PRACE

PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE

2017

summerofhpc.prace-ri.eu

A long hot summer is time for a break, right? Not necessarily! PRACE Summer of HPC 2017 reports by participants are here.

HPC in the summer?

Leon Kos

There is no such thing as lazy summer. At least not for the 21 participants and their mentors at 10 PRACE HPC sites.

Summer of HPC is a PRACE programme that offers university students the opportunity to spend two months in the summer at HPC centres across Europe. The students work using HPC resources on projects that are related to PRACE work with the goal to produce a visualisation or a video.

This year, training week was in Ostrava and it seems to have been the best training week yet! From MPI to Vectorization and good food, the week was a blast! It was a great start to Summer of HPC and set us up to have an amazing summer!

At the end of the summer videos were created and are available on Youtube as PRACE Summer of HPC 2017 presentations playlist. Together with the following articles interesting code and results are available. Dozens of blog posts were created as well. At the end of the activity, every year two projects out of the 21 participants are selected and awarded for their outstanding performance. The winners of this year, Mahmoud Elbattah and Arnau Miro Janea, presented their experience at the award ceremony in IT4I supercomputing centre, Chech Republic.

Therefore, I invite you to look at the articles and visit the web pages for details and experience the fun we had this year.

What can I say at the end of this wonderful summer. Really, autumn will be wonderful too. Don't forget to smile!

Contents

PRACE SoHPC2017 Coordinator
Leon Kos, University of Ljubljana
Phone: +386 4771 436 E-mail: leon.kos@lecad.fs.uni-lj.si

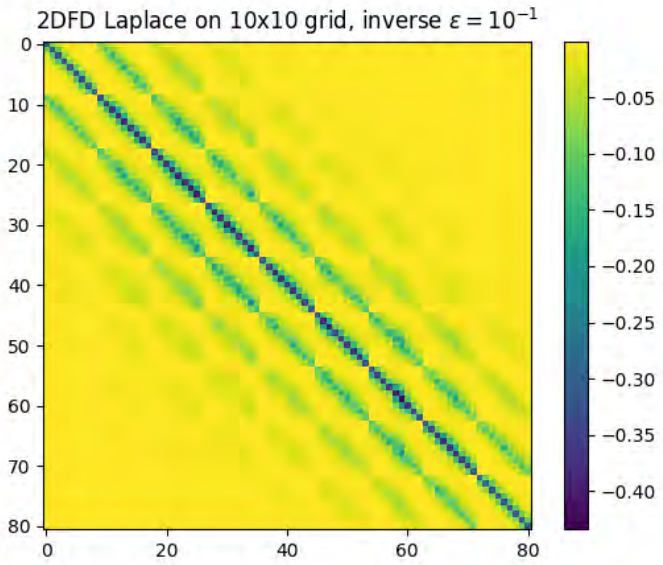
PRACE SoHPCMore Information
http://summerofhpc.prace-ri.eu


Leon Kos

Accelerating stochastic methods for the solution of linear systems of equations.

$(MC)^2MI$ for matrix computations

Anton Lebedev



We accelerated a stochastic method for the computation of preconditioners and compared it to a different, state-of-the-art stochastic method. We show that our method is a better choice. Furthermore, a sequence of truly random numbers has been compared to pseudorandom numbers confirming that pseudo random numbers are good enough for our purposes.

Whether we are carrying out simulations of the world's climate, of a car crash or of the sound of drums spinning at the speed of light, linear systems of equations are omnipresent in science and technology. Even if the problems are considered to be non-linear they will still rely on the solution of linear systems! The solution of such systems is therefore the bedrock of scientific computing. The size of the systems varies from four equations and unknowns (as in computer graphics) to millions or even billions (as in car or air-plane simulations). Small systems are solved using direct methods, whilst big systems require iterative methods. If we can accelerate these methods even a tiny bit, it is worth every effort. Especially since the solution process is often carried out many times over which leads to accumulation of savings.

Acceleration is possible using preconditioners and the goal of this project was:

> Accelerating iterative solvers by accelerating the computation of stochastic preconditioners.

Iterative Solvers and Preconditioners

An iterative method is used to solve a system of linear equations (signified as $Ax = b$ in matrix form) by essentially performing the operation Ax, with a modified matrix A, over and over again until a desired precision of the solution is obtained. As the number of steps needed increases, so does the computation of this expensive matrix-vector product. It is thus imperative to reduce the number of steps to be carried out!

This can be accomplished using a preconditioner P to obtain

$$x \approx PAx = Pb \quad .$$

This will generally result in fewer steps of the iterative solver and thus shorter run times. Computations of preconditioners are not easy, since they have to fulfil the following three Properties:

1. P should be easy to compute.
2. P should approximate A^{-1} well.
3. P should be cheap to apply (P should be a sparse matrix).

In general, one can fulfil two of the above requirements and these need to be chosen carefully.

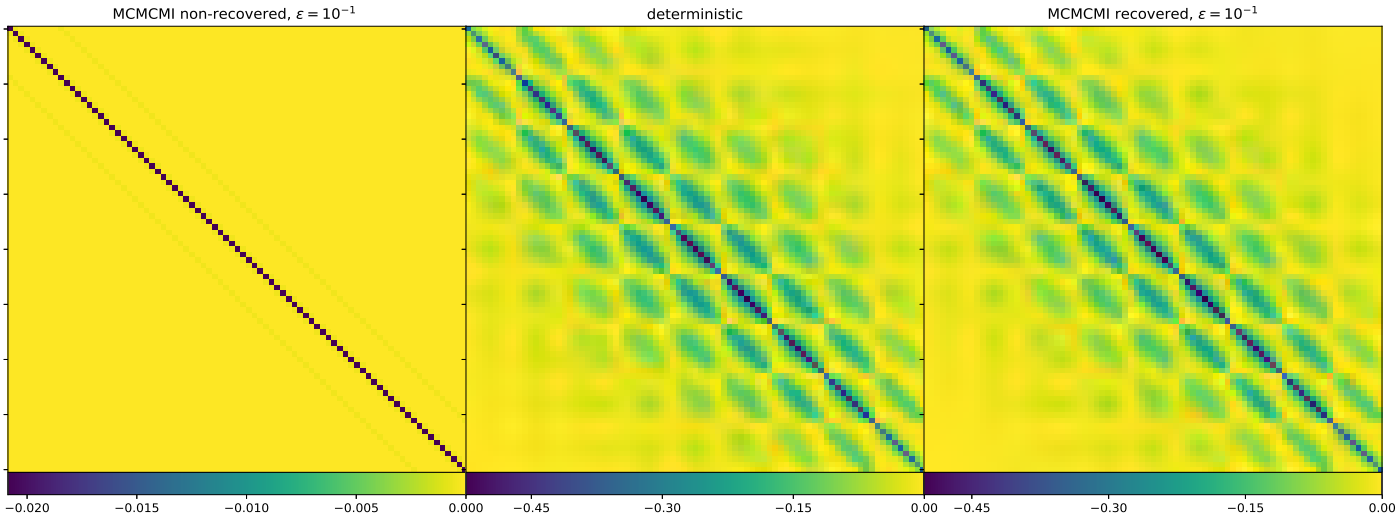


Figure 1: Inverse of a finite-differences Laplacian on a 2D 10×10 grid. *Left:* Inverse computed using $(\mathrm{MC})^2$ MI without the application of the costly recovery procedure. *Centre:* Inverse computed using a deterministic method. *Right:* Inverse computed using $(\mathrm{MC})^2$ MI **with** application of the recovery procedure. Using the method without recovery (left figure) yields a bad inverse matrix, but it is *good enough* as a preconditioner!

At the Frontier of Science

Non-randomized (deterministic) preconditioners are either easy to compute but ineffective (such as point-Jacobi or D-ILU variants) or effective, computationally expensive (ILU, IC) and in general not easy to parallelise.[4]

In this project we focused on the "Markov chain Monte Carlo matrix inversion ($(\mathrm{MC})^2$ MI)" method by Alexandrov and Straßburg[1,2] which computes an *effective* preconditioners and is easy to parallelise.

Additionally, stochastic projection (SP)[3] has been considered as a parallelisable alternative to deterministic iterative solvers. Its benefit is that it is very easy to implement and does not require a multi-stage solution process of the linear system, as in the case of $(\mathrm{MC})^2$ MI.

Both methods are stochastic, i.e., they rely on chance. There is no such thing as true chance in software - even though its behaviour often seems to be random. We thus additionally investigated the effects of pseudo-random numbers on our method of choice.

The Nuts and Bolts

$(\mathrm{MC})^2$ MI essentially treats a matrix as a sort of city map on which many random walks are performed. Hence, each worker which computes part of the preconditioner P, must have access to the whole matrix A (the map). If the matrix is large, the time to distribute a copy of the matrix to each worker dominates the computation by a very large margin, no matter whether CPUs or GPUs are used.

Figure 2: The network of a 2D FD Laplacian (common in physics simulations) after a few rows (and columns) have been cut out. This is no longer a suitable Markov chain.

At the beginning of the project, there was an MPI implementation of $(\mathrm{MC})^2$ MI, which suffered of the aforementioned communication overhead.

We began with a collection of ideas of how to reduce the amount of data to be distributed. Chief among them were the distribution of only a small subset of rows and the dropping of the entries of the matrix smaller than a prescribed threshold.

Both approaches lead to a preconditioner which will *never* fulfil Property 2! That is the property we drop and consider approximations of A^{-1} which are not good, but *good enough*!

Performance begins on paper! The usefulness of both ideas was first checked using pen and paper. We chose representative sample matrices (diagonal, 1D and 2D Laplacian), transformed them in accordance with the algorithm into graphs (read: city maps) and checked what happens if we apply our ideas.

To verify the conclusions obtained from theoretical analysis, and to facilitate future experimentation $(\mathrm{MC})^2$ MI as well as SP have been implemented in Python in serial and parallel versions. Both were then been modified to implement the proposed ideas.

To see whether the use of a pseudo-random number generator (PRNG) impairs our methods, a random number generator was written in Python to use the sequence of random numbers provided by the Fusion Group at the BSC. $(\mathrm{MC})^2$ MI was then run, for a chosen set of matrices, once using the PRNG and once using the random sequence.

The assessment of the usefulness of the resulting preconditioners was carried out using the generalized minimal residuals (GMRES) iterative solver.

Numerical Experiments - Evaluated

The reduction in the amount of data shipped to workers by using only a subset of the rows of the matrix was no good. The resulting preconditioners fulfilled none of the criteria stated above. The preconditioners were thus to be computed using matrices modified by dropping entries smaller than a chosen threshold. The modified matrices were used to compare the performance of $(MC)^2$ MI to its chief rival, modified sparse approximate inverse (MSPAI).

Prior to launching these numerical experiments, two problems in the original C code were rectified. One being a non-conformity to the MPI standard - which prevented compilation using the INTEL toolchain, and the other being the use of a PRNG not suited for parallel execution. The latter was replaced by adding an interface to Tina's random number generator (TRNG), which is a PRNG designed specifically for parallel applications.

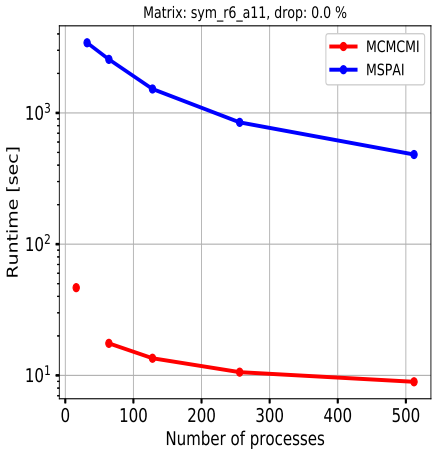


Figure 3: Scaling behaviour of $(MC)^2$ MI and MSPAI for a large matrix. Note the logarithmic time scale.

For large matrices, the use of $(MC)^2$ MI yields solutions which are a lot faster than when MSPAI is used, with the difference largest when 7% of the smallest (in magnitude) entries of the original matrix are dropped (exact number depends on the matrix). The resulting approximate inverse matrices are not good approximations of A^{-1}, but sufficiently good preconditioners.

The comparison of a sequence of true random numbers and pseudo random numbers has shown that for the purposes of stochastic methods there is no immediate benefit in using truly random numbers. A good PRNG will do the job just as well and is easier to obtain than a true random number generating device.

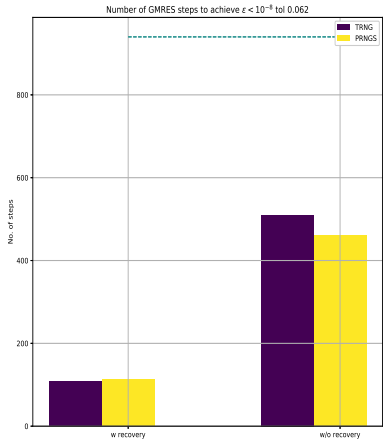


Figure 4: Number of steps required for GMRES to achieve a desired precision with a preconditioner computed using a true random number generator (TRNG) and a pseudo-random number generator (serial implementation, PRNGS).

Finally, a version of the SP method was implemented in CUDA, demonstrating its feasibility. This implementation has also demonstrated the usefulness of dynamic parallelism in GPU programming, limiting the hardware suitable for this method to GPUs of the Maxwell generation and younger.

Conclusion and Outlook

We have accelerated, using cold-blooded theoretical analysis and simple numerical experiments, the $(MC)^2$ MI method. Additionally, a set of scripts implementing $(MC)^2$ MI, SP as well as a whole slew of tests necessary for the assessment of the quality of the preconditioners have been created. These, along with the procedures established for the reproducibility of the results, will continue to serve as a basis for theoretical analysis and practical experimentation.

Though the PRACE Summer of HPC programme is now over, the development of the GPU version of $(MC)^2$ MI and investigation of the benefits of truly random numbers will continue in the upcoming months. Due to the high bandwidth and low latency of the PCIe bus, a GPU implementation promises to provide further speed-up for the computation of preconditioners using $(MC)^2$ MI.

Multiple possibilities to accelerate the method by modifying the matrix A have been devised by the end of the Summer of HPC and will be investigated by future generations of Master's and Bachelor's students.

Since the methods presented above are stochastic, meaning they require randomness, additional numerical experiments will have to be carried out to obtain statistically significant results.

References

[1] Alexandrov, V. N. (1998) Efficient parallel Monte Carlo methods for matrix computations

[2] Straßburg, J. and Alexandrov, V. (2014) Enhancing Monte Carlo Preconditioning Methods for Matrix Computations

[3] Sabelfeld, K. and Loshchina, N. (2010). Stochastic iterative projection methods for large linear systems

[4] Ferronato, M. (2012) Preconditioning for sparse linear systems at the dawn of the 21st century: History, current development, future prospects

PRACE SoHPC Project Title

Hybrid Monte Carlo Method for Matrix Computation on P100 GPUs

PRACE SoHPC Site

Barcelona Supercomputing Centre, Spain

PRACE SoHPC Authors

Anton Lebedev, Universität Tübingen, Germany


Anton Lebedev

PRACE SoHPC Mentor

Vassil Alexandrov, BSC-CNS, Spain

PRACE SoHPC Contact

Anton, Lebedev, Universität Tübingen
Phone: +49 176 96984852
E-mail: anton.lebedev@student.uni-tuebingen.de

PRACE SoHPC Software applied

NumPy, SciPy, Tina's Random Number Generator, Paraver

PRACE SoHPC More Information

www.numfocus.org
TRNG
Paraver

PRACE SoHPC Acknowledgement

I would like to extend my thanks to Vassil Alexandrov for his mentoring during the SoHPC as well as to Aleksander Wennersteen who has been my compatriot during my time at BSC. Additional thanks are due to Diego Davila, for his original parallel implementation of the MCMCMI method.

PRACE SoHPC Project ID

1701

Learning to move crowds

Aleksander Wennersteen

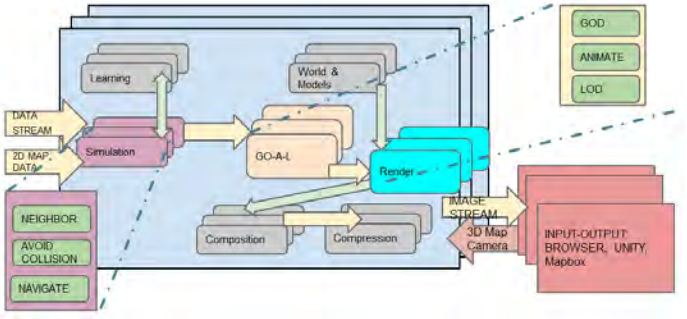
In an ever urbanised and densified world, the ability to predict the behaviour of very large crowds will become increasingly important. We have shown that Monte Carlo and Deep Learning methods used to play games and drive cars are also capable of controlling agents in a crowd.

Deep Reinforcement Learning, i.e. combining neural networks with reinforcement learning, has been adopted quickly in the last few years. First, Deepmind researchers used it to play Atari games with superhuman performance; then they combined it with a tree search algorithm to play Go — beating the best human player; lastly, it is used in conjunction with various computer vision techniques in self driving cars! These are the mighty tools we have chosen for our crowd simulation.

It is well known that adding Monte Carlo methods to deep learning can be very beneficial, both in terms of using Stochastic Gradient Descent (SGD) versions and randomising the minibatch samples. Similarly, Monte Carlo Tree search is superior to breadth- and depth-first search in the case of large graphs, such as a playing field, and in particular when there is a stochastic element to it such as in our crowd simulation. Monte Carlo methods also have an amazing benefit in that they are almost trivially parallelisable both in sub-problems and sub-areas. In the case of tree search and machine learning problems, it also greatly reduces the amount of work that needs to be done.

Crowd Simulation

Crowd simulations are interesting for several reasons. An interesting application is modelling how people react to external stimuli and potential problems that can occur when evacuating people from buildings and cities. Moreover, it is an interesting computer graphics problem because you have to render and control a lot of individuals, as well as the world they're in, with very limited resources for the scale. Below we see a diagram of the entire process.



Previous work done on the topic by the group at the BSC has enabled us to create visualisations with very high levels of detail and automatically generated diversity – all whilst keeping computational and memory costs low. The program is distributed across several CPU nodes with one master node required to combine everything, and utilises GPUs for the navigational aspect. They have also done work on the behaviour of different figures according to age, gender etc.

Artificial Intelligence

Artificial Intelligence (AI) can be defined as the study of intelligent agents; a device that perceives its environment and takes actions to maximise its chance of success at some goal.

As laid out in Russell and Norvig, a machine needs to possess several capabilities in order to be able to pass the Turing test for AI. Among these are understanding natural languages, storing information and using this information to reason, and to be able to learn and adapt to new situations. This last ability is what we often call Machine Learning (ML) – learning something without being told explicitly what to do

in each case. There are many approaches to this but the one we will talk about is Neural Networks (NNs). The basic idea of NNs is to model the structure of the brain, and to mimic how humans learn. We split the task into several smaller tasks and then combine all the small parts in the end.

With NNs, we try to enable the computers to learn from previous experiences and understand the world in terms of a hierarchy of concepts. Each concept is defined using those concepts further down the hierarchy. This hierarchy of concepts model allows the algorithm to learn complicated concepts by building them out of simpler ones.

The basic idea behind reinforcement learning is that we mimic how humans and animals are trained. If the agent does something good, say, doesn't crash, it gets a reward. If it does something bad, e.g. crashes or breaks a rule, it gets punished. There are many approaches to solve this, which all centre around giving the machine a function which assigns a predicted score for every action in a state. Q-learning and Deep Q-Learning discussed below are examples of reinforcement learning techniques.

Deep Q-Learning

Before we consider Deep Q-Learning (DQL) we need to explain regular Q-learning. Q-learning is a technique for solving Markov Decision Processes (MDPs). They provide a framework for modelling decision making in situations where the outcomes are partly random and partly under the control of a decision maker. It involves a Q-function which assigns a numerical value - the "reward" R, for taking a certain action a in state s. By keeping track of tuples of information of the form $< s, a, R >$, i.e the "state, action taken and reward", we can construct a so called (action) policy (function).

A policy is a rule which tells you that in a certain state you take some action. We are usually interested in the optimal policy which gives us the best course of action. A simple solution, but not very applicable in real world situations, is explicitly storing a table specifying which action to take. In the case of a tabular approach, we simply update the table whenever we receive a better reward than the one we already have for the state.

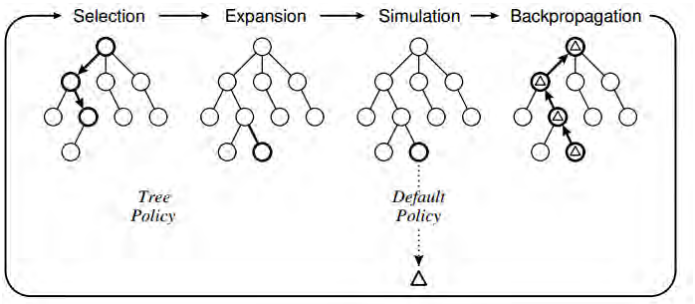
Compared to the dynamic programming problem of Q-learning, we do "neuro-dynamic programming" in DQL, meaning we replace the function from the dynamic programming problem with a neural network. We then use the stochastic gradient descent algorithm, together with the reward, to guide our policy function towards the optimal policy. An important thing to mention here is that we do not always want to follow the optimal policy, as this would lead to us not obtaining new information about our Q-function. Therefore, every now and then, we take some random action instead of the one given by the optimal policy. It has been shown to be beneficial to start with executing almost exclusively random actions in the beginning and then increase the number of predicted actions later. We can perhaps see the analogy to growing up. Children often do things we, as adults, know not to do. This is the way they learn - by exploring the state space and seeing if they are rewarded or punished.

DQL incorporates many interesting techniques for finding the optimal policy – such as a version of the stochastic gradient descent and minibatch updating. What is interesting with the minibatch update version utilised is the experience replay mechanism. Minibatch processing means that during training we not only take into account the latest event, but several. With experience replay, we store experiences in memory and randomly sample from these to fill our batch instead of the n latest experiences. This is advantageous as we decouple the experiences, and in this way bad decisions do not propagate through. This is also done by the brain's hippocampus, where recent experiences are being reactivated and processed during rest periods, like sleeping.

Tree Search

In the AlphaGo program, Monte Carlo tree search is implemented - as in most other Go playing programs, but guided by two deep neural networks. A very nice feature with the Monte Carlo tree search is that it allows us to almost trivially decompose our problem into smaller workloads. When the workload becomes large enough, this can make a huge difference.

Monte Carlo tree search is superior to breadth- and depth-first search in the case of large graphs, such as a playing field, and in particular when there is a stochastic element to it such as in our crowd simulation. The reason is simply that we explore the most likely best options first. This way, we can actually put a time limitation on how long we want to run the tree search for!



In the crowd simulation scenario, this gives the benefit of being able to explore finer grained paths than without. You might want to ask how we choose exactly, and for crowd simulation how we can know which paths are better. We can either adopt the approach taken with AlphaGo and let DL provide the intuition, or we can get the density of people, and use that as a probability distribution.

Here we see how MCTS has run different searches in the state space and identified the best path. At each stage, the result is stored so that the computational load is minimised by not having to re-evaluate the same node. There are several fundamentally different methods of executing MCTS in parallel: Leaf parallelization, i.e. parallel execution of many playouts from one leaf of the game tree; root parallelization, i.e. building independent game trees from the root; tree parallelization, i.e. parallel building of the same game tree, protecting data from simultaneous writes.

Sadly, MCTS parallelisation generally works best on CPUs and less well on GPUs. This is due to the lower GPU core processor speeds which lack the capability to effectively handle operations other than arithmetic operations.

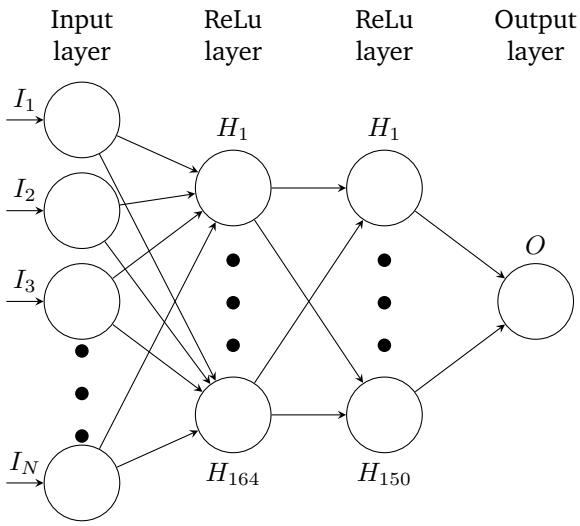
What have I done?

The original neural net used in the Atari paper started off with several convolutional layers to process images because the only input they gave it was an image of the game. The program then calculated its own reward function by seeing how the score increases. These of course, are features we have no need for. We found various blog posts modifying DQL to various uses and ended up modifying the collision avoidance source code by Matt Harvey.[3]

As such, getting the deep learning part up was very easy, and GPU support comes automatically with any deep learning framework. Of course, as with any code you find on the internet, it was far from optimised, and a couple of hours of work on critical parts of the code gave good performance increase. We also wanted several extra features, such as moving towards a goal and accelerating. This involved simply experimenting with the reward function, which is where the P100 GPU cluster at the BSC came in handy.

