Università di Pisa
Dipartimento di Informatica

Technical Report: TR-09-17

Minimizing Communications with $\mathcal{Q}-transformations$ in Uniform and Affine Stencils

Massimiliano Meneghin, Marco Vanneschi

October 6, 2009

Address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

Minimizing Communications with $\mathcal{Q}-transformations$ in Uniform and Affine Stencils

Massimiliano Meneghin, Marco Vanneschi

October 6, 2009

Contents

1 Introduction **2**

2 Related Works **3**

3 *HUA* Model **4**

3.1 A General Model 5

3.2 Defining the *HUA* Model 6

3.3 Example of a nine point stencil in HUA Model 7

4 $\mathcal{Q}-transformations$ in the HUA model **9**

4.1 q_shift method: combination of $\mathcal{Q}-transformations$ and $shift$ method 11

4.2 Example of a positive $\mathcal{Q}-transformation$ on a nine point stencil 12

5 Combining Positive and Negative $\mathcal{Q}-transformations$ **13**

6 Breaking the Toroidal Constrain **15**

7 Experimental Results **16**

8 Conclusion and Future Works **17**

Abstract

In stencil based parallel applications, communications represent the main overhead, especially when targeting a fine grain parallelization in order to reduce the completion time. Techniques that minimize the number and the impact of communications are clearly relevant. In literature the best optimization reduces the number of communications per step from 3^{dim}, featured by a naive implementation, to $2 * dim$, where dim is the number of the domain dimensions. To break down the previous bound, in the paper we introduce and formally prove $\mathcal{Q}$–$transformations$, for stencils featuring data dependencies that can be expressed as geometric affine translations. $\mathcal{Q}-transformations$, based on data dependencies orientations though space translations, lowers the number of communications per step to dim.

1 Introduction

Data parallelism is a well known paradigm of parallel programming, characterized by replication of functions and partition of data over a set of virtual processing nodes.

One of the most powerful data parallel paradigm is represented by the stencil paradigm; some examples are found in image processing [13] and partial difference equation solvers [11, 1, 6].

A stencil-based data parallel application is characterized by a certain number of iteration steps; at each step the processing nodes calculate a new value for all the elements of the partitioned domain, complying some data dependencies over sets of elements. Data dependencies are implemented by proper data exchanges (communications) between processing nodes.

In stencil based applications, communications represent the main overhead, especially when targeting a fine grain parallelization in order to reduce the completion time. Techniques that minimize the number and the impact of communications are clearly relevant.

To have an idea about the order of magnitude we are working with, we consider a worst case configuration in which every processing node needs data from all its neighbours. In this kind of stencil a *naive* method of implementation performs at each step $3^{dim} - 1$ incoming communications and so much outgoing ones, where dim is the number the domain dimensions.

Techniques to reduce the number of communications have been studied [10, 3, 8]. Plimpton [10] first presented a technique, which we refer to as $shift$ *method*, to reduce data exchanges with diagonal neighbours. The method cuts down the total number of communications in a step from $3^{dim} - 1$ to $2 * dim$ (see Fig. 1).

We focus our study to break down the previous bounds introducing the *homogeneous uniform affine* model (HUA) and $\mathcal{Q} - transformations$, a class of stencil transformations which lowers the number of communications per step to dim. HUA models a wide class of stencils where data dependencies can be expressed as geometric affine translations over a toroidal space. Exploiting HUA features we describe and proof $\mathcal{Q} - transformations$ which are based as well on geometric translations. In the paper, we report some performance tests to compare the communication overhead of different methods. Tests were performed on top of both cluster and multi-core architectures. The results show that for a fine grain parallelization the methods based on $\mathcal{Q} - transfomrations$ feature the best results with considerable speed up.

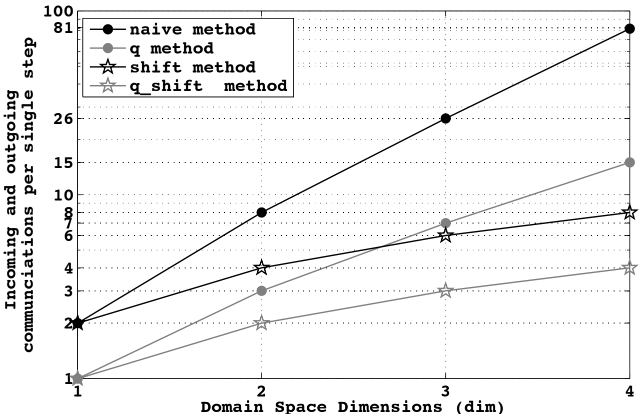


Figure 1: Number of communications per single step exploiting different methods

In section 2 we present the works in literature with which we compare our work. In section 3 the *HUA* model is described while the section 5 the $Q-transformations$ are presented. Section 6 introduces some extension to the *HUA* model to support not toroidal working domains. Section 7 reports performance test to compare communication overhead introduced by the method presented in the paper. Finally in section 8 we resume the obtained results and we point out some interesting ideas for future works.

2 Related Works

Stencil based applications, because of their relevance in parallel processing, have been the subject of many studies, which are focused on different aspects: from the minimization of the communication impact to the maximization of the cache performance.

The total number of dimensions over which a stencil works is $dim + 1$ if we consider, besides the dim space dimension, the one of the time. In literature, we can distinguish two main classes of studies: those that focus on time-space transformations and those which studies only space ones.

In the time-space class we find all the works applying tuned tiling techniques to stencil computations [12, 9, 4, 5, 2, 7]. Differently form those works, we focus only on *space transformations.*

Works on optimizing stencil without considering time dimension are relative less numerous. Plimpton [10] first presented a technique, which we refer to as $shift$ method, to reduce communications with diagonal neighbours. The $shift$ method cuts down the total number of sent messages per step from $3^{dim} - 1$ to $2 * dim$ (see Fig. 1). The technique is based on indirect communications with the diagonal neighbours. A single step can be seen as composed of a set of micro-steps, one for each space dimension. Differently from the *naive* method, the $shift$ technique requires, in order to support micro-steps, the interleaving of send and receive operations within the same step. This characteristic can produce some difficulties while managing overlapping of communications and computations as we will discuss later on. In the paper we consider the $shift$, with the *naive* method, as a yardstick for $\mathcal{Q}-transformations$ performance.

Ding and He [3] propose what they call *ghost expansion* method. They target the reduction of communications per step exploiting a kind of $oversending$ technique. The idea is to expand the data size exchanged between processing units, in such a way that communications are not required at each step. This technique force a processing unit to compute its own partition and in some steps, to update previous received data instead of communicating. With this method, the mean data exchanged per step is equal to *naive* and $shift$ ones, but the mean number of send and receive operations per step is reduced.

As the $shift$, the $oversending$ method too can be interpreted as a stencil transformation, where one step of the modified stencil is a kind of macro-step that corresponds to two or more steps of the original stencil.

