# Optimization of Seismic Imaging Code, DSFDM/FFWI (Direct Solver Finite Difference Method / Frequency Full Waveform Inversion), on a Shared Next Generation Node, MESCA II

Djibril Mboup (djibril.mboup@aims-senegal.org)
African Institute for Mathematical Sciences (AIMS)
Senegal

Supervised by: Abdou Khadr Gaye and Benjamin Pajot
ATOS GDC, Senegal and ATOS, France

26 August 2017
*Submitted in Partial Fulfillment of a Masters II at AIMS*

# Abstract

A short, abstracted description of your essay goes here. It should be about 100 words long. But write it last.

An abstract is not a summary of your essay: it's an abstraction of that. It tells the readers why they should be interested in your essay but summarises all they need to know if they read no further.

The writing style used in an abstract is like the style used in the rest of your essay: concise, clear and direct. In the rest of the essay, however, you will introduce and use technical terms. In the abstract you should avoid them in order to make the result comprehensible to all.

You may like to repeat the abstract in your mother tongue.

## Declaration

# Contents

# Introduction

# 1. Preliminary

## 1.1 Motivation and Context

The last decade has witnessed real progress in the geophysical research area. This fact is due to the development of new approach in seismic modeling such as Full Waveform Inversion (FWI), and the opportunity that offers the High Performance Computing (HPC). The rapid increasing of the HPC technologies allows to deal with large test cases with significant performance. In this insight, a group of researcher working in Geoazur laboratory and Observatory of Côte d'Azur (OCA) has developed advanced seismic imaging methods. To share the benefit of their work in the scientific community, they make in open-source distribution many of their packages. At the same time, These researchers are involved the SEISCOPE consortium sponsored by 9 companies Gas and Oil (see http://seiscope2.osug.fr) from which their work is promoted.

In this perspective, the SEISCOPE team has developed an efficient code for full waveform inversion, called DSFDM/FFWI (Direct Solver Finite Difference Method / Frequency FWI). The DSFDM/FFWI code was applied successfully in real seismic dataset collected in ocean-bottom cable data from the Valhall oil field (North Sea) in the visco-acoustic vertical transverse isotropic (VTI) approximation []. For its running, this code uses different packages especially MUMPS (MULtifrontal Massively Parallel sparse direct Solver) to solve the wave propagation equation with direct-solver method which is the most expensive part of the inversion process.

The modelling in a frequency domain is a promising method because it naturally takes into account the attenuation, a physical characteristic that can greatly impact the seismic imaging quality in certain cases, but also because of its speed of calculation for simulations with multiple right-hand side. The backwards of this method, which has prevented its expansion rapidly, is its memory consumption which limits it to medium-sized use cases.

Bull, a company of Atos group, has developed its second generation of server x86 with large shared memory, called MESCA-II (Multiple Environments on a Scalable Csi-based Architecture. Technically, a Mesca node can have 8 sockets ....

## 1.2 Geophysical Seismic Imaging

**1.2.1 Introduction.** The Earth is characterised by seismic activities. Human has been always animated by the curiosity to understand these activities. During years, human used different techniques to explore the basement depths. Recently, this exploration is more motivated by multiples reasons that can be economics, social, environment, scientific etc. For instance, the study of the nature the soil is useful fol civil engineers, landslide imaging is very useful for estimating the risk of gravity, the detection of oil and mineral resources present a major economic challenge, the monitoring of radioactive waste landfill areas or CO2 injection represents an new environmental challenge.

With the advents of Industrial revolution accompanied by the higher need to find new oilfield by hydrocarbon industry, the seismic imaging technique has successfully used from the mid of 19th. By definition, Seismic imaging is a tool that bounces sound waves off underground rock structures to reveal the possible tectonic structures []. In seismic modelling, one need data coming from seismic wave propagation to create good quality images of the subsurface (imaged by acoustic waves). The acoustic waves are generated under the Earth by sources that can be man-made devices or by earthquakes. Receivers or seismometers acquire information that can provide details of the velocity and the geometric structure of the Earth.

During years scientists invent several techniques of seismic imaging which include Electrical Resistivity Tomography, Ground penetrating radar, Induced polarization and Seismic Tomography and Reflection seismology.

- **Electrical Resistivity Tomography (ERT)**. The ERT technique is invented by Schlumberger brothers and developed by the work of Andrey Nikolayevich Tikhonov []. This method is an electrical testing method where electrical current is induced in the ground between one pair of electrodes and the voltage is measured between another pair. These measurements are used to estimate lateral and vertical variations of resistivity values of the earth. ERT can be used to map geologic variations (soil lithology, presence of ground water, fracture zones, variations in soil saturation, areas of increased salinity or, in some cases, ground water contamination), bedrock depths and geometry, mapping cavities such as caves, karst and/or evaporite dissolution sinkholes. Like other seismic technique, ERT has the capacity to yield either 1D (sounding), 2D (profile) or 3D (volume) imaging.

- **Ground Penetrating Radar (GPR)**. GPR systems work by sending a tiny pulse of energy into a material via an antenna. An integrated computer records the strength and time required for the return of any reflected signals. Subsurface variations will create reflections that are picked up by the system and stored on digital media []. This technique has many applications: civil engineering, resources explorations.

- **Induced Polarization (IP)**. Similar to electrical resistivity tomography, this technique measure the electrical chargeability of subsurface materials. IP provide additional information about the spatial variation in lithology and grain-surface chemistry. It can be made in time-domain and in frequency-domain. IP method is widely used in mineral exploration and mining industry and it has other applications in hydrogeophysical surveys, environmental investigations and geotechnical engineering projects.

- **Reflection Seismology (RF)**. This is the most wide used technique when we talk about hydrocarbon exploration, or mineral exploration. RF is a form of echo sounding, detecting echoes from seismic interfaces. It can yield results that are the closest of any geophysical technique to a conventional geological section. The waves propagation under the Earth require to know some features of the wave: transmission, absorption, and attenuation in the earth materials and its reflection, refraction, and diffraction characteristics at discontinuities.

  In the context of onshore, geophysicists use geophones as receivers to collect the speeds of particles on the ground. Accelerometer sensors can be used to record these data. These sensors are directional and measure the speed or acceleration in a spatial direction. Sometimes, devices with multi-components are used to measure wavefields in the horizontal directions (parallel to the Earth's surface) and vertical direction, thus identifying the different types of waves whose depend on the directions []. The figure 1.1 shows the scenario of ....

  In the off-shoring scenario, the hydrophones, placed on the surface of the sea, are often used to collect the seismic wavefields information. In the figure 1.2, a ship generates the seismic source by an air gun and drag behind, under the water surface, a raw of hydrophones measuring the pressure field which indicates the waves propagated in the structure of the subsoil. Others sensors named *Ocean Bottom Seismometer* (OBS) are designed to record the earth motion under oceans

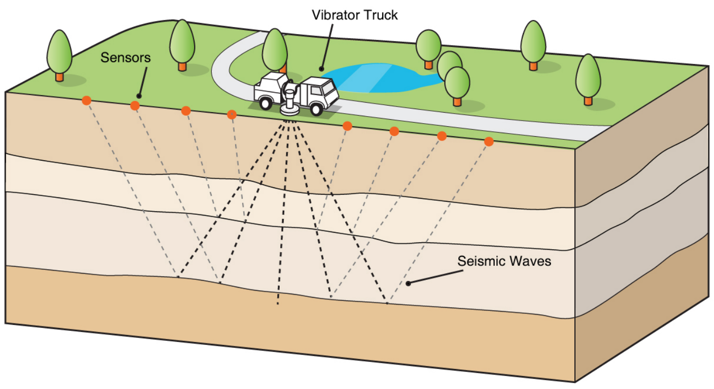


Figure 1.1: Seismic imaging.

and lakes from man-made sources. These devices can stay several days under the water. Each OBS receives different acoustic waves for each generation of sound arriving at different times. There exists another device called Ocean Bottom Cable (OBC).The data acquisition is performed by multi-component sensors bound on the cables.

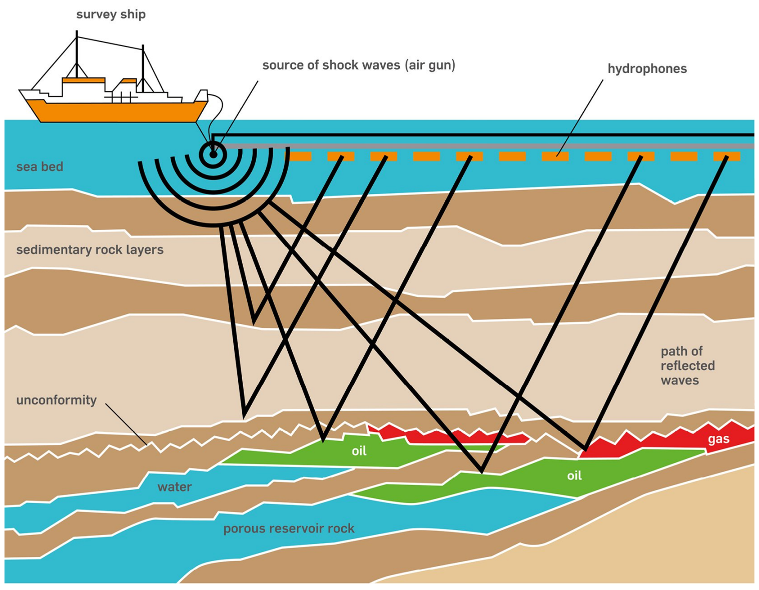


Figure 1.2: Seismic imaging.

**1.2.2 Seismic model.** These techniques need numerical methods to solve the seismic imaging problem. In this study, we just focus on Frequency-domain seismic modeling based on the sparse direct solver (DSFDM)and Full Wave Inversion method used by Seiscope Consortium to develop the DSFDM/FWI code.

