Experiences developing and running numerical simulations on HPC platforms: BSIT and GeNESiS

Claudia Rosas

13th - 14th June - HPC Knowledge Meeting ’19 - Barcelona
Where’s Wally?

One of the almost 600 workers of BSC!
Out of the 4 Departments, I’ve collected:
- 4 years of experience in CS Dept.
  - Performance Analysis
- Almost 3 years of experience in CASE Dept.
  - Porting and Optimization
  - + Performance Analysis ;}
What are the main requirements for HPC software?
Efficient software development

Codebases
Millions of lines of code

- simple iPhone game app
- Unix v1.0 1971
- Win32/Simile virus
- average iPhone app
- Pacemaker
- Photoshop v1.0 1990
- Camino web browser
- Quake 3 engine 3D Video game system
- Space Shuttle

Note: there are more systems being built BSC

http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
Efficient software development

Bigger problems

Software is not getting any simpler

Maintain? Extend?

Heterogeneous hardware
Wave propagation simulation
Wave propagation simulations

• Geophysical apps are among the largest computer simulations
  • Popular family that involves propagation of electromagnetic and seismic waves.
  • Wave + materials: wave propagate through the Earth’s subsurface and get deviations/reflections from rocks.

• Geophysics focuses on understanding the physical fields and infer the properties of the subsurface.
  • The physics of geophysical waves is relatively simple when compared to the geometrical complexity and the different scales of geological structures
  • …Thus the need for computer simulations.

• Computer generated map of the subsurface
  • A priori knowledge gathered by geologists
  • Need 3D simulations to explain recorded data (no symmetrical)
  • Highly dependent on the wavelength (sensitivity of the wave)
The reason behind optimizing wave propagation

Forward FM Modelling

Reverse-Time RTM Migration

Full Waveform FWI Inversion

Slide from: M. Hanzich 2015
Why Elastic?

- Elastic Anisotropic
- Elastic Isotropic
- Acoustic

More Detail & higher computational cost
Heterogeneous target architectures
Some of the current target architectures

- **Intel Skylake Platinum**
- **Intel Xeon Phi**
- **Nvidia Volta**
- **IBM Power 9**

48 racks with 3456 nodes
- 2 sockets Intel Xeon Platinum 8160 CPU with 24 cores each @ 2.10GHz
- 216 nodes with 12x32 GB DDR4 (8GB/core)
- 3240 nodes with 12x8 GB (2GB/core)
Some of the current target architectures

- Intel Xeon Phi
- Intel Skylake Platinum
- Nvidia Volta
- IBM Power 9

16 compute nodes
- 1 Intel(R) Xeon Phi(TM) CPU 7230 @ 1.30GHz 64-core processor
- 96 GB main memory distributed in 6x 16GB DDR4
- 16 GB HBM distributed in 8x 2GB MCDRAM
Some of the current target architectures

52 compute nodes:
- 2 x IBM Power9 8335-GTH @ 2.4GHz (20 cores and 4 threads/core)
- 512GB of main memory distributed in 16 dimms x 32GB
- 4 x GPU NVIDIA V100 (Volta) with 16GB HBM2.
Barcelona Subsurface Imaging Tool – BSIT
BSIT-I in Numbers (Kaleidoscope)

2 Kernels
- acoustic.2d.cell.rtm
- acoustic.3d.cell.rtm

4 years of development

6 Production Chain Binaries
- rtm_acoustic
- rtm_acoustic_kernel.sh
- rtm_acoustic_post
- rtm_acoustic_proc
- rtm_chkp
- RTMkernel.sh

General
- Lines of Code: > 40K
- Doc Lines: > 10K
- Releases: ~2
- Local version: ~200
- People: 5

Supported Hardware:
- Cell/B.E.

Parallel Paradigms
- MPI
- Cell/B.E. SDK
The Framework

Environment for Assessing Performance (EAP)

- A single wave propagation simulation level
- Provides common pieces related to a parallel application

Environment for Assessing Confidence (EAC)

- Workflow level
- Distributes tasks to different computational resources and gathers results

BSIT-II in Numbers (RBRC) [2015]

68 Kernels
FM/RTM/FWI
Acoustic/Elastic/CSEM

Only 4 years of development!!!