A stencil transformed with the $oversending$ method can be then optimized with the $shift$ technique in order to delete diagonal communications as presented in Ding and He paper. Moreover the authors propose an optimized *ghost expansion* method for PDE problems that, playing with some algorithmic aspects of PDE solver, reduces also the mean data size exchanged.

The *ghost expansion* method and those based on $\mathcal{Q}-transfor-$ *mations* are independent, we postpone a deep comparison between the two techniques to other works.

Finally Palmer and Nieplocha [8] summarize the previous methods and present the result of their implementations on different distributed architectures exploiting the message passing library *MPI*. The main result is that no single algorithm provides optimal performance on all the platforms. We analyze the communication overhead for different computation grains and for two and three dimensional cases. Our result can extend Palmer and Nieplocha one: considering one specific target architecture, there is no one optimal algorithm for all the possible configurations of the space dimensions and number of element of the working domain. Nevertheless for fine grain parallelization the method achieving better performance is one of those based on $\mathcal{Q}-transformations$.

3 *HUA* Model

In this section we first introduce a general model to describe any kinds of stencil, than we pass to the presentation of some restrictions of the general model and consequently to the definition of the *HUA* model. The *HUA* model is also exploited to describe an example of a nine point stencil in a two dimension space.

For the easy of explanation, we focus our description on regular domain space, i.e. regular data partitions. Nevertheless all the considerations presented can be easily extended also to irregular cases.

3.1 A General Model

A stencil based application can be expressed as a set of consecutive steps, each one characterized by one initial and one final state of the domain elements. An example in a two dimensional space is the well known Jacobi iterative algorithm, based on finite difference approximations for solving partial differential equations.

Jacobi method consists of a number of iterations over a working domain given by a matrix J, which is initialized with the estimated value of a target function over a regular mesh. A pseudo code representation of the algorithm is:

```
for $i = 1$ to $MAX\_STEP\_NUM$ do
  for $x = 0$ to $X_{max}$ do
    for $y = 0$ to $Y_{max}$ do
      $J_{TMP}[x][y] = (J[x][y+1] + J[x][y-1]$
                  $+J[x+1][y] + J[x-1][y])/4$
    end for
  end for
  $J = J_{TMP}$
end for
```

With the aim of defining a general formalism, let $\mathcal{M}$ be a vector representation of a regular working domain. More precisely let $\mathcal{M}$ be a subset of $\mathcal{N}^{dim}$ containing the origin. All the element e of $\mathcal{M}$ are vector defined regarding the normal basis of $\mathcal{N}^{dim}$. In the Jacobi example $\mathcal{M}$ correspond to J, while a vector element e is associated with one value of the index couple (x, y).

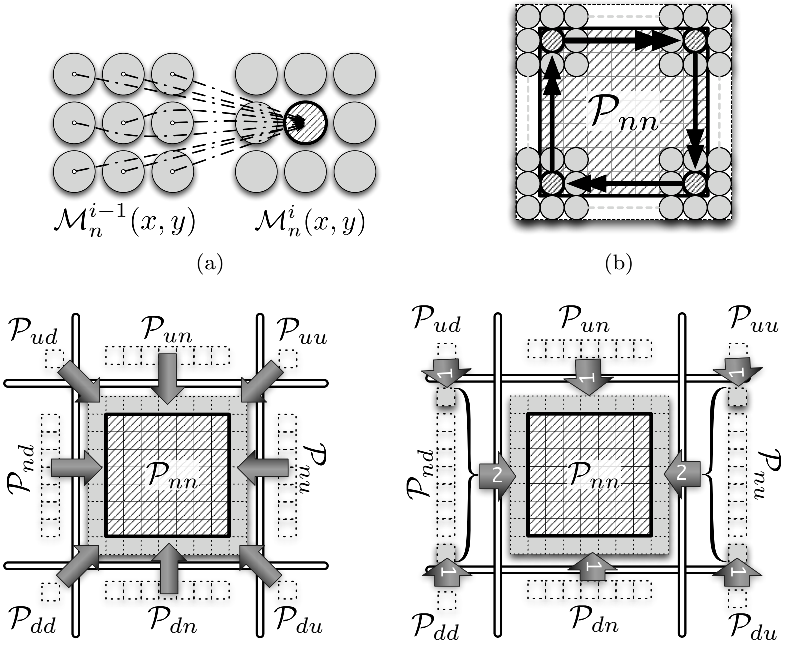
$\mathcal{M}^i$ denotes the domain status at the beginning of the step i, and $\mathcal{M}^i[e]$ the value associated to the vector e. Moreover as the domain state at the end of a step is the state at the beginning of the following step, $\mathcal{M}^{i+1}$ represent the state at the end of step i. In the Jacobi example $\mathcal{M}^i[e]$ is the value of $J[x][y]$ at the beginning if the iteration i.

We can model the transformation of a generic step i, which modifies the status of the domain $\mathcal{M}$ from $\mathcal{M}^i$ to $\mathcal{M}^{i+1}$, as follow:

$$\begin{array}{rcl} \forall e \in \mathcal{M} & \overset{step_i}{\rightarrow} & (\mathcal{F}_{ie}, \mathcal{S}_{ie}) \\ \mathcal{S}_{ie} & = & \{g_{1e}, g_{2e}, \ldots, g_{ne} |\ \forall \alpha\ g_{\alpha e} \in \mathcal{M}\} \\ \mathcal{M}^{i+1}[e] & = & \mathcal{F}_{ie}(\mathcal{M}^i[g_{1e}], \ldots, \mathcal{M}^i[g_{ne}]) \end{array}$$

The step transformation associates each element e of the domain to a couple formed by a function $\mathcal{F}_{ie}$ and an ordered set of elements $\mathcal{S}_{ie}$. The value of a domain element at the end of the step i, $\mathcal{M}^{i+1}[e]$, is found applying the function to the values of elements in $\mathcal{S}_{ie}$ evaluated at the beginning of the step i. In the Jacobi case the function $\mathcal{F}_{ie}$, which is equal for all the domain elements and for all the steps, calculates the mean over the element of the set $\mathcal{S}_{ie}$, which for the generic element (x, y) is defined as $\{(x, y+1), (x, y-1), (x+1, y), (x-1, y)\}$.

Because both the function $\mathcal{F}_{ie}$ and the set $\mathcal{S}_{ie}$ can be specified differently for each step and for each element of the domain, it is clear that this model can



(a) (b) (c) (d)

Figure 2: Graphical representation of a naive implementation of nine point stencil modeled with *HUA*: 2(a) virtual processors dependencies, 2(b) dependencies between partitions of virtual processors, 2(c) computing processing communication pattern performed in a naive method and 2(d) communication pattern with *shift* method, numbers on the arrows indicate a time order on different communications

express any kind of stencil.