The DSFDM is a finite-difference frequency-domain method, which has aimed to solve linear systems resulting from the discretization of the time-harmonic wave equation with sparse direct solvers. This method with 27 stencils has developed in a visco-accoustic vertical transverse isotropic (VTI) [Operto et al. (2007), Brossier et al. (2010), Operto et al. (2009) and Operto et al. (2014)]. It is governed by the linear systems resulting by 3D visco-acoustic VTI wave equation:

$$Ap_h = b \tag{1.2.1}$$

$$p_v = Bp_h + s^{'}, \tag{1.2.2}$$

$$p = (2p_h + p_v)/3 \tag{1.2.3}$$

where the matrices $A$ and $B$ result from the discretization of operators.

$$A = \omega^2 \left[ \frac{\omega^2}{\kappa_0} + (1+2\epsilon)(\mathcal{X} + \mathcal{Y}) + \sqrt{1+2\delta}\mathcal{Z}\frac{1}{\sqrt{1+2\delta}} \right] \tag{1.2.4}$$

$$B = \frac{1}{\sqrt{1+2\delta}} + \frac{2(\epsilon - \delta)\kappa_0}{\omega^2\sqrt{1+2\delta}}(\mathcal{X} + \mathcal{Y}) \tag{1.2.5}$$

Similarly, the source terms are given by:

$$b = \frac{\omega^4 s_h}{\kappa_0}s - \omega^2\sqrt{1+2\delta}\mathcal{Z}\left(s_v - \frac{1}{\sqrt{1+2\delta}}s_h\right)s, \tag{1.2.6}$$

$$s^{'} = \left(s_v - \frac{1}{\sqrt{1+2\delta}}s_h\right)s \tag{1.2.7}$$

The wavefields $p_h = \sigma_{xx} = \sigma_{yy}$ and $p_v = \sigma_{zz}$ are the so-called horizontal and vertical pressure wavefields, respectively (Plessix and Cao, 2011). The angular frequency is denoted by $\omega$, $\kappa_0 = \rho V_0^2$ where $\rho$ is the density and $V_0$ is the vertical wavespeed, $\delta$ and $\epsilon$ are the Thomsen's parameters. The seismic source vector is compactly denoted by $s$. For an explosion and vertical force, the expression of the source vector is given by

$$s_{explosion} = s(\omega)\hat{\delta}(x_s - x) \tag{1.2.8}$$

$$s_{vertical_force} = s(\omega)d_z\hat{\delta}(x_s - x) \tag{1.2.9}$$

where $s(\omega)$ denotes the Fourier coefficient of the temporal source wavelet for the modeled frequency, $d_z$ the derivative with respect to $z$,$\hat{\delta}$ the delta function, $x_s$ the coordinates of the source position.

The $s_h$ and the $s_v$terms are given by the following formulas.

$$s_h = (2(2+\epsilon) + \sqrt{1+2\delta})/D, \tag{1.2.10}$$

$$s_v = (1 + 2\sqrt{1+2\delta})/D, \tag{1.2.11}$$

$$D = 4\sqrt{1+2\epsilon} + 4\sqrt{1+2\delta} + 1 \tag{1.2.12}$$

The expression of $D$ was corrected relative to the one provided in Operto et al. (2014).
The second-order differential operators $\mathcal{X}$ , $\mathcal{Y}$ and $\mathcal{Z}$ are given by

$$\mathcal{X} = \partial_{\tilde{x}} b \partial_{\tilde{x}}, \mathcal{Y} = \partial_{\tilde{y}} b \partial_{\tilde{y}}, \mathcal{Z} = \partial_{\tilde{z}} b \partial_{\tilde{z}},$$

where $b = 1/\rho$ is the buoyancy and the complex-valued coordinate system $(\tilde{x}, \tilde{y}, \tilde{z})$ is used to implement perfectly-matched layers absorbing boundary conditions (BÃ©renger, 1994; Operto et al., 2007). If we assume that $\delta$ is slowly varying and hence can be considered as locally homogeneous, the operator $A$ can be simplified as

$$A_a = \omega^2 \left[ \frac{\omega^2}{\kappa_0} + (1+2\epsilon)(\mathcal{X}+\mathcal{Y}) + \mathcal{Z} \right] 2\mathcal{Z}\kappa_0(\epsilon-\delta)(\mathcal{X}+\mathcal{Y}) \tag{1.2.13}$$

In this case, the wave operator within the bracket describes wave propagation in an elliptic medium, while the second term accounts for anellipticity.

In *elliptic* media $(\delta = \epsilon \neq 0)$, the source become

$$b = \frac{\omega^4 s_h}{\kappa_0} s \tag{1.2.14}$$

$$s^{'} = 0 \tag{1.2.15}$$

The authors mention this point because the second term of the source $b$, equation 1.2.6, involves a second-order vertical derivative which can be tricky to implement if the source does not match the grid point. Therefore, it is worth assuming that the medium is elliptic at the source positions if this is a reasonable assumption.

In isotropic media $(\delta = \epsilon = 0)$, one can easily check that Equations 1.2.1-1.2.3 reduces to

$$\left[ \frac{\omega^2}{\kappa} \mathcal{X} + \mathcal{Y} + \mathcal{Z} \right] p = \frac{\omega^2}{\kappa} s \tag{1.2.16}$$

where the operator $\left[ \frac{\omega^2}{\kappa} \mathcal{X} + \mathcal{Y} + \mathcal{Z} \right]$ represents the operator $A_{iso}$

Similarly, the operator $A$ can be rewritten considering the NMO velocity $(V_{NMO} = V_0\sqrt{1+2\delta})$ in the diagonal term. This introduces the $\eta = \frac{\epsilon-\delta}{1+2\delta}$ parameter in the coefficients of the matrix.

$$A_{NMO} = \omega^2 \left[ \frac{\omega^2}{\kappa_{NMO}} + (1+2\eta)(\mathcal{X}+\mathcal{Y}) + \sqrt{1+2\delta}\mathcal{Z}\frac{1}{\sqrt{1+2\delta}} \right] + 2\sqrt{1+2\delta}\mathcal{Z}\frac{\kappa_{NMO}}{\sqrt{1+2\delta}}(\mathcal{X}+\mathcal{Y}) \tag{1.2.17}$$

**Algorithm.**
The system of three equations 1.2.1-1.2.3 is solved with the following sequence.

- Solve the linear system, equation 1.2.1, for $p_h$ with a sparse direct solver.
- Compute explicitly $p_v$ from $p_h$, equation 1.2.2.
- Compute explicitly $p$ from $p_h$ and $p_v$, equation 1.2.3.

The computational cost of the last two steps is negligible relative to that of the first step.

Note that wave-equation operators 1.2.4, 1.2.13, 1.2.16 and 1.2.17 are implemented in DSFDM code. (See Documentation)

**1.2.3 Full Wave Inversion (FWI).** The Full Wave Inversion is a non linear-data fitting method that enable to get detailed estimates of subsurface properties from seismic data, which can be the result of either passive or active seismic experiments []. Basically, initialising a guess of the subsurface parameters (a model), the data are predicted from the solution of the wave propagation equation. And then, the model is always updated to minimize the misfit function between the observed and predicted data. The figure 1.3 illustrates the FWI procedure. (1.2.1)

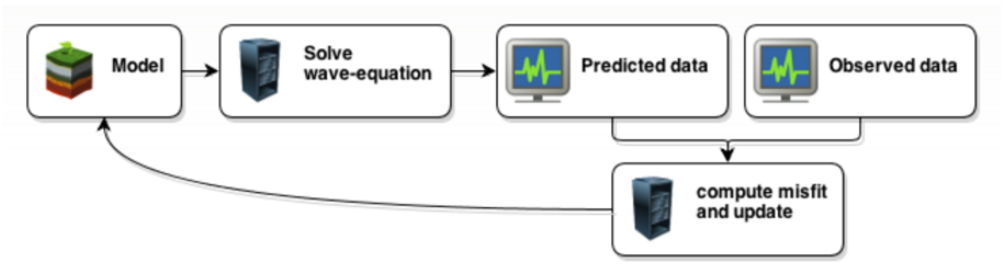


Figure 1.3: Full Wave Inversion (FWI) procedure. from []

Mathematically, the FFWI can be formulated as an optimization problem. In [], Virieux, J. and S. Operto give the main theoretical aspects of FWI based on a least-squares local optimization approach. In a data-domain frequency-domain, FWI is an iterative reduction of the misfit function C defined as the least-squares norm of the difference between recorded and modelled monochromatic pressure data, $d_{obs}$ and $d(m)$, respectively.

$$\min_m C(m) = \min_m \sum_\omega \|d_{obs} - d(m)\|$$

The subsurface model updated at iteration $k+1$ is given by:

$$m_{k+1} = m_k - \gamma_k \mathbf{H}_k \nabla_m C_k$$

where $\mathbf{H}_k \nabla_m C_k$,the direction of gradient is given by the product of the gradient of C by an approximation of the inverse Hessian, H. The step length $\gamma_k$ defines quantity of descent. The descent direction can be preconditioned by a diagonal approximation of the Hessian, in our case, the diagonal elements of the so-called pseudo-Hessian matrix (Shin et al., 2001), the aim of which is to balance the gradient amplitudes with respect to depth by removing geometrical spreading effects.

The expression of the gradient preconditioner is given by:

$$H = 1/[P + \epsilon max(P)], \tag{1.2.18}$$

where $P = diag\Re\left\{(\frac{\partial A}{\partial m}p_h)^\dagger, (\frac{\partial A}{\partial m}p_h)\right\}$ , $\frac{\partial A}{\partial m}p_h$ are the so-called virtual sources (Pratt et al., 1998) and the damping factor $\epsilon$ should be chosen with care to balance properly in depth the gradient without generating instabilities. For more details see Virieux, J. and S. Operto 2009, An overview of full-waveform inversion in exploration geophysics. Geophysics 74(6),. The FFWI code is implemented with SEISCOPE optimization toolbox which contains different optimization algorithms (steepest-descent, conjugate gradient, l-BFGS, truncated Newton) including the line search [].

Figure 1.4: Full Wave Inversion (FWI) procedure. from []

**1.2.4 Application: Valhall.** The Seismic imaging was applied in Vallhal dataset with Full Waveform Inversion method in the context of SEISCOPE project (by Operto et al.,by Operto et al.). The Valhall oil field is an offshore field located in North Sea in 70 m of water (see figure 1.4). In this oilfield, the presence of gas cloud is observed in the overburden. The reservoir is approximatively located in 2.5 km depth (Barkved et al. 2010). The data acquisition was making possible by wide aperture/azimuth acquisition covering a surface of 145 $km^2$. A layer of 12 cables contains 2302 hydrophones, which record 49,954 explosive sources located 5 m below the surface of the water (Figure 1.9.a). A distance of 300 m separate two cables except two outer cables at 600 m. The figure 1.9.a shows the gas cloud intersecting the cables (area covered by the 50,000 explosive sources) and the receiver position, whose records are shown in panel. In figure 1.9.b, the solid black arrow indicate precritical reflection from the reservoir, the dashed black arrow points to the critical and postcritical reflection, and the white arrow points to the reflection from the top of the gas [].

In the study of Operto et al., a case study of valhall data set shows approximately the presence of gas at 2.5km in depth []. With initial condition $V_0$, vertical-velocity, $\epsilon$ and $\delta$, Thomsen's parameters, and the density $\rho = -0.0261V_0^2 + 0.373V_0 + 1.458$ are used to apply frequency-domain FWI on the OBC data set []. With discrete frequencies chosen in the 3.5-10 Hz frequency band, the DSFDM/FWI code was performed in computer nodes that are equipped with two 2.5 GHz Intel Xeon IvyBridge E5-2670v2 processors with 10 cores per processor. the shared memory per node is 64 GB. The connecting network is InfiniBand fourth data rate (FDR) at 56 Gb/s. The operations are performed in single-precision complex arithmetic, for which the peak performance of the machine is 10 Gflops/s/core (which corresponds to a double precision peak of 20 Gflops/s/core). Figures 3.9a 3.9b show the 2D and 3D visualisations in different levels of the reservoir of the gas cloud. According to the authors, this study reveals memory consumption in the step of LU factorisation. They showed that the complexity can reach $O(n^6)$ [].