36 Production Chain Binaries
FM/RTM/FWI
Acoustic/Elastic/CSEM

65 Propagation Libraries
Framework
FM/RTM/FWI
Acoustic/Elastic/CSEM

General
Lines of Code: > 200K
Doc Lines: >100K
Releases: ~12
Local version: ~3500
People: 10

Supported Hardware:
General Purpose
GPUs
Xeon Phi’s
Blue Gene / Q
Tegra

Parallel Paradigms
MPI
OpenMP
SSE/AVX
CUDA
3D Elastic Data Generation: Petascale Simulation - Stress a complete HPC cluster (1 Petaflop) *

- **General:**
  - Elastic, 3D, isotropic, with free-surface
  - 10 Hz peak, 25 Hz maximum
  - 6 points per wavelength
  - **5041 shots (simulations)**
    - vmin=1089m/s, vmax=6000m/s
    - 16km x 16km (3.2 km depth), 6 sec.
  - **Output: 2.8 TB synthetic SEGY traces**

- **Each shot (kernel):**
  - 440x2201x2201 cells, 11279 iterations
  - 40 cell-long ABCs. Total **121 GB** without DD
  - ~2.2 hours in 20 nodes (**3 days** for the whole set of shots using the whole **Marenostrum 3**)
  - 6241 seismograms, 4 channels, 39 GB output

*Note: MareNostrum III
Slide taken from M. Hanzich 2016
Petascale Modeling: CPU usage

* Note: MareNostrum III
Slide taken from M. Hanzich 2016

Normal Marenostrum Workload

Elastic Modeling
Petascale Modeling: Network Usage

*Note: MareNostrum III Slide taken from M. Hanzich 2016*

![Graph showing network usage](image)

- **Normal Marenostrum Workload (0.55 TB/sec.)**
- **Elastic Modeling (1.55 TB/sec.)**
Petascale Modeling: Total Load

Resource utilization:

• Each cell is a computational node (~3000 nodes).

• **Pink-colored** cells means that the CPUs in that node are being used at over 99% (running jobs).

• **Blue-colored** cells means that the CPUs in that node are being used around 95% (starting jobs).

• **Green cells** were not allocated to our execution (free nodes).

* Note: MareNostrum III
Slide taken from M. Hanzich 2016
Additional optimizations for BSIT

• The produced code is easily portable and for each representative kernel (problem + physic), we evaluated several architectures and the potential of optimizations in each one:

  • Intel Xeon Phi, both KNC and KNL
  • Intel Skylake – Xeon Platinum 8160 (General Purpose of MN4)
  • NVIDIA Volta V100
  • Power9 (preliminary results)
Intel Xeon Phi First Contact (KNC)

Absolute performance of the elastic propagator optimizations on an Intel Xeon Phi (in blue) compared with an Intel Xeon E5-2697v3 processor (in red)

Intel Skylake

- Portability of code from KNL to Skylake is almost straightforward
- Expensive chips
- No on-package memory

NVIDIA Volta V100

Farrés et al. (2019). “Performance evaluation of fully anisotropic elastic wave propagation on NVIDIA Volta GPUs”. 81st EAGE Conference & Exhibition 2019
BSIT final remarks

• It is flexible and robust enough (more than 5 years of evaluation) to build several numerical simulations.

• Has shown a consistent scalability and capacity to run at Petascale

• Each kernel opens additional windows for improvement:
  • You are able to measure and optimize almost in every new available architecture

• Its learning curve is steep, it is not object oriented, it depends fully on C and the user has to built hers own libraries.

• Thus we proposed GeNESiS!!!
General Numerical Engine and Simulations System – GeNESiS
GeNESiS

**General Numerical Engine and Simulation System:**
- Software system / Ecosystem;
- Elaborate any numerical simulation;
- HPC platforms.

- Quick and elegant deployment of uniform reusable code.
- Flexible: layered and incremental.
- Written in C++11 and Python.
GeNESiS Overview

- Building blocks
- Workflow
- Main internal procedures
  - Resource Allocation
  - Highly parallel simulation
- Domain decomposition
- Communication protocols
- Heterogeneous hardware support
- Efficiency metrics
GeNESiS Overview

- Basic pieces of relevant numerical methods
- Solutions of PDEs (FEs and FDs)
  - Basic functions (FE)
  - Elemental matrices (FE)
  - Coefficients (FD)
  - Stencils (FD)
GeNESiS Overview

- Combines building blocks to procedure to solve physical equation (wave propagation, heat diffusion, compressible and incompressible flows...)
- User can choose predefined implementations or implement its own.
GeNESiS Overview

**PROBLEM**

- Convergence of computation and management of the minimal processing unit
  - Handles communication among workers (if used)
  - Definition and assignment of tasks
- I/O procedures
- Combines equation with the pre and post processing data
EnVironment for Acceleration (EVA):

• Framework to build numerical simulation kernel
  • Program or library (PDE)
  • Modeling and Inversion problems
  • Supports different HPC Hardware architectures
  • Manages domain decomposition
CAAd INput (CAIN)