3.2 Defining the *HUA* Model

The previous model is too general for most of the stencils like those present in PDE, image processing and other applications. We therefore consider some constraints in order to define what we call the *Homogeneous Uniform Affine* model (*HUA*).

Definition 1 (**Homogeneous Uniform Affine model**). *The HUA models applications represented by a* **homogeneous** *sequence of the same step, which* *is characterized as follows:*

$$\begin{aligned} \forall e \in \mathcal{M} &\overset{step_i}{\rightarrow} (\mathcal{F}_i, \mathcal{S}_i) \\ \mathcal{S}_i &= \{g_1, g_e, \ldots, g_n | \; \forall \alpha \; g_\alpha = (e + \beta_\alpha) \in \mathcal{M}\} \\ &\overset{\Downarrow}{=} e + \{\beta_1, \beta_2, \ldots, \beta_n\} \\ \mathcal{M}^{i+1}[e] &= \mathcal{F}_i(\mathcal{M}^i[g_1], \mathcal{M}^i[g_2], \ldots, \mathcal{M}^i[g_n]) \\ &\overset{\Downarrow}{=} \mathcal{F}_i(\mathcal{M}^i[e + \beta_1], \ldots, \mathcal{M}^i[e + \beta_n]) \end{aligned} \tag{1}$$

In *HUA* the same function $\mathcal{F}_i$ is **uniformly** associated to all the domain elements as well as $\mathcal{S}_i$ that is parametrically described. In $\mathcal{S}_i$ each item is defined as an **affine** translation of the element e. Considering again the Jacobi example, $\mathcal{S}_i = (x, y) + \{(0, 1), (0, -1), (1, 0), (-1, 0)\}$

The parametric definition of $\mathcal{S}_i$ leads to a strong consequence. If $\mathcal{M}$ is a limited space, $\mathcal{S}_i$ associated to the element of the space boundary contains elements that are not in $\mathcal{M}$. We therefore apply the *HUA* model **only over a toroidal space**. This is an important feature which is fundamental to define $\mathcal{Q}-transformations$. As discussed in section 6, it is possible to introduce a relaxation to the model in order to remove this restriction.

3.3 Example of a nine point stencil in HUA Model

From now on we consider, as tutorial example, a classical case of nine point stencil in a two dimensional and toroidal space of length l_x and l_y. This example can be used as a worst case stencil, in fact data dependencies require that processing nodes communicate with all their neighbours. We define $\mathcal{M} = (Z/l_x)\text{x}(Z/l_y)$ a toroidal subspace of N^2.

Representing a single step transformation of the example exploiting the *HUA* model, we obtain the following characterization:

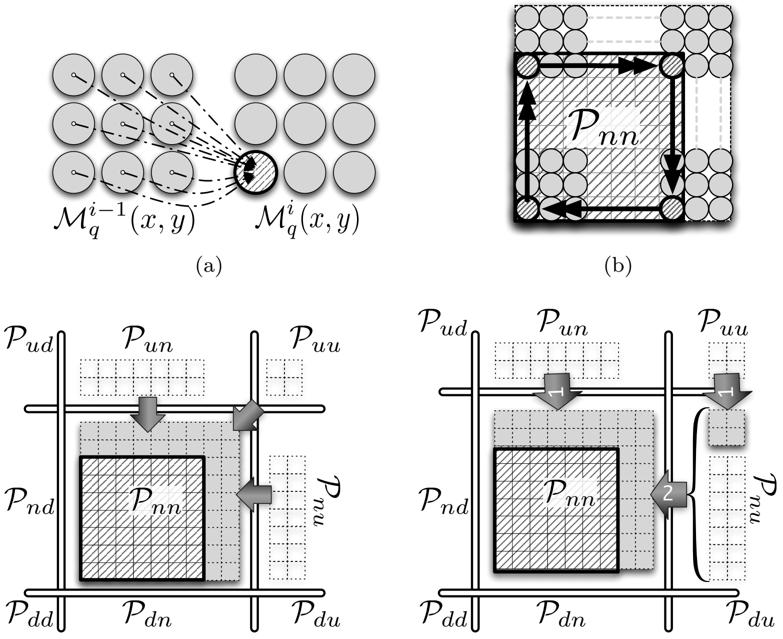
$$\begin{aligned} \forall e = (x, y) &\in \mathcal{M} \overset{step_i}{\rightarrow} (\mathcal{F}_i, \mathcal{S}_i) \\ \mathcal{S}_i &= e + \big\{(a, b) | a, b \in \{-1, 0, 1\}\big\} \\ \mathcal{M}^{i+1}[(x, y)] &= \mathcal{F}_i\big(\mathcal{M}^i[(x+1, y-1)], \mathcal{M}^i[(x+1, y)], \\ &\quad , \mathcal{M}^i[(x+1, y+1)], \ldots, \mathcal{M}^i[(x, y)]\big) \end{aligned}$$

A graphical representation in fig. 2(a) visualizes the data dependencies between two consecutive steps expressed by the model.

In $\mathcal{S}_i$, the information provided by the elements $\beta_\alpha \in \{(a, b) | a, b \in \{-1, 0, +1\}\}$ is sufficient to calculate all the dependencies of a generic partition $\mathcal{P}_{nn}$. Ideally it is enough to apply the stencil to the elements of the partition bounds as represented in 2(b).

From the dependencies knowledge between partitions (*partition dependencies*), the definition of a communication schema is straightforward.

In a *naive* implementation each step is based on the following phases: 1) the processing node sends data to the eight neighbours, 2) it computes all the



(a) (b) (c) (d)

Figure 3: Graphical representation of a q method for a nine point stencil modeled with *HUA*: 2(a) virtual processors dependencies, 2(b) dependencies between partitions of virtual processors, 2(c) computing processing communication pattern performed in the q method and 2(d) communication pattern with both q and $shift$ method; the numbers on the arrows indicate a time order on different communications

partition elements with only local dependencies, and finally 3) after receiving data form the eight neighbours, it computes what remains.

The schema of the program is simple and does not introduce any particular difficulties about managing both computation and communication, also when targeting architectures supporting their overlapping. Considering that in a regular domain the number of possible neighbours is $3^{dim} - 1$, the simplest *naive* approach features $3^{dim} - 1 = 8$ incoming and outgoing communications per step.

An optimization of the previous schema can be applied exploiting the Plimpton's *shift method*. The strength of the approach is to avoid direct communication with diagonal neighbours; all data shift along the main axes as represented in fig. 2(d). Because only two communications for each space dimension remain, the *shift* method lowers the number of communications per step from $3^{dim} - 1 = 8$ to $2 * dim = 4$.

The reduction of the number of communications is obtained by the *shift method* at the expense of program structure management. Differently from the *naive* schema, the *shift* method interleaves send and receive operations within the same stencil step. A processing node first sends along both directions of a dimensions and then receives from the same ones. This base pattern is repeated for all the space dimensions.