## 1.3 High Performance Computing (HPC)

**1.3.1 Introduction to HPC.** Supercomputing has developed early in the 1970s with CDC 6600, ILLIAC-IV and the generation of CRAY. The high performance computing (HPC) was born in context of scientific computing. In fact, It was a challenge to be fast and more precise when were talking about scientific computing such as simulation purposes like weather forecasting, numerical mechanics for car crash tests, financial market prediction or other various complex phenomena modelings. The HCP's development

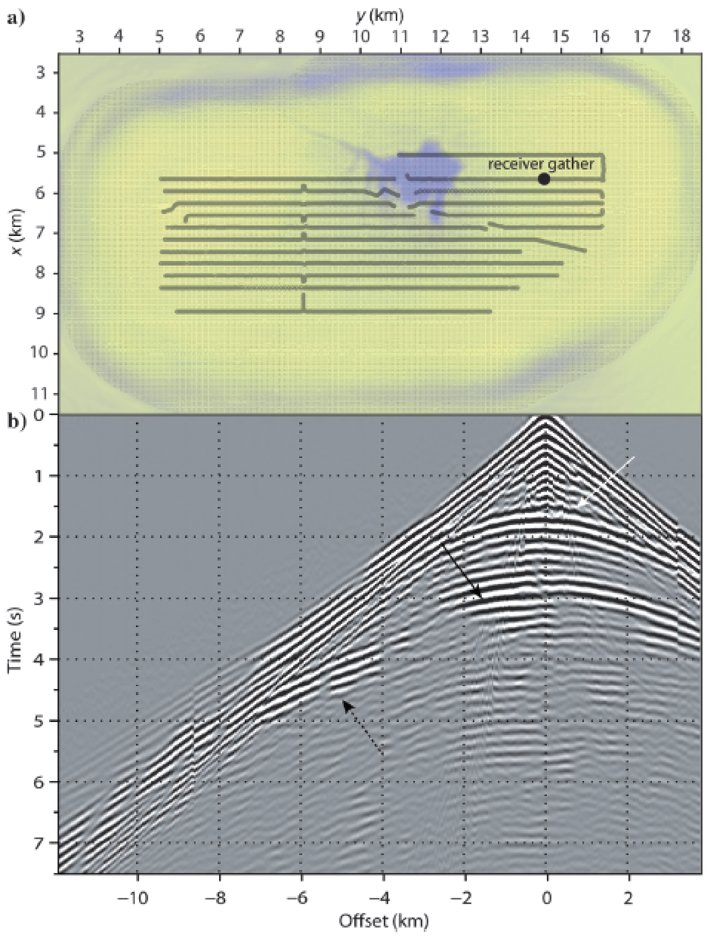
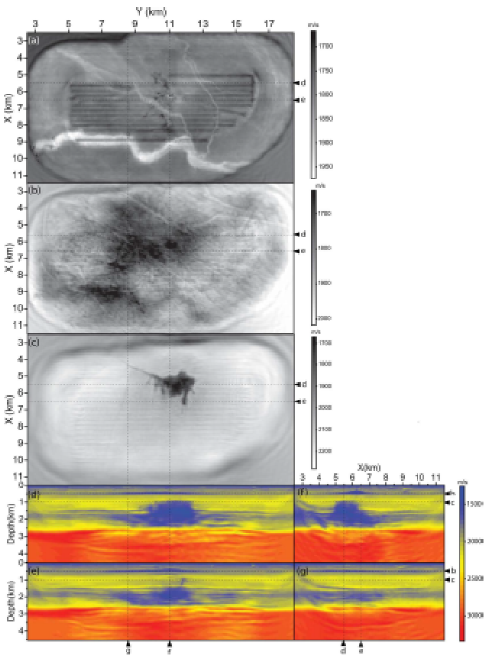
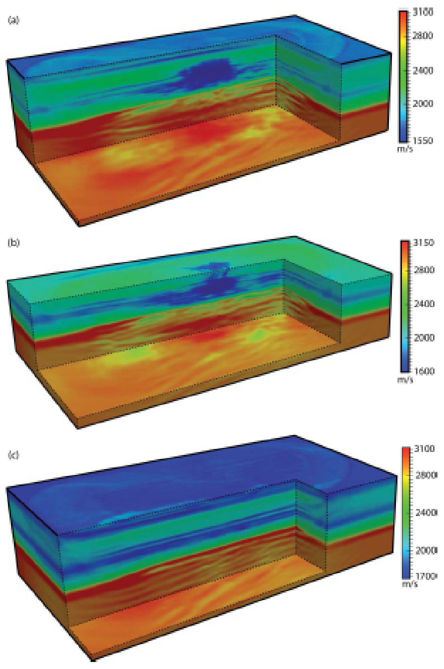


Figure 1.5: Full Wave Inversion (FWI) procedure. from []

was also accompanied with new paradigm in the computer design in hardware and software levels. It takes more advantages in the concepts of parallel system and parallel program, shared and distributed memory. Thus, the high performance computers as machines with a good balance among the following major elements []:

- Multistaged (pipelined) functional units.
- Multiple central processing units (CPUs) (parallel machines).
- Multiple cores.
- Fast central registers.
- Very large, fast memories.
- Very fast communication among functional units.
- Vector, video, or array processors.
- Software that integrates the above effectively.

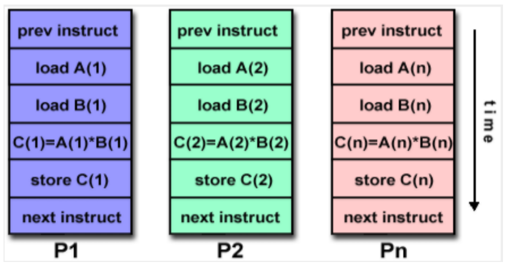

(a) Picture 1

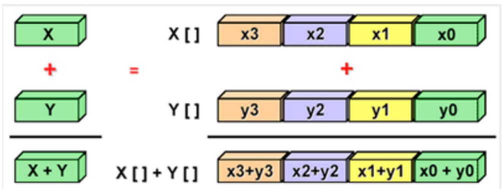

(b) Picture 2

Today, with Moore's Laws, the performance of computer is continuously increasing. In 2017, the Top500[1], website ranking the list of the top 500 supercomputers, classifies the Sunway TaihuLight (a system developed by *China's National Research Center of Parallel Computer Engineering & Technology* (NRCPC)) to be the most power computer in the world. The Sunway TaihuLight has a note of 93 Petaflops in Linpack Performance, where a Petaflops is $10^{15}$ floating-point operations per second. Recently, the Atos group has launched a new generation of quantum learning machines that can reach 24 TB of memory and 16 CPU, with power of from 30 to 40 Qubits (see 1.7). These computers allows researchers, engineers and students to develop and experiment with quantum software.

Figure 1.7: ATOS quantum learning machine

[1]https://www.top500.org





!h (a) Picture 1 (b) Picture 2

**1.3.2 HPC architecture.** Several effort has been doing in single processor performance by the ever-increasing density of transistors-the electronic switches-on integrated. The small dimension of transistors in an integrated circuit has increased the speed of this circuit. Therefore, this fact implies the increase of the heat of these transistors due to the power consumption. However, the air-cooled integrated circuits are reaching the limits of their ability to dissipate heat, in the first decade of the twenty-first century[]. Therefore, it is not sure to continue to increase the speed of integrated circuits. The chip's industries have another option instead to spend much money for building ever-faster, more complex, monolithic processors. They decide to put multiple, relatively simple, complete processors on a single chip, called multicore[]. Each core represents a CPU. Now, Programmers have to take into account the parallel architecture to be more efficient and more faster. In the parallel systems, different concepts are used : SIMD (*Single Instruction, Multiple Data*), MIMD (*Multiple Instruction, Multiple Data*), Shared Memory, Distributed Memory, Interconnection networks

- **SIMD**: A concept of Flynn's taxonomy, SIMD indicates the fact that multiple processing units (core) execute the same instruction on multiple data streams. Each processing unit can perform on different data elements. The figure 1.8a gives an example of SIMD execution on three CPU.The SIMD is used for specialized problems characterized by a high degree of regularity, such as graphics/image processing (GPUs). In addition, SIMD is used in the architecture of vector processors which enable to vectorize loop in matrix calculation (see figure 1.8b). Example of SIMD Processor Arrays and Vector Pipelines : Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV, IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10.

- **MIMD** As the name suggests, MIMD or multiple instruction streams on multiple processors (cores) operate on different data items concurrently. MIMD systems typically consist of a collection of fully independent processing units or cores, each of which has its own control and logical unit []. The execution process can be synchronous or asynchronous, deterministic or non-deterministic []. The MIMD architecture is found in most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs. The figure shows an example of MIMD execution.

- **Shared Memory** A shared-memory system is a collection of autonomous processors (cores) connected to memory via an interconnection network, and processors share physical address of memory []. There are two different memory access in shared-memory system: Uniform Memory Access (UMA) and cache-coherent Non-uniform Memory Access (ccNUMA).