Now for the tree search. After battling for a while with the same game interface as I used for the deep learning, I moved over to a CPU-only version where we only ran searches on a grid. This is after all the same as in the final crowd simulation.

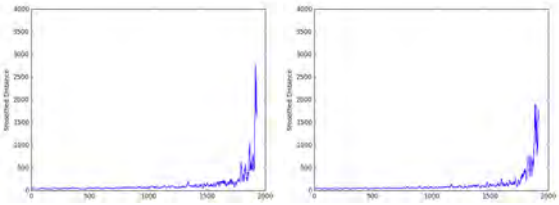
Now, the crowd simulation originated in computer graphics research. This means python goes out the window. C++, here I come! Recent work on the collision avoidance and navigation problem has also put a lot of CUDA code into the project, as well as the MPI parallelism across nodes (different computers). What does this mean? Well, we are moving the MCTS and deep learning to C++ and CUDA! These are words you never want to say. It is the reason why everyone does DL research in languages like Python. Even interfacing code is very much non trivial.



This is the final network I ended up with. Indeed quite similar to the original Deepmind DQL network without the image processing convolutional layers. The input to our network is a binary vector which gives us the location of the obstacles and the output is the direction to move in.

What did I find out?

My results are by definition hard to quantify. It is one of the few times in science where you simply have to look. For the deep learning, we had two objectives: move towards a goal and avoid colliding into both moving and stationary obstacles. Indeed, the agent learnt each skill very quickly and on average improved well with time. Surprisingly, although the loss-function indicated less learning with combined goal-searching and collision avoidance, with the right reward function, over-fitting ceased to be a problem. With collision avoidance only, the agent tended to prefer rectangular paths, and took a lot of training to care about the moving ones. The fact that the learning easily adapts to different situations suggests that it is stable.



Although both networks learn, here we really see the stochastic nature of deep learning come into play as the left agent ended up with a much better score where the networks and environments were identical.

As expected, MCTS works very well for navigating in complex circumstances such as finding the way out of a maze. It does however need some fine tuning for collision avoidance. However, this seemed to be something that improved with scale. Wanting to move on to trying the algorithms on a full scale, this was not pursued to great depth.

Whilst hard to transfer performance between Python + Pygame to heavily optimised C++ & CUDA code, the computational cost looks manageable. But how will it compare in the end against the current MDP solving strategy of using CUDA operations to find the least dense direction to go in?

References

[1] Silver, Huang et.al., Nature 529 (2016). Mastering the game of Go with deep neural networks and tree search

[2] Minh et.al.,Nature 518 (2015). Human-level control through deep reinforcement learning

[3] Matt Harvey, https://github.com/harvitronix/reinforcement-learning-car

PRACE SoHPC Project Title

Monte Carlo and Deep Learning Methods for Enhancing Crowd Simulation

PRACE SoHPC Site

BSC, Barcelona, Spain

PRACE SoHPC Authors

Aleksander Wennersteen,
The University of Edinburgh, UK

PRACE SoHPC Mentor

Vassil Alexandrov, BSC-CNS, Spain
Isaac Rudomin, BSC-CNS, Spain

PRACE SoHPC Acknowledgement

Firstly I would like to thank Vassil and Isaac for the project and all their assistance along with their group members. I would also like to thank the many different people I have met at the BSC for interesting conversations and for pointing me to sources where I found help.

PRACE SoHPC Project ID

1702

Combining the benefits of native MPI-like approaches with Big Data tools advantages to bridge the gap between HPC and Big Data

Bridging the gap between HPC and Big Data

Adrián Rodríguez-Bazaga

One of the hardest challenges of the current Big Data landscape is the inability to process huge volumes of information in an acceptable amount of time. The goal of this work is to ascertain if it is useful to use typical Big Data tools to solve High Performance Computing problems. We do this by exploring and comparing two distributed computing frameworks implemented on commodity cluster architectures. We compare the Apache Spark Big Data framework written in Scala with 'traditional' approaches. These use the distributed memory model with MPI on a distributed file system such as HDFS (Hadoop Distributed File System), and native C libraries that create the interface to encapsulate this file system functionalities. To be more precise, we have chosen the K-means clustering algorithm that will be executed on variable size datasets and will be compared in terms of computational time and failure resilience for both approaches.

Apache Spark [1] applications are written in Scala [2] and run on top of the JVM (Java Virtual Machine). Because of this, they cannot match the FPO performance of Fortran/C(++) MPI programs compiled to machine code. Despite this, it has many desirable features of (distributed) parallel application. These include fault-tolerance, node-aware distributed storage, caching or automated memory management (see Figure 1 for an overview of the Apache Spark architecture).

Yet we are curious about the performance limits of Apache Spark applications in High Performance Computing problems. By writing and executing referential code in C++ comparisons can be made.

We do not expect the resulting code to be, in terms of performance, truly competitive with MPI (or GPI-2) in production applications. Still, this kind of experiment may be valuable for Big Data engineers and programmers who may implement computationally demanding algorithms, such as Machine Learning or Clustering algorithms.

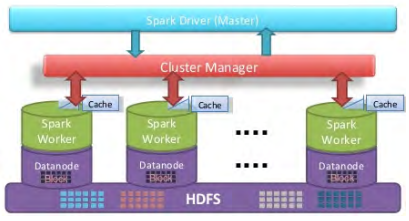


Figure 1: Apache Spark architecture

1 Introduction

1.1 Apache Spark vs MPI

Previous work [3] has shown that HPC frameworks based on MPI outperform Apache Spark or HDFS-based Big Data frameworks by usually more than by an order of magnitude for a variety of different application domains (see Figure 2 for an overview of the HDFS architecture).

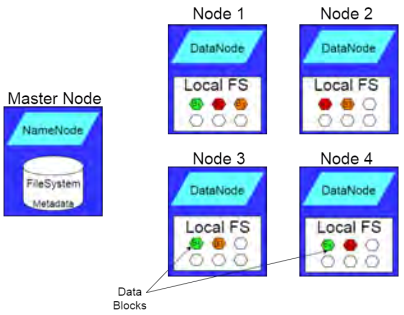


Figure 2: HDFS architecture

Unlike previous approaches related to the topic, we propose a fair comparison of similar implementations of algorithms in Spark, C++ with MPI and mixed MPI/GPI-2, using HDFS distributed data storage on top of cluster architecture based on commodity hard-

ware. The I/O from/to the distributed file system is arranged using native C libraries which create an interface to encapsulate the file system functionalities, while trying to preserve the fault tolerance characteristics of the HDFS architecture.

2 Proposed Fault Tolerance approach: Tools & Libraries

In this section we describe some of the tools which are used to provide our MPI and Spark-like applications with fault tolerance features.

2.1 GASPI

GASPI [3] is a communication library for C/C++ and Fortran. It is based on a Partitioned Global Address Space (PGAS) style communication model where each process owns a partition of a globally accessible memory space. PGAS programming models have been considered as an alternative to MPI for some time. The PGAS approach offers developers an abstract shared address space which simplifies the programming task and at the same time facilitates data-locality, thread-based programming and asynchronous communication.

2.2 GPI-2

GPI-2, which is the implementation of GASPI, takes full advantage of the hardware capability to perform remote direct memory access (RDMA). More importantly, it focuses on providing truly asynchronous communication to allow for overlap of computation and communication. The available thread-safe communication also allows multi-threaded applications with a fine-grained communication and asynchronous execution capability (see Figure 3 for an architecture overview).

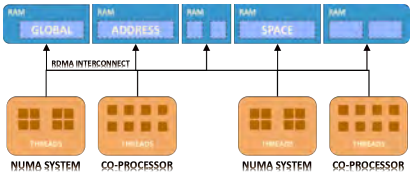


Figure 3: GPI architecture overview

GASPI was designed for the increasing need for fault tolerant applications and it supports application-driven fault tolerance on the process level.

Along with other mechanisms available in GPI-2, we used a ping-based fault tolerance and checkpoint-based extension to GPI-2 in order to compete with Apache Spark fault tolerance features to MPI-based applications. The procedure gaspi_proc_ping tests the availability of a particular GPI-2 rank. As the name indicates, a ping message is sent to a particular process. If a problem is detected, a GASPI ERROR is returned to which the application can react.

2.3 Libhdfs3

The Hadoop Distributed File System (HDFS) is a highly fault-tolerant distributed file system designed to run on commodity "low cost" hardware. HDFS provides high throughput access to application data and is suitable for applications with large data sets.

The Libhdfs3 library is an alternative implementation of traditional libhdfs, and is implemented based on the native Hadoop RPC protocol and HDFS data transfer protocol. It gets rid of the drawbacks of JNI, and it has a lightweight, small memory footprint code base. In addition, it is easy to use and deploy, and we use it to read and write directly from/to HDFS in our C++ implementations of the benchmarked algorithms.

3 Case Study 1: K-Means

K-Means clustering is a technique commonly used in machine learning to organise observations into k sets, or so called "clusters", which are representative of the set of observationse. Observations (S) are represented as n-dimensional vectors, and the output of the algorithm is a set of k n-dimensional cluster centers that characterize the observations. Cluster centers are chosen to minimize the within-cluster sum of squares, or the sum of the distance squared to each observation in the cluster:

$$min \sum_{i=1}^{k} \sum_{\vec{x}_j \in S_i} ||\vec{x}_j - \vec{\mu}_i||$$

where S_i is the set of observations in the cluster i and μ_i is the mean of observations in S_i.

This problem is NP-hard and can be solved with complexity O(n^{dk+1} log n). In practice, approximation algorithms are commonly used to get results that are accurate enough to within a given threshold and which terminate prior to convergence.

The main steps of K- means algorithm are as follows:

1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stabilizes.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers.

Figure 4 shows an illustration of the K-means algorithm on a 2-dimensional dataset with three clusters.

4 Experiments architecture

The experiments were run on the HPC cluster from the Matej Bel University in Banská Bystrica, Slovakia. The cluster constitutes of 24 computing nodes made up of IBM System x iDataPlex dx360 M3 servers. Each computing node is embedded with two Intel Xeon X5670 processors, 48 GB of RAM, and two hard drives from which a RAID 0 with a total capacity of 1.2 TB is created. Two computational nodes include nVidia Tesla M2070 GPUs with 448 CUDA cores and 6GB of RAM. The computing nodes are interconnected with 40Gbps InfiniBand. The InfiniBand network also provides connection of computational nodes to data repositories. The cluster also has 6 IBM iDataplex dx360 M4 hubs embedded with Intel Xeon E5-2670, 128GB RAM, 2x900GB HDD and three IBM iDataplex dx360 M4 nodes with Intel Xeon E5-2670, 64GB RAM, 2x900GB HDD and two graphics NVidia Tesla K20 cards.

There are in total 560 CPU cores with HyperThreading support. In addition there are 2 nVidia Tesla M2070 graphics accelerators each with 448 CUDA cores and 6 nVidia Tesla K20 graphics accelerators.

We deployed HDFS over the nodes to be accessible from Spark and C++ programs, where the input files were stored.

5 Spark vs Mixed MPI/GPI-2 implementations

In the Apache Spark MLlib implementation, each of the K-Means iterations is packed in one job, and that job is divided in two stages. In Figure 5 we

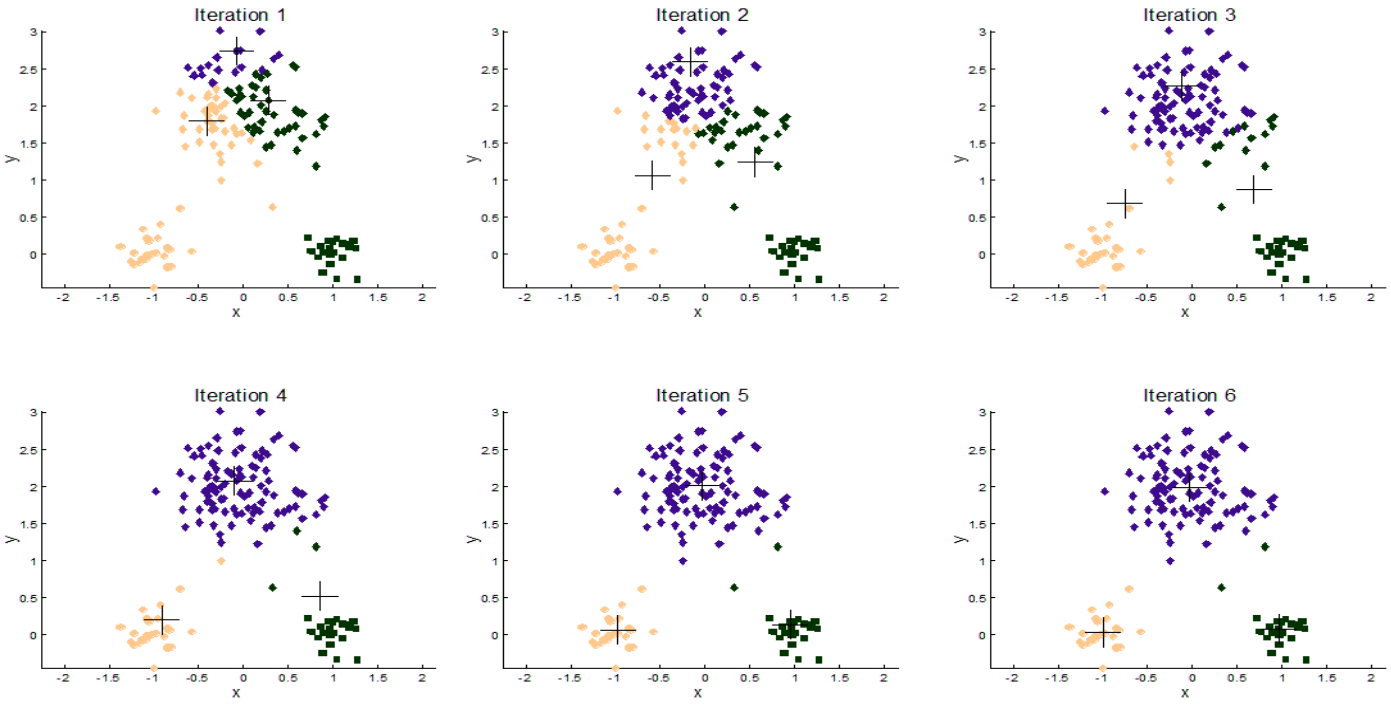


Figure 4: Illustration of the K-Means algorithm with three clusters (differentiated with colours). In each iteration the cluster centers (crosses) are getting more precise until convergence on iteration 6.

illustrate the Directed Acyclic Graph for the K-Means main job, with the two stages and their tasks.

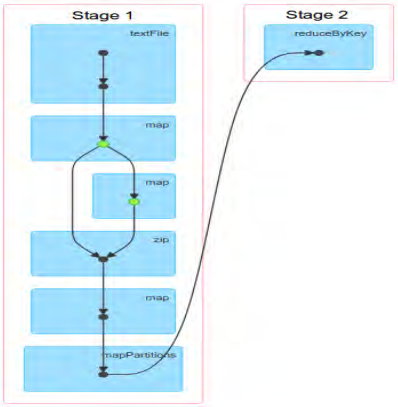


Figure 5: Directed Acyclic Graph for the K-Means main job in Spark.

The first stage is the map partitions task where input data is mapped. This means that with this step we convert each element of the RDD source into a single element of the RDD result by applying a split function to get the correct format to load the algorithm data with it. After that, the data is zipped and finally a mapPartitions is applied. This converts each partition of the source RDD into multiple elements of the result.

The second stage is a reduction phase, where the output from the first stage is used to generate a new RDD where all values for a single key are combined into a tuple - the key and the result of executing a reduce function against all values associated with that key.

5.1 Mixed MPI/GPI-2

In order to have a fault-tolerance approach, we have used a checkpoint-based methodology, saving the state at certain points of the execution, to allow recovery of that state in case of failure. The application needs to decide when it is more reasonable to perform a checkpoint. It must also detect a failure and proceed to a recovery process. In this recovery process, a spare node replaces the node that reported a failure, the last saved checkpoint (state) is read and the application loads it in order to continue it's execution without exiting. To have this feature, the application needs to start with a set of available spare nodes (the greater the number of spare nodes, the higher time resilience the application will show). These spare nodes are a set of nodes that at an initial stage are idle while the rest of the nodes are executing the application.

In the initialisation phase, the application provides a segment, offset and a size where the backup data will be placed in each checkpoint iteration. This is application specific and, in our case, we save the K-Means cluster assignments in that backup data.

A checkpoint policy and group must also be defined. Using the in-memory checkpoint approach, we follow an asynchronous, coordinates checkpoint approach. The coordination is achieved by ensuring global consistency of a snapshot using a GASPI barrier collective operation. The goal of this barrier is to ensure that there's at least one particular snapshot that is consistent on all processes. When a checkpoint is performed, the saved data is transferred to the mirror using GASPI asynchronous communication.

Carrying out a checkpoint is a two step procedure. First the checkpoint needs to start and then it has to be committed. To commit a checkpoint a global operation is used, ensuring the completion of a previously initiated checkpoint operation on all nodes. At each execution point, a valid snapshot thus exists, so the application can return to it if required. This commit operation has timeout to avoids blocking.

After the GPI-2 environment is set up, the fault detection and recovery process operations need to be set up.

The fault detection method we implemented, relies on a global GASPI barrier operation after each algorithm iteration. This barrier waits for all processes to reach the end of an iteration before reaching a timeout. If some node doesn't make it to this point then a failure has occured. In order to detect which nodes have failed and which not, we retrieve the GASPI ranks state vector,

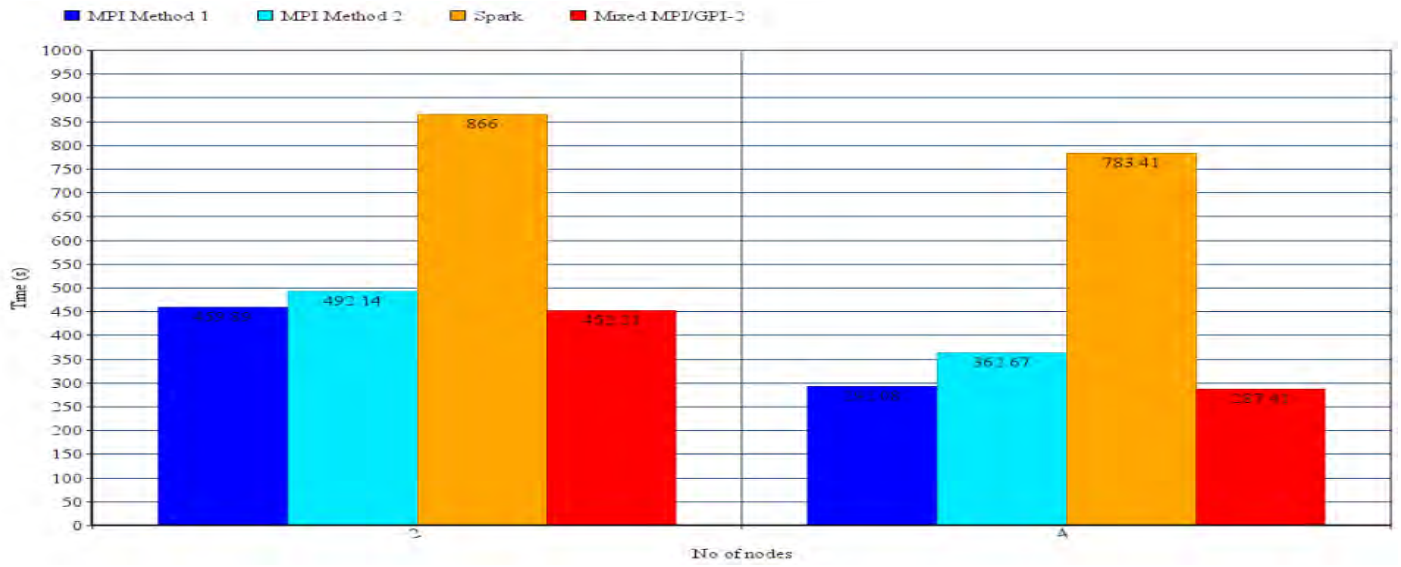


Figure 6: Execution time of K-Means on a varying number of nodes with 1 million points, 1000 centroids and 300 iterations over 1 million 2-dimensional points.

which returns the healthy or corrupt status of each node. After the problem is detected, the failure is communicated to all other running processes, so all the remaining healthy processes can enter in a consistent way to the recovery process.

The recovery process is divided into 3 actions, starting with bringing up a spare node to take the place of the failed one, creating a new ranks group and restoring the data from the consistent checkpoint saved at the GASPI working segment.

6 Results

In Figure 6 we depict the results for running K-Means with Apache Spark, first and second MPI methods and mixed MPI/GPI-2 method, with different number of cluster centers and maximum number of iterations, using the 1 million 2-dimensional dataset. We used 2 and 4 nodes for this benchmarking.

This experiment ran for 1000 centroids and 300 iterations. Apache Spark works, as expected, significantly slower, with a decrease in speed of 2.5 times in comparison with other approaches.

The first MPI approach turned to be slightly better in terms of computation time than the second MPI approach.

Also, the Mixed MPI/GPI-2 method, even with the added fault-recovery capability, it is slightly better than the second best implementation (the first MPI approach). GPI-2 features do not add any appreciable delay in the execution due to the fact it uses, in the logical level, the GASPI asynchronous methodology to perform all the checkpoint savings and fault detections.

References

[1] http://spark.apache.org/

[2] http://www.scala-lang.org/

[3] Anderson, M., Smith, S., Sundaram, N. and L. W., Theodore. Bridging the Gap Between HPC and Big Data Frameworks. *43rd International Conference on Very Large Data Bases, Munich, Germany. Procedia Computer Science, 53:121 – 130, 2015.*

[4] A. Raveendran, T. Bicer, and G. Agrawal. A framework for elastic execution of existing MPI programs. *In IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW), pages 940–947, May 2011.*

[5] S. Kamburugamuve, P. Wickramasinghe, S. Ekanayake, G. C. Fox. Anatomy of machine learning algorithm implementations in MPI, Spark, and Flink. Technical report, January 2017.

[6] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita. Big data analytics in the cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf. Procedia Computer Science, 53:121 – 130, 2015.

[7] A. Gittens, A. Devarakonda, E. Racah, M. Ringenburg, L. Gerhardt, J. Kottalam, J. Liu, K. Maschhoff, S. Canon, J. Chhugani, P. Sharma, J. Yang, J. Demmel, J. Harrell, V. Krishnamurthy, M. W. Mahoney, and Prabhat. Matrix factorizations at scale: A comparison of scientific data analytics in Spark and C+MPI using three case studies. In 2016 IEEE International Conference on Big Data (Big Data), pages 204–213, Dec 2016.

[8] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, and B.-G. Chun. Making sense of performance in data analytics frameworks. In Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, NSDI 2015, pages 293–307, Berkeley, CA, USA, 2015. USENIX Association.

[9] A. Raveendran, T. Bicer, and G. Agrawal. A framework for elastic execution of existing MPI programs. In IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW), pages 940–947, May 2011.

[10] Xiaoyi Lu, Fan Liang, Bing Wang, Li Zha, and Zhiwei Xu. DataMPI: Extending MPI to Hadoop-like Big Data Computing. IEEE 28th International Parallel & Distributed Processing Symposium, 2014.

PRACE SoHPC Project Title

Are Big Data tools applicable in HPC?

PRACE SoHPC Site

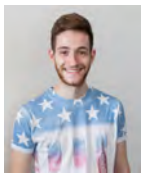
Computing Centre of the Slovak Academy of Sciences, Slovakia

PRACE SoHPC Authors

Adrián Rodríguez-Bazaga, Spain

PRACE SoHPC Mentor

Michal Pitoňák, CCSAS, Slovakia



Adrián Rodríguez-Bazaga

PRACE SoHPC Contact

Adrián Rodríguez Bazaga
Phone: +34 676 024 363
E-mail: adrianrodriguezbazaga@gmail.com

PRACE SoHPC Project ID

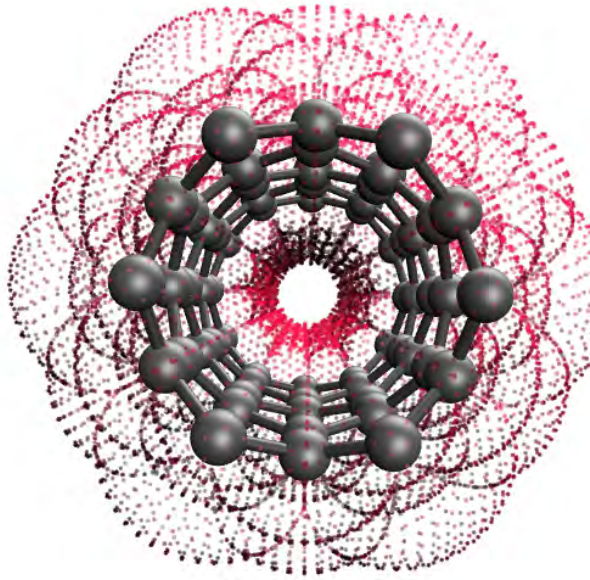
1703

PRACE SoHPC Acknowledgements

This work has been funded by the Summer of HPC program by PRACE (Partnership for Advanced Computing in Europe).