In the resulting program structure, overlapping of communication and computation is more difficult than with *naive* method; computations with only local dependencies has to be smartly interleaved between the send and receive operations of a base pattern.

4 $\mathcal{Q}-transformations$ in the HUA model

One of the most used techniques exploited to describe stencil computations is the *owner-computes* rule. In a stencil description with virtual processors (*VP*) abstraction, each *VP* is the owner of a domain element. The element can be read from other *VP*s but only the owner can modify it. Although *owner-computes* rule makes stencil description easier, we discover that other techniques can lead to optimizations that reduce the number of dependencies between domain partitions.

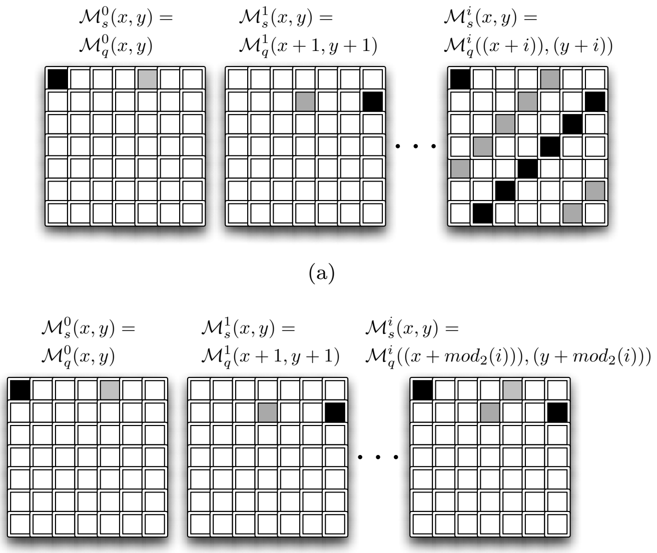
We studied $\mathcal{Q}-transformations$ to automatically transforms stencils into a new optimized form. The *HUA* model is the formalism that provide the right features to describe how the transformations work. $\mathcal{Q}-transformations$ are divided into two subclasses: positive and negative. We focus first on the positive one.

Definition 2 (**Positive** $\mathcal{Q}-transformation$)**.** *A positive* $\mathcal{Q}-transformation$ *transforms a generic stencil described with the HUA model (see eq. 1) as follows:*

$$\begin{array}{rcl}
\forall e \in \mathcal{M} & \overset{\mathcal{Q}^+}{\rightarrow} & \left(\mathcal{F}_i, \mathcal{Q}_i^+\left(\mathcal{S}_i\right)\right) \\
\mathcal{Q}_i^+(\mathcal{S}_i) & = & \mathcal{S}_i + q^+ \\
 & \overset{\Downarrow}{=} & \{g_1, \ldots, g_n|\, \forall\alpha\; g_\alpha = (e + \beta_\alpha)\} + q^+ \\
 & \overset{\Downarrow}{=} & e + \left\{\gamma_1, \gamma_2, \ldots, \gamma_n | \gamma_\alpha = \beta_\alpha + q^+\right\} \\
q^+ & = & < q_1^+, \ldots, q_{dim}^+ > \\
q_i^+ & = & min\left\{(\beta_\alpha + q^+) * \epsilon_i \geq 0 \forall\alpha\right\} \\
\mathcal{M}_Q^{i+1}[e] & = & \mathcal{F}_i(\mathcal{M}_q^i[e+\gamma_1], \ldots, \mathcal{M}_q^i[e+\gamma_n])
\end{array} \tag{2}$$

where $\epsilon = \{\epsilon_1, ...\epsilon_{dim}\}$ *is the set of the vectors in the natural basis of* N^{dim}.

With respect to the original stencil, just the set of elements S_i has been changed into $Q(S_i)$, adding to each vector element of S_i the constant vector q^+. This vector is defined in such a way that $Q(S_i)$ can be represented in the form $e + \{\gamma_1, \gamma_2, \ldots, \gamma_n | \gamma_\alpha = \beta_\alpha + q^+\}$ where the components of the generic vector γ_α are all not negative, i.e. the result of the scalar product between generic vector γ_α and one vector of the N^{dim} basis is not negative ($\gamma_\alpha * \epsilon_i =$



(a)

(b)

Figure 4: Evolution of elements location when exploiting positive $\mathcal{Q}-transformation$ 4(a) and when interleaving positive and negative $\mathcal{Q}-transformation$ 4(b)

$(\beta_\alpha + q_+) * \epsilon_i \geq 0 \forall\alpha\forall i)$. In the case of Jacobi example, we find $q^+ = (+1,+1)$, therefore $Q(S_i) = e + \{\gamma_1, \gamma_2, \ldots, \gamma_n | \gamma_\alpha = \beta_\alpha + q^+\}$
$= (x,y) + \{(+1,0),(0,+1),(+2,+1),(+1,+2)\}$

The semantic correctness of the *positive* $\mathcal{Q}-transformations$ is ensured by the following property.

Propriety 1 ($\mathcal{Q}-transformation$ correctness)**.** *Considering a generic domain space* $\mathcal{M}$*, let* $\mathcal{M}_s^i$ *be the state of the domain when applying* i *times a generic stencil* s*. Let* $\mathcal{M}_{\mathcal{Q}}^i$ *be the value of the domain when applying the stencil* $\mathcal{Q}^+(s)$*, which is the result of* $\mathcal{Q}-transformations$*.*

In a generic step i *the values of* $\mathcal{M}_{\mathcal{Q}}^i$ *are the same of* $\mathcal{M}_s^i$*, apart from a space translation.*

Proof. Let the $\mathcal{M}^0$ be the status of the domain before applying any stencil, we therefore write

$$\mathcal{M}^0 = \mathcal{M}_S^0 = \mathcal{M}_Q^0 \tag{3}$$

If we define the element $a = e + q^+$, by eq. 2 we can describe the state domain after the first step as:

$$\begin{aligned} \mathcal{M}_Q^1[e] &= \mathcal{F}_0(\mathcal{M}_Q^0[e+\gamma_1], \ldots, \mathcal{M}_Q^0[e+\gamma_\alpha]) \\ &= \mathcal{F}_0(\mathcal{M}_Q^0[a+\beta_1], \ldots, \mathcal{M}_Q^0[a+\beta_n]) \end{aligned}$$

Therefore for eq. 3 we have:

$$\mathcal{M}_Q^1[e] = \mathcal{F}_0(\mathcal{M}_S^0[a+\beta_1], \dots, \mathcal{M}_S^0[a+\beta_n])$$

At this point, according to the definition of the *HUA* model (see eq. 1), it comes that:

$$\mathcal{M}_S^1[a] = \mathcal{M}_S^1[e+q^+] = \mathcal{M}_Q^1[e]$$

Iterating the previous reasoning in various steps, we find that:

$$\mathcal{M}_S^i[a] = \mathcal{M}_S^i[e+i*q^+] = \mathcal{M}_Q^i[e] \tag{4}$$

$$\mathcal{M}_S^i[e] = \mathcal{M}_Q^i[e-i*q^+] \tag{5}$$

□

The key feature of the transformed stencil is reflected into a stronger geometric propriety of the γ_αs with respect to the β_αs: all the γ_αs present only not negative components. When calculating $\mathcal{M}_Q^{i+1}[e]$, all the element required by $\mathcal{F}_i$ are reachable from e with only positive movement in the domain space, i.e. movements along positive directions in the space dimensions. We can say that the result of the $\mathcal{Q}-transformation$ is in same way an "*orientation*" of the communication .