  The UMA can be considered a 'flat' memory model, because latency and an bandwith are the same for all cpu and all memory location. Sometimes called Symmetric Multiprocessor (SMP), UMA system enables to have equal access and equal time to memory. The general problem of UMA machines is that bandwidth bottlenecks are bound to occur when the number of sockets is larger

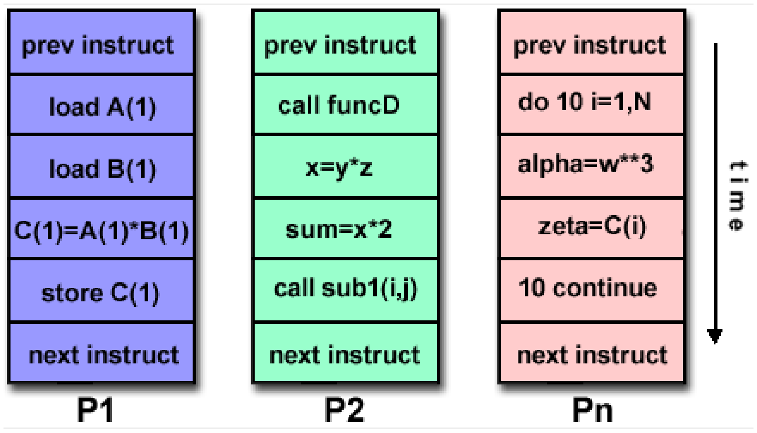
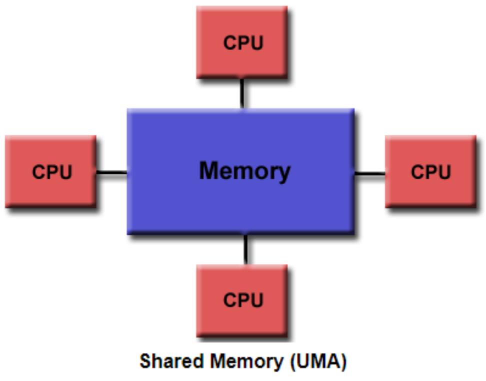


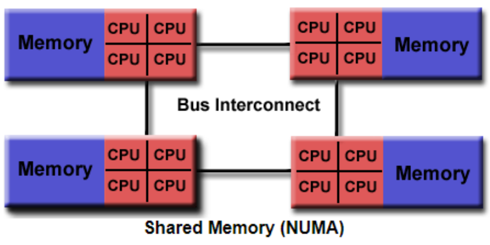
Figure 1.9: Picture []

than a certain limit []. The figure 1.10a show a representation of UMA architecture. In ccNUMA, memory is physically distributed but logically shared (see figure 1.10b). This architecture seems to be distrubuted-memory system, but network logic makes the aggregated memory of the whole system appear as one single address space []. This simplify the memory access without resorting to a network of any kind. The locality domain (LD), set of processor cores connected locally to memory,can be considered as a UMA "building block". The ccNUMA principle gives scalable bandwidth for very large processor counts. It can be inexpensive small for two- or four-socket nodes frequently used for HPC clustering [].

The LD can be, sometimes, an obstacle for high performance software on ccNUMA. The second problem is potential contention if two processors from different locality domains access memory in the same locality domain, fighting for memory bandwidth. Both problems can be solved by carefully observing the data access patterns of an application and restricting data access of each processor to its own locality domain [].



(a)



(b)

- **Distributed Memory** In the distributed-memory system, each processor is connected to exclusive local memory and network communication connect also inter-processor memory (see 1.13). The memory is scalable with the number of processor and we are a rapid access of each processor with its own memory. However, the programmer must explicitly define how to access data when a processor need it in another one. Besides, the programmer is also responsible to handle the synchronization task [].

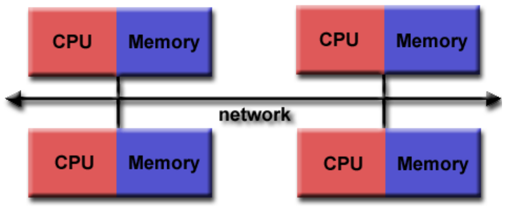


Figure 1.11: Distributed Memory

According to [], there are actually no distributed-memory systems any more that implement such a layout where we HPC clustering. Most of parallel systems couple at same time shared and distributed systems (Hybrid Systems) i.e. there are shared-memory building blocks connected via a fast network. This Hybrid Distributed-Shared Memory make the system to take advantages of the two architectures and the increase of the scalability. The figure ?? shows an Hybrid architecture with CPU and GPU (*Graphical Processing Unit*) cores.

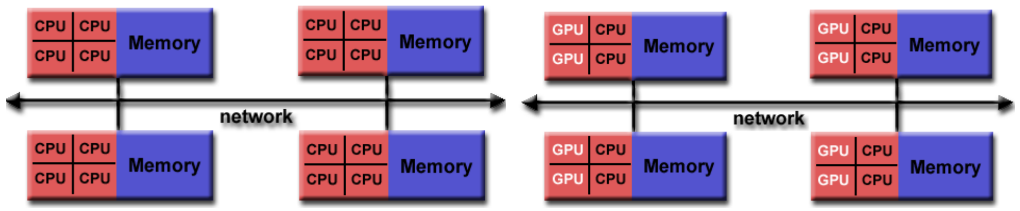


Figure 1.12: Hybrid System with CPU/GPU cores

- **Interconnection networks** In HPC domain, the interconnect plays a crucial role for latency and bandwidth performance of both distributed- and shared-memory systems.

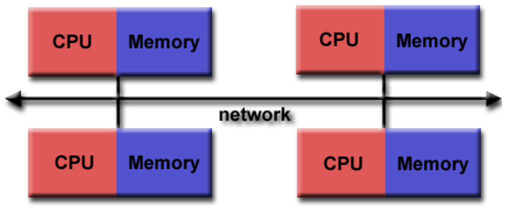


Figure 1.13: Interconnected Networks with different topologies

### 1.3.3 Parallel Programming.

- **Speedup**
- **Amdhal's Law**
- **Gustafson's Law**
- **Scalability**

### 1.3.4 Shared Memory Parallel Programming: OpenMP.

### 1.3.5 Distributed Memory Parallel Programming: MPI.

### 1.3.6 Challenges of HPC in the context of 3D Seismic Imaging.

# 2. Optimization Techniques in a Parallel System

## 2.1 Compiler Optimization

## 2.2 Threading/Multi-threading

## 2.3 Vectorization

## 2.4 Placement of MPI processes

## 2.5 Code Profiling

## 2.6 Optimization Libraries

## 2.7 Hardware Accelerators

# 3. Optimization of DSFDM/FFWI code in shared node, MESCA II

## 3.1 DSFDM/FFWI

**3.1.1 Description.** The DSFDM/FFWI is a package which contains parallel computer codes to perform seismic modeling and fullwaveform inversion in the frequency domain. These codes have have been designed to execute modeling/inversion in 1, 2, 3 dimension media. According to the authors, only the 3D version is tested and Documented [doc DSfDm]. The DSFDM code is the seismic modeling code, while the FFWI code performs the inversion, using most of the subroutines implemented in DSFDM.Moreover,the DSFDM/FFWI codes assume visco-acoustic submarine supports in which vertical transverse isotropy can be taken into account.

Seismic modeling is performed in the frequency domain with a finite-difference method on a uniform Cartesian grid ([Operto et al., 2007][Brossier et al., 2010]; [Operto et al., 2014]). The linear system resulting from the discretization of the time-harmonic wave equation is solved with a sparse direct solver. Sources and receivers, which can be processed in a reciprocal way, can be considered at arbitrary positions in coarse finite-difference grids with the sinc parameterization developed by Hicks (Hicks, 2002). Absorbing boundary conditions are perfectly-matched layers (PMLs) ([Bérenger, 1994]; [Operto et al., 2007]). A free-surface boundary condition can be used along arbitrary tomography by forcing the pressure wave-field to 0 along this boundary. All of the tasks performed during seismic modeling (building impedance matrix, building right-hand side vectors, call of MUMPS subroutines, writing of solutions and extraction at receiver solutions) are implemented in the DSFDM code.

FFWI code is implemented with various local optimization methods. Recorded data are provided in the frequency-domain using an adaptation of the Seismic Unix (SU) format. The data are complex-valued (in single precision) and the frequency interval is uniform. The first frequency, the number of frequency and the frequency interval must be provided at specific locations in the SU headers. The numerical optimization is performed with the SEISCOPE optimization toolbox and allows for steepest-descent, conjugate gradient,l-BFGS and truncated Newton optimizations ([Métivier and Brossier, 2016]). Seismic modeling and inversion are performed on the same Cartesian grid. Source wavelet estimation can be performed at each iteration of the FWI. Only the preconditioned steepest-descent optimization was tested, although conjugate gradient and l-BFGS optimizations are interfaced with the FFWI code. Interfacing Gauss-Newton and Newton truncated optimizations is the aim of ongoing work. The code has been tested for the update of the vertical wavespeed only. An application to real data is shown in [Operto et al., 2015]. Multi-parameter gradients are implemented but still need to be validated.

## 3.2 MUMPS Solver

MUMPS ("MUltifrontal Massively Parallel Solver") is a package for solving systems of linear equations of the form $Ax = b$, where $A$ is a square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric, on distributed memory computers. The MUMPS package is designed to solve linear equations by using a direct method based on a multifrontal approach which performs a Gaussian factorization

$$A = LU$$

where $L$ is a lower triangular matrix and $U$ an upper triangular matrix. If the matrix is symmetric then the factorization

$$A = LDL^T$$

where D is block diagonal matrix with blocks of order 1 or 2 on the diagonal is performed.

MUMPS solvers the result of collaboration between different partners working in the MUMPS project since 1996. Todays, the MUMPS team include CERFACS, CNRS, ENS-Lyon, INRIA, INPT, University of Bordeaux. In the recent years, the popularity of MUMPS package has increased in different countries and different scientific projects. It has known different version during the years with real improvement of the LU factorization methods. The recent research of MUMPS project is focused on the Block-Low rank option that allows decreasing the complexity of sparse direct solvers on problems arising from partial differential equations is provided for experimentation purpose.

A matrix A of size $m \times n$ is said to be low rank if it can be approximated by a low-rank product $\tilde{A} = XY^T$ of rank $k_\epsilon$, such that $k_\epsilon(m+n) \leq mn$ and $\|\tilde{A} - A\| \leq k_\epsilon$. The first condition states that the low-rank form of the matrix requires less storage than the standard form, whereas the second condition simply states that the approximation is of good enough accuracy. Using the low-rank form also allows for a reduction of the number of floating-point operations performed in many kernels (e.g., matrix-matrix multiplication).

Thanks to the low-rank compression, the theoretical complexity of the factorization is reduced from $O(n^6)$ to $O(n^{5.5})$ and can be further reduced to $O(n^5 \log 5)$ with the best variant of the BLR format ([Amestoy et al., 2016b]).

## 3.3 Tools

As we mentioned above, this DSFDM/FFWI version use MUMPS package to solve the linear system of discretized wave propagation equation. We need to install different packages for MUMPS dependencies as well as DSFDM/FFWI. MUMPS can be installed in sequential version. In the same way, it can be used with multithreaded machine (with OpenMP) or with parallel version (distributed memory MPI based). We need these packages for MUMPS installation:

- **BLAS library**: BLAS (*Basic Linear Algebra Subprograms*) are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations. BLAS is written in Fortan and it has an optimized sequential or multithreaded version.
- LAPACK library: LAPACK or Linear Algebra PACKage routines are written so that as much as possible of the computation is performed by calls to the Basic Linear Algebra Subprograms (BLAS). LAPACK is designed at the outset to exploit the Level 3 BLAS. Because of the coarse granularity of the Level 3 BLAS operations, their use promotes high efficiency on many high-performance computers, particularly if specially coded implementations are provided by the manufacturer.
- ScaLAPACK library: Similar to LAPACK, ScaLAPACK is a library of high-performance linear algebra routines for parallel distributed memory machines. It solves dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems. The use of ScaLAPACK provides efficiency, scalability, reliability, portability, flexibility, and ease of use (by making the interface to LAPACK and ScaLAPACK look as similar as possible).