The computing was performed in the High Performance Computing Center of the Matej Bel University in Banska Bystrica using the HPC infrastructure acquired in project ITMS 26230120002 and 26210120002 (Slovak Infrastructure for High-Performance Computing) supported by the Research & Development Operational Programme funded by the ERDF.

Modelling Nanotubes in Parallel



Andreas Neophytou

Using MPI to parallelise the calculation of nanotube band structures.

Abstract

The software used in this project utilises the Hartree-Fock method to calculate the electronic structure of nanotubes. MPI was successfully implemented into the existing code in order to parallelise the diagonalisation of the Fock matrix. Distributing this part of the Hartree-Fock method across multiple nodes will enable larger nanotubes to be modelled.

Introduction

The band structure of a solid describes its electronic properties (for instance conductors and insulators have different band structures). Carbon nanotubes have been shown to display different electical properties depending on their structure, hence there has been much interest in modelling them.

By making use of the helical symmetry, it is possible to simplify the band structure of the nanotubes and the corresponding calculations. The Hartree-Fock method is used to calculate the band structure and energy of the nanotubes by approximately solving the Schrödinger equation. One step of the Hartree-Fock algorithm involves the diagonalisation of what is known as the Fock matrix. The goal was to implement MPI in the routine that performs this diagonalisation.

Helical Symmetry and *k*-Space

I'll spare you the full mathematical formalism of constructing a nanotube using its helical symmetry (for the purposes of this report I don't think it is necessary). Instead I'll try to explain why we do this and its relevance to my project.

Imagine you have a ribbon (the kind you might use to wrap Christmas presents with) and you wrap it around your finger to make a tube. This is the basis behind the helical symmetry of nanotubes. It is possible to say that any nanotube can be formed by rolling up a sheet of atoms. Now we can take a small unit cell (for instance 2 atoms) and replicate it N times in such a way that it follows the path of the ribbon. In this way, we can construct a nanotube using very small components. Then, using what we call **k**-points, define a particular symmetry representation for this nanotube as an infinite system. Their number is infinite in principle, though, in our case we can map the nanotube of N unit cells to N k-points. However, due to the symmetry of the nanotube, we need only consider $(N+1)/2$ as some of the **k** points are degenerate.

This is essentially, the crux of why we use the helical symmetry to define the nanotube. By doing so, we can construct any nanotube from small components and greatly reduce the number of **k** points we need to consider. This in turn will greatly reduce the size of the calculations carried out during the Hartree-Fock method.

Now, if you're not a chemist or physicist you're most likely confused as to what these **k** *points are. All you need to know is that we need them in order to calculate the band structure of the nanotubes and that we only consider (N+1)/2 of the N* **k** *points in our calculations.*

Solving the Fock Equations

Now I won't go into the theoretical background of the Hartree-Fock method, instead I'll skip right to the end. The final step of each Hartree-Fock iteration involves diagonalising **k** Fock matrices ($\mathbf{F}^{\mathbf{k}}$) using the following equation:

$$\mathbf{F}^{\mathbf{k}}\mathbf{C}^{\mathbf{k}} = \varepsilon^{\mathbf{k}}\mathbf{S}^{\mathbf{k}}\mathbf{C}^{\mathbf{k}}$$

Solving these equations gives us the information we need to calculate the band structure and energy of the nanotube being modelled.

The key point to understand is that we have **k** matrix equations and that each of these equations are independent of one another. Given this, it is possible to split up the problem into M blocks of **k** points. Our problem can then be spread over a number of independent nodes, with each node diagonalising M $\mathbf{F}^{\mathbf{k}}$ matrices and sending the results to the Master node. The goal of my project was to use MPI to do just this.

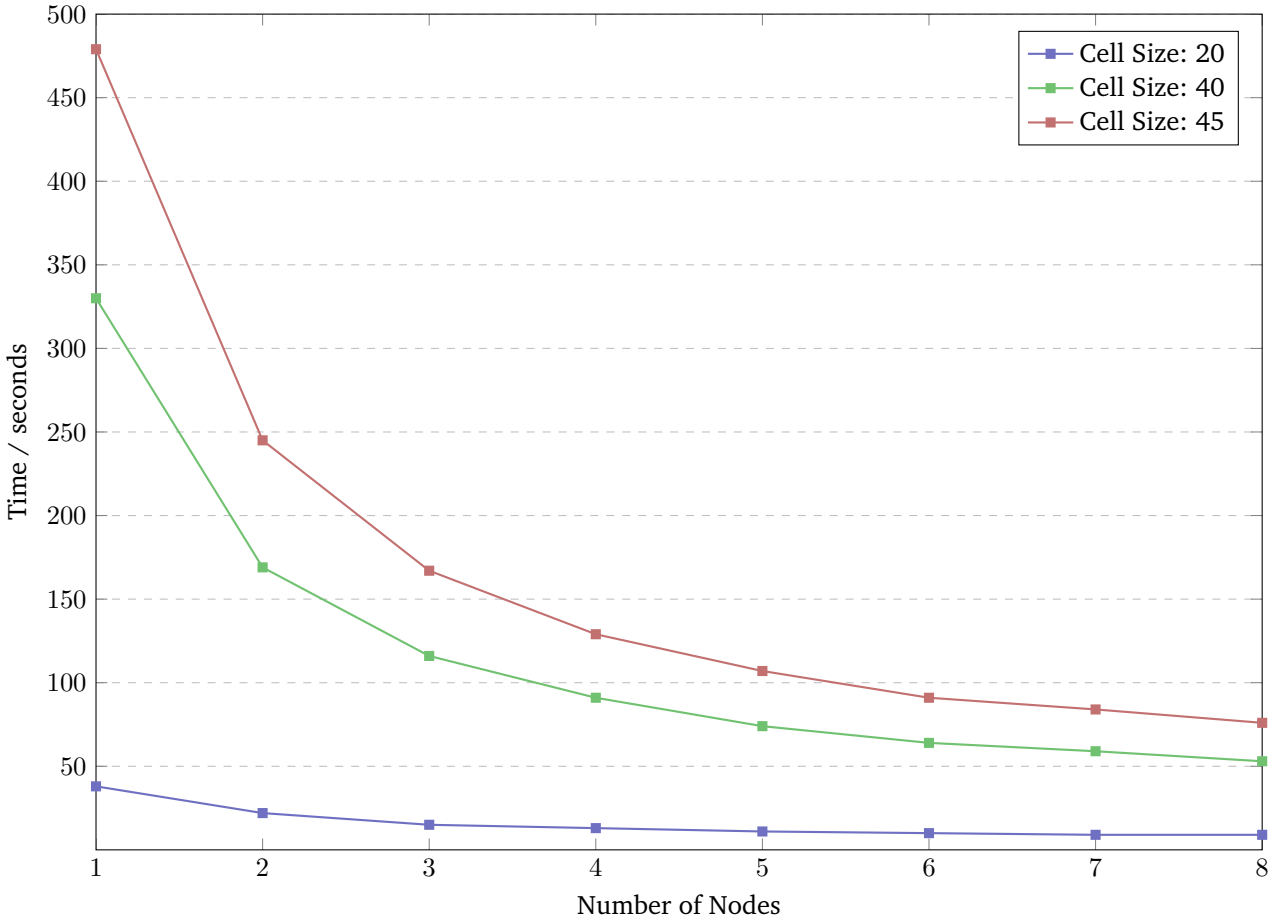


Figure 1: Time taken to diagonalise 813 Fock matrices on 1-8 nodes, with varying unit cell size.

A Master and his Slaves

In principle this is a simple project, but (as is always the case) it wasn't in practice. The most difficult part of this project was getting acquainted with the code I was working with, mainly due to the shear size of it (it has roughly 3,500 different routines). Slowly, I managed to get to know the parts of the code relevant for my project.

Now the code is also quite old - it is written in FORTRAN77 and I'm pretty sure it's at least a decade older than me, so that means it uses MPI that is also quite old. The code uses a Master/Slave system where the Master must manually 'wake' each Slave to run the code on the extra nodes. The Master calls a routine which allocates memory for the arguments of the subroutine you want to run in parallel, and then has each Slave call that subroutine. So, before I could get going with actually parallelising the diagonaliation of the Fock matrices, I had to create the Master diagonaliser and his Slaves. Once that was done, it was more or less (probably less) smooth sailing.

Results

To really see the benefit of parallelising the diagonalisation of the Fock matrices I would have to model a nanotube which has thousands of unique **k**-points. Unfortunately it would take too long to collect the data for such large systems (we're talking days for a single run to complete) so instead I ran the program for nanotubes that have 813 unique **k**-points, but have quite large unit cells (a larger unit cell corresponds to larger Fock matrices). The results can be seen in the figure above. As the unit cell becomes larger the time required to diagonalise each Fock matrix increases and so the effect of the MPI on the total runtime becomes more prominent. In summary the MPI works, but it works better for larger nanotubes.

What Next?

Clearly the MPI implementation has decreased the time required to diagonalise the Fock matrices, however it is also clear that it is only really needed for very large nanotubes (much larger than those used to collect the data shown above). It's now just a case of making use of the MPI to model much larger nanotubes.

References

[1] P. Baňacký, J. Noga, and V. Szöcs. 2013. "Electronic Structure of Single-Wall Silicon Nanotubes and Silicon Nanoribbons: Helical Symmetry Treatment and Effect of Dimensionality". *Advances in Condensed Matter Physics*. **2013**. Article ID 374371

PRACE SoHPC Project Title

Using the Helical Symmetry of Nanotubes to Calculate their Band Structure

PRACE SoHPC Site

Computing Centre, Slovak Academy of Sciences, Slovakia

PRACE SoHPC Acknowledgements

I would like to thank everybody at the SAS Computing Centre for their help, and for making the summer so very enjoyable. I would also like to thank Professor Noga for his help with the project.

PRACE SoHPC Authors

Andreas Neophytou, University of Birmingham, England

PRACE SoHPC Mentor

Jozef Noga, Slovak Academy of Sciences, Slovakia

PRACE SoHPC Project ID

1704

Andreas Neophytou

Visualising HPC System's Load

Petr Stehlík

Energy efficiency is one of the most timely problems in managing HPC facilities and this can be addressed at different scales and perspectives. Using Internet of Things technologies this project focuses on visualising data collected from the Galileo supercomputer in a web application.

The current monitoring system[1] consists of several layers which allow to aggregate at a single point, heterogeneous data sources which consist of computing elements, nodes, job scheduler and facility telemetry of the Galileo supercomputer located at CINECA, Bologna, Italy.

The system was named *ExaMon* (shorthand for Exascale Monitoring) and is built on top of the MQTT protocol.[2] It allows measured metrics to be sent to a central broker where received data are processed and stored in the KairosDB database which utilises the Cassandra cluster.

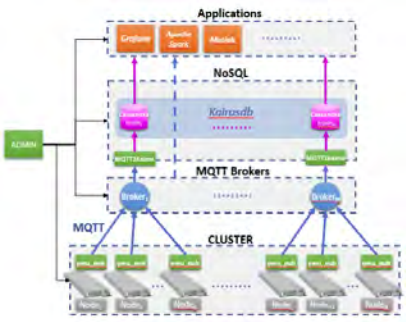


Figure 1: Examon Architecture

It allows post-processing of data in a time-oriented fashion, visualising them on a time-line and as a single number.

The current implementation uses the Grafana framework to visualise data stored in KairosDB. Grafana will be replaced by the results of the project which will create a dedicated web application for defined use-cases. It will also include a 3D model of a cluster room, showing various metrics of the whole HPC system with focus on energy consumption and efficiency.

Methods

The whole project can be separated into three phases.

Data Analysis

First phase is data analysis where the whole dataset of available metrics was presented, how they are distributed and eventually processed on the back-end. Datasets can be divided into multiple levels of aggregation:

Per-core level

The most low-level data can be found in a core's registers - such as IPS and Lx-cache misses. It also provides information about its load and temperature.

Per-CPU level

The two CPUs of nodes can each provide data about its C-states, energy counters and its frequency.

Per-node level

Most of the information available on node-level basis come from IPMI.[3] With this interface, we can access information about a node's utilization, multiple temperature sensors and average power consumption.

Per-cluster level

Galileo's cluster room was equipped with several environmental sensors. This dataset is not currently available due to technical problems.

Per-job level

This data is gathered using the PBS scheduler's hooks. This dataset is aside from previous ones since it points to allocated and used resources of a job submitted to the queue. This data are stored directly to the Cassandra cluster omitting KairosDB.

With each level, we can aggregate the lower levels (except job-level data). This is especially useful for core-level

data which are mostly too dense for any comprehensible visualisation.

Visualisation

The second phase was to visualise data stored in KairosDB in a simple yet insightful way using a lightweight web application. The ExaMon Web application, uses the Angular framework as its base on top of which several other libraries were used. Two libraries worth mentioning are Dygraphs and Bootstrap. Dygraphs produces powerful time-oriented charts utilising the canvas element in a web browser. Bootstrap is a CSS framework to produce a uniform user interface across the whole application.

Compared to Grafana, the web application feels more lightweight, faster and easier to use because of the prepared datasets which are being used. The balance between configurability and ease of use must have been found. We concluded the best way to achieve this was to enable time selection on given datasets but restrict configurability of the charts. In this way, the user is not bogged down with configuration and only focuses on prepared data.

If there is such desire to see other metrics, the Grafana framework is still available right next to the ExaMon Web. As an additional feature, compared to Grafana, we can perform more advanced queries using the KairosDB REST API.

Live Data

The last phase was to utilize the live stream of MQTT messages right in the ExaMon Web. Two use-cases were defined for the MQTT messages depending on their origin.

PBS Jobs

Each PBS job is assigned a unique job ID and goes through a specific set of events during its lifecycle. Using the ID, the user can subscribe to such MQTT messages and view various information on the ExaMon Web job dashboard. The dashboard also uses the Cassandra cluster in case the job is already finished and stored in the cluster. In this way, the user can see additional data about their job.

Using the job data, a user can view detailed information about allocated resources of a given job such as CPU load, system utilisation and others, as seen in Figure 2. With this information, the user can assess some conclusions about their program. How effective it is, where the slow parts are and even perform a top-down analysis for performance issues. They can also view how the program performed in terms of energy efficiency.

3D Model of Cluster Room

The second use-case is for the general public, partly for system administrators and is separated in two different parts. The first is very similar to the job dashboards where data are displayed using aggregated cluster level time-series charts. This allows to easily display, for example, the cluster's CPU load.

The other part is the most crucial in terms of interactive data visualisation. An accurate 3D model of the Galileo cluster was created using Blender and with the help of Blend4Web incorporated into the ExaMon Web. More integration was achieved utilizing WebSockets (using Socket.io library) that enables us to create a reactive paradigm model instead of a polling-based one. The model inside the page receives live data that has been published by the nodes and sent to the broker. A subscription model was developed to accommodate large amounts of visitors. The model then colours each node based on the minimum and maximum value of all received data. Weighted moving average was used in order to accommodate for sudden spikes in data using the given formula:

$$v_{new} = v_{current} + v_{previous} \times (1 - \alpha)$$

where $v_{previous}$ value is set to the first available value and $\alpha = 0.75$ as a default value was chosen based on short-term evaluation.

Results

ExaMon Web can be split to two major parts: 1) A tool for overseeing jobs submitted to PBS queue and 2) Cluster-level visualisation and analysis. Each of them are designed with a prepared use-case scenario.

Job Visualiser

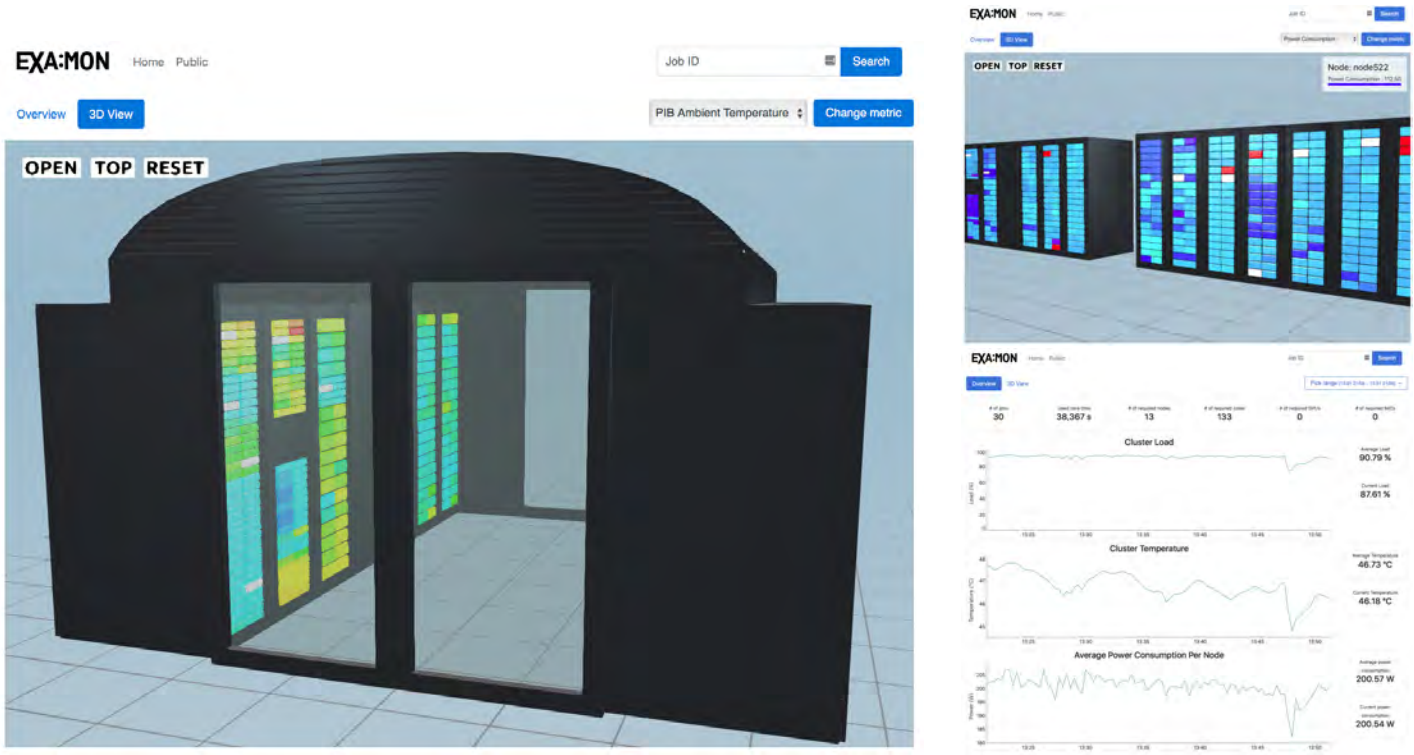
The main task of the job visualiser is to inform a user about their submitted job. The job ID is then used in the ExaMon Web job lookup. UI allows to query by manual input or, the list of active jobs and the last finished job are shown.

In order to capture and keep this data lifecycle, a new tool in the form of a Python class had to be developed. The `JobManager` subscribes to MQTT topics that send job-related data. This data is then stored in an internal volatile database. This way it can keep track of all submitted jobs. The `JobManager` is aware of the job's state and sorts the incoming job data according to their lifecycle.

The class is designed to be expandable and configurable. This means we can add callback functions to certain points of a job's lifecycle. This is used when a user subscribes to a live job and with each new received message with the same job ID, the job record in the database is processed and sent to the user via a WebSocket.

If the job is finished, the application will assess all information in the same way as for an active job and add additional information to the page as seen

Figure 2: From left to right: intro page with jobs lookup, currently running jobs and last finished job; job's info dashboard with a finished job; job's energy dashboard.



The public 3D model, its other possible arrangement and a public overview dashboard with a 30 minutes pre-selected time range.

in Figure 2.

In both cases, a user can view the performance and energy usage of their job. Each of the dashboards provide pre-selected interactive time-series charts.

Cluster Visualiser

The cluster visualiser is very similar to the job visualiser in terms of used components and the form of data. The main difference between them is that the cluster visualiser is mainly designed for the general public which does not run jobs on the cluster but is interested on how a HPC facility performs.

The introductory dashboard shows several charts aggregated to cluster level with averaged and last values right next to each chart. The user can also select a time range in which the data will be shown.

The second part is a precise 3D model developed in Blender and used in the Blend4Web framework which is incorporated into the ExaMon Web application. The model utilizes the same class as Job Visualiser to capture incoming MQTT messages but this time for metrics published by the cluster's sensors and tools. The manager computes the weighted moving average which are then available for querying.

Once a user opens the 3D model, the application subscribes to given metrics and waits for available data. A minimum and maximum value is computed and according to these values each node is colour-coded in usual colours ranging from red to blue in the HSL colour model.

The application then receives new data for each node as soon as they are made available by the manager and recolours nodes.

Users can operate the 3D model in the usual way (panning, rotation and zooming) and can see detailed information about each node by clicking it. This highlights the node and shows a legend.

Discussion & Conclusion

The ExaMon Web application was successfully developed and plans for public deployment are arranged. The application is already running on one of CINECA's virtual machine in a staging environment. The expected web application was delivered with several improvements and additions which will help the team at UNIBO to further develop the whole ExaMon system.

The application can be further developed with new dashboards such as *System Administrator* dashboard. Such a dashboard can help the system administrator of Galileo to quickly assess valuable insight of the whole cluster which would otherwise be very complicated and time-consuming.

Using this application, CINECA can show the general public how a supercomputer performs and what it takes to run it. The users of Galileo can comfortably and easily view detailed information about their jobs and how they behave in real-time and adjust their programs to perform better and more efficiently.

References

[1] Beneventi, Francesco, et al. "Continuous learning of HPC infrastructure models using big data analytics and in-memory processing tools." 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2017.

[2] Locke, Dave. "Mq telemetry transport (mqtt) v3.1 protocol specification." IBM developerWorks Technical Library (2010).

[3] Kaufman, Gerald J. "System and method for application programming interface for extended intelligent platform management." U.S. Patent No. 7,966,389. 21 Jun. 2011. APA

PRACE SoHPC Project Title

Web visualization of Energy load of an HPC system

PRACE SoHPC Site

CINECA, Italy

Petr Stehlík

PRACE SoHPC Authors

Petr Stehlík, BUT, Czech Republic

PRACE SoHPC Mentor

Dr. Andrea Bartolini, UNIBO, Italy

PRACE SoHPC Software applied

Angular, Dygraphs, Bootstrap, Blender, Blend4Web

PRACE SoHPC Acknowledgement

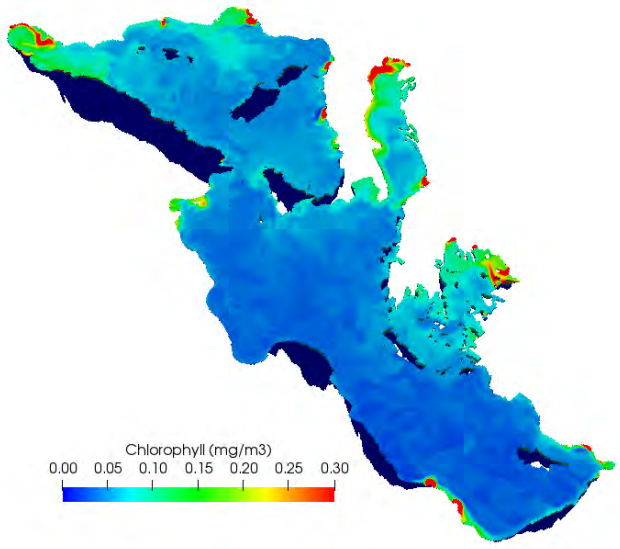
I would like to express my gratitude to all people at CINECA and UNIBO who made this project possible and to all people who helped during the development of ExaMon Web. I would like to also thank my family and close friends for all the support I received.

PRACE SoHPC Project ID

1705

Viewing the Mediterranean

Arnau Miró



ParaViewWeb is a very interesting tool for data visualisation. This project explores the capabilities of generating 3D visualisations using ParaView web capabilities for off-line browsing of archived data. A custom web application as well as a number of ParaView plugins have been developed in order to explore OGSTM-BFM data sets.

Most people have an idea of the ocean as a surface, however, the third dimension, i.e., depth, is extremely important since the sunlight penetrates down to a certain profundity. Moreover, vertical transport processes are key to biological dynamics.

Researchers at the Istituto Nazionale di Oceanografia e di Geofisica Sperimentale (OGS) use a highly optimised 3D transport reactions non-linear PDE applied to biogeochemical problems. Using OGSTM-BFM[1] (OGS Tracer Model - Biogeochemical Flux Model) code, OGS researchers are able to perform simulations to study the major biogeochemical properties in marine ecosystems, with a special focus on the Mediterranean.