The benefits of this "*orientation*" are evident when moving from data dependencies to partition dependencies. In a worst case, where every processing node needs data from all its neighbours, $\mathcal{Q}-transformations$ produce an equivalent stencil where the processing nodes needs data only from those neighbours that are reachable moving along positive directions. We call this new technique q method. Therefore q method cuts down the number of communication from $3^{dim}-1$, featured by the *naive* method, to $2^{dim}-1$. The improvement is obtained without changing the structure of a step as it happens for the $shift$ method. A stencil program based on the q method does not interleave send and receive operations within the same steps as the *naive* method.

4.1 q_shift method: combination of $\mathcal{Q}$–$transformations$ and *shift* method

The $shift$ method is independent from $\mathcal{Q}-transformations$; they can be combined in what we call q_shift method. The result is the elimination of the diagonal communications from the ones featured by the q method. The communications we have to take into account are those with neighbours that can be reached with movements parallel to the space axes and only along positive directions. Therefore the q_shift method reduces the number of communications per step from $3^{dim}-1$ to dim. The combination of the q and $shift$ provide the best result with respect to all the other methods; this is evident analyzing the chart in Fig. 1.

4.2 Example of a positive $\mathcal{Q}-transformation$ on a nine point stencil

We resume the tutorial nine point stencil introduced for the *HUA* model. From the form of $\mathcal{S}_i = e + \big\{(a,b)|a,b \in \{-1,0,1\}\big\}$, it comes that $q^+ = (1,1)$. Therefore the positive $\mathcal{Q}-transformation$ transforms $\mathcal{S}_i$ in

$$\begin{aligned}
\mathcal{Q}(\mathcal{S}_i) &= \mathcal{Q}\left((x,y) + \{(a,b)\,|a,b \in \{-1,0,1\}\}\right) \\
&= (x,y) + \{(a,b)\,|a,b \in \{-1,0,1\}\} + q^+ \\
&= (x,y) + \{(a,b)\,|a,b \in \{-1,0,1\}\} + (1,1) \\
&= (x,y) + \big\{(a,b)|a,b \in \{\{-1,0,1\}+1\}\big\} \\
&= (x,y) + \big\{(a,b)|a,b \in \{\{0,1,2\}\}\big\}
\end{aligned}$$

A graphical representation about the changes on the stencil shape comes from the comparison of the original stencil in Fig. 2(a) with the transformed one in Fig 3(a). It is clear that the input data value of the function $\mathcal{F}_i$ are the same in the two cases, what changes is the position in which the resulted value is stored.

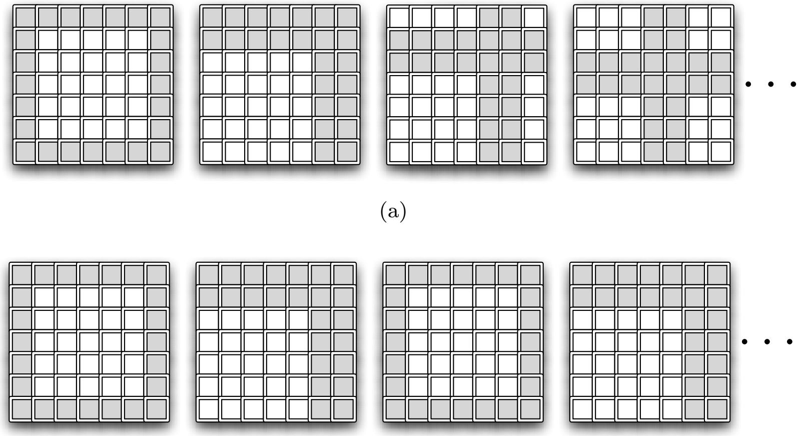
Exploiting the abstraction of virtual processors, in the original stencil, if a VP holds an element e, according to the owner-computes rule, changes to the value of e are performed only by the owner VP. In the case of the transformed stencil, the association between domain element and VPs changes at each step. If, at the beginning of a step, a VP holds the element e, at the end it holds $e + q^+$.

In other words, there are two different mappings we have to take into account. Domain elements are regularly mapped onto virtual processors which are than regularly grouped and mapped onto processing nodes. In an original stencil, all the mappings are fixed and do not change at run time. In the transformed stencil at each step, there is a constant remapping of the domain element over the VPs.

Figure 4(a) presents, in the nine point stencil example, a matrix of VPs in different steps; two domain elements are highlighted in black and gray. It is evident how they move each step; the movement is static in the sense that at each step it is possible to know where the domain element is mapped, as expressed by eq 4.

The benefits of the "orientation" of the communication introduced by $\mathcal{Q}-transformations$ are well-rendered when analyzing partitioned dependencies as presented in fig. 3(b). Shifting the stencil on elements of the partition bounds, it comes that the partition dependencies are concentrated only over two edges of the partition. Therefore only three communications are required. As illustrated previously for the $\mathcal{Q}$–$transformations$, the three neighbours are reachable from the examined partition with movement along positive directions and parallel to x or y axes.

In the *naive* and the q implementations of the nine point example, the amount of exchanged data per step is the same; this is not true in stencils modelled with the *HUA* that does not imply communications with all its neighbours. An example is the Jacobi one, where the difference of transferred data per step



(a)

(b)

Figure 5: Evolution of border elements in a non toroidal domain space: 5(a) with positive $\mathcal{Q}-transformation$, 5(b) with the interleaving of positive and negative $\mathcal{Q}-transformation$

is negligible and consist of just one domain element. We are currently working on a model to study which is the ratio between the improvement for the lower number of communications and the possible detriment triggered by a greater amount of exchanged data per step. Finally applying the q_shift method, which is the combination of both $\mathcal{Q}-transformations$ and the $shift$ technique, only two communications are required as represented in fig. 3(d). Removing diagonal communication, the data are exchanged with only those neighbours that are reachable with movements parallel to the axes and along positive directions

5 Combining Positive and Negative $\mathcal{Q}-transformations$

An important issue when exploiting q method is the configuration of the matrix at the end of the computation. The elements of the domain space are not placed in their initial (or original) positions, or also denoted as original positions, because of the shifts introduced by the $\mathcal{Q}-transformations$.