- Manages (preprocessing) input data
  - CAD meshes
- Produces clean and usable information for other frameworks
  - Serves to EVA, ADAN or both
  - For example: Preprocesses input unstructured mesh for checking, refinement, domain decomposition, etc.
Advanced Block to Engineer knowLedge (ABEL)

- Tackles preprocessing of input data besides meshes
- Post processing of any result (temporal or definitive) produced by other frameworks in GeNESiS.
  - Handle libraries to manage, filter, interpolate input data and binary results form a modeling kernel
  - Binary results from a modeling kernel in EVA could be translated to a standard format,…
Application Distribution Among Nodes (ADAN)

- Build workflows to process many simulation on a set of HPC resources.
  - A simulation kernel built in EVA that needs to be executed many times (Monte Carlo method, complex inversion or minimization systems)
- The workflow comprehends from the kernel to pre and post processing, check pointing, and so on.
Highlights of GeNESiS

• As GeNESiS is mostly written in C++11 and Python
  • Functionalities can be extended in both languages

• To provide python-based extensions to test research ideas (quickly) and, once properly validated, built a C++ based version for efficiency and production.

• Porting from one to another will imply much less effort

• The goal is to reduce the time and cost of the Research & Development cycle
What do we have available now?

• EVA and CAIN, that is: Kernels and Meshes
• SWIG “translates” between Python and C++
• One can use Design Patterns for C++:
  • Singleton, Factory Method, …
• There is Exception handling

Only EVA is covered in this talk 😊

A Workflow in EVA

Automatic Hardware Detection

A Workflow in EVA

A Workflow in EVA

from genesis import eva, cain

definition of own matrix-free solver
class MySolver(eva.Cloneable,
   eva.MatrixFreeSolver):
   ...

try:
   # EVA Instance
eva = eva.EVA()

   # Create & add solver instance
   mysolver = MySolver()
   eva.add_solver(mysolver)

   # Create & add mesh instance
   mesh = cain.Mesh.get('Unstructured2DMesh')
   eva.set_mesh(mesh)

   # Configure with an input parameters file
   eva.config(sys.argv[1])

   # Start the simulation.
eva.start()

   # Finalize the processing.
eva.finish()

except Exception as e:
   ...

To Extend EVA according to your profile

• EVA- User, must extend red circle

• EVA- Developer, could extend blue circles

Case of Study: 2D High Order Finite Elements, Discontinuous Galerkin

• HOFE, DG: In-house developed prototype
  • Elastic waves in a 2D media
  • Spatial domain discretized with unstructured triangular meshes
  • Reported in the deliverable of GAP (High Performance Geophysics, Acronym in Spanish) ID. TIN2016-80957-P supported by the Spanish government under a National Plan.

Weak scalability

<table>
<thead>
<tr>
<th>Mesh discretization</th>
<th>Elements</th>
<th>Iterations</th>
<th>Nodes</th>
<th>MPIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

79% of efficiency
Strong scalability

- Mesh discretization: 12.5
- Elements: 230400
- Iterations: 456
- 10 runs, increasing the number of MPI processes (nodes) – 3 tries
- 1 MPI to 512 MPIs

Final remarks
Take-home message...

- GeNESiS is currently on development: ADAN and ABEL are work in progress.
- Our prior knowledge on Software Development mature the idea behind of GeNESiS
  - BSIT is mature and has proved to be scalable and efficient.
- The time span between prototyping and implementing scalable code is reduced.
  - Research in progress for EVA
- The binomial Python and C++ brings the advantages of both worlds to developers’ hands
  - Flexibility, broad availability of libraries and robustness and efficiency.
Thank you Questions?
crosas@bsc.es

13th - 14th June
HPC Knowledge Meeting ’19 - Barcelona
The reason behind optimizing wave propagation

Forward FM Modelling

Reverse-Time RTM Migration

Full Waveform FWI Inversion

Slide from: M. Hanzich 2015
Why Elastic?

Acoustic

Elastic Isotropic

Elastic Anisotropic

Slide from: M. Hanzich 2015
Our way of doing things

Conventional Grid Cell

SSG Cell Acoustic

FSG Cell

Velocity Node
$u_1, v_1, w_1, u_2, v_2, w_2$

Location of material properties

Stress Nodes
$\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$
$\sigma_{xz}, \sigma_{xy}, \sigma_{yz}$

Velocity Nodes
$u, v, w$

Stress Node
$\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xz}, \sigma_{yz}, \sigma_{xy}$

Velocity Node
$u, v, w$

Slide from: M. Hanzich 2015
Many arrays..

12 Velocity arrays (variables)

24 Stress arrays (variables)

22 Parameter arrays

58 arrays
Data Layout (SoA)