- BLACS library: In the context of distributed parallel machines, the BLACS (Basic Linear Algebra Communication Subprograms) routines is created to make easy linear algebra oriented message passing interface that may be implemented efficiently and uniformly across a large range of distributed memory platforms.
- OpenMP library: As we discuss in the section 1.3.4, this package allows parallel programming in shared memory multi-core platforms. The openMP directives improves strongly the performance of MUMPS. OpenMP parallel regions of MUMPS can be set through the OMP_NUM_THREADS environment variable.
- MPI library:
- Others

## 3.4 Installation of DSFDM/FFWI

### 3.4.1 Description of nodes.

## 3.5 Methodology

In this study, our aim is to evaluate the capacity of the DSFDM/FFWI code to process large test cases on MESCA-II node and to improve the performance of the code in this architecture. This implies how to keep the code scalable when the memory consumption and MPI communications are increasing. In [], the authors show that the memory consumption and elapsed time are mostly based on LU matrix factorisation doing by MUMPS package. Their results explain in more details this fact []. The MUMPS team has developed the BLR option to improve the performance of the storing of LU matrix values (Amestoy et al. []). As we discuss above (see 2), the optimization of DSFDM/FFWI can be seen in multiple levels. In this study, we will focus on scalability study (strongly and weakly) and different profiling methods to analyse the performance of the code.

## 3.6 Scalability study

The scalability, as we discussed in section 1.3.3, is the coarse grained of the study of performance. It can be measured in two scales: strong scaling and weak scaling. Here, we study these two scales in different nodes cluster such as Mesca and classical nodes. When we say classical nodes it refers the other nodes different to Mesca.

**3.6.1 Strong Scalability.** This study allow us to understand how much faster our DSFDM/FFWI code is with N processors by fixing the size of the problem. Firstly, we perform different test cases in classical nodes to be comparing the performance with Mesca. To study the scalability, we use the homogeneous isotropic model that aims to validate DFSFDM against an analytical solution computed in an homogeneous infinite acoustic isotropic ($ianiso = 0$) medium (Docffwi). According to the documentation, we fix, firstly, the size of the problem to be grid $(n1, n2, n3) = (41, 41, 121)$ ie. $4km \times 4km \times 12km$ (depth,X,Y). The others parameters are the same as the documentation. The wavespeed and the density are respectively equal to $1.5km/s$ and $1000km/m3$. The grid interval is $100m$. The frequency is $3.72Hz$. The shot coordinates in meters are $(x1, x2, x3) = (1000, 2000, 2000)$.

The results obtained in Figures 3.1-3.5 show the comparison of **DSFDM** with the analytical solution (**FDTD**).

In the first test case, we run the code in a classical node such that the architecture is given in Table 3.1

| | |
|---|---|
| Number of sockets (CPUs) | 2 sockets Intel(R) Xeon(R) CPU E5-2697 v2 |
| Number of Core(s) | 2 * 12 cores per socket at the nominal frequency 2.70GHz |
| Hyperthreading | activated |
| Number of Thread(s) | 2 * 24 threads per cpu |
| Shared Memory | 2*32 GB DDR* (* unknown) |
| L1d/L1i cache | 32K |
| L2 cache | 256K |
| L3 cache | 30720 KB |

Table 3.1: Architecture of classical nodes

The table 3.2 shows the computational resources used when the code is performing in this node.

| Grid dimensions | npml | #u | #n | #MPI | #th | $Mem_{LU}(Gb)$ | $T_{LU}(s)$ | $T(s)$ |
|---|---|---|---|---|---|---|---|---|
| $41 \times 41 \times 121$ | 8 | 1 | 1 | 2 | 10 | 6,7 | 17.76 | 23.77 |

Table 3.2: Homogeneous Isotropic running on classical node: #u($10^6$): number of unknowns. #n: number of nodes. #MPI: number of MPI process. #th: number of threads per MPI process. $Mem_{LU}(Gb)$: Memory for LU factorization in Gbytes. $T_{LU}(s)$: Elapsed time for factorization in s. $T(s)$: Running time

Secondly, we execute the same test case with the same parameters in the Mesca node. The architecture of this node is showing in table 3.3.

| | |
|---|---|
| Number of sockets (CPUs) | 8 sockets Intel(R) Xeon(R) CPU E7-8890 v4 |
| Number of Core(s) | 8 * 24 cores cores at the nominal frequency 2.20GHz |
| Hyperthreading | deactivated |
| Number of Thread(s) | 8 * 24 threads per cpu |
| Shared Memory | 8*340 GB DDR* (* unknown) |
| L1d/L1i cache | 32K |
| L2 cache | 256K |
| L3 cache | 61440K |

Table 3.3: Architecture of Mesca II node

These results are shown in the table **??**

By using two MPI processes and ten threads, the classical node shows almost the same performance as Mesca( $T_{LU}(s) \approx 17.7s$). In addition, the total memory allocated for factorization was 6.7 Gigabyte.The time of LU factorization takes 74 % and 70 % respectively in the classical node and Mesca. This type

| Grid dimensions | npml | #u | #n | #MPI | #th | $Mem_{LU}(Gb)$ | $T_{LU}(s)$ | $T(s)$ |
|---|---|---|---|---|---|---|---|---|
| $41 \times 41 \times 121$ | 8 | 1 | 1 | 2 | 10 | 6,7 | 17.73 | 25.048 |

Table 3.4: Homogeneous Isotropic running on Mesca II node

of node will be used for rest of the simulation. The results are shown in tables **??** and 3.2.

In the following sentences, we define three test cases depending on the size of the grid to study the strong scalability of DSFDM code. This grid is a 3D dimension, $n1, n1, n1$, with PMLs. The chosen sizes of the grid are $(n1, n2, n3) = (81, 81, 121)$, $(n1, n2, n3) = (81, 121, 121)$ and $(n1, n2, n3) = (121, 121, 121)$. The grid interval is 100. We perform these tests in node cluster which contains the classical nodes and Mesca node described above. The resolution of wave propagation equation is handled by MUMPS solver. Here, we are more interested in factorization phase performing by MUMPS solver. This factorization can be doing in two ways: Full Rank and Low-Block Rank. In the following simulation, we will focus on the Full Rank option. In all the tests, we consider the number of MPI processes is always less or equal than the number of sockets (or CPU).

The Figures 3.6a and 3.6c show the elapsed time of the three different test cases running on classical nodes. As we can see in Figure 3.6a, the Time of LU factorization depending on numbers of processes decreased by following a power law decay. We remark also, we obtain a good approximation when the numbers of processes is equals to a power of 2 (see Figure 3.6b). A log-log plot, (Figure **??**), shows the strong scalability of the tests. In the case of $(n1, n2, n3) = (121, 121, 121)$, we observe a rapid drop of elapsed time between 2 and 8 processes and smooth descent of the curve in 8 and 16. The table 3.5 shows the curve fitting of the the log scale plots (ie. $Y = A * log(N) + B$, with N number of processes).

| Size | A | B |
|---|---|---|
| $81 \times 81 \times 121$ | -0.65 | 2.17 |
| $81 \times 121 \times 121$ | -0.71 | 2.5 |
| $121 \times 121 \times 121$ | -0.75 | 2.86 |

Table 3.5: Parameters of the fitting curves of Elapsed time in classical nodes

Similarly, we get the same remark in Mesca node (see Figure 3.6c). Here, we see a rapid growth between 2 and 4 processes different of the classical nodes. The elapsed time is fitted in log scale plot such that the slope and the intercept are given by the table 3.6. For comparing the performance of Mesca and the classical nodes, we can use the execution time in the two architectures.

| Size | A | B |
|---|---|---|
| $81 \times 81 \times 121$ | -0.62 | 2.16 |
| $81 \times 121 \times 121$ | -0.74 | 2.5 |
| $121 \times 121 \times 121$ | -0.72 | 2.81 |

Table 3.6: Parameters of the fitting curves of Elapsed time in Mesca

..............................
Comment ...................... ..............................

..............................

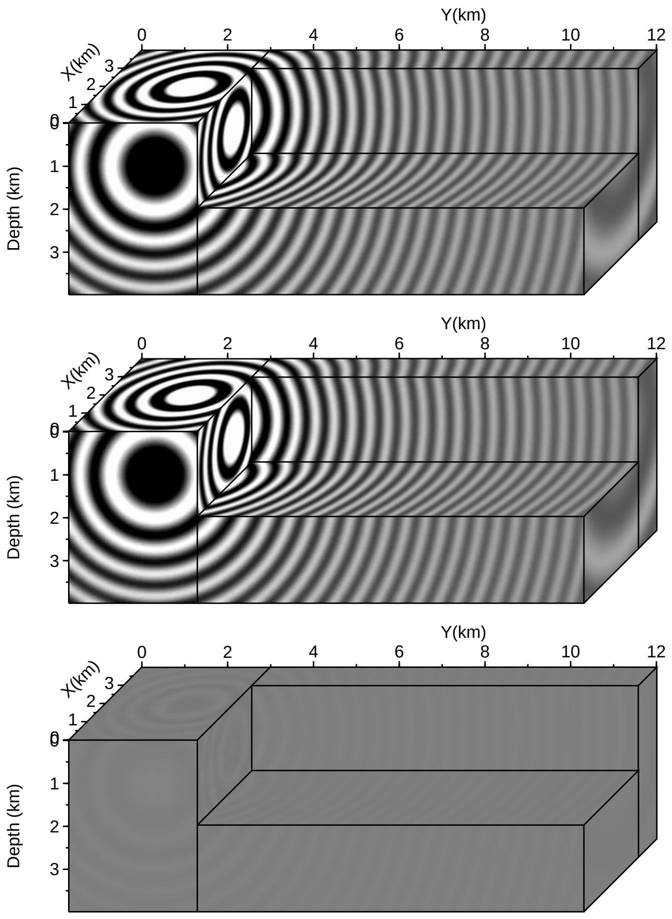


Figure 3.1: Homogeneous Isotropic. Infinite homogeneous isotropic medium. Validation against analytical solution. a) DSFDM solution. b) Analytical solution. c) Difference.