A possible solution is a sequential post processing computation to rearrange elements in their original positions according to eq. 4.

Another solution consists in exploiting the features of toroidal space; a number of steps, after the end of the computation, can be performed without modifying any value, until the elements are again in the initial positions. This approach implies the execution of a number of ”empty” steps that depends both on the lengths of the domain and on the number of steps previously performed by the application.

A powerful alternative is possible for the class of those stencils featuring a scape with a central symmetry with respect to the application point e. Most of the stencils presented in literature belong to this class, as Jacobi and the nine point stencil. The alternative solution, which requires at most one "empty" step to rearrange the domain elements, consists in the definition of a transformation featuring antithetical characteristics with respect to positive $\mathcal{Q}-transformation$.

Definition 3 (**Negative $\mathcal{Q}$–$transformation$**)**.** *A negative $\mathcal{Q}$–$transformation$ transforms a generic stencil described with the HUA model (see eq. 1) as follows:*

$$
\begin{array}{rcl}
\forall e \in \mathcal{M} & \overset{\mathcal{Q}^-}{\to} & \left(\mathcal{F}_i, \mathcal{Q}_i^-\left(\mathcal{S}_i\right)\right) \\
\mathcal{Q}_i^-(\mathcal{S}_i) & = & \mathcal{S}_i + q^- \\
 & \overset{\Downarrow}{=} & \{g_1, \ldots, g_n |\ \forall\alpha\ g_\alpha = (e + \beta_\alpha)\} + q^- \\
 & \overset{\Downarrow}{=} & e + \left\{\gamma_1^-, \gamma_2^-, \ldots, \gamma_n^- | \gamma_\alpha^- = \beta_\alpha + q^-\right\} \\
q^- & = & < q_1^-, \ldots, q_{dim}^- > \\
q_i^- & = & min\left\{(\beta_\alpha + q^-) * \epsilon_i \le 0 \forall\alpha\right\} \\
\mathcal{M}_Q^{i+1}[e] & = & \mathcal{F}_i(\mathcal{M}_q^i[e + \gamma_1^-], \ldots, \mathcal{M}_q^i[e + \gamma_n^-])
\end{array}
$$

where $\epsilon = \{\epsilon_1, ...\epsilon_{dim}\}$ is the set of the vectors in the natural basis of N^{dim}.

The key idea is the same of positive $\mathcal{Q}-transformations$, but the fundamental geometric feature of the negative ones is that all the components of the γ_α^- are not positive instead of not negative. The proof of the semantic correctness of the negative $\mathcal{Q}-transformations$ is identical to the one presented for the positive ones and leads to the following equations:

$$
\begin{array}{l}
\mathcal{M}_S^i[a] = \mathcal{M}_S^i[e + i * q^-] = \mathcal{M}_Q^i[e] \\
\mathcal{M}_S^i[e] = \mathcal{M}_Q^i[e - i * q^-]
\end{array}
\tag{6}
$$

Stencils with central symmetric shapes present the nice characteristic that $q^- = -q^+$: a component of q^- is the negative of the corresponding one of q^+. This feature can be exploited to reduce the effect of the element movements, introduced by $\mathcal{Q}-transformations$, as presented in the following property.

Propriety 2 (Interleaving Negative and Positive
$\mathcal{Q}-transformation$)**.** *Considering a generic domain space $\mathcal{M}$, let $\mathcal{M}_s^i$ be the state of the domain when applying i times a generic stencil s, featuring a central symmetry. Let then $\mathcal{M}_Q^i$ be the state of the domain when applying the stencil $\mathcal{Q}^+(s)$ on odd steps and the stencil $\mathcal{Q}^-(s)$ on even ones.*

At the beginning of odd steps all elements are in their original positions, while at the beginning of even ones all elements are translated of a quantity q^-, always with respect to their original positions.

Proof. Let the $\mathcal{M}^0$ be the state of the domain before applying any stencil, we have

$$\mathcal{M}^0 = \mathcal{M}_S^0 = \mathcal{M}_Q^0$$

At the firs step we apply the positive $\mathcal{Q}-transformation$: $\mathcal{M}_Q^1 = \mathcal{M}_{Q^+}^1$. Considering eq 5, we obtain the following equation:

$$\mathcal{M}_S^1[e] = \mathcal{M}_Q^1[e - q^+]$$

After the first step the domain elements are translated with respect to the original positions of a quantity q^-, in fact, because of the central symmetry of the stencil s, we have $-q^+ = q^-$.

At the second step the negative $\mathcal{Q}-transformation$ is applied and so from eq. 6 we get:

$$\mathcal{M}_S^2[e] = \mathcal{M}_Q^2[(e - q^+) - q^-] = \mathcal{M}_Q^2[e + q^- - q^-] = \mathcal{M}_Q^2[e]$$

The domain elements after two steps, with a positive and a negative $\mathcal{Q}-transformations$, are back in the original positions.

Iterating the reasoning, we obtain:

$$\mathcal{M}_S^i[e] = \mathcal{M}_Q^i[e + (mod_2(i) * q^-)]$$

At the end of even steps the data are in the original positions, while in odd steps they are translated of a quantity q^-. □

A graphical presentation of the Property 2 is given in fig. 4(b) for the case of the nine point stencil in a two dimensional space.

From the previous property it is clear that, exploiting the interleaving of positive and negative $\mathcal{Q}-transformations$, the domain does not need any rearranging when the application ends after an even number of steps . In the other cases, one ”void” step is sufficient.

6 Breaking the Toroidal Constrain

The constraint of toroidal domain space has been introduced to make translations exploited by $\mathcal{Q}-transformations$ always possible. More precisely, the toroidal space guarantees that, referring to eq 1 and 3, the generic element $e+\gamma_\alpha$ (which is the element e translated by q^+) is always in $\mathcal{M}$.

As previously observed, we work with two different mappings; domain elements are regularly mapped into VPs which are then regularly mapped into processing nodes.

Exploiting the two mapping feature, we can force the toroidal characteristic directly to the VP space in order to break the constraint on the working domain.

We consider that each element of the working domain is mapped into a VP and the association does not change during the applications. All the translations introduced by $\mathcal{Q}-transformations$ can be described directly with respect to VPs instead of elements. VPs are translated inside the VP space which, being toroidal, guarantees that a translated VP is still into the VP space.

With those considerations the *HUA* can model also stencils in non toroidal space domains, and all the optimizations based on $\mathcal{Q}-transformations$ can be exploited.

A peculiar feature of non toroidal working domains is the presence of bounds. Elements close to space bounds are associated to specific stencils with a restricted area. Therefore it is important to track step by step the positions of the VPs where bauds elements have been mapped, in order to apply the specific stencil. This tracking action does not introduce any problem because by Property 1 and 2, translations introduced by $\mathcal{Q}-transformations$ are know statically; it is possible to calculate where a certain VP has been moved at the end of a certain step.