For this reason, they consider that efficient three dimensional visualisation of numerical data is crucial to understand the dynamics of the sea[2] (Figure 1). Three dimensional views of phosphate,[3] which is considered one of the most important nutrients in the Mediterranean, and chlorophyll,[4] which is an important indicator of biological activity, are useful to understand the biogeochemical processes of the marine system. In addition, sea temperature and water mass circulation help us understand transport processes in terms of vortex and strain structures that can be detected by three dimensional visualisation methods.

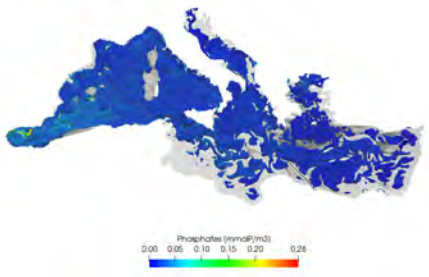


Figure 1: Chlorophyll iso-surfaces (0.25 - 0.30 mg/m^3) coloured by phosphates $mmolP/m^3$. Credits: "Generated with E.U. Copernicus Marine Service Information".

This is where ParaView comes to play. ParaView provides researchers with an open-source data analysis and visualisation tool. Data exploration can be done either interactively in 3D or programmatically using multiple python interfaces. ParaView can be easily customised using plug-ins written in XML language that have python code embedded. Recently, ParaView developers have begun to explore the possibility of using web browsers to visualise and explore data. The new API is called ParaViewWeb and it offers an easy access point to the data, for both researchers and general public.

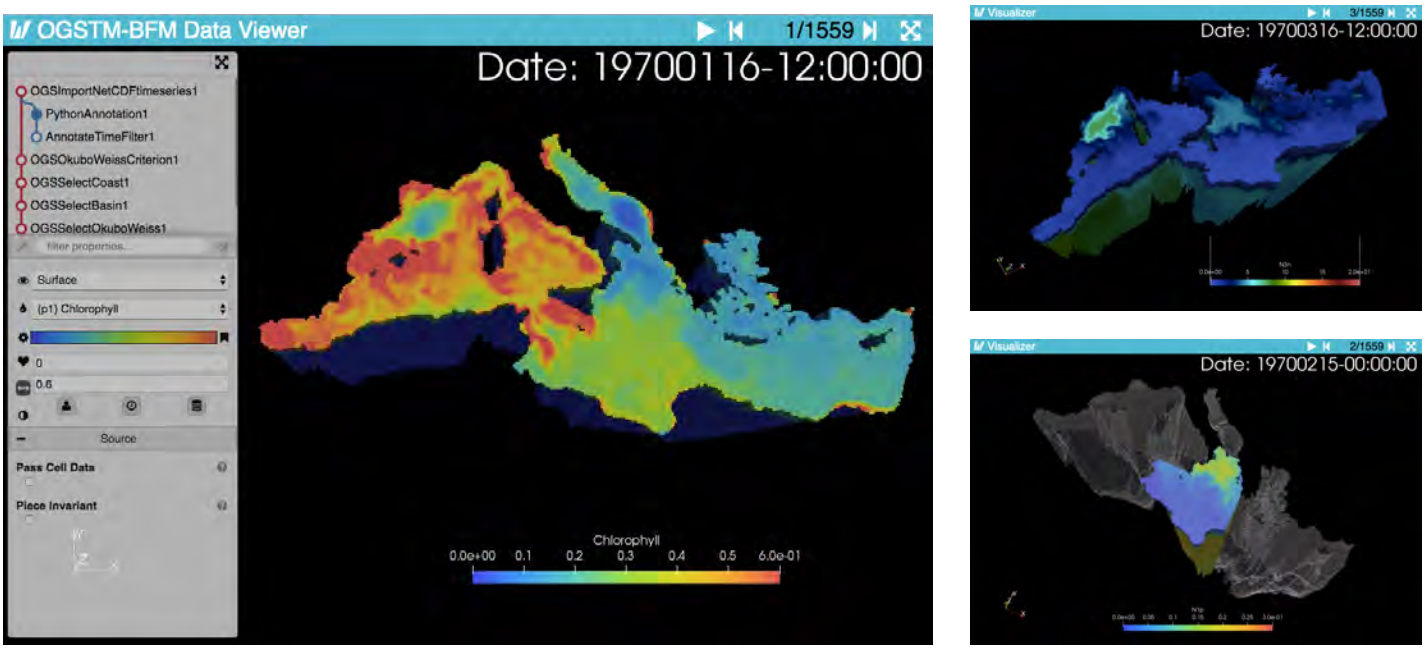
Unveiling ParaViewWeb

ParaViewWeb is a next-generation JavaScript library (React) web framework to build applications with interactive scientific visualisation inside a Web browser. These applications can use a ParaView backend for large data processing and rendering.

Precisely, ParaViewWeb basically consists of two approaches:

- Rendering at the server.
- Rendering at the client.

The first approach is better for those applications that require flexibility and large data processing, however, the application runs slower and for the largest datasets a powerful server is needed. Whereas the second approach uses WebGL techniques and runs faster on the client but requires the data to be preprocessed beforehand.



Examples of the ParaViewWeb tool developed for OGS. Left, chlorophyll concentration (mg/m^3) on the Mediterranean. Right top, nitrate concentration($mmolN/m^3$) on the *open sea* region. Right bottom, phosphate concentration ($mmolP/m^3$) on the Ionian basin.

ParaViewWeb workflow

The easiest way to understand how ParaViewWeb works is with a simple example. Let us imagine a person going to a restaurant for a meal, for example, a pizza. The client walks in and sits on the table and, at some point, the waiter will ask what the client wants. The client will place their order, ask for a specific pizza in the menu, and wait until it is done. Meanwhile, the waiter will take the order to the kitchen. The chef will look at the order and prepare the pizza. When the pizza is ready, the chef will request the waiter to bring the pizza to the client. The waiter will take the pizza and bring it to the client's table, where the client will be able to enjoy it.

The four players in this simple example: the client, the waiter, the order and the chef, are similar to those in a ParaViewWeb deployment. Figure 2 sketches the deployment of ParaViewWeb and illustrates the four players. There is a client, a researcher, with his personal computer connecting to a virtual machine that serves the webpage using Apache2. Apache2 plays the role of the waiter, it annotates the client connection in the file `proxy.txt` and communicates with the launcher. The launcher is a python code that runs in the background of the virtual machine. It plays the role of the chef, i.e., prepares the content that Apache2 requested by sending ParaViewWeb jobs to cluster machines through PBS jobs. It also annotates which port is using `proxy.txt` so it can be related with the user connecting to the website through web socket interface.

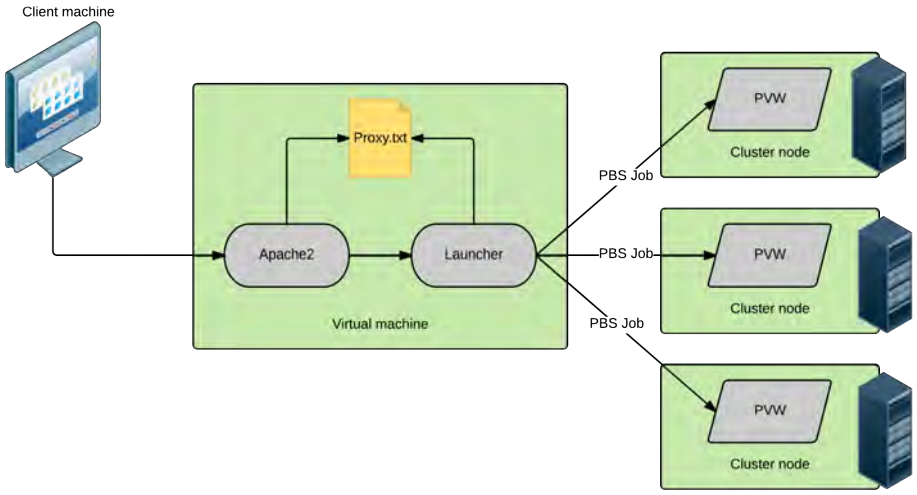


Figure 2: ParaViewWeb deployment in CINECA.

OGS specific plugins

Before visualising data from OGSTM-BFM, the output data-sets need to be imported into ParaView. In addition, other post processing operations either to generate new fields or to visualise a certain region of the Mediterranean are needed. Python plugins have been created to fill these tasks.

The pipeline must be built taking into account that the data is imported to a rectilinear grid for easier post processing. All operations involving a rectilinear grid must be placed right after the data is loaded. When selecting among different regions, the data is converted to an unstructured grid. For this reason, the selection plugins should be placed last in the pipeline.

NetCDF import plugins

The NetCDF importer plugin lets the user load a full OGSTM-BFM dataset and select the biogeochemical and physical variables to visualise. The mesh resolutions (either low, medium or high) must be previously preprocessed in order to use them with this plugin. The plugin selects the current time step by referring to the file date and time.

Rectilinear grid operations

So far, the only rectilinear grid operation plugin developed is the computation of the Okubo-Weiss[5,6] criterion for

oceanic turbulence. The plugin lets the user choose over which variable apply the filter and select an adequate filter coefficient, defined as

$$W_0 = k\sigma_W, \qquad (1)$$

where W_0 is the filtered Okubo-Weiss criterion, k is the filter coefficient and σ_W the spatial standard deviation of W.[7]

Selection plugins

Sometimes, OGS researchers need to visualise the data in a specific region of the Mediterranean. Selection plugins are provided to cover this need. Different zones of the Mediterranean can be selected - such as the coastline or the open sea, where the water is deep. Moreover, the different sub basins depicted at Figure 3 can be selected individually.

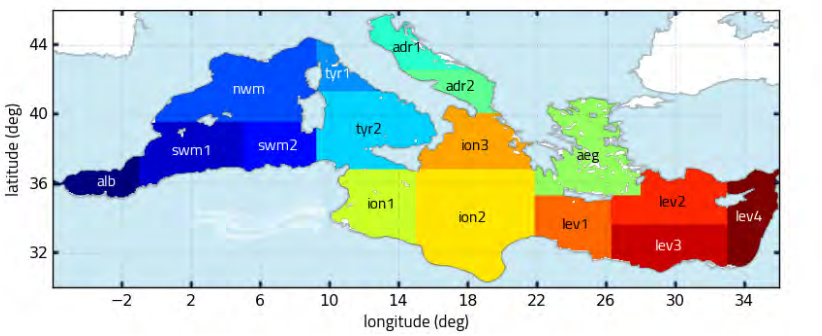


Figure 3: Sub basins in the Mediterranean. Credits: http://medeaf.inogs.it/nrt-validation.

The web application

A web application tailored to OGS's needs has been developed as part of this project. The application, named OGSTM-BFM Data Viewer is a fork of the Visualizer application, developed by Kitware.

The ParaView Visualizer is an interactive web data viewer, where the users can access the data remotely and build their own pipelines. It uses the render at the server approach, connecting to a python script running ParaView, which sends the visualisation to the web browser. ParaView Visualizer is entirely built in React.

However, ParaView Visualizer is too complex, therefore, the need for a specifically tailored application arises. OGSTM-BFM Data Viewer is build on ParaView Visualizer and has many of the functions deleted or simplified. A specific pipeline using OGS plugins is fed to OGSTM-BFM Data Viewer and cannot be changed, i.e., the pipe elements (or proxies) cannot be deleted or added. Proxies can be hidden or viewed by clicking on the coluored ball in the pipeline. Moreover, their properties can also be modified and affect the whole pipeline. Similar to the Visualizer, the colour map and the variable to be visualised can be easily selected and tailored to the user's needs.

OGSTM-BFM Data Viewer also features two annotations. The first one, at top right of the screen displays the date and time of the current visualisation, whereas the bottom right one the port in which the application is running. The latter is particularly useful in order to connect directly to the visualisation.

Conclusion

ParaViewWeb provides a number of features that are worth exploring. At one hand, the render on the server approach is used to directly render large data sets on the cluster machine. This is specially useful for viewing data without the need to download the whole dataset. The amount of interactiveness is also adequate to provide a valid scientific tool.

On the other hand, the python backend running ParaView provides enough flexibility to build complex pipelines and provide custom tools for specific sets of data. Due to the render on the server approach, the full power of ParaView can be used without penalties for the user's computer.

Moreover, being programmed in React offers reusable code for other similar applications.

ParaViewWeb is still under development, providing modifications and improvements in each new version. It is worth to explore in the future, a possible connection between ParaView Catalyst and ParaViewWeb in order to visualise in real time the data sets in a browser.

References

[1] Lo Vichi M., Lovato T., Lazzari P., Cossarini G., Gutierrez Mlot E., Mattia G., Masina S., McKiver W. J., Pinardi N., Solidoro C., Tedesco L., Zavatarelli M. (2015). The Biogeochemical Flux Model (BFM): Equation Description and User Manual. *BFM version 5.1*. BFM Report series N. 1, Release 1.1, July 2015, Bologna, Italy, http://bfm-community.eu, pp. 104

[2] Lazzari, P., Teruzzi, A., Salon, S., Campagna, S., Calonaci, C., Colella, S., Tonani, M., Crise, A., (2010). Pre-operational short-term forecasts for the Mediterranean biogeochemistry. *Ocean Sciences*, 6:25–39. doi:10.5194/os-6-25-2010.

[3] Lazzari, P., Solidoro, C., Salon, S., and Bolzon, G. (2016). Spatial variability of phosphate and nitrate in the Mediterranean: A modeling approach. *Deep Sea Research Part I: Oceanographic Research Papers*, 108: 39–52.

[4] Lazzari, P., Solidoro, C., Ibello, V., Salon, S., Teruzzi, A., Béranger, K., Colella, S., Crise, A., (2012). Seasonal and inter-annual variability of plankton chlorophyll and primary production in the Mediterranean: a modelling approach. *Biogeosciences*, 9:217–233. doi:10.5194/bg-9-217-2012.

[5] Okubo, A. (1970). Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences. *Deep-Sea Research*, 17:445–454.

[6] Weiss, J. (1991). The dynamics of enstrophy transfer in two-dimensional hydrodynamics. *Physica D*, 48:273–294.

[7] Isern-Fontanet, J., García-Ladona, E. and Font, J. (2006). Vortices of the Mediterranean: An Altimetric Perspective. *American Meteorological Society*, 36:87–103.

PRACE SoHPC Project Title

Web visualisation of the Mediterranean

PRACE SoHPC Site

CINECA, Italy

Arnau Miró

PRACE SoHPC Authors

Arnau Miró, UPC ESEIAAT, Spain

PRACE SoHPC Mentor

Dr Paolo Lazzari, OGS, Italy

PRACE SoHPC Contact

Luigi Calori, CINECA
Phone: +39 051 6171 509
E-mail: l.calori@cineca.it

PRACE SoHPC Software applied

ParaView

PRACE SoHPC More Information

www.paraview.org

PRACE SoHPC Acknowledgement

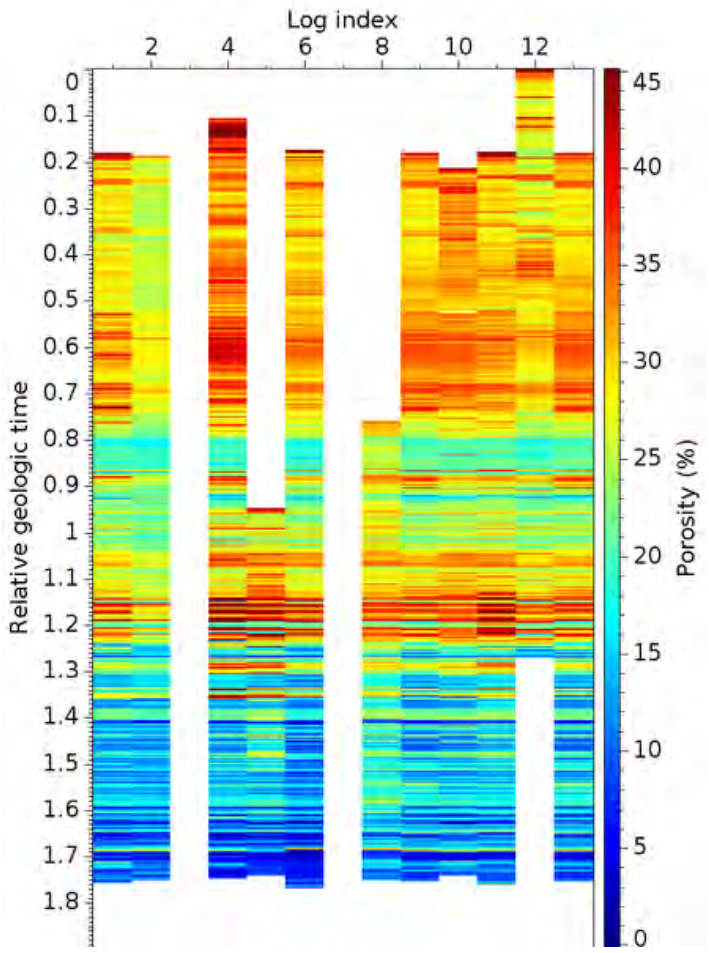
The author would like to acknowledge the help of the OGS researchers Dr. Paolo Lazzari, Dr. Stefano Salon and Dr. Cosimo Livi, as well as the CINECA mentors Dr. Luigi Calori, Dr. Massimiliano Guarrasi and Francesca Delliponti for their invaluable help during this project.

PRACE SoHPC Project ID

1706

Automated Well-Log Correlation Method visualized with real data

Dimitra Anevlavi



The aim of this project is to evaluate an existing state of the art Well-Log Correlation method through visualisation techniques with real borehole data. This effort will contribute to the creation of a Python-based Machine-Learning application that will facilitate geologic interpretation of basin areas of interest.

The main purpose of this project was to develop utilities that would enable the pre-processing of a wide range of real borehole data so that the visualisation of the well-log correlation among the wells would be substantially facilitated. The correlation was conducted using an already existing method suitable for the simultaneous correlation of any number of logs. The suitability of the well-log correlation for our project, was based not only on the fact that it gave promising results, but also due to the fact that it is open source and well documented. The evaluation of this method through the visualised results and the implementation on realistic case studies was the final output.

Introduction

The idea of using automated algorithms to facilitate the work done by geologists - for geological interpretation and lithology identification purposes, is not new. The prevalence of machine-learning research and the application of neural network methods in a wide variety of research fields proves that it is time to revisit the topic. In any case, before we can implement any of these new technologies, we should first familiarise ourselves with the challenges of the work conducted by geologists nowadays.

One aspect of geological interpretation includes facies classification methods. By facies, we refer to "the character of a rock" expressed by its formation, composition, and fossil content. Facies classification methods are widely used by geologists, since a basin of interest can be characterised by its lithology formation. With a variety of well log recordings at their disposal, a group of researchers have already implemented a novel method of facies classification using an inception convolution network[1]. The output of the machine-learning implementation produced during their study has been compared to the work of geologists, but the results were not very accurate. These may be due to human factors and the uncertainties of well-log data measurements which cannot be completely avoided. Furthermore, a lack of

neural network algorithm training data, seems to have been the main reason for the loss of accuracy. A suggestion for improvement, would be to train existing neural networks with realistic case studies. In general, the problem of efficient and accurate Facies Classification is considered to be one of the most challenging problems geologists need to solve, and up to date methodologies that take into consideration the wide variety of parameters that affect the results have not been fully developed.

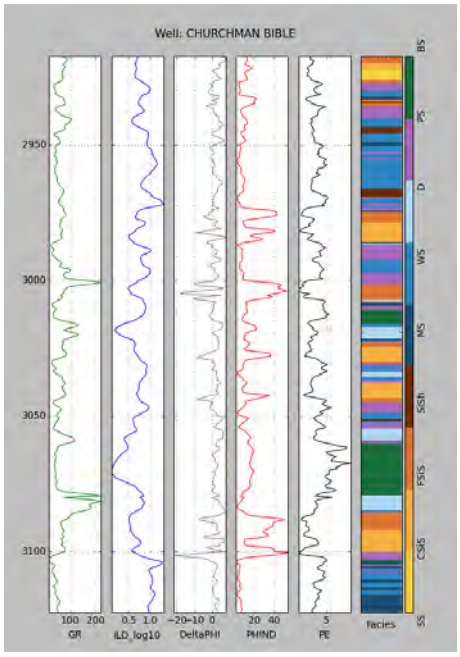


Figure 1: Example of colour-coded facies classification output based on the measurements of geologic functions of depth[m]

Another aspect of the work that geologists conduct, concerns well-log correlation methods[2]. During the process of well logging, scientists record various properties of the rock/fluid mixtures penetrated by drilling into the earth's crust. After this process is finished, it is up to geologists to determine corresponding depths among well logs, that are both geographically and geologically related to each other. It is often the case that corresponding depths represent a single geologic time in which sediments of similar properties were deposited over large areas. This interpretation is crucial as far as the identification of basin areas of interest are concerned, in the search for petroleum, gas, minerals or other substances. Recent research in well log correlation showed that an improved method of simultaneous correlation of multiple well logs is possible. An example of this procedure is shown in the graph above, where in a colour-coded manner, 13 boreholes have been correlated in the means of relative geologic time.

At this point, if we take into account the results of recent geologic interpretation research, we can properly introduce the concept of this project. Learning from experience, machine-learning algorithms are able to discover abstract representations, and to understand the data in terms of a hierarchy of concepts. This concept seems very suitable for the high dimensional data of well-log measurements that geologists need for both pre/post-processing.

Final Product

The aim of this project is to contribute to the preliminary concept and design of a Python-based application of Machine-Learning methods that will facilitate geologists in the process of well-log correlation. The machine learning method used for the identification of facies could be used as an effective solution to the well log correlation problem. An idea at this point would be for researchers and developers to combine previous work in this field.

A crucial first step, and the final goal of this project, would be the evaluation of already existing state of the art Well-log Correlation methods based on real well-log data. After this step is complete, researchers would be able to make alterations to the convolution network formerly used to implement facies classification, so that it would solve the problem of well-log correlation.

The Methods that I used

For the purpose of the project, well-log data from the Netherlands and Dutch sections have been analysed and pre-processed.

The pre-processing was conducted with code developed in Python that takes into account different file formats and parameters. This is important as borehole data are typically found in .las, .lis formatting. The identification of neighbouring groups of well-logs that are of greater interest to geologists proved to be challenging, as well as the selection of parameters upon which data extraction were to be implemented.

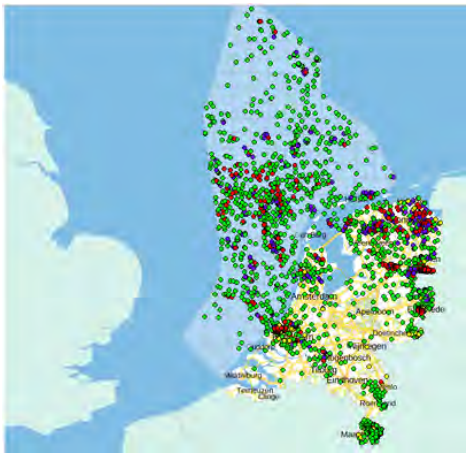


Figure 2: Map of borehole data available in the Netherlands and the Dutch sector of the North Sea continental shelf.