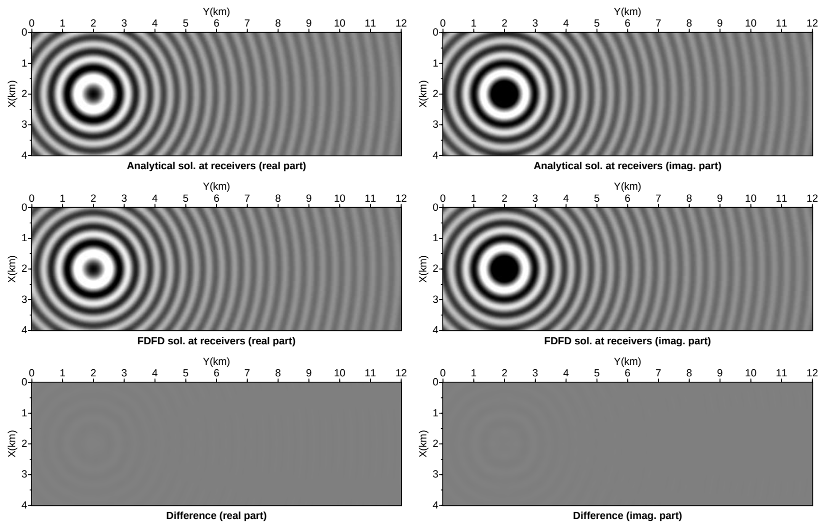


Figure 3.2: Homogeneous Isotropic. Infinite homogeneous isotropic medium. Validation against analytical solution. Solution at receiver positions, 1km below the surface.

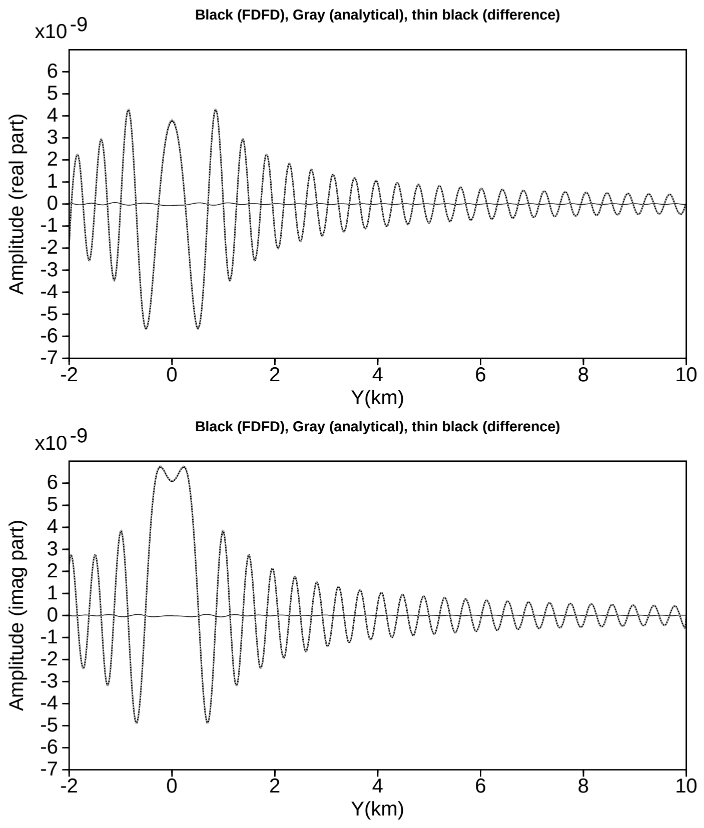


Figure 3.3: Homogeneous Isotropic. Infinite homogeneous isotropic medium. Validation against analytical solution. Solution at receiver positions along the Y profile running across the shot position.

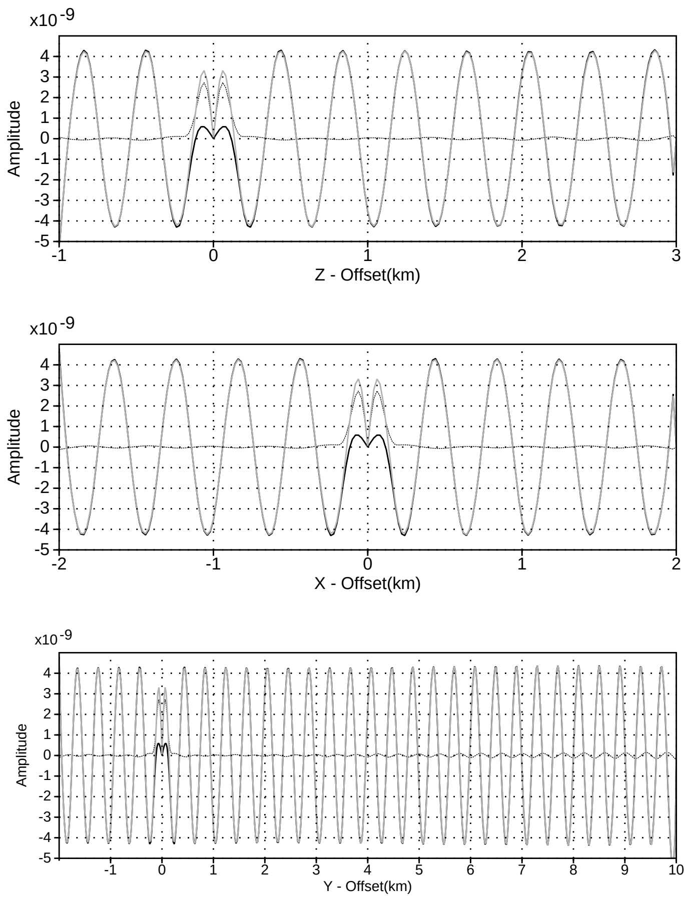


Figure 3.4: Homogeneous Isotropic. Infinite homogeneous isotropic medium. Validation against analytical solution. Logs across shot position with correction for geometrical spreading

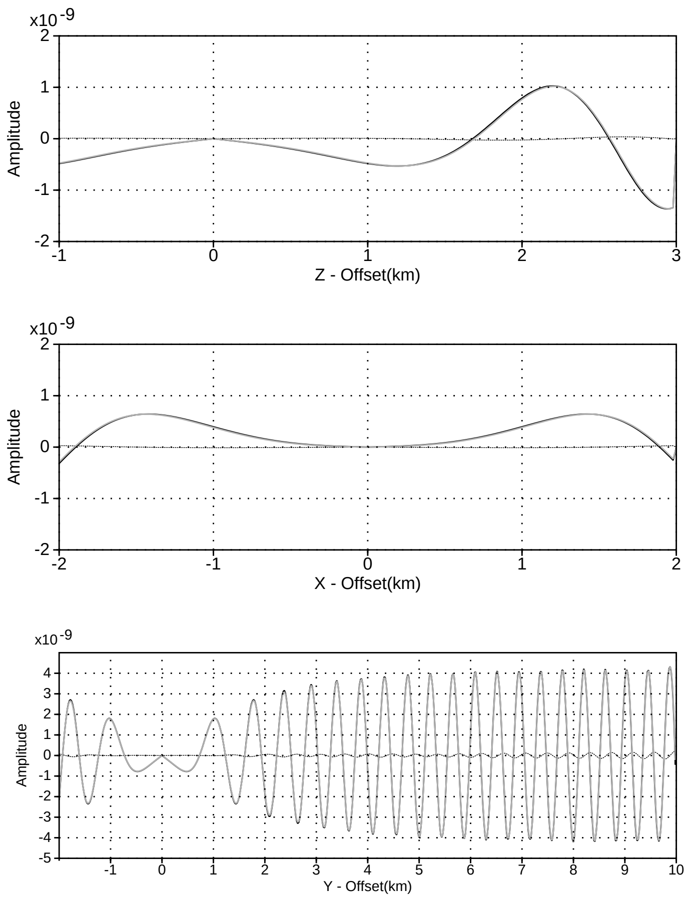
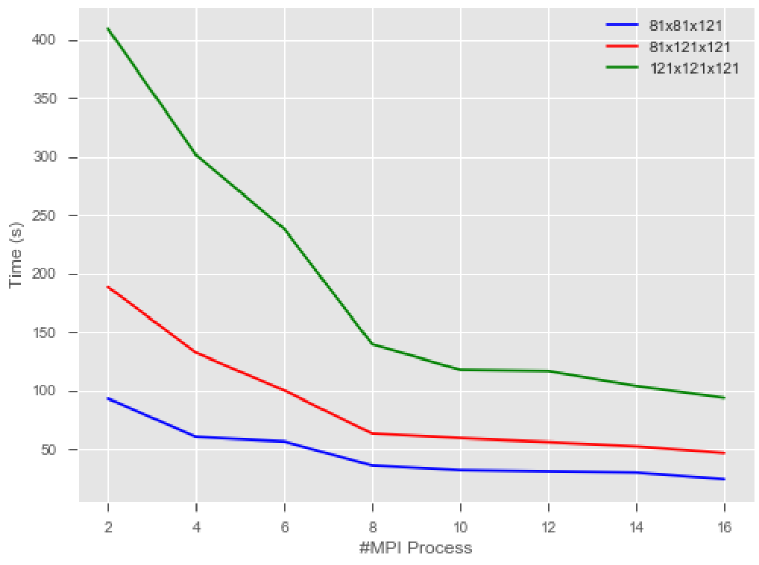
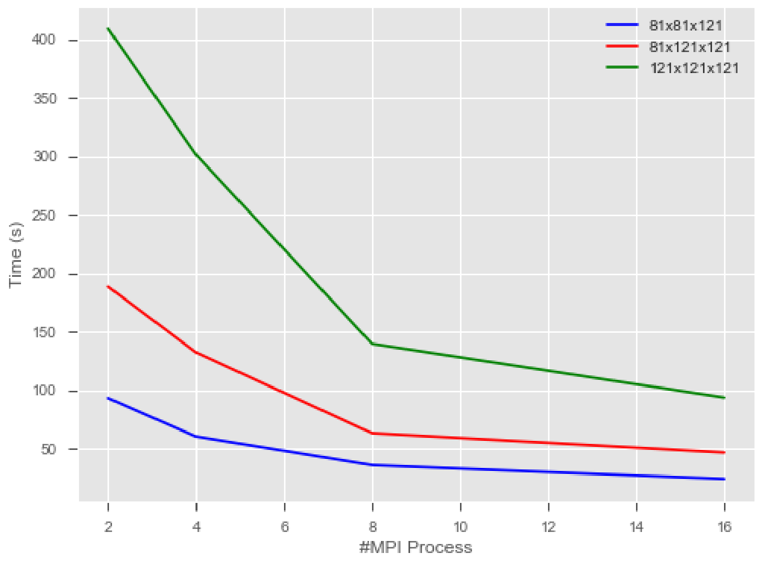


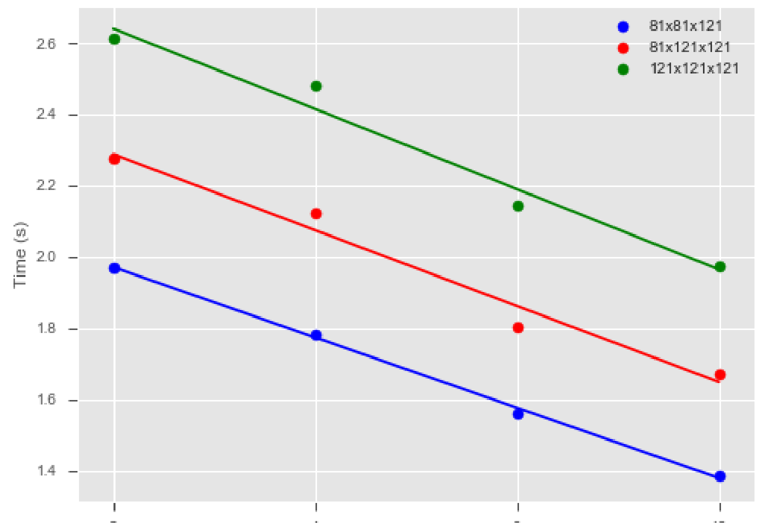
Figure 3.5: Homogeneous Isotropic. Infinite homogeneous isotropic medium. Validation against analytical solution. Logs along the slices of Fig. ??b with correction for geometrical spreading