Figure 5 represents the evolution of domain bounds in the case of nine point stencil.

7 Experimental Results

We considered a nine point stencil and its extension in a three dimensional space (twenty-seven points stencil) as tests for comparing the four presented methods: *naive*, $shift$, q, and q_shift.

Because we want to focus only on communication overhead, our tests consider only the time spent in send, receive and synchronization operations. As Palmer and Nieplocha [8] we exploited MPI as support for asynchronous communications.

The experiments are parametric with respect to the number of partition elements and exploit only square and cubic partitioning; each partition features the same length in all directions.

As in *HUA* stencil interactions between partitions are limited in number, in our experiments we targeted a configuration with the minimum number of processing nodes such that in a neighbour list no one node compares more than one time. This feature is guaranteed when at least three partitions are considered per dimension. We therefore worked on a grid of nine processing nodes for the two dimension case and twenty-seven nodes in the three dimensional one.

Charts in fig. 6 report test results on a eight core Intel(R) Xeon(R) CPU E5420 @ 2.50GHz exploiting the MPICH version on shared memory.

It is interesting to observe that on this chip-multiprocessor architecture, featuring intra-chip communications, the results strongly respect the forecasts that can be extracted from the chart in fig. 1.

The measured performances are characterized by the following behaviour: the method which features a lower number of communications also supports a lower communication overhead. This observation is stressed by $shift$ and q methods when passing from the two dimension case to the three dimension one. In two dimensions, q method features a number of communications lower than the $shift$ ones and this is also reflected by performance test where q presents a lower overhead. When passing to three dimensions, the difference of the number of communications is inverted and as well the performances: the $shift$ method performs a step faster than q.

A comment has to be done on the jumps featured by all the methods in both performance charts. The causes are caching effects on message copying phase; the size of the exchanged messages rises with the increasing of the number of partition elements. The jumps are experienced in advance by methods based on $\mathcal{Q}-transformations$ because, for a fixed number of partition elements, those cases feature bigger messages.

Charts 6(b) and 6(b) represent the speed up of $shift$, q and q_shift with respect to the *naive* method. Regardless the number of space dimensions, the q_shift features the best speed up for fine grain parallelization; it is more than four times faster than the *naive* in the two dimension case and nine times in the three dimensional case.

Charts in fig. 7 report communication overhead measured on dedicated thirty node cluster with Intel(R) Pentium(R) III CPU 800MHz and Ethernet Pro 100.

In the two dimensional case all the methods present a comparable overhead, but optimization based on elimination of diagonal communications, $shift$ and q_shift methods, perform worst than the other two.

In the tree dimension case the differences between the methods are relevant. Performance chart in 7(d) shows that q_shift method reaches a speed up of seventy with respect to *naive* method and more or less ten with respect to $shift$ one.

Analyzing all the previous performance results on both cluster and multi-core architectures, we can assert that optimizations based on $\mathcal{Q}-transformations$, q and q_shift methods, perform the lowest communication overhead, regardless the number of dimension space and the target architecture.

8 Conclusion and Future Works

In this paper we presented and formally proved the powerful features of $\mathcal{Q}-transformations$: a set of transformations applicable to stencils whose data dependencies can be represented in terms of affine space translations.

In stencil based parallel applications, communications represent the main overhead, especially when targeting a fine grain parallelization in order to reduce the completion time. Techniques that minimize the number and the impact of communications are clearly relevant.

We proved that the reduction of the number of communications featured by $\mathcal{Q}-transformations$ based optimizations is greater than those provided by methods presented in literature.

Moreover our experiments, both on multi-core and cluster architectures, show that implementations exploiting $\mathcal{Q}-transformations$ perform the lowest communication overhead when targeting fine grain parallelizations.

Our feature works are going to be focused on studying $\mathcal{Q}-transformation$ in combinations with other methods in literature, as tilling and $oversending$, for improvements on both communications and caching mechanisms.

References

[1] B.D. Acunto. *Computational Methods for DPE in Mechanics.* World scientific, 2004.

[2] Kaushik Datta, Mark Murphy, Vasily Volkov, Samuel Williams, Jonathan Carter, Leonid Oliker, David Patterson, John Shalf, and Katherine Yelick. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[3] C Ding and Y He. A ghost cell expansion method for reducing communications in solving pde problems. *Supercomputing*, Jan 2001.

[4] Jia Guo, Ganesh Bikshandi, Basilio B. Fraguela, and David A. Padua. Writing productive stencil codes with overlapped tiling. *Concurrency and Computation: Practice and Experience*, 21(1):25–39, 2009.

[5] Shoaib Kamil, Parry Husbands, Leonid Oliker, John Shalf, and Katherine Yelick. Impact of modern memory subsystems on cache optimizations for stencil computations. In *MSP '05: Proceedings of the 2005 workshop on Memory system performance*, pages 36–43, New York, NY, USA, 2005. ACM.

[6] K.Morton and D.Mayer. *Numerical Solution for Partial Differential Equations.* Cambridge Univ. Press, 2005.

[7] Sriram Krishnamoorthy, Muthu Baskaran, Uday Bondhugula, J. Ramanujam, Atanas Rountev, and P Sadayappan. Effective automatic parallelization of stencil computations. In *PLDI '07: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, pages 235–244, New York, NY, USA, 2007. ACM.

[8] B Palmer and J Nieplocha. Efficient algorithms for ghost cell updates on two classes of mpp architectures. *14th IASTED International Conference on Parallel and Distributed Computing and Networks*, Jan 2002.