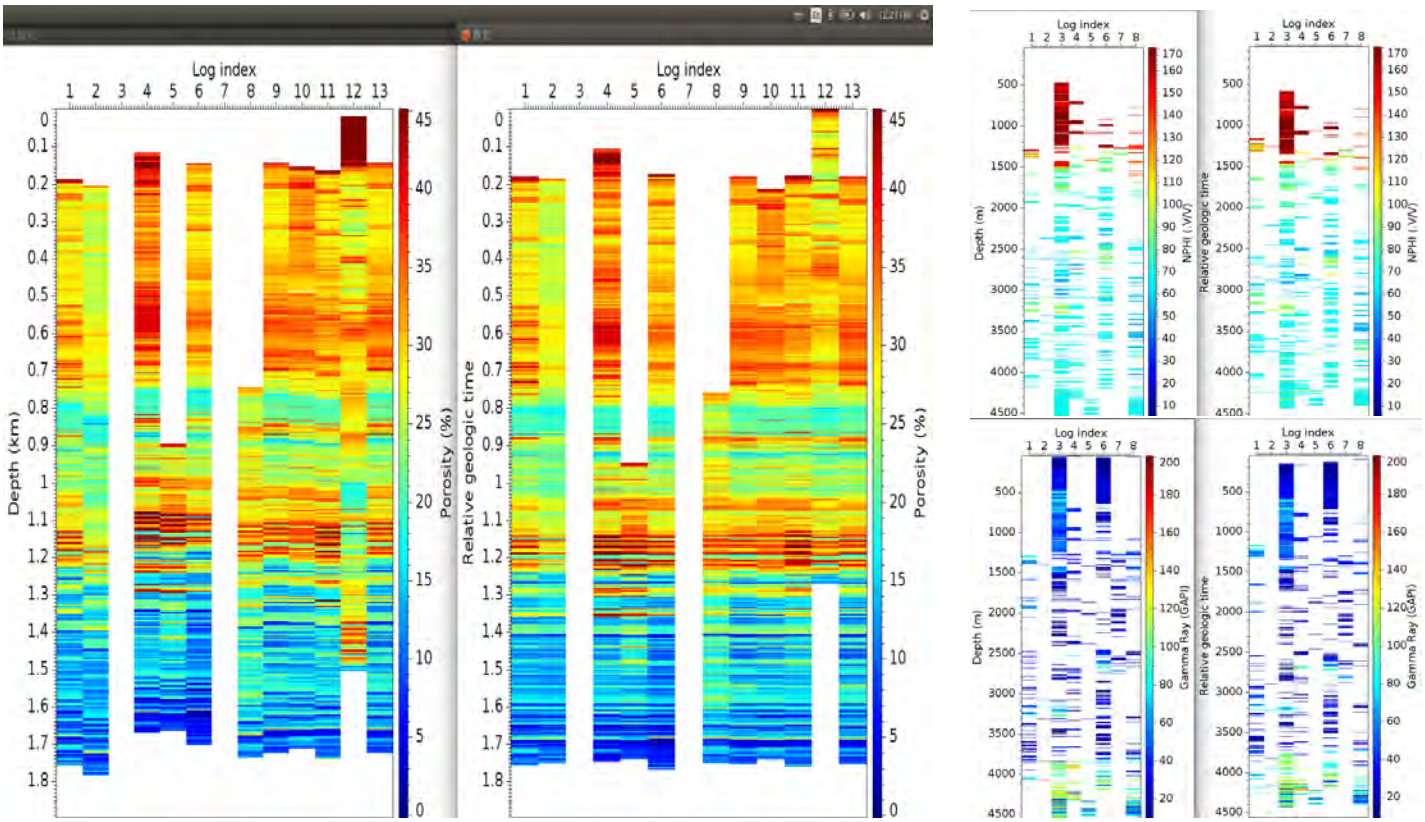
After pre-processing data in a format that would allow for the input files for the Well-Log Correlation code to be easily prepared, we were able to produce visualisations of the Correlation among the wells.The pre-processing utilities provide geologists with a wide variety of options:

- Data extraction based on specific geologic parameters and storage both in different formats as well as in binary form.
- Plotting options with interpolation features.
- Merging of borehole data for data entry suitable for the Well-Log Correlation method in the Mines Java Toolkit open source repository.

This enabled the visualisation of the simultaneous and automated well-log correlation of real borehole data. It was also done in a way which could allow researchers to produce case studies which could later evaluate the correlation method itself. Whether this Well-Log correlation method would be the base for the neural network algorithm or not would be decided by the experience of geologists. In any case, stay tuned for the results!!

Results – What did I find out?

The generation of case studies of well-log groups was significantly facilitated by the utilities developed in Python during this project. Only a small amount of well-log data-sets has been examined so far, but nevertheless the visualisation of the correlated wells has proven to be successful. It is understandable that the overall evaluation of the method,



The graphs above depict examples of well-log correlation. On the left we have in a colour-coded manner the measurements along the depth, and on the right the measurements as a function of relative geologic time. The right part of each of the three graphs are actually the results of geological interpretation!

as far as limitation and accuracy are concerned, will be conducted by experienced geologists in the near future.

Discussion & Conclusion

It is important to note that the resources available in the open source Mines Java Toolkit are not suitable, without alterations, for the correlation of a large amount of data due to memory limitations in the application. The variety of features that facilitate geologic interpretation was impressing, but the documentation proved to be challenging to users unfamiliar with these methodologies.

The compatibility of the final Well-Log correlation application should be improved as well to provide for more features and development. The final results of our work is available as an open source repository linked with the Mines Java Toolkit library. In this way, improvements and additions are more than welcome.

Suggestions for future research include the concept of machine-learning and pattern recognition techniques. As stated above, researchers have already implemented methods of deep convolution networks for the challenging problem of facies classification and the results are quite promising.

Machine-Learning Methods

A great challenge for researchers would be the adaptation of the already existing neural networks, so that they solve the well-log correlation problem. Literature and open-source projects that focus on such machine-learning methodologies, suggest that lack of realistic case studies could be the reason for poor performance and results.

Our project in this way would prove to be a good generator of various case studies based on real data, suitable for the training of newly developed neural networks. If you would like more information, feel free to check out my video presentation as well.

References

[1] Loralee Wheeler & Dave Hale (2014) Simultaneous correlation of multiple well logs

[2] Valentin Tschannen, Matthias Delescluse, Mathieu Rodrigeuz and Janis Keuper (2017) Facies classification from well logs using an inception convolutional network

PRACE SoHPC Automated Well Log Correlation Method visualized with real data

Official title of the project

PRACE SoHPC Edinburgh

University of Edinburgh, EPCC, Scotland

PRACE SoHPC Author

Dimitra Anevlavi, [National Technical University of Athens] Greece

PRACE SoHPC Mentor

Dr Amy Krause, EPCC, Scotland

Dimitra Anevlavi

PRACE SoHPC Software applied

Python, Jython, Java

PRACE SoHPC Acknowledgement

I graciously acknowledge the welcoming members of EPCC, my mentor for her support, as well as Ms. Rosa V. Fulgueira(BGS) and Mr. Stavros Arsenikos(BGS) for their help and enthusiasm for the project at EPCC as a collaboration with the British Geological Survey.

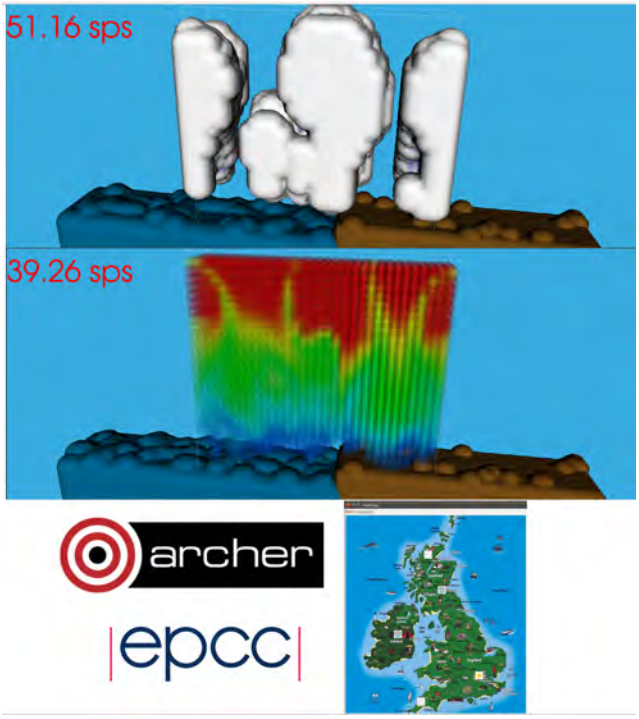
PRACE SoHPC Project ID

1707

A new atmospheric model which the scientific community can use to simulate clouds has been developed. A demo of this model has been created as an interactive outreach tool.

Weather forecasting for SoHPC

Sam Green



An interactive weather forecasting demo is a useful tool to allow the public, undergraduates, and even some scientists to appreciate and understand how their decisions on the initial conditions of a simulation can have an impact on the simulation's outcome.

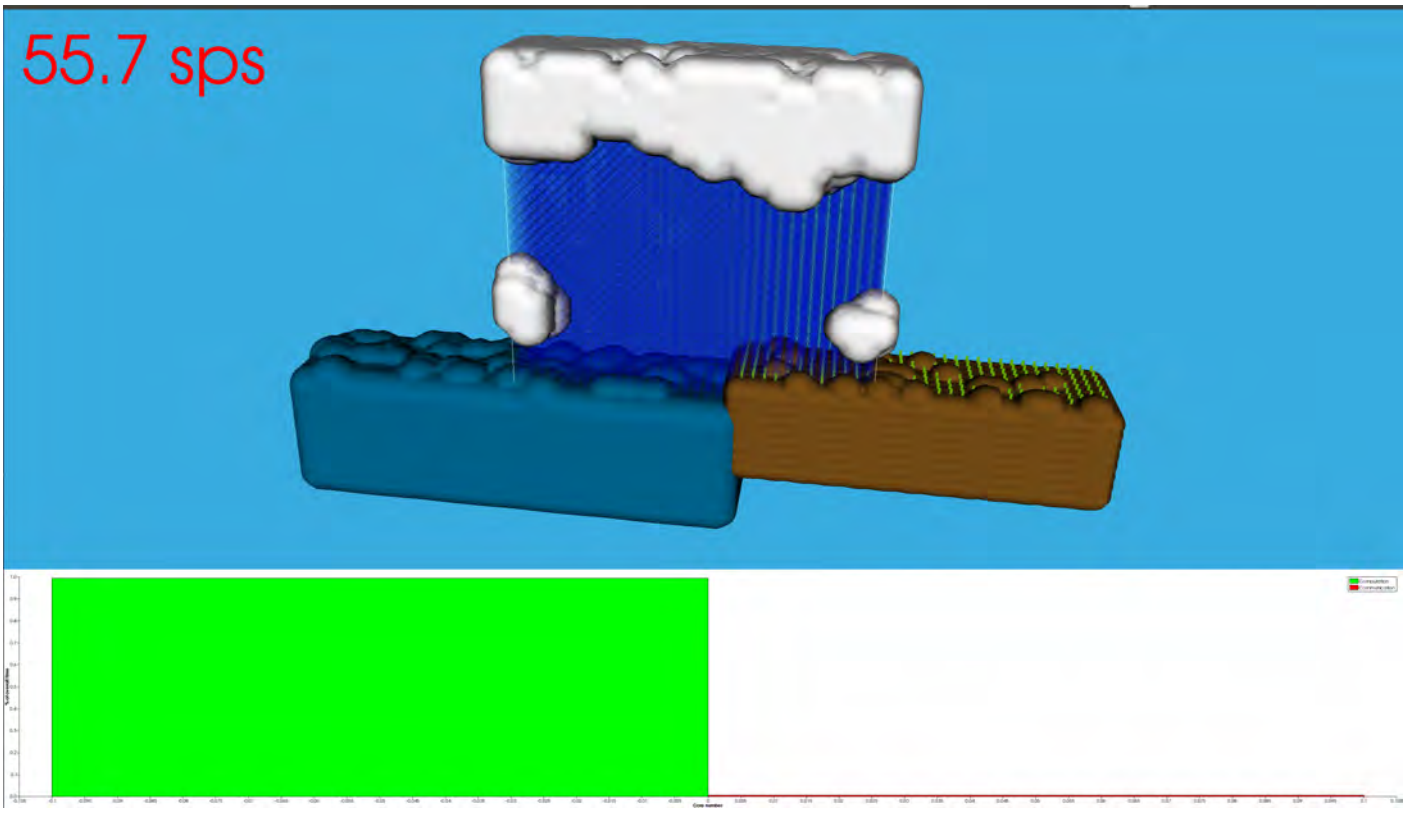
MONC (Met Office NERC Cloud model) is a new atmospheric model developed in conjunction with the UK Met Office. It is being used by the scientific community to simulate the atmosphere, clouds and turbulent flows. It is a highly scalable Large Eddy Simulation (LES) model that can simulate the atmosphere at high resolution, with an accuracy of up to 10s of meters. The model was built to run on thousands of cores on supercomputers like ARCHER, but can also run on smaller systems like Wee-Archie (a mini supercomputer EPCC built out of Raspberry Pis, see Figure 1) for outreach purposes. An interactive outreach demo of this model was created during the 2016 SoHPC programme (by Tomislav Subic) to show how choosing different initial conditions (such as temperature, pressure, accuracy, scientific method, etc) can affect the outcome of the simulation. This demo is then run on Wee Archie where the user can visualise the development of the weather/simulation in real time.

Figure 1: Wee-Archie: a small 64 core cluster made out of 18 Raspberry Pis.

This current demo is considered successful for public engagement and showing the public how a weather simulation is set up and run. However, it could also be very useful for atmospheric, computational and HPC education targeted at undergraduates. A tool which illustrates the impact of the different choices of calculation accuracy, what scientific method to use (for modelling flows in the atmosphere), and how many computer cores/hours to use would be hugely beneficial. Especially showing users the simulation in real time, both in terms of the actual progression of the weather but also the performance and scalability of the system.

Therefore, this current outreach demo was modified into an interactive visualisation tool which can be used for education in weather prediction and illustrating the impact that different scientific choices can have upon the fidelity of results and performance of the models. The initial feedback from events in the previous year was used to expand the outreach demo.

Several upgrades have been made to this outreach demo throughout the course of the project. These include a new window - separated from the visualisation window, that now contains all the settings and a map of the UK, live weather data being fed into the simulation, and a temperature and pressure view of the atmosphere and clouds.



The visualisation generated from the outreach demo showing the formation of the clouds and rain over land and sea. The bottom plot shows the ratio of computation and communication time for each core.

Development

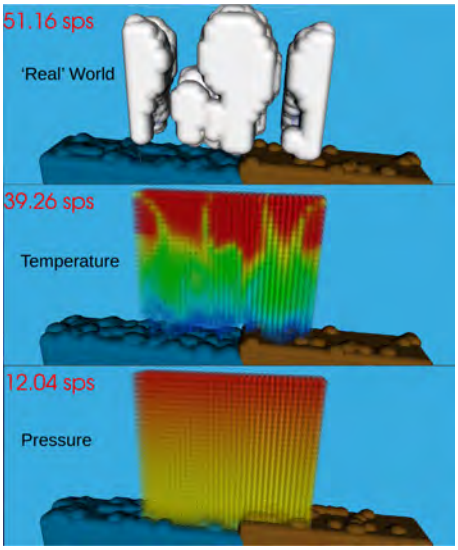
EPCC have developed a framework to run demos on Wee Archie which includes the transfer of configuration files, executing a job and getting the generated data back. This outreach weather demo was created by a student during the Summer of HPC 2016 so the basic framework was already in place for me. A few of the tools that were used to create this framework were a Python (2.7 and 3), Visualization Toolkit (VTK), and the WxWidgets GUI framework. My task was to use these tools to update the core demo and make it more user friendly and useful for outreach events. At the beginning of the project, the aim was to update the core demo for it to be used for education. However, once the demo got accepted to feature at the NERC showcase event (a public outreach event), it was decided to focus on outreach instead.

I will now describe the general pipeline of the application and the changes that have been made. The demo visualisation now takes up the whole display window (previously one third of the window was occupied with buttons) and allows the user to see the simulation in more detail. When the simulation is started, Wee Archie (or whatever system is running the demo, i.e. a laptop or supercomputer) creates a data file which is used to generate the visualisation. This occurs when the data file is used by VTK to render all the objects that are specified by the render script.

Clouds are rendered using filters that create a surface around points in the atmosphere that have a certain amount of cloud mass. Rain is set-up as a separate quantity and the points in the atmosphere that contain rain mass are rendered as blue spheres. To distinguish between the different amount of rain mass at each point, transparency and colouring are applied from light to dark blue. Most of the rain begins within the clouds (similar to clouds in our atmosphere) so you can only see it when it starts raining. This visualisation is meteorologically accurate and we can see how clouds form and move and how rain will start to accumulate around the clouds and then fall. Also, land (brown) and sea (blue) have been rendered to give a landscape for the weather to evolve over. An animation has also been included which simulates the effects of rainfall on crops. If enough rain falls, crops will grow out of the land, but too much rain will kill the crops. A decomposition grid is also rendered in the simulation to show how the atmosphere is split up amongst the cores. The plot at the bottom of the visualisation window shows the ratio of the computation (green) and communication (red) time for each core.

Before the simulation can be started and the visualisation made, the user first needs to open the new interactive settings window. There is a button built into the the top of the visualisation window that does this, and once the settings window opens, you are greeted with a map of the UK and Ireland. Four buttons, with icons, have been placed

over the UK (and one in Ireland) and when one of them is pressed, a weather simulation will start. The buttons are all linked to a live MetOffice data feed, known as DataPoint. This is a service to access freely available MetOffice data feeds, from hundreds of weather stations all over the UK, in a format that is suitable for application developers. The button's images relate to whatever this data feed is saying. So, a picture of a cloud relates to cloudy weather, rain to rainy weather, and sun to sunny weather. Also, the four buttons over the UK are positioned in the Highlands, Edinburgh, London and Cornwall. This positioning is to give the demo the option of different landscapes upon which to simulate the weather. So if you click on the icon in the Highlands, it will create a mountainous landscape. If you click on the icon in Edinburgh, it will create a city next to the sea landscape. London, a city with a river running through it. And in Cornwall, land and sea. There is also an 'Advanced' tab in the settings window that contains several buttons to change some of the simulations input parameters (for example wind power, temperature, number of cores), however since I didn't set these up I won't discuss them further.



We are now also able to see the temperature profile of a cloud and rain system. This gives us a whole new outlook into the temperature of different parts of the atmosphere as the clouds form and the rain begins to fall. As the temperature profile evolves with time, you can see that the heat in the atmosphere is undergoing some turbulence. We can also see the pressure profile of the system too. Throughout the course of the simulations I ran, nothing much seems to change with the pressure over time. However, with other simulation setups (for example a mountainous terrain) the pressure will be more interesting to visualise. A useful feature of this is that we can switch between the temperature, 'real' world, and pressure view to see what parts of the atmosphere and clouds are associated with high or low temperatures and pressures. MONC was already developed to calculate the temperature and pressure of the simulation but the data values were not set up to be passed into the outreach demo. This data was then set-up as another separate quantity and the points in the atmosphere that contain temperature/pressure mass are rendered as spheres. To distinguish between the high or low temperatures or pressures at each point, colouring is applied from blue to red (blue being low values and red being high). The addition of these two views is a critically important part of the visualisation as it provides the user with extra information on how the weather is evolving. They add a lot of extra realism to the demo and are more interesting to visualise than the existing real world view.

Discussion and conclusions

During the course of this project I have further developed a demo that can be used at outreach events to help the general public understand the weather, simulations, HPC and parallelisation. It gives an idea of what kind of simulations run on HPC systems, how large workloads are distributed between cores, and what kind of factors can influence performance. At the start of the project, the idea was to simply expand the current capabilities of the outreach demo. It was not until after a few weeks and discussions that the idea of bringing live weather data into the project was thought of. Another important aspect of being able to access live weather data from DataPoint that hasn't been looked into is that not only can we have an image of the current weather, but we can feed this data (it contains values such as temperature, pressure, wind speed, humidity, etc.) into MONC to simulate the current outside weather. So when you click on the button over Edinburgh, whatever the weather the button's image is representing is the weather that will be simulated (i.e. a sun image will produce no clouds initially). However, this has not yet been done and is a task that can further advance this project.

This outreach demo also has the potential to be used as an education tool - which was one of the hopes at the start of this project. It can be used in classes, training courses, with early stage meteorologists (or any stage depending on their background), other scientists, and with college students who want to use and understand HPC and its computational methods. This will help them understand the impact of the various input parameters on the results and performance of HPC codes. Most scientists/PhDs/developers who start to use HPC systems are not fully aware of the trade-offs they have to make and this demo can give them an inside look into the fact that HPC performance depends on a number of different factors. Also, the decomposition grid is an interesting way of showing how data and the workload is distributed between all the cores being used. It is also useful to demonstrate that having more cores is not always the solution if your workload isn't intelligently distributed between the cores.

Since this demo is still really only in it's early stages of development and because of the way it is written and the tools it uses, it is simple to improve and change many aspects of it in the future. Also to set up different versions targeting different audiences. It will be interesting to see how this application will be used and what other outreach applications it can inspire. It has already taught me many things about HPC and Python that I can bring forward into other simulation projects throughout the course of my PhD and personal hobbies.

PRACE SoHPCProject Title

Interactive weather forecasting on supercomputers as a tool for education.

PRACE SoHPCSite

EPCC, University of Edinburgh, Scotland

PRACE SoHPCAuthors

Sam Green, [DIAS] Ireland

PRACE SoHPCMentor

Dr. Nick Brown, EPCC, Scotland


Sam Green

PRACE SoHPCContact

Ben, Morse, EPCC
E-mail: B.Morse@epcc.ed.ac.uk

PRACE SoHPCSoftware applied

VTK, wxPython

PRACE SoHPCAcknowledgement

Thanks to my mentor Nick Brown for giving me access to his code MONC and for all his help during the course of the project.

PRACE SoHPCProject ID

1708

ARCHER, UK National Supercomputing Service - online visualisation of current and historic supercomputer usage

HPC Usage Data

Jakub Nurski

The ARCHER supercomputer has an active user base of around 3000 users. It is desirable to be able to analyse its usage data and provide some visualisation to help track what is working well and what needs attention.

Edinburgh Parallel Computing Centre is a supercomputing centre based at the University of Edinburgh. It houses the UK national supercomputer service called ARCHER. There are a lot of projects and communities that rely on this machine in their everyday work. It is a crucial, scientific instrument for many different people, not only academics of the University of Edinburgh, but its extended 3000 active user base!

EPCC would really like to keep track of usage data of ARCHER. This will allow them to check that everything on ARCHER is running as expected, and will also allow them to analyse changes and trends in the community as time has progressed.

The goal of this project was to use historical ARCHER usage data and illustrate them in a way that is easy to understand, and accessible - ideally through an online visualisation tool.

Final Product

Built using Angular, the final result of the project is an online visualisation tool that allows ARCHER users to explore, sort, group and compare the usage data.

Project Title Online visualisation of current and historic supercomputer usage EPCC University of Edinburgh, United Kingdom Authors Piotr Nurski

Climate Change Visualisation

Edwige Pezzulli

Visualisation plays a crucial role for sharing climate results in an easy-to-understand manner. It is thus important to improve graphic tools and visualisation concepts. Presenting climate results to the general public in an appealing and accessible manner will also help in raising public awareness on a crucial issue such as global warming. This project aims to develop tools capable of visualising climate data, producing 2D graphs and animation over time.

Climate is changing. The global temperature is rising - about 1.1 degrees Celsius since 1880.[1] The oceans also absorb the heat, so they are warming,[2] with a growing acidity of surface waters, which has increased by 30% since the Industrial Revolution.[3] The arctic sea ice has declined rapidly both in its extent and thickness over the last several decades.[4,5] As a consequence, the global sea level is rising - about 20 cm in the last century.[6] Glaciers are retreating everywhere,[7] and the snow is melting earlier in the Northern Hemisphere.[8] Moreover, the number of extreme events, such as high-temperature peaks and intense rainfall events, is increasing.[9–11]

Generally speaking, the climate has always slowly changed, and it varied for several reasons. Its changes have been caused by biotic processes, astrophysical variability (such as differences in solar radiation received by the Earth), or volcanic activities. However, the scientific community jointly agrees that the main cause of the current global warming trend is anthropogenic meaning it is human induced. In fact, human activity has increased the concentration greenhouse gases - such as carbon dioxide, methane and nitrous oxide, which are produced by burning fuels like coal and oil and by clearing land for agriculture and industries. Why are greenhouse gases so important? Well, the heat-trapping nature of carbon dioxide and other gases was demonstrated in the mid-19th century, so there is no question that increased levels of greenhouse gases must the Earth to warm up in response. To give you an idea, carbon dioxide levels raised from 280 part per million to 400 part per million in the last 150. years[1].

What's going to happen in the near future? To answer this question, we need climate models. The real world is a complex system made up of a large number of interacting components

[1] https://climate.nasa.gov/

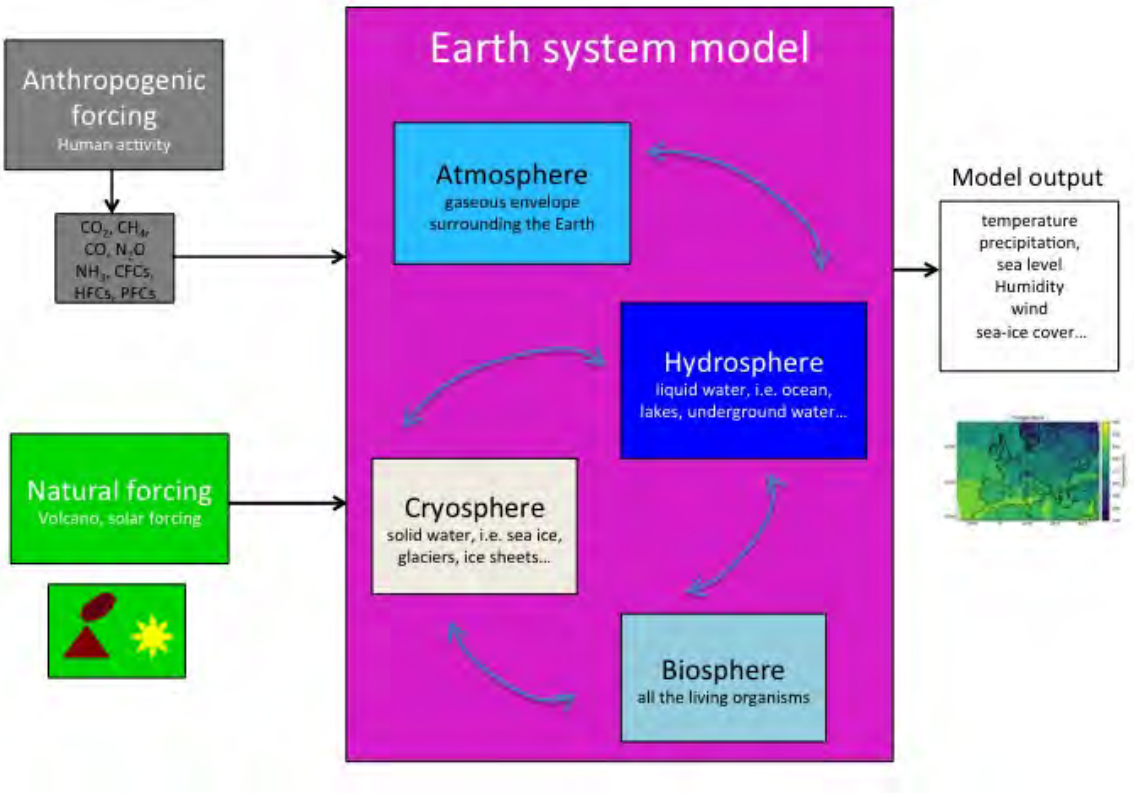


Figure 1: Schematic example of a climate model.

whose behaviour is non-linear. Therefore, it is necessary to put all the physical components together and see how the entire system interacts and evolves.