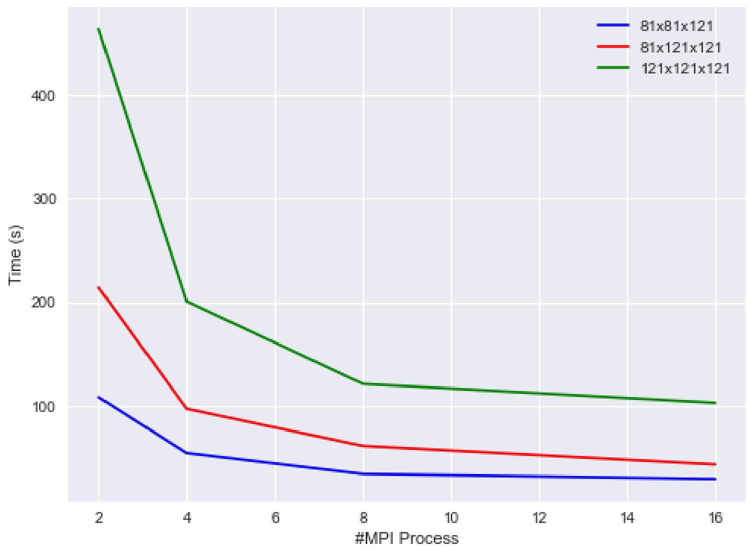


(a) Time Elapsed for LU factorization performing in classical nodes

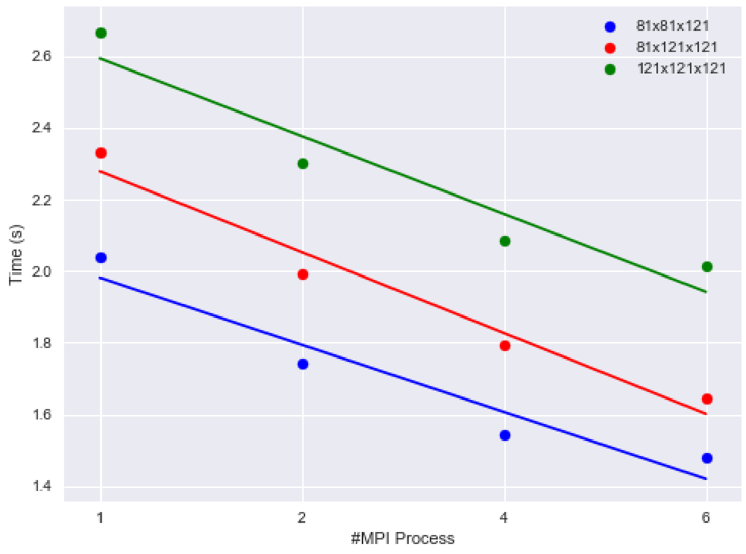


(b) Elapsed time for LU factorization in log scale performing in on classical nodes

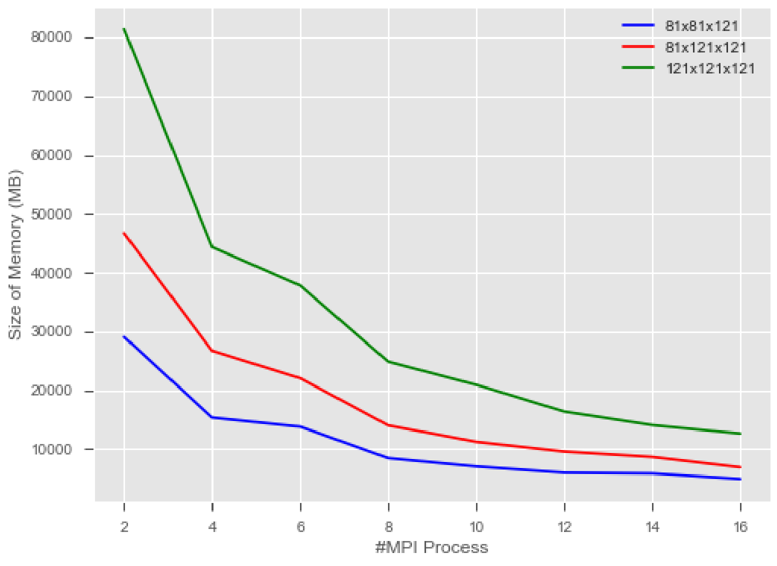




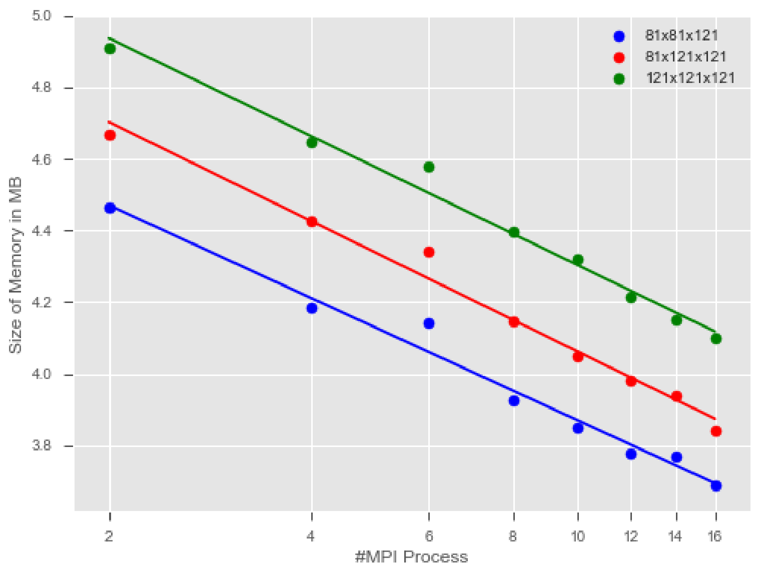
(a) Time Elapsed in LU factorization in performing in Mesca node



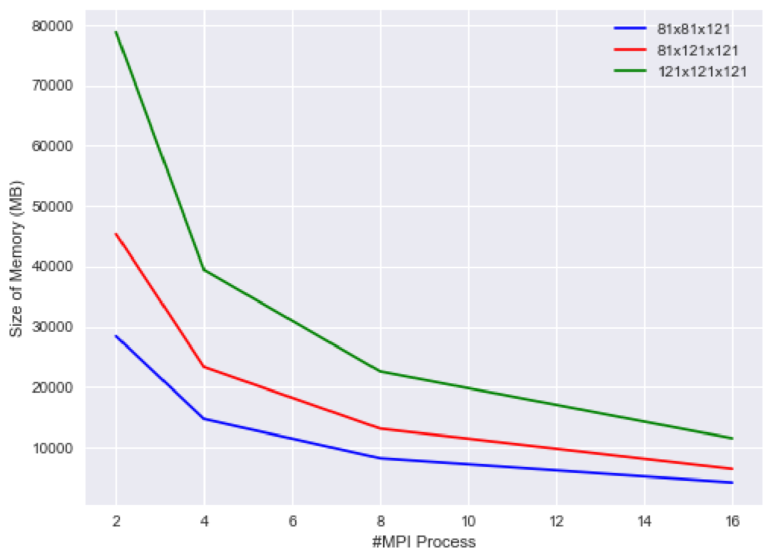
(b) Elapsed time for LU factorization in log scale performing in Mesca node



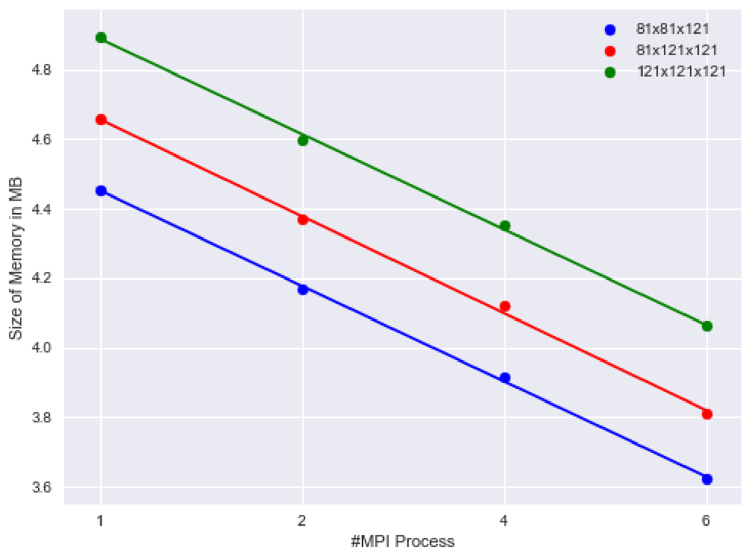
(a) Avg. Space in MBYTES per working proc for LU factorization on classical nodes



(b) Avg. Space in MBYTES per working proc for LU factorization in log scale on classical nodes



(a) Avg. Space in MBYTES per working proc in LU factorization



(b) Avg. Space in MBYTES per working proc in LU factorization in log scale

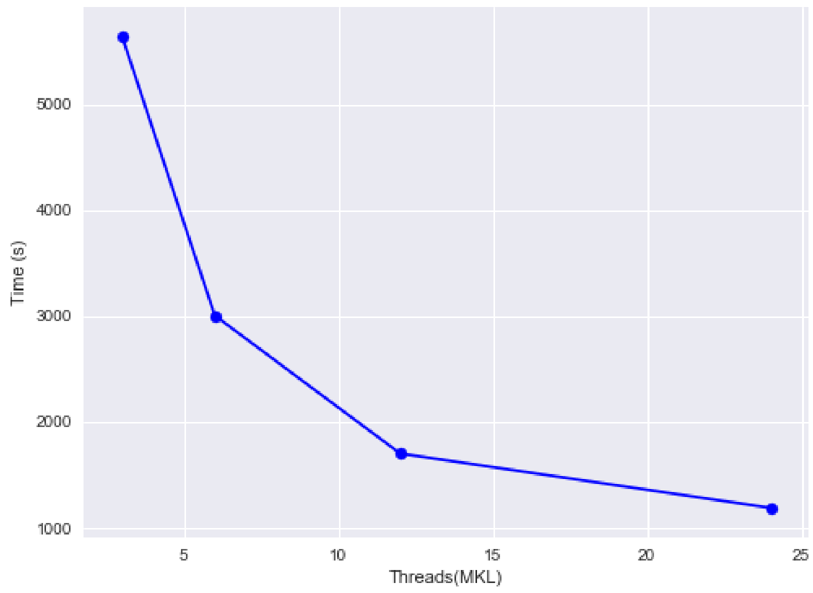


Figure 3.10: Elapsed Time for LU factorization function of number of threads in Mesca node

Comment ..................... .............................