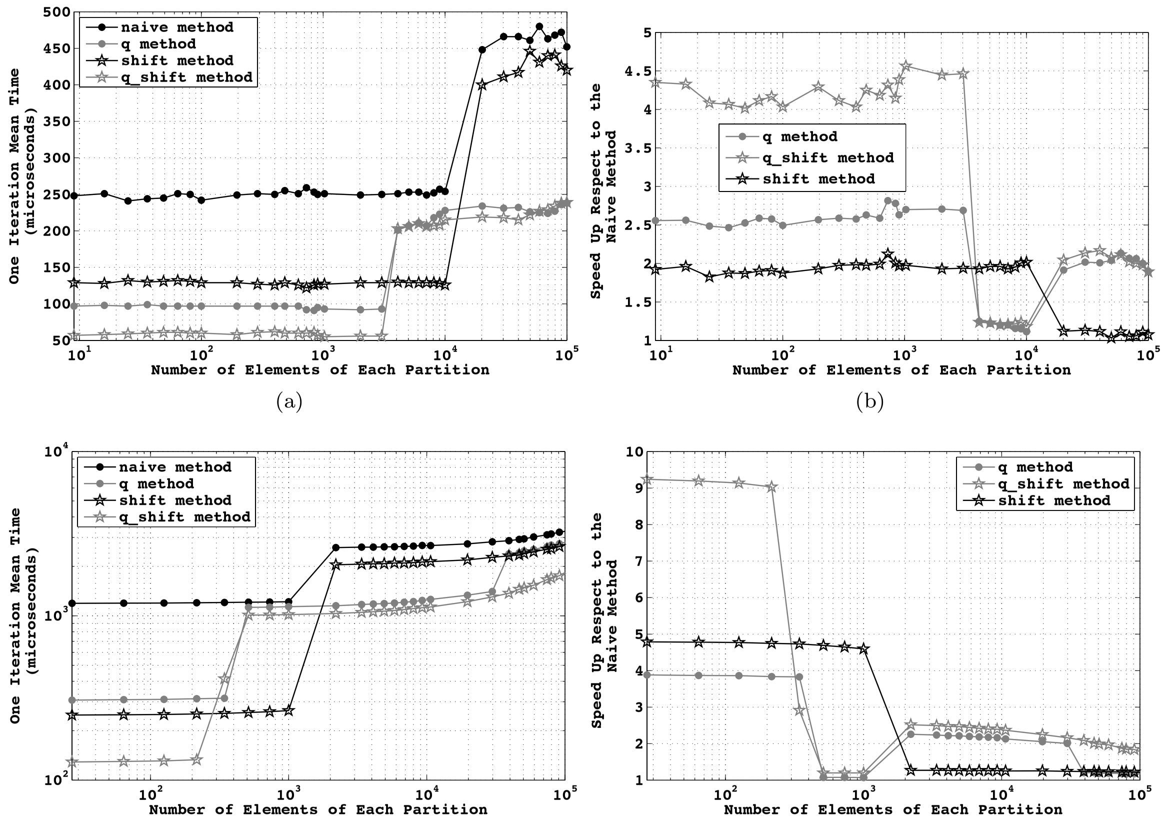
[9] Craig J. Patten, H.A. James, K.A. Hawick, and A. L. Brown. Stencil methods on distributed high performance computers. Technical report, 1997.

[10] S Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, August 2007.

[12] Lakshminarayanan Renganarayanan, Manjukumar Harthikote-Matha, Rinku Dewri, and Sanjay V. Rajopadhye. Towards optimal multi-level tiling for stencil computations. In *IPDPS*, pages 1–10, 2007.

[13] B. Wilkinson and M.Allen. *Parallel Programming.* Pearson Prentice Hall, 2005.



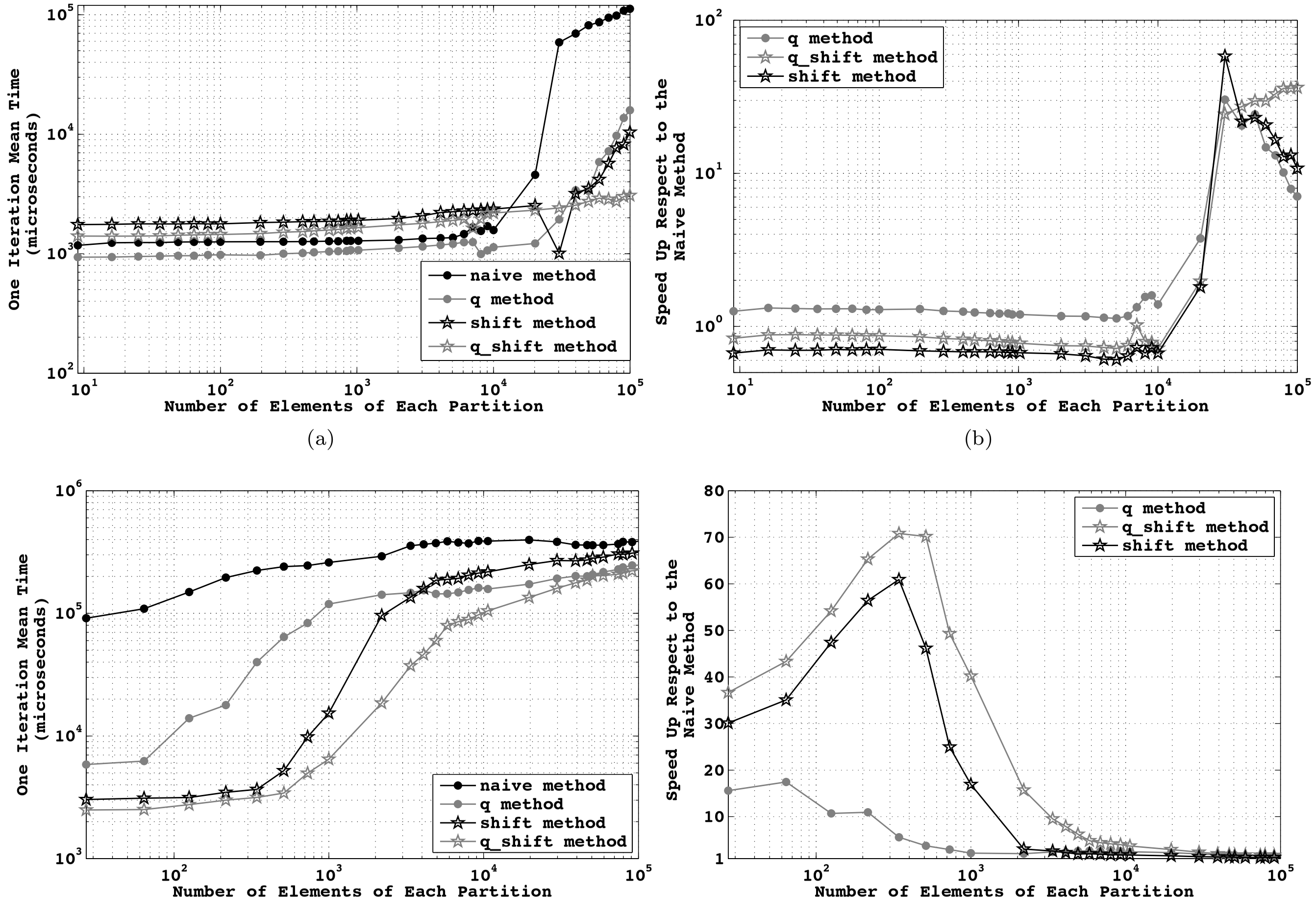
(a)

(b)

(c)

(d)

Figure 6: Nine point stencil in a two dimension space 6(a),6(b) and twenty-seven point stencil in a three dimension space 6(a),6(b) performed on top of a eight core Intel(R) Xeon(R) CPU E5420 @ 2.50GHz exploiting shared memory MPICH.



(a)

(b)

(c)

(d)

Figure 7: Nine point stencil in a two dimension space 7(a),7(b) and twenty-seven point stencil in a three dimension space 7(a),7(b) performed on dedicated thirty node cluster with Intel(R) Pentium(R) III CPU 800MHz and Ethernet Pro 100 exploiting MPICH library.