Climate models

A climate model consists of a set of equations, which describe the interaction between the solar radiance, the atmosphere, oceans, land surface and ice (see Figure 1). But if the human activity plays a fundamental role in climate change, how can scientists make credible future climate projections? They assume four different future emission scenarios- the so-called Representative Concentration Pathways (RCP). This consists of 4 possible greenhouse gas concentration futures.[12] The optimistic curve assumes that greenhouse gas emissions peak now, with a subsequent declining trend, while the in the pessimistic case emissions continue to rise throughout the 21st century.

In my project I am handling output data from the regional climate Weather Research and Forecasting (WRF) model. WRF is a numerical weather prediction system for both atmospheric research and operational forecasting needs, which can generate atmospheric simulations using real data (observations, analyses) or idealised conditions. It has been developed by the National Center for Atmospheric Research (NCAR), the National Oceanic, the Air Force Weather Agency (AFWA), the Naval Research Laboratory, the University of Oklahoma, and the Federal Aviation Administration (FAA)[2].

WRF model outputs are stored in the NetCDF (Network Common Data Form) format, which is very common in the climate community because of its advantages. For instance, it includes information about the data it contains, it can be accessed by computers with different ways of storing integers, characters and floating-point numbers. It also includes useful metadata - such as units of the quantity they contain, data timestamps, type of modifications/post-processing data has undergone, name of author(s) etc.[13]

From data output to visualisation

The use of visual representations has always been an integral part of climate science, as it helps scientists to better understand complex climate phenomena. Visualisations bring intuition to researchers on how climate systems behave and are also important in communicating the problem of climate change to the public. In fact, possible advancements towards a more green future start from recognising the effect of our actions on the Earth's climate and ecosystem.

I revisited the relevant literature on climatology to understand which quantities are important to visualise from a physical point of view. Then, under the supervision of my mentor, we decided to focus on temperature variation, which is the most evident feature of climate change.

I then wrote some Python pipelines to extract all the information from different NetCDF output data from various WRF pessimistic simulation. In particular, the model forecasts the evolution of temperature over Central and Southern Europe, from 2006 to 2099. I wrote Python code for producing temperature colour maps over this region for different years (see the top right panel of Figure 2), with a final animation of local surface temperature change over time.

[2]https://www.mmm.ucar.edu/weather-research-and-forecasting-model

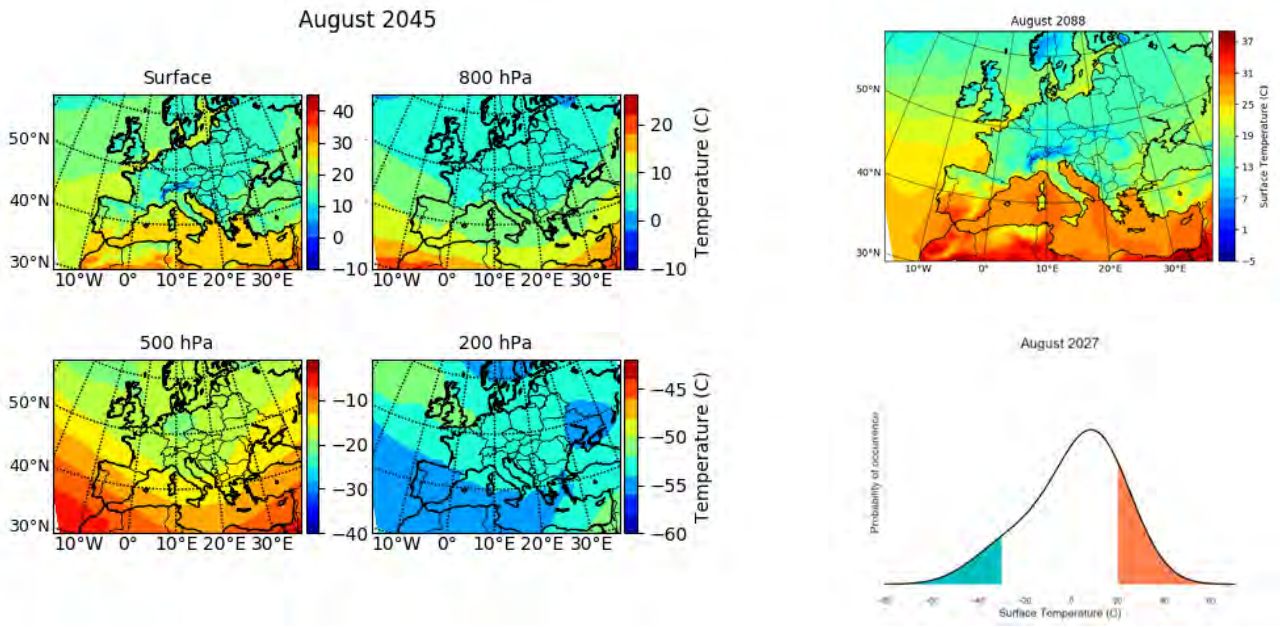


Figure 2: WRF model prediction. *Left* panel: Colour map of local temperature for different altitudes, August 2074. *Top right* panel: Contour map of local surface temperature, August 2088. *Bottom right* panel: probability distribution of surface temperature, August 2027.

However, different altitudes carry different information.

In fact, global warming often occurs faster at higher altitudes such as mountains. And what happens on mountains has a deep impact on critical aspects of our economic and social system, such as access to water (re)sources. In fact, warming causes the snow to melt, with a subsequent increasing temperature of the ground. This implies that snow will accrue slowly in the winter, and will melt faster in the spring, with less fresh water flowing down the mountains during the summer. If more than one billion people don't have daily access to clean water, climate change will definitely aggravate this situation. It will also have serious implications upon agriculture. For this reason, we also produced the animation of temperature evolution for different altitudes (see left panel of Figure 2).

Another way to extract information from data is through visualising the probability distribution function (PDF) of a variable. Therefore, we focused on the PDF of surface temperature over time, shown in the bottom right panel of Figure 2.

Thanks to the collaboration between science and informatics, by building complex models and running simulations upon supercomputers, we can foresee what is expecting us and our planet. It is up to us to intervene to prevent from what may happen.

Acknowledgement

I acknowledge Eleni Katragkou for her supervision, Ioannis Liabotis and Dimitris Dellis for their friendly hospitality. Thanks to Giannis Sofiadis, Dimitris Akritidis, Maria Karypidou for their kind help and a special thank to my traveling companion Mahmoud Elbattah, for his amiable support.

References

[1] https://crudata.uea.ac.uk/cru/data/temperature

[2] Levitus, et al, "Global ocean heat content 1955–2008 in light of recently revealed instrumentation problems," Geophys. Res. Lett. 36, L07608 (2009).

[3] http://www.pmel.noaa.gov/co2/story/What+is+Ocean+Acidification%3F

[4] L. Polyak, et.al., "History of Sea Ice in the Arctic," in Past Climate Variability and Change in the Arctic and at High Latitudes, U.S. Geological Survey, Climate Change Science Program Synthesis and Assessment Product 1.2, January 2009, chapter 7

[5] R. Kwok and D. A. Rothrock, "Decline in Arctic sea ice thickness from submarine and ICESAT records: 1958-2008," Geophysical Research Letters, v. 36, paper no. L15501, 2009

[6] https://www.ipcc.ch/pdf/assessmentreport/ar5/syr/AR5_SYR_FINAL_SPM.pdf

[7] http://nsidc.org/cryosphere/sotc/glacier_balance.html

[8] C. Derksen and R. Brown, "Spring snow cover extent reductions in the 2008-2012 period exceeding climate model projections," GRL, 39:L19504

[9] "Attribution of Extreme Weather Events in the Context of Climate Change," National Academies Press, 2016 https://www.nap.edu/read/21852/chapter/1

[10] Kunkel, K. et al, "Probable maximum precipitation and climate change," Geophysical Research Letters, (12 April 2013) DOI: 10.1002/grl.50334

[11] Kunkel, K. et al, "Monitoring and Understanding Trends in Extreme Storms: State of the Knowledge," Bulletin of the American Meteorological Society, 2012.

[12] https://en.wikipedia.org/wiki/File:All_forcing_agents_CO2_equivalent_concentration.png

[13] http://www.unidata.ucar.edu/software/netcdf/docs/netcdf_introduction.html

PRACE SoHPC Project Title

Visualizing European Climate Change

PRACE SoHPC Site

Aristotle University of Thessaloniki, Greece

PRACE SoHPC Authors

Edwige Pezzulli, University of Rome "La Sapienza", Italy

PRACE SoHPC Mentor

Eleni Katragkou, Aristotle University of Thessaloniki, Greece

Edwige Pezzulli

PRACE SoHPC Software applied

Virtuoso

PRACE SoHPC More Information

www.virtouso.org

PRACE SoHPC Acknowledgement

Write any requested acknowledgements or thanks here. Mentors should be asked for them too.

PRACE SoHPC Project ID

1710

Metadata Extraction from Climate Simulations

Mahmoud Elbattah

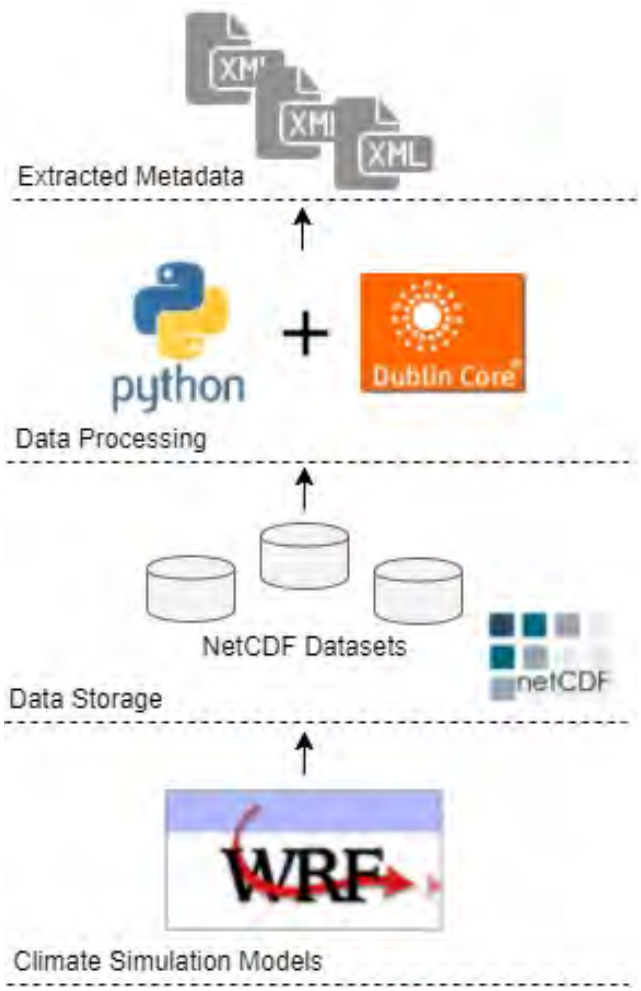


Figure 1: Project overview.

The goal of our project was to extract descriptive metadata from NetCDF-based output generated by climate simulation models. The metadata should also be formatted in a query-able format, so that search and query tasks can be conducted effectively. In this manner, climate researchers can easily share, discover, and use NetCDF datasets.

As the project worked intensively with NetCDF datasets, this section serves as a brief background to the NetCDF format and its underlying data structure. NetCDF stands for "Network Common Data Form". The NetCDF creators (Rew, and Davis, 1990) defined it as a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. It actually emerged as an extension to the NASA's Common Data Format (CDF). NetCDF was developed and is maintained within the Unidata organisation.

Unidata is a diverse community of education and research institutions with the common goal of sharing geoscience data and the tools to access and visualise that data. Unidata aims to provide data, software tools and support, to enhance Earth-system education and research. Funded primarily by the National Science Foundation (NSF), Unidata is one of the University Corporation for Atmospheric Research (UCAR)'s Community Programs (UCP).

The NetCDF data abstraction, models a scientific dataset as a collection of named multi-dimensional variables along with their coordinate systems, and some of their named auxiliary attributes. Typically, each NetCDF file has three components which are: i)Dimensions, ii)Variables, and iii)Attributes. Dimensions describe the axes of the data arrays and each dimension has a name and a length. A typical NetCDF variable has a name, a data type, and a shape described by a list of dimensions. Variables in NetCDF files can be one of six types (char, byte, short, int, float, double). Scalar variables have an empty list of dimensions. Any NetCDF variable may also have an associated list of attributes to represent information about the variable.

Figure 2 illustrates The NetCDF abstraction with an example of dimensions/variables that can be contained in a NetCDF file. The variables in the example represent a 2D array of surface temperatures on a latitude/longitude grid, and a 3D array of relative humidities defined on the same grid, but with a dimension representing atmospheric

level.

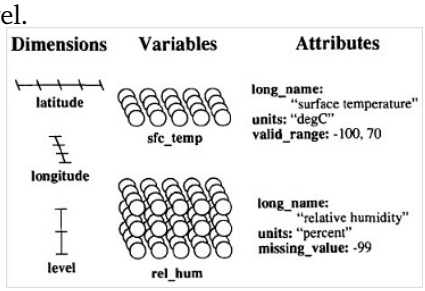


Figure 2: NetCDF data structure.

Problem Description

DSpace is a data repository where climate researchers and institutions can easily share their datasets (e.g. NetCDF files). However, shared files can be considered as a "black box", which always needs to be opened first in order to know what is inside. In fact, climate simulation models generate vast amounts of data, stored in the standard NetCDf format. A typical NetCDF file can contain a set of many dimensions and variables. With so many files, researchers can spend a lot of time trying to find the appropriate file (if any).

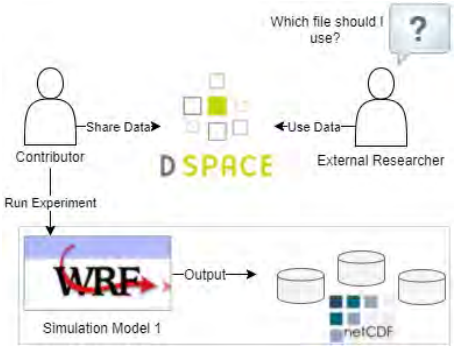


Figure 3: Problem description.

Project Objectives

The goal of our project is to produce explanatory metadata that can describe NetCDF datasets. The metadata should also be stored and indexed in a queryable format, so that search and query tasks can be conducted effectively. In this manner, we can facilitate the search and query of NetCDF datasets uploaded to the DSpace repository, so that researchers can easily discover and use climate data. Specifically, a set of objectives were defined as below:

- Defining the relevant metadata structure to be extracted from NetCDF files.
- Extraction of metadata from the NetCDF files.
- Storage/indexing of extracted metadata.
- Extending search/querying functionalities.

The project was developed in a collaboration between GrNet and the Aristotle University of Thessaloniki. GrNet provided us with access to the ARIS supercomputing facility in Greece, and they also manage the DSapce repository. The ARIS supercomputer is usually utilised to run the computationally intensive climate simulation models. The output of simulation models was also stored on ARIS.

Data Source

As already mentioned, the DSpace repository contains the main data source of NetCDF files. DSpace is a digital service that collects, preserves, and distributes digital material. Our particular focus is on climate datasets provided by Dr Eleni Katragkou from the Department of Meteorology and Climatology, Aristotle University of Thessaloniki. The datasets are available through the following URL:
https://goo.gl/3pkW9n

Methodology

The project was mainly developed using Python. A set of packages was utilised as follows: i)NetCDF4., ii)xml.etree.cElementTree., iii)xml.dom.minidom., iv)glob,and v)os.
Subsequently, the extracted metadata was encoded using the standard Dublin Core XML-based schema. The Dublin Core Schema is a small set of domain-independent vocabulary terms that can be used to describe information or data in a general sense. The full set of Dublin Core metadata terms can be found on the Dublin Core Metadata Initiative (DCMI) website. The full implemented Python code can accessed on GitHub via:
https://goo.gl/LFAeGz

Results

The project outcomes included the following:

- More than 40K metadata fields were extracted.
- 940 DublinCore-based XML files.Figure 4 provides an example of the extracted metadata.

Figure 4: Example of extracted metadata.

Acknowledgements

First, I would like to thank my mentors Ioannis Liabotis and Eleni Katragkou for their kind support and help. Furthermore, many thanks to Dimitris Dellis from GrNet who provided a lot of technical support during the project development. Last but not least, thanks to Edwige Pezzulli for her kind collegiality and companionship.

References

Rew, R., and Davis, G. (1990). NetCDF: An Interface for Scientific Data Access. IEEE Computer Graphics and Applications,10(4), 76-82.

PRACE SoHPC Project Title
European Climate Model Simulations

PRACE SoHPC Site
Aristotle University of Thessaloniki, Greece GrNet, Greece

PRACE SoHPC Authors
Mahmoud Elbattah, National University of Ireland Galway, Ireland

PRACE SoHPC Mentor
Eleni Katragkou, Aristotle University of Thessaloniki, Greece
Ioannis Liabotis, GrNet, Greece

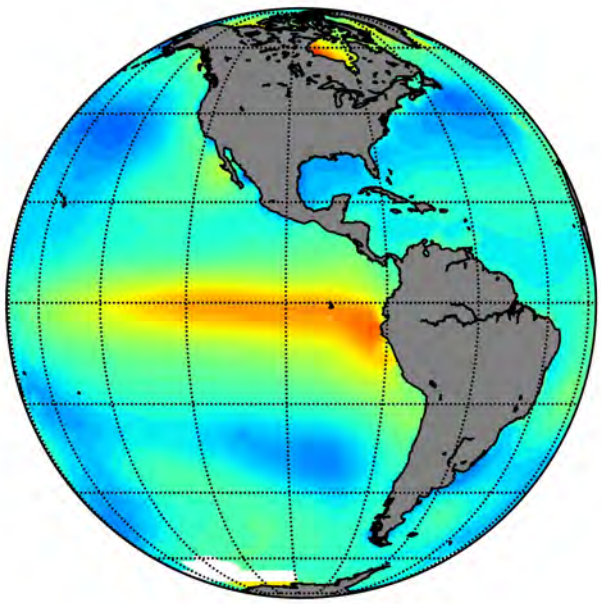

Mahmoud Elbattah-photo

PRACE SoHPC Software applied
Virtuoso

PRACE SoHPC More Information
www.virtouso.org

PRACE SoHPC Project ID
1711

El Niño around the world



Ana Maria Montero Martinez

El Niño is a periodic climate event that causes abnormally warm surface weather in the equatorial Pacific Ocean. We will see what its effects are in different parts of the worl.

El Niño-Southern Oscillation (ENSO) is a major climate pattern that causes oscillations in the meteorological parameters in the equatorial Pacific ocean. Normal weather conditions of this region include the following:

- Low pressures (L) in the Pacific islands, causing rainy weather and warm ocean waters.
- High pressures (H) near the eastern American coast, causing colder ocean waters and less rain.

In these normal conditions, the higher amounts of water taken to the western part of the Pacific by trade winds raises ocean levels in Indonesia to half a meter, when compared to Peru. The difference between surface water temperatures in the above locations can be up to 8 degrees. Tthe lower coastal temperatures in Peru occur due to rising deep water which is rich in nutrients and helps maintain the marine ecosystem in Peru - upon which many fishermen are dependant.

When ENSO occurs, the trade winds that normally flow from west to east start to slow down, causing warm waters to move towards the west (since the winds are no longer pushing them to the east). This changes the the climate pattern in the Pacific Ocean. The main characteristic of these ENSO events include:

- Areas of low and high pressures switch places (this is why it is called Southern Oscillation), causing the rain to move towards the west.
- Because of this, warm ocean waters move towards the west, causing alterations in in South America fishing patterns.

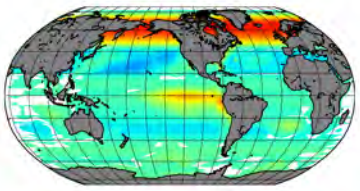


Figure 1: Sea surface temperature anomalies during the winter season for the 1982/83 El Niño event.

ENSO is a phenomenon that involves the ocean, the atmosphere and the land upon such a huge scale that it affects weather in the entire world - afeecting precipitation, temperature and wind flows. As a consequence various events occur which include assive forecast fires in Indonesia - since it stops raining in that area, flooding in Peru - caused by a severe amount of rain in a short period of time, drought in India - because of abnormally light monsoon months, and an overall temporal warming of the global climate.

All of these events highlights ENSO as an important part of the weather climate cycle. It is a phenomenon which is worth studying and that is what I worked on at ICHEC, Dublin this summer.

Initially, I analysed previous El Niño events and whether it is possible to find common trends between them which could help in predicting future El Niño events. Secondly, I studied how El Niño impacts other parts of the world, what is the correlation between all these events and for how long they last. Finally, I looked into all the available forecast systems one season ahead, to check their accuracy and precision in detecting future El Niño events.

Why is it called El Niño?

El Niño is a phenomenon that occurs throughout the year, but mostly when water in the Peru coast is most warmest - and this occurs in December. Fisherman associated this phenomenon with the Christmas holidays and called it "El Niño" that literally means "The boy" after the nativity.

El Niño... and also La Niña?

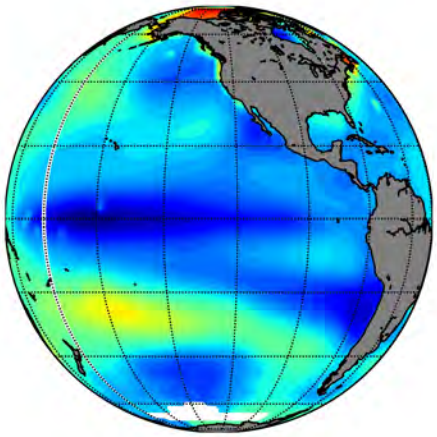


Figure 2: Sea surface temperature anomalies for La Niña event in 1973/74 during winter.

La Niña is a phenomenon that is also part of the ENSO oscillation and occurs naturally in the world's climate cycle. The El Niño phase is also called the warm phase - due to the abnormally warm water in the east coast of South America. Conversely, La Niña is the cold phase when abnormally strong trade winds from the west cause lower than usual temperatures in the equator, as shown in Fig. 2

Similar to El Niño, la Niña conditions change global temperatures and precipitations patterns - even though its consequences are drastically different to those of El Niño. Among its consequences, we can highlight greater precipitations in Asia and some parts of Africa and Brazil, more hurricanes in the United States and severe droughts in Mexico and the west of America.

Consequences around the world

El Niño events affects the weather in many places around the world. The main areas affected experience changes in rainfall patterns, as seen in Fig. 3 and 4.

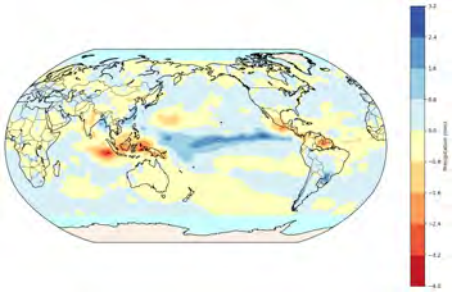


Figure 3: El Niño rainfall pattern changes.

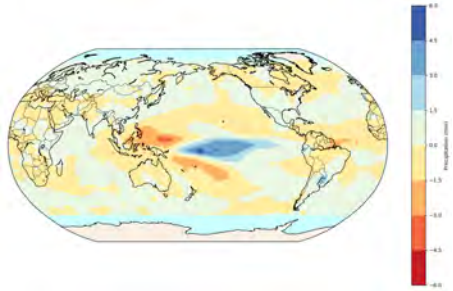


Figure 4: El Niño rainfall pattern changes.