.............................
Comment .....................
.............................

.............................
Comment ..................... .............................

Placement of Threads: KMP AFFINITY

| KMP_AFFINITY | Grid dimensions | npml | #MPI | #th | #Cores | $T_{LU}(s)$ | $Avg.Mem_{LU}/proc(M$ |
|---|---|---|---|---|---|---|---|
| scatter | $92 \times 181 \times 321; dz = 50$ | 8 | 8 | 24 | 192 | 396.3693 | 73402 |
| compact | $92 \times 181 \times 321; dz = 50$ | 8 | 8 | 24 | 192 | 435.1387 | 73402 |

Table 3.7: Homogeneous Isotropic running on Mesca node

.............................
Comment ..................... .............................

Case study: Ovethrust.

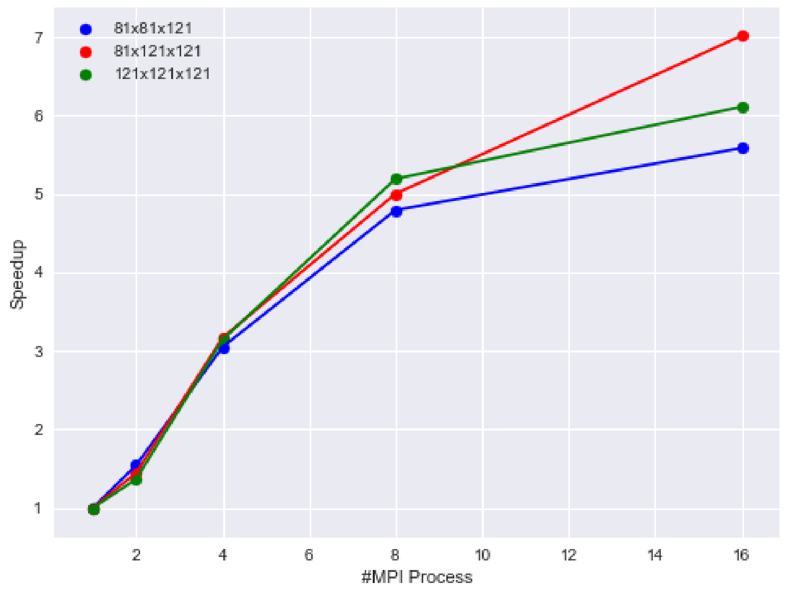


Figure 3.11: Speedup in Mesca node

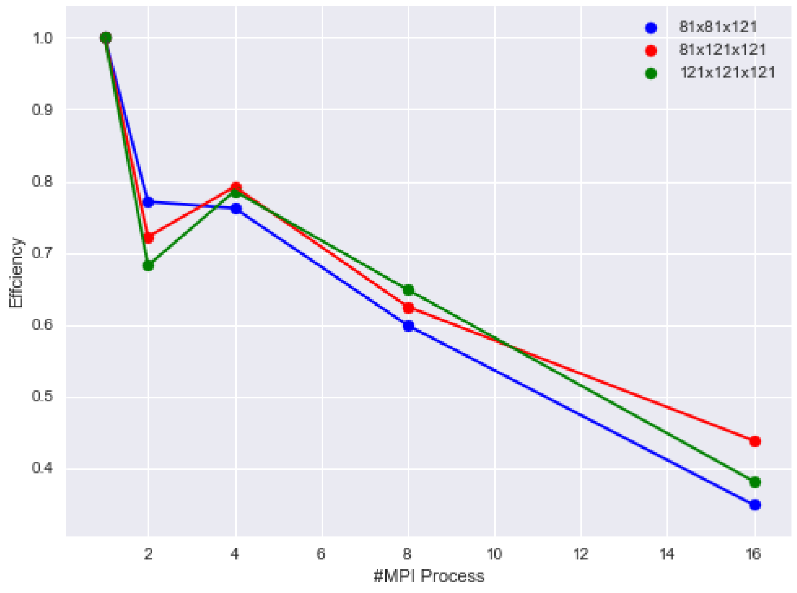


Figure 3.12: Efficiency in Mesca node

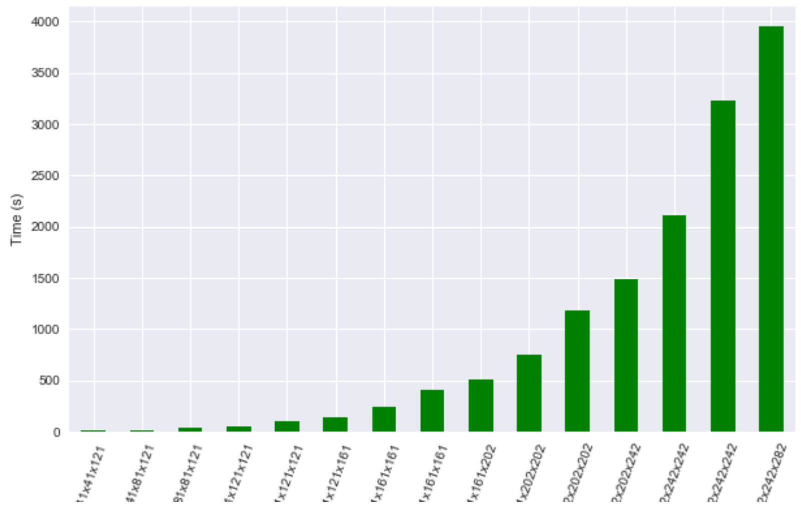


Figure 3.13: Time in LU factorization in Mesca node

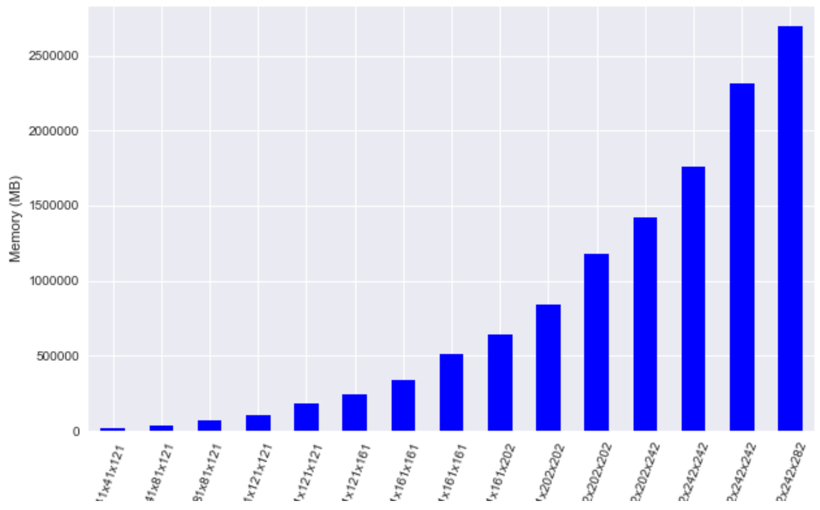


Figure 3.14: Memory consumption in LU factorization in Mesca node

| Freq | Grid dimensions | npml | | #MPI | #th | #Cores | $T_{LU}(s)$ |
|---|---|---|---|---|---|---|---|
| 3.5, 4, 4.5, 5 | $66 \times 130 \times 230; dz = 70$ | 8 | 24 | 10 | 240 | 78 | |
| 7 | $92 \times 181 \times 321; dz = 50$ | 8 | 32 | 10 | 320 | 322 | |
| 10 | $131 \times 258 \times 458; dz = 35$ | 4 | 68 | 10 | 680 | 1153 | |

Table 3.8

| Freq | Grid dimensions | npml | #MPI | #th | #Cores | $T_{LU}(s)$ | $T(s)$ |
|---|---|---|---|---|---|---|---|
| 3 | $66 \times 130 \times 230; dz = 70$ | 8 | 5 | 24 | 120 | 194.0989 | 3m42.937s |
| 3,5 | $66 \times 130 \times 230; dz = 70$ | 8 | 8 | 15 | 120 | 98.9756 | 2m4.808s |
| 7 | $92 \times 181 \times 321; dz = 50$ | 8 | 8 | 20 | 160 | 439.0829 | 8m21.590s |
| 7 | $92 \times 181 \times 321; dz = 50$ | 8 | 8 | 24 | 192 | 396.3693 | 7m39.295s |
| 10 | $131 \times 258 \times 458; dz = 35$ | 4 | 8 | 24 | 192 | 2182.6717 | 38m58.811s |

Table 3.9

## 3.7 Profiling

## 3.8 Results

# Discussion and Perspectives

# Acknowledgements

This is optional and should be at most half a page. Thanks Ma, Thanks Pa. One paragraph in normal language is the most respectful.

Do not use too much bold, any figures, or sign at the bottom.

## References

[AST92] Alan Adolphson, Steven Sperber, and Marvin Tretkoff (eds.), *$p$-adic methods in number theory and algebraic geometry*, Contemporary Mathematics, no. 133, American Mathematical Society, Providence, RI, 1992.

[Bea06] Alan F. Beardon, *From problem solving to research*, 2006, Unpublished manuscript.

[Dav99] M. C. Davey, *Error-correction using low-density parity-check codes*, Phd, University of Cambridge, 1999.

[Lam86] Leslie Lamport, *LATEX: A document preparation system*, Addison-Wesley, 1986.

[Mac86] D. J. C. MacKay, *Statistical testing of high precision digitisers*, Tech. Report 3971, Royal Signals and Radar Establishment, Malvern, Worcester. WR14 3PS, 1986.

[Mac95] D. J. C. MacKay, *A free energy minimization framework for inference problems in modulo 2 arithmetic*, Fast Software Encryption (Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms) (B. Preneel, ed.), Lecture Notes in Computer Science Series, no. 1008, Springer, 1995, pp. 179–195.

[MN95] D. J. C. MacKay and R. M. Neal, *Good codes based on very sparse matrices*, Available from www.inference.phy.cam.ac.uk, 1995.

[Sha48] C. E. Shannon, *A mathematical theory of communication*, Bell Sys. Tech. J. **27** (1948), 379–423, 623–656.

[Sha93] ______, *The best detection of pulses*, Collected Papers of Claude Shannon (N. J. A. Sloane and A. D. Wyner, eds.), IEEE Press, New York, 1993, pp. 148–150.

[Web12] Webots, *Commercial mobile robot simulation software*, 2012, www.cyberbotics.com.