w_i \cap R_j$.*

Actually, a synchronization can only be needed by the data read in a stencil computation, and only if this data has been modified before, which means that it has been written before. This synchronization is needed because the neighborhood function $\mathcal{N}$ of the stencil computation involves values computed on different resources.

Definition 13 *A synchronization between two computations $c_i \ll c_j$ is defined as a specific computation update$(w_i \cap R_j)$.*

We denote an function to insert at a precise position of a given ordered list, a computation c_{up}, as

$$\text{insert} : [c_k]_{0 \leq k \leq n-1} \times \text{position} \times c_{up} \rightarrow [c_0, \ldots, c_{\text{position}-1}, c_{up}, \ldots, c_{n-1}] \quad (1)$$

The new ordered list of computations Γ' is obtained from the insertion of the synchronizations at the good position. Algorithm 1 illustrates how Γ' is build.

Algorithm 1: Creation of Γ' from Γ

```
Γ' = Γ
offset = 0
for c_j ∈ Γ do
    if c_j is a stencil then
        if Γ[c_0 : c_j] == [] then
            insert(Γ', 0, update(R_j)
            offset + +
        else
            for c_i ∈ Γ[c_0 : c_j] do
                if c_i ≪ c_j then
                    insert(Γ', j + offset, update(w_i ∩ R_j)
                    offset + +
```

The final step of this parallelization is to run Γ' on each resource. Thus, for each resource $0 \leq k \leq p-1$ a multi-stencil program defined by

$$\mathcal{MSP}_k(T, \mathcal{M}_k, \Delta_k, \Gamma'), \quad (2)$$

is runned.

We denote this parallelization technique a coarse-grain data parallelization in contrast with the same technique applied to the finer level of a single computation. In this case, for a computation $c(R, w, D, \text{exp})$, the local domain D is divided for p resources $\{D_0, \ldots, D_{p-1}\}$, and each resource k is responsible for a sub-computation $c_k(R, w, D_k, \text{exp})$. This finer data parallelization does not require the addition of synchronizations because each computation . This finer data parallelization technique is not directly adressed by the work presented in this paper as it will be explained later.

Task parallelism.

3 Component model and transformation

3.1 A primitive component model

Definitions of a primitive component (ports, interfaces), a component instanciation, an assembly + simple semantics on the behavior of a primitive component. Cite L2C as an implementation of such a model + introduction of notations (use,provide,use-multiple,sync).

3.2 Control components

Definition of three control components, and of the kernel component. Explain the behavior.

3.3 Transformation to a parallel assembly

Transformation from a parallel algorithm using the directives: PARALLEL, PARALLEL SECTIONS, SECTION and FORALL, to a component assembly

4 Component Stencil Language

4.1 CSL language and example

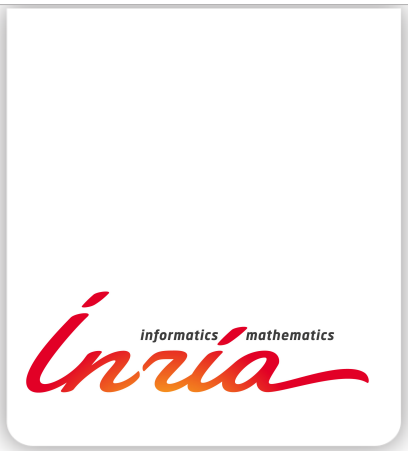
4.2 CCSL compiler and example

4.3 Evaluation

5 Related work

stencil compilation: Pochoir, PATUS etc.
stencil programs: Lizst, OP2
control in components: X-MAN (The New Component Model), STCM, Kell-calculus

6 Conclusion



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399