North America

During ENSO events, precipitation in Mexico and the south of the US - such as California and Florida, increases drastically. This is usually good for the region as it mostly suffers from drought.

However, although this rainfall can be beneficial, the risk of extreme rainfall is high, and this can cause lead to higher instances of flooding.

Somalia

In Somalia, the risk of flooding is severe and it causes - according to the United Nations, up to 55,000 people hit by this hihg level of precipitation. The floods causes roads to be impassable and prevents thousands of people from receiving aid. As a consequence, river banks need to be strengthened and people need to move to higher ground.

India

During El Niño years, the monsoon in India is weaker. As a result, precipitation is lower and as a consequence, there is a risk of drought in the country.

Conclusions

At the end of this project, we can conclude that El Niño is an event that affects the weather in many parts of the Earth. It is also very difficult to predict since it involves the atmosphere, the ocean and the land in such a way that a lot of different scenarios can happen under similar initial conditions. Nowadays, it is important to correctly predict these events so that we can minimise the impact on property and human life. But predicting when and how an El Niño event in going to happen cannot be done accurately more than three months before the event.

Nowadays, most El Niño research is moving towards its prediction and its evolution under climate change conditions. Given that global Earth temperatures are rising El Niño events may be more frequent. At the moment, it is impossible to find any correlation between the periodicity of El Niño and climate change because no statistically significant trend can be found.

References

[1] Xu, Kang & Tam, Chi-Yung & Zhu, Congwen & Liu, Boqi & Wang, Weiqiang. (2017). CMIP5 Projections of Two Types of El Niño and Their Related Tropical Precipitation in the 21st Century. Journal of Climate. 30. 849-864. 10.1175/JCLI-D-16-0413.1.

PRACE SoHPC Project Title

El Niño, its periodicity and impact on world weather

PRACE SoHPC Site

ICHEC, Ireland

PRACE SoHPC Authors

Ana Maria Montero Martinez, University of Extremadura, Spain

PRACE SoHPC Mentor

Emma Hogan, ICHEC, Ireland

PRACE SoHPC Software applied

Python, Sony Vegas Pro.

PRACE SoHPC Project ID

1712

Ana Maria Montero Martinez

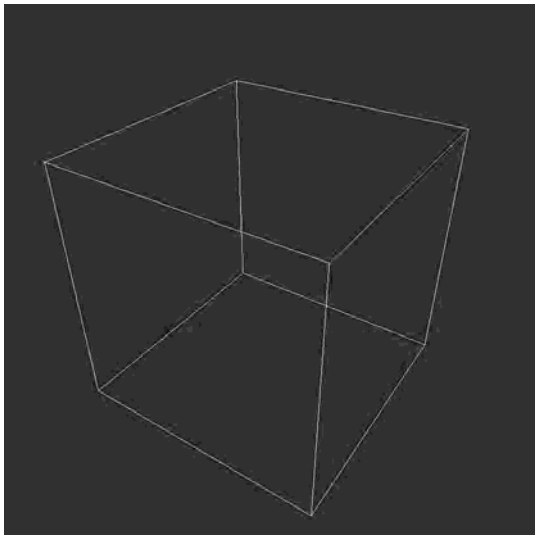
Radiosity in Computer Graphics

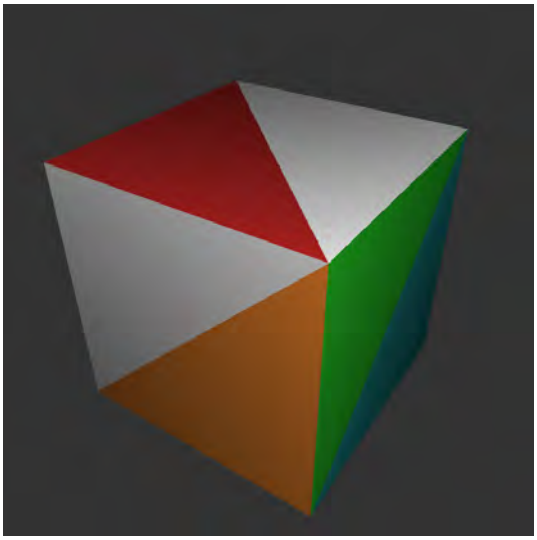
Jamie Quinn

Be it in films, visual arts or computer games, photorealistic rendering of a virtual scene has been a hot topic for decades. The ultimate aim here is to use physically motivated techniques to realistically render a simple scene similar to the one shown.

Firstly, what is rendering? Well, stored in the computer is only the description of the geometry. For the box shown, only the location of the faces are stored. Rendering is simply the process of turning that description of geometry into an actual image you can see. The standard way to render a scene is to choose a virtual camera position, take the geometry that the virtual camera is "seeing" and draw it to the screen.

For a simple box, using perspective to make the box appear like it would in the real world looks like the image below. What about colour? Simply drawing solid colours looks a little strange, as seen in the upper right image. What we really want is to be able to add a virtual light to the scene and give the box some kind of shading. Adding a light behind the camera produces the lower right image.





What is Radiosity?

The shading used for that last image uses a simple method called **Gouraud shading**.[4] It's easy and efficient, but all it does is calculate how much each bit of the geometry is directly affected by each *light* in the scene. Take a look at the room shown on the first page, there's only one light, a white light, and yet the box on the left is coloured a little bit red because of the red wall. Here we have a case of light bouncing off one object and affecting another. Gouraud shading doesn't (easily) take this into account, so we turn to radiosity.[1]

To give a quick definition, **radiosity** is a measure of how much light a patch of a virtual scene is giving off. This amount of light depends on two things, how much light is hitting the patch and bouncing off, and how much light the patch itself is emitting. We use this to render a scene by considering how the light given off by each patch in a scene affects every other patch. We can then properly colour a scene using the resultant radiosity.

How Do We Mathematically Find Radiosity?

Calculating the radiosity really depends on calculating the spatial relationship from one patch to another - called the **form factor**. It is a number between 0 and 1 describing how much one patch can "see" another patch. If two patches are very far away from one another the form factor between them will be near 0, and if they are very close the form factor will be closer to 1. If one patch can't be seen at all from another patch, the form factor will equal 0. From these form factors, we can find how the radiosity is propagated through the scene from patch to patch.

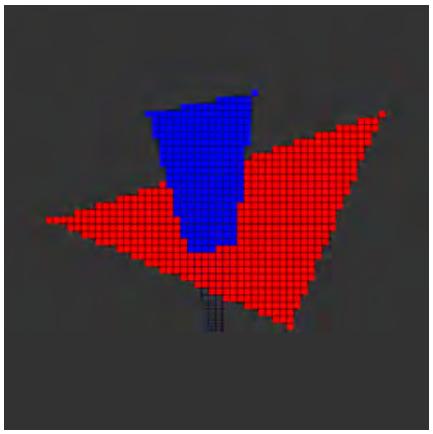
How Do We *Actually* Find Radiosity?

A few different techniques and algorithms are used in the process of calculating radiosity. To calculate the form factors, a **model-view-projection** transformation must be set up. This allows the placement of a virtual camera on a patch in order to see which other patches that patch "sees".

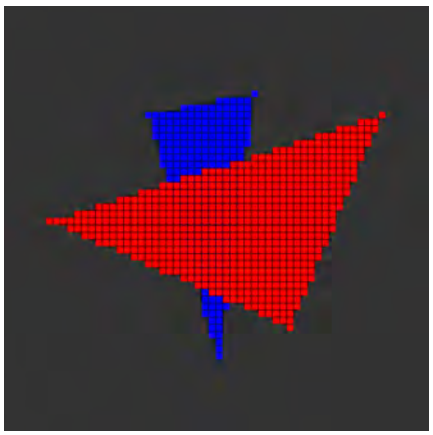
This transformation is used to project the geometry onto a half-cube sticking out of the patch. This is known as the **hemi-cube approximation**.[3]

After projection, each triangle of the geometry is **rasterised** to the hemicube faces. That simply means that we figure out which pixels on the faces should come from which triangles. That usually happens in **scanlines**, where we scan across the triangle, figure out which pixels are inside the triangle and colour them accordingly. That looks like this.

When rendering the triangles that make up the entire geometry, we come to a problem. All the triangles are simply kept in a list and rendered in an arbitrary order. So what if we've moved the camera so that a red triangle should be in front of a blue triangle, but the blue triangle just happens to come after the red triangle in the list? The blue triangle will rewrite the correct red colour with a false blue colour.



We solve this problem by using the **z-buffer** along with an algorithm that for every pixel we're trying to rasterise asks, has it already been taken by a triangle that was closer?



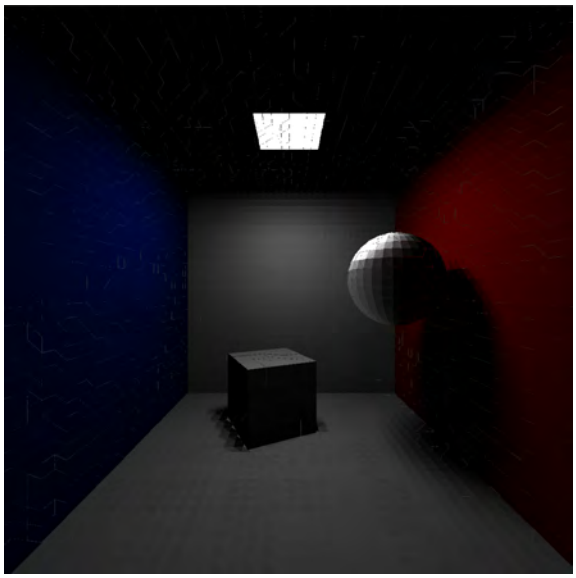
We then find the area the projection each patch covers on the hemi-cube. This determines the form factor.

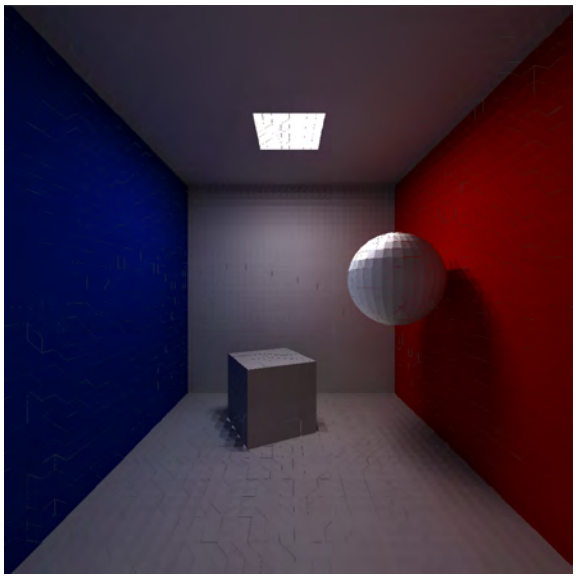
To calculate the final radiosity, we use a technique known as **progressive refinement**.[2] This is where we know the initial radiosity in the scene, i.e. that given off by the lights we've specified. We then propagate this radiosity through the scene in one of two ways. First using **the gathering method**. We select the i-th patch and find the form factors between it and every other patch. Then, from each j-th patch we gather how much radiosity is being sent from that patch to ours. This depends on the total amount of radiosity of patch j, r_j, the form factor, F_{ij}, and the **reflectivity** of patch i, ρ_i. Reflectivity is a measure of how much light bounces off our patch. It equals 1 if all the light bounces and 0 if none of it does. The radiosity of patch i from patch j is given by

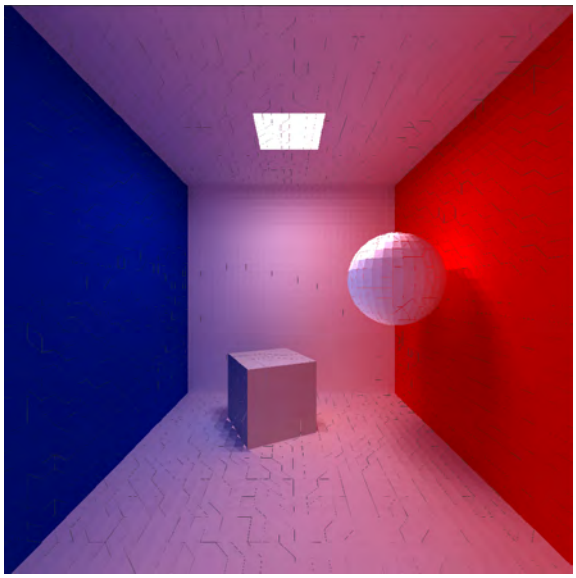
$$r_i = \rho_i F_{ij} r_j.$$

The second method, **the shooting method**, is essentially the opposite of the gathering method. We select the i-th patch, work out form factors, and then shoot the radiosity from our patch into every other patch. The gathering method deals better with low resolution hemi-cube faces, however the shooting method may be stopped sooner if it detects that the radiosity has reached an acceptable accuracy.

Snapshots of the progressive refinement process:







The Code

Coding the entire project essentially from scratch was a wonderful learning experience. Although there are many excellent vector maths libraries out there, I decided to roll my own as an educational exercise. **Vectorisation**, letting the computer know that certain things can be done all at one time instead of one operation after another and treating the cache well by making sure my little data structures fit well into the very fast memory right beside the processor, were two concepts I kept in my mind while developing the vector maths parts of my code. In the future I would most definitely use a library, it was good for my learning but there are faster and better designed libraries out there.

The same was similarly true for the rendering process. Though the original code used entirely my own algorithms, parallelised for one single machine using OpenMP, I ended up managing to code a nearly complete alternative solution using the low-level graphics library OpenGL with a surprisingly small amount of code. OpenGL dealt with the entire rendering process - it was designed for games after all and no games programmer wants to reimplement forty years of graphics research! The only caveat is that OpenGL runs exclusively on graphics cards and although such cards are becoming more popular - mainly because they're very good at applications like machine learning, they're reasonably rare in high-performance computing centres.

Futher Work

Amongst standard polishing of a rushed codebase (there was only 7 weeks to do all this after all) there are a few other features/implementations to try, which include:

- Parallelisation on cluster with MPI.
- Complete GPU implementation.[5]
- Substructuring around shadow boundaries.
- Fixing of minor graphical glitches and artifacts.
- Implementation of 3D viewer for final scene.

References

[1] Cindy Goral, Kenneth E. Torrance, Donald P. Greenberg and B. Battaile, *Modeling the interaction of light between diffuse surfaces*, Computer Graphics, Vol. 18, No. 3.

[2] Cohen, Michael F., et al. "A progressive refinement approach to fast radiosity image generation." ACM SIGGRAPH computer graphics 22.4 (1988): 75-84.

[3] "The Hemi-cube - a radiosity solution for complex environments" Michael F Cohen and Donald P Greenberg ACM SIGGRAPH Computer Graphics 1985 Vol 19 Issue

[4] H. Gouraud, "Continuous shading of curved surfaces," IEEE Transactions on Computers, 20(6):623–628, 1971.

[5] Toshiya Hachisuka, "High-Quality Global Illumination Rendering Using Rasterization", https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter38.html

Jamie Quinn

PRACE SoHPC Project Title

Radiosity in Computer Graphics

PRACE SoHPC Site

ICHEC, Ireland

PRACE SoHPC Authors

Jamie Quinn, University of Glasgow, Scotland

PRACE SoHPC Mentor

Oisin Robinson, ICHEC, Ireland

PRACE SoHPC Software applied

C++, Python, OpenGL, OpenMPI, Three.js, OpenMP, Blender

PRACE SoHPC Acknowledgement

I'd like to acknowledge the whole ICHEC team for being extremely welcoming and wonderful in a strange new city.

PRACE SoHPC Project ID

1513

Skeletal Motion Tracking

Report written by mentor Petr Strakos

Motion capture technology can be used to identify and track the human frame. Using new tools developed by IT4Innovations to process DICOM images, it was possible to develop a tool for real time visualisation of human skeletal motion.

Motion capture technology has advanced rapidly in recent years. Using a combination of depth sensors and cameras, with sufficient resolution and frame-rate, it is possible for software to identify and track a human skeletal frame.

The goal of this project was to develop a program which renders and displays a human skeleton, whose pose corresponds to a human subject of motion capture. This program could be a useful tool for athletes, sports coaches and physicians. It may also be a useful for education of the human anatomy.

Computed Tomography

Computed Tomography (CT) is a medical imaging technique which takes X-ray measurements from several different angles, to provide a set of cross sectional images of a scanned object. The picture below is an example of a CT image.

While the picture above is two dimensional, various tools can be used in conjunction with CT data to generate 3D models of various organs and calculate the volume of various tissues in the human body.

The scope of this project though was to use CT images to generate 3D models of bones.

Blender

Blender is a professional, free and open-source 3D computer graphics software that can be used for 3D modelling.

Given its open-source nature, Blender can be extended through plugins. Blender plugins are Python scripts that the user interacts with using the Blender GUI.

A team at IT4Innovations developed an interesting Plugin for Blender which is useful for processing Computed Tomography (CT) images. The actual image processing algorithms are implemented in C++ with OpenMP, with PyObject interfaces so they can be called from Python.

This plugin allows somebody with little knowledge of programming to use a high performance cluster for Digital Imaging and Communications in Medicine (DICOM) image processing.

At the click of a button, the user can load in a series of ordered DICOM images, view the loaded images by using a slider to move through them and select various orientations to view the data. Furthermore, the user can apply various denoising filters to view the data in greater clarity, or to improve the effectiveness of later image processing.

The K-Means Segmentation Algorithm

The K-Means Segmentation Algorithm is a simple, yet powerful, algorithm which solves the well known clustering problem.

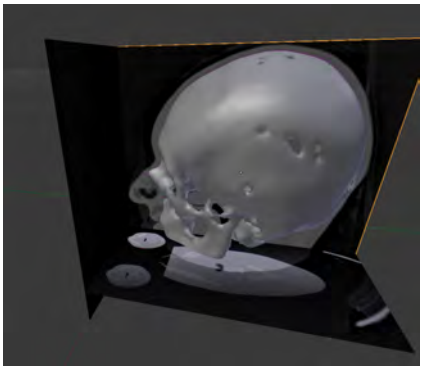
It was used within the project to segment skeletal image data into a set of images representing individual bones.

Although the algorithm requires a long time to run on a large dataset, IT4Inovations researchers have implemented a parallel version of this algorithm using OpenMP, which when run on the Xeon Phi co-processors of the Salomon cluster allows one to generate models of individual bones of the skeleton in much less time.

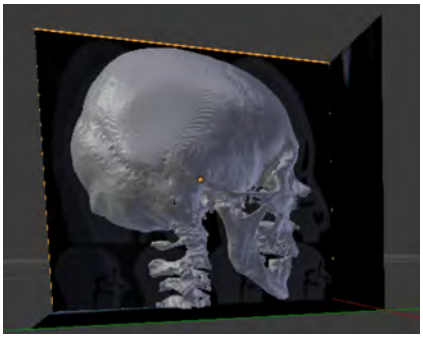
From 2D to 3D

Even though CT images are 2D data, image processing operations can be applied to all selected slices given as output by the K-Means algorithm.

Thus, once the data has been segmented, it is only necessary to select a segment to be converted into a structural 3D model called a “mesh model”. The picture below is one such model.



The level of detail of the mesh model generated by the tool can be controlled by various parameters. Below is a higher detail model generated from the same images as the previous model.



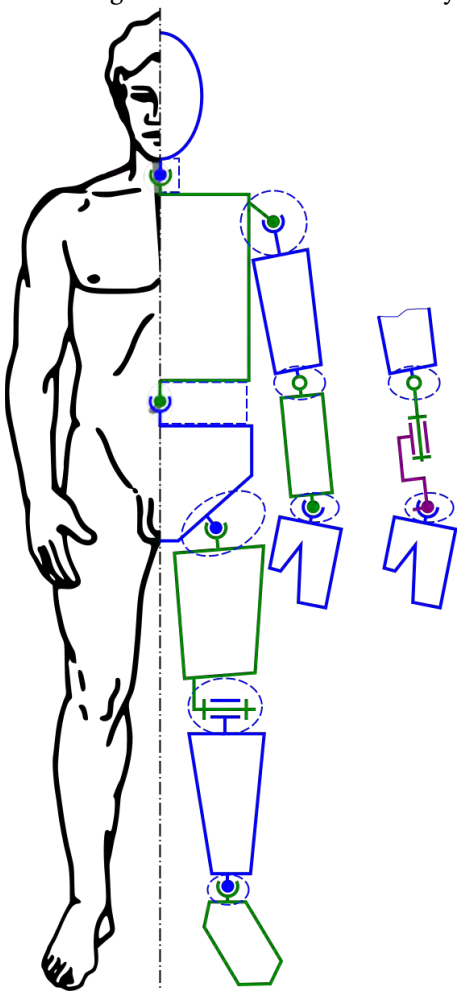
The appropriate detail of a model produced using this tool will vary, depending on what it will used for. Highly detailed models will be computationally expensive to render, and as such, may not be suitable for games or interactive tools that require the model to be rendered in real time. However, for generating pre-rendered animations, a highly detailed model may be more impressive, and worth the computational resources required for production.

Kinematic Chains and Kinect 2

Once image data have been processed to generate mesh models, a 3-dimensional kinematic chain based on motion capture of a moving human can be created.

A kinematic chain is simply a set of rigid bodies, connected by fixed points and various constraints defining how the fixed bodies can move relative to the rigid bodies they connect to.

The picture below is an example of a kinematic chain modelling the human body.



Kinematic chains are often used in robotics, but can also be useful in other fields including structural engineering and physiotherapy.

The Microsoft Kinect 2 (K2) is a relatively inexpensive device which can be used to generate a 3-dimensional kinematic chain based on motion capture of a moving human.

Code was developed to identify and track a human kinematic frame using the K2 and when the structural models are applied to the corresponding bodies of the kinematic chain, a model of a human skeleton can be rendered in real time.

Conclusion

Using all the above mentioned tools and methods, an application can be developed that identifies and tracks human skeletal motion from CT data.

PRACE SoHPC Project Title

Visualization of Real Motion of Human Body Based on Motion Capture Technology

PRACE SoHPC Site

IT4Innovations, Czech Republic

PRACE SoHPC Authors

Report written by mentor Petr Strakos

PRACE SoHPC Mentor

Petr Strakos, IT4Innovations, Czech Republic

David J. Bourke carried out part of this work

PRACE SoHPC Contact

Karina Pesatova, IT4Innovations
E-mail: karina.pesatova@vsb.cz

PRACE SoHPC Software applied

Blender

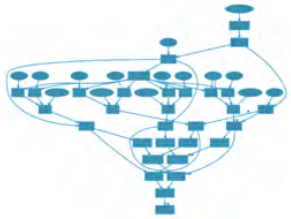
PRACE SoHPC More Information

www.blender.org, http://blender.it4i.cz

PRACE SoHPC Project ID

1714

Go! Data Visualisation



Shukai Wang

If you are a big fan of data visualisation, you are in the right place. Here I will introduce a number of visualisation techniques to turn the results of a machine learning pipeline to interactive heatmaps.

The identification of bioactive compounds is one of the most important steps in drug discovery. In the past, scientists could only measure bioactivity once a particular compound was made. Some predictions could be made based on their structures (e.g. weights, functional groups, etc), however, in many cases, predictions was limited and inaccurate. Nowadays, there is enormous development in proteomics and genome sequencing technology. In the field of biochemistry, there have been several recent publications which make some predictions by applying Machine-learning (ML) algorithms.[1,2] At the same time, there are many readily available chemical databases which contain experimental structure-bioactivity information. Examples of these include PubChem,[3] BidingDB,[4] Reaxys,[5] etc. ML algorithms can be applied to these to get insights into structure-bioactivity relationships. This will be illustrated in this project report.

In short, ML is a set of approaches which use a series of complex models and algorithms to make data predictions. The key idea of ML is to train the algorithm using learning data, for it to later make relevant predictions. However, based on the nature of the data, we often need to have a bit of understanding of the data, as well as to select features and algorithms and to adjust the evaluation criteria to optimise the prediction accuracy. Algorithms refer to the training methods, such as whether it is a classification or clustering problem. Evaluation criteria means how you define the threshold to make the prediction. A ML workflow in this project will be illustrated later.

Due to the quantity of data and variations of algorithms and criteria, ML in this case would require a large amount of computational time. However, this computational process can be optimised through parallelisation of independent tasks. HyperLoom[6] is a package developed by a team at IT4Innovations in the Czech Republic. It is designed to define and execute workflow pipelines in large-scale distributed environments and optimise the performance of the supercomputers through a simple API. In this way, the computation time when using ML will be greatly reduced. One of the key steps in this process is to visualise the communication between each worker, which has been completed in this project.

Methods

Existing ML code was submitted to the Salomon supercomputer at IT4I. The ML and HPC performance results were sent back to the user. The ML pipeline performed gridsearch combined with cross-validation using the libSVM library - a support vector machine implementation. The effect of the gamma and the cost values on prediction accuracy were visualised by a Python code I wrote using Bokeh.[10] I also completed the traffic visualisation using Bokeh[10] as well as Graph-Tool[7] and compiled to the major HyperLoom testing code.

Results and Discussion

Machine Learning Pipeline

As shown in Figure 1, the ML pipeline in this project had a total of 6 steps. Each data set (e.g. setA) was selected based on some categories from the data base (the outer loop) and it went through the ML pipeline via the cross-validation process (the inner loop). The cross-validation process divided each data set into 20% for validation and 80% for training. As one data set can be separated into a total of 5 distinctive validation sets, this training/validation process would be iterated for five times. The advantage of cross-validation is that all data would be validated as well as participating in the training process. For each iteration, the training set would be further segmented to D1-D5, each of which contained an 80/20 training/validation ratio. The data would be used for training using training parameters (TP) 1-n. In this case, it was

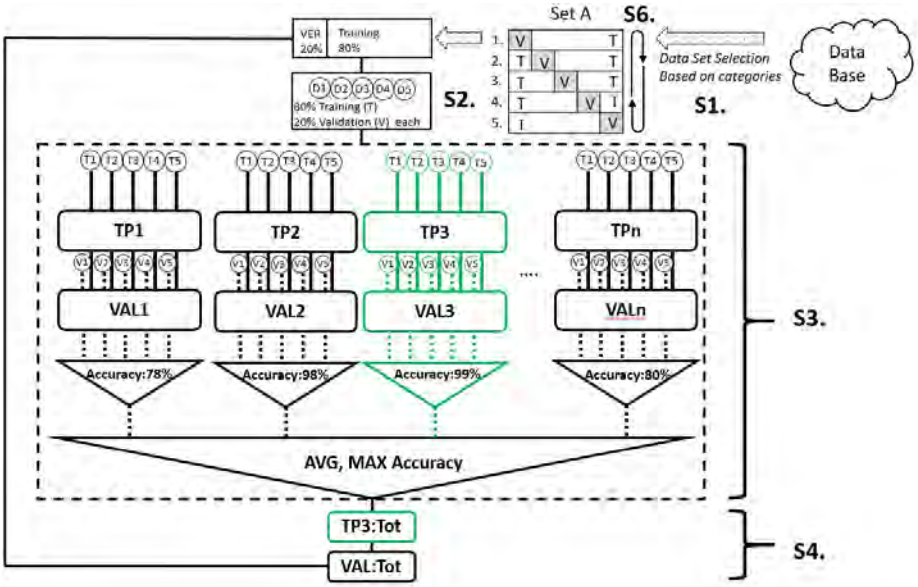


Figure 1. ML pipeline workflow: S1: Select data set A according to requirements. S2. Segment data as set A1 and further segment the training set as D1-D5. S3: Training and validation on segmented data set using TP1-TPn and identification of the best TP (highlighted in green), based on accuracy. S4: Training and validation of the whole data set using the best TP. S6. Iterate S2-4 when set A is segmented to A2-A5, known as cross validation. ST: training, V: validation, TP: training parameter, VAL: validation.

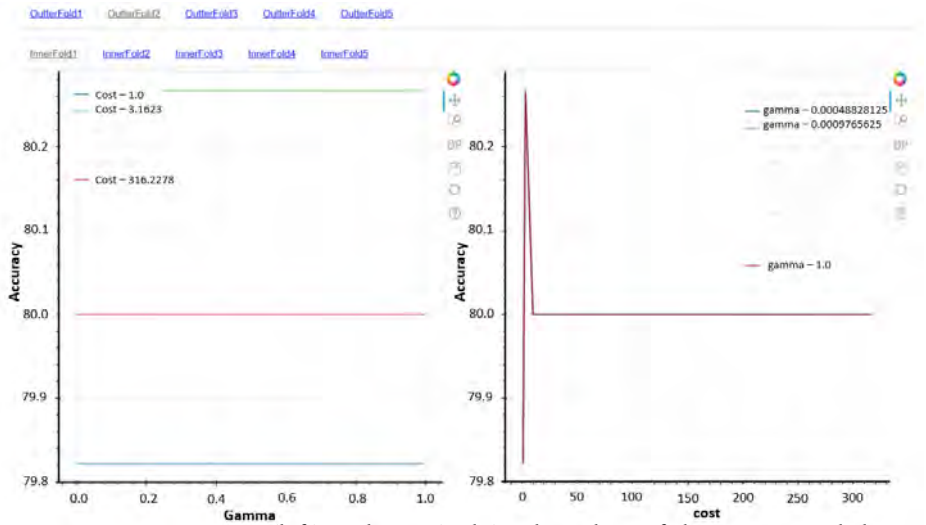


Figure 2. Prediction accuracy against gamma (left) and cost (right). The values of the gamma and the cost were picked based on experimental experience.

the gamma and the cost. Both are parameters for a nonlinear support vector machine (SVM) with a Gaussian radial basis function kernel for classification problems. SVMs are used for classification and regression analysis in supervised learning models. They are normally associated with a number of algorithms. Kernel methods are named after Kernel functions, which deal with high-dimensional, implicit featured statistical problems. After the TP with the highest accuracy was identified, the D1-D5 would be merged into the original 80% training set, trained again using the best TP and followed by an overall validation. This method can greatly reduce the computational cost.

The effect on accuracy of gamma (left) and cost (right) was visualised using interactive diagrams as shown in Figure 2. The outer loops were the data set selected based on the features from the database. The inner loop represents the result from each cross-validation iteration. Normally, a high gamma leads to high-bias (high accuracy) and low-variance (high precision) models, but in this case the gamma had no influence on the accuracy. Hence all lines overlapped each other on Figure 2 (right). On the other hand, the cost defines the soft/hard margin in the classification process. In this case, some certain cost value is better than the rest. By visualising the accuracy against training parameters, it is very easy to find the best parameters for the overall training set.

HPC Performance Visualisation

The performance of the HPC system was also monitored while running the ML. Here, all ML pipelines were scheduled using the HyperLoom package (Figure 3).

As can be seen from Figure 3, the scheduler receives tasks from the client, schedules them and sends them to the server in an optimised order. The server will then distribute them to different workers. The workers communicate with each other by sending the results of its own task as the input file of tasks of another worker. Finally, the server receives the individual results from the worker and returns both the results and the performance trace files to the client.

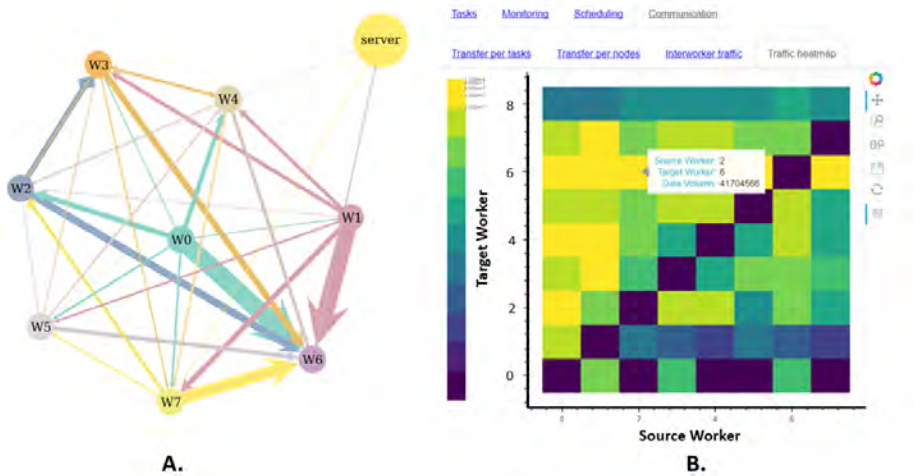


Figure 4. The visualization of traffic between workers and the server using: A. A network diagram - where edge width, data volume, diamond arrows indicate the actual data volume is less than 1/200 of the maximum data volume (after rescaling) and W means workers.. B. An interactive heatmap using a log scale colour scheme - with bright yellow as the maximum.

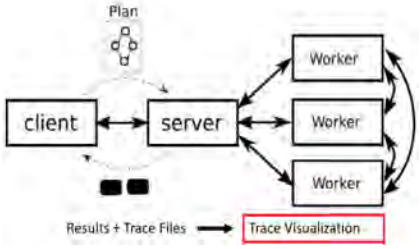


Figure 3. The architecture of HyperLoom, including a client, a server and several workers. The interconnections are between the client and the server, the server and the workers and between each worker.

During the project, two ways were developed to visualise the communication between the workers and the user: A network diagram (Figure 4A) and a heatmap (Figure 4B). Figure 4A was constructed using Graph-Tool[7] after comparing its presentation with NetworkX[8] in combination with Matplotlib[9] or the Dot language. The network diagram generated by Graph-Tool[7] was of high quality and its coding was easier. Figure 4 shows the traffic between each computational worker when a ML pipeline is completed. It is clear that W6 received the most results compared to any other worker whereas W1 has the highest export. Finally, the server only received data from workers and the amount was negligible. Figure 4B tells the same story as 4A.

Conclusion

Two months are very short in terms of delivering academic research, but my time has been very productive. Although my project hasn't gone deeper in one specific area, I have covered a range of topics from machine learning, HPC and data visualisation. I have understood how a general ML pipeline works, and got to learn some theory about ML, such as Vector Support Machines. I have also improved my coding skills considerably.

Acknowledgements

The author would like to express her appreciations to Jan Martinovič, Vojtěch Cima and Stanislav Bohm for their academic guidance, Lukas Vojacek for his technical support and Karina Pešatová in IT4I for all her help in settling in the Czech Republic for the past two months. The author would also like to thank Vladimir Chupakhin from Johnsen & Johnsen for providing experimental data for my project and finally the PRACE program for the funding and providing this opportunity.

References

[1] Liu, J.; Patlewicz, G.; Williams, A.; Thomas, R. S.; Shah, I. Chem. Res. Toxicol. (2017). Predicting Organ Toxicity Using in Vitro Bioactivity Data and Chemical Structure.

[2] Sun, J.; Jeliazkova, N.; Chupakhin, V.; Golib-Dzib, J.-F.; Engkvist, O.; Carlsson, L.; Wegner, J.; Ceulemans, H.; Georgiev, I.; Jeliazkov, V.; Kochev, N.; Ashby, T. J.; Chen, H. J. Cheminform. 9 (1), 17 ExCAPE-DB: An Integrated Large Scale Dataset Facilitating Big Data Analysis in Chemogenomics. (2017)

[3] Wang, Y.; Suzek, T.; Zhang, J.; Wang, J.; He, S.; Cheng, T.; Shoemaker, B. A.; Gindulyte, A.; Bryant, S. H. Nucleic Acids Res. 42 (D1), D1075–D1082. (2014) PubChem BioAssay: 2014 Update.

[4] Gilson, M. K.; Liu, T.; Baitaluk, M.; Nicola, G.; Hwang, L.; Chong, J. Nucleic Acids Res. 44 (D1), D1045–D1053. BindingDB in 2015: A Public Database for Medicinal Chemistry, Computational Chemistry and Systems Pharmacology. (2016)

[5] Reaxys database. https://www.reaxys.com. (2016)

[6] HyperLoom https://code.it4i.cz/ADAS/loom. (2017)

[7] Peixoto, T. de P. Graph-Tool https://graph-tool.skewed.de. (2017)

[8] Hagberg, A.; Schult, D.; Swart, P. NetworkX. https://networkx.github.io. (2017)

[9] Hunter, J. D. Comput. Sci. Eng. 9 (3), 90–95. https://matplotlib.org. (2007) Matplotlib: A 2D Graphics Environment.

[10] Bird, S.; Canavan, L.; Mari, M.; Paprocki, M.; Rudiger, P.; Tang, C. ; Ven, V. B.; Bokeh https://bokehplots.com/pages/team.html. (2017)

PRACE SoHPC Project Title

Performance visualization for bioinformatics pipelines

PRACE SoHPC Site

IT4Innovations, Czech Republic

Shukai Wang's photo

PRACE SoHPC Authors

Shukai Wang, Imperial College, UK

PRACE SoHPC Mentor

Jan Martinovič, IT4I, Czech Republic

PRACE SoHPC Contact

Karina, Pešatová, IT4Innovations
Phone: +420 597329587
E-mail: karina.pesatova@vsb.cz

PRACE SoHPC Software applied

Python, Bokeh, Pandas, Scikit Learn, Graph-tool

PRACE SoHPC More Information

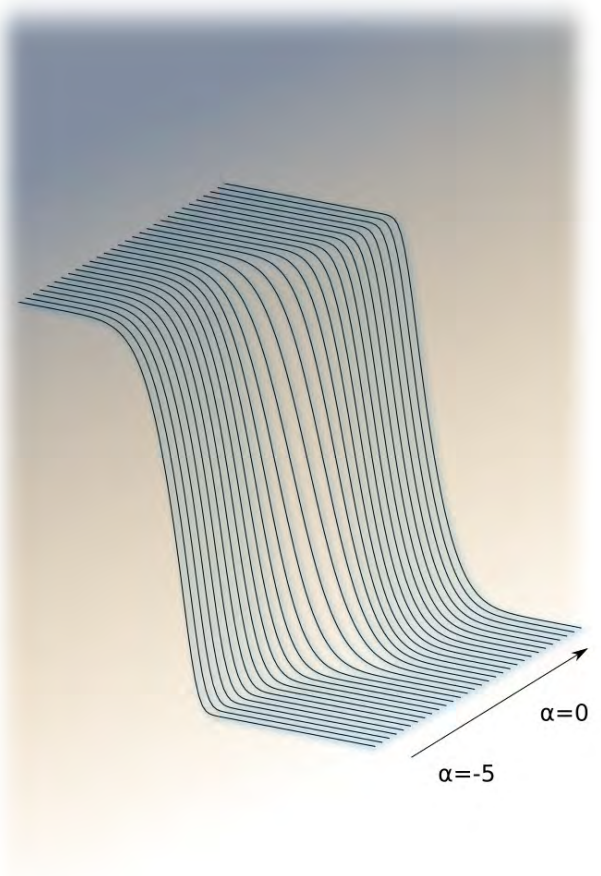
https://youtu.be/Bn32p-4PhW8

PRACE SoHPC Project ID

1715

Cude colors on a phine grid for SoHPC

Philippos Papaphilippou



The project aims to help eliminate the simulation time of a Quantum Monte Carlo simulator for carbon nanotubes by using data science techniques. This was achieved by creating an online optimization algorithm that balances the accuracy of the simulator for better performance.

Simulations of Lattice QCD (Lattice Quantum Chromodynamics) are used to study and calculate properties of strongly interacting matter, as well as the quark-gluon plasma - which are very difficult to explore experimentally due to the extremely high temperatures required for these interactions.[1] The simulator we are trying to optimise uses Quantum Monte Carlo Calculations (which have originally been used for LaticeQCD) to study the electronic properties of carbon nanotubes.[2]

My main activity was to help eliminate the simulation time of the simulator by using data science techniques. The method I will describe here is independent to the parallelisation and vectorisation of the code for fully exploiting the computing resources. The code of the simulator is already parallelised and vectorised (perhaps not yet optimally), but as we are going to see, some algorithmic decisions can further accelerate its execution.

Background Information

The simulator uses leapfrog integration to integrate differential equations. The equations that are used for the integration look like recurrence relations. This means thats each next value is calculated using a previous value. Specifically, the smaller the time difference t, the more accurate the integration is. t is inversely proportional to the number of MD (Molecular Dynamics) steps (Nmd). Therefore, the greater Nmd gets, the more accurate the simulation is, but also the more time it takes to finish. If the value of Nmd is low, the acceptance rate of the new trajectories is too low to achieve results in reasonable time - even on a supercomputer. If Nmd is too high, the acceptance rate is 100% but it introduces a significant time overhead. The goal is to tune this parameter on runtime to select the value that would yield an acceptance rate close to 66%.

Building a model

The idea is to create a model of the relationship "Acceptance Rate versus Nmd" each time a new data point comes in. We can then select the best Nmd for the next run. In Figure 1 we can see a set of observations from an experiment.

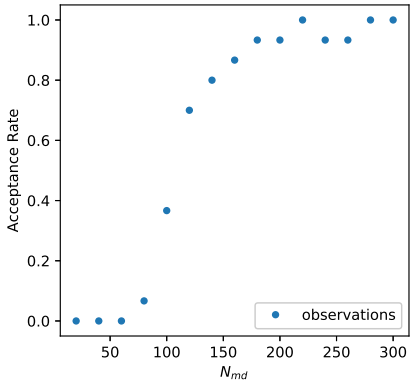


Figure 1: Observations for different number of MD steps

In computer science, when an algorithm is processing data serially as soon as it becomes available, it is called an online algorithm. In our case, we need such an online algorithm for tuning Nmd - because each data point is expensive to produce and we do not have the whole dataset from the beginning. Also, we are not aware of the exact relationship of the acceptance rate with the other parameters, such as the lattice size. We could do an exhaustive exploration of the other parameters, but this would take an unreasonable amount of time for the same computing resources. Of course, there are shortcuts to creating models when exploring a big design space. But the dynamic approach makes it less complicated and more general for future experiments where new parameters could be included, without knowing their amount of orthogonality.

We need to tune the Acceptance rate (the dependent variable) by manipulating the number of MD steps (the independent variable), whose relationship is unknown. Not only is it unknown, but there are some other parameters, such as the lattice size, that we know can influence this relationship for different simulations. Therefore, we concluded that an online algorithm would make things simpler, as the model would have to be created from observations using the same combination of all other parameters.

However, we are not completely unaware of the relationship we are looking for. We already know that:

- The data almost has the shape of a sigmoid function (S-shaped)
- The acceptance rate (dependant variable) starts from 0.0 for few steps and ends at 1.0 for a big number of steps.

How about other approaches?

There are many ways to create a model, such as regression by using a Muli-Layer Perceptron (MLP) (an Artificial Neural Network type). The problem with such an approach is that neural networks are very agnostic to the shape of the data we would like to model. It can take longer to train the network and it is also prone to overfitting (the error of the training data contributes to creating an inaccurate model).

The biggest disadvantage of using the neural network approach, is that since it doesn't take into consideration what we already know about the shape, it can give a model that fits the current data well, but makes very bad new predictions (such as with overfitting). Imagine the orange line of Figure 2 to be the result of training an MLP, but for any Nmd greater than 300 the acceptance rate falls back to values near 0. This is very possible with MLPs and with other approaches, and it is unwanted. That is the main reason for choosing to fit the data to a specific function by using least squares fitting instead.

Selecting a function for fitting

My code uses the scipy library which implements the Levenberg-Marquardt algorithm for the least squares functionality. To the user, it is very simple to tell this algorithm what parameters need to be found. From high school mathematics we already know that we can shift, stretch or shrink the graph of a function with a set of simple operations:

- If we want to move f(x) to the left by a, we use f(x+a)
- If we want to shrink f(x) by b in the x-axis, we use f(b*x)
- If we want to move f(x) up by c, we use f(x)+c
- If we want to stretch f(x) by d in the y-axis, we use d*f(x)

After we select the function, we can give all these freedoms to the algorithm. However, it would be very nice if we could take some of them away. This would make the fitting faster and increase its accuracy for predicting new data. In other words, by removing some of these freedoms, we are instructing the algorithm about the shape we are expecting it to find.

Sigmoid

A mathematical function that has a specific S shape curve is named sigmoid. My first selection for a function was the logistic function 1/(1+exp(x)), which ranges from 0 to 1.

Since we already know that the acceptance rate ranges from 0 to 1, then we do not need the last two operations described above. Our model then has the form 1/(1+exp((x+a)*b)), which is relatively accurate even if we only have a few points (beginning of the simulations) and also fast to fit.

In Figure 2, we can see the result of fitting the logistic function on the previous dataset.

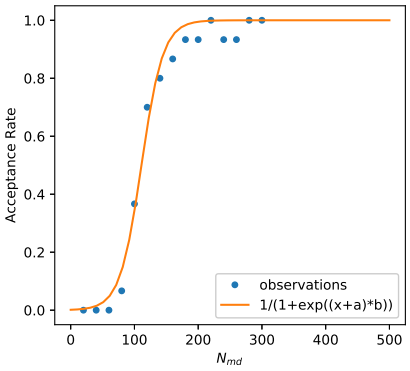


Figure 2: Fitting a sigmoid function

Cumulative Distribution Function

As we can observe, using a sigmoid function was a relatively good attempt. However, we can see that the orange line is more "pointy" than needed at the top part of the figure and less "pointy" than needed at the lower part. This is due to the fact that the sigmoid function is symmetric. The task was to look for functions that look like a sigmoid function but allow for a degree of asymmetry. The solution was to use the Cumulative Distribution Function (CDF) of the skew normal distribution.

The CDF gives the area under the Probability Density Function (PDF) and it has the shape we are looking for. The skew normal distribution allows skewness which can make the model fit our data better. In Figure 3, you can see a comparison of the normal distribution and the skew normal distribution.

As we can see, by manipulating the alpha parameter, we can change the shape of the PDF of the distribution and consequently the shape of the CDF as well. Of course, as the selection of the function was done visually, it does not necessarily mean that the physics gives a skew normal distribution form to the data, but it seems to suit our cases well. In the following figure, we can see a

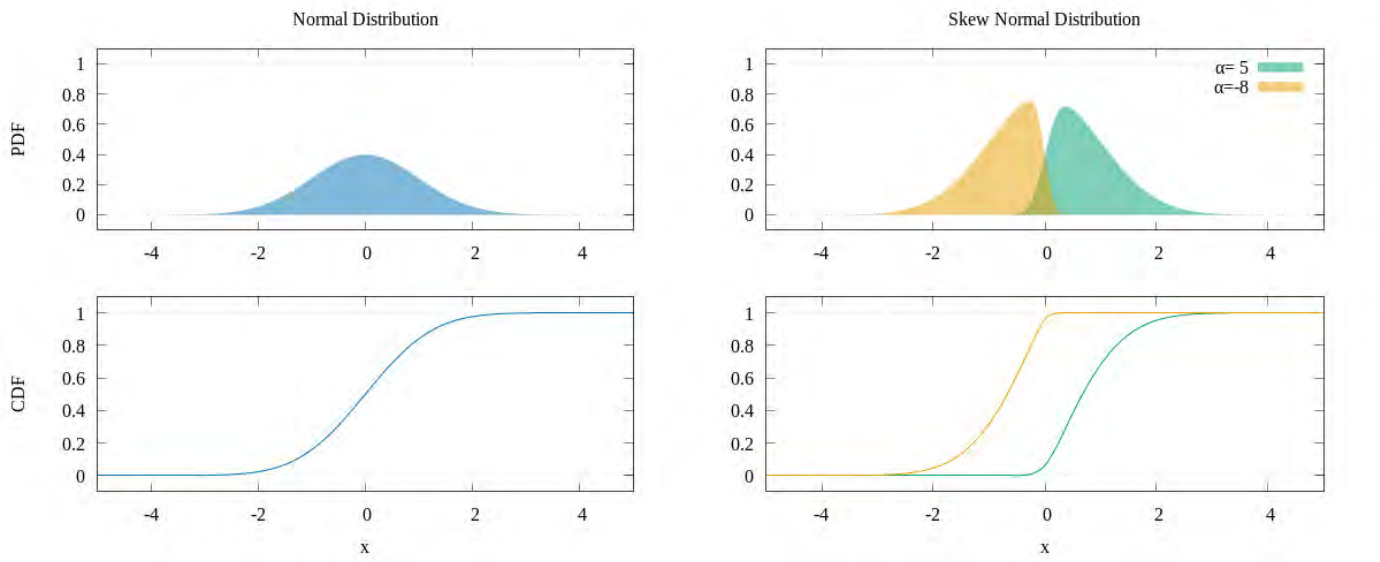


Figure 3: Comparison of the Probability Distribution Function (PDF) and the Cumulative Distribution Function (CDF) for the normal and skew normal distributions.

demonstration of the different CDFs we can obtain by varying the alpha parameter.

This "trick" gave a quick and accurate methodology for creating a relatively good model, even with a couple of observations. Another way of increasing the stability of the model was to insert two artificial points - one on each side, which would most likely be valid, such as f(0)=0.0 and f(600)=1.0. In this way we could get a seemingly valid shape for even one observation - good enough at least to allow the online algorithm to select a meaningful new Nmd, closer to the optimal.

Vizualisation of the algorithm

You can watch the video presentation by clicking on following link: 1716 Philippos Papaphilippou, Cude colors on a phine grid. At the 3:56 timestamp, you can see the resulting visualisation of the tuning in action.

The video visualization demonstrates the two states in which my auto tuner is at during tuning. At the first state, the Nmd is constant and a number of trajectories are calculated. This is continued until the confidence interval[3] gets sufficiently small for increased accuracy in measurements. At the second state, the fitting tooks place, (for a negligible time compared to the simulation time) and the new Nmd is selected according to the created model. This continues for multiple iterations until the end conditions are met.

In addition, there is a series of blog posts at the SoHPC website, where I have illustrated some of the above material using animated images.

Discussion & Conclusion

The tuning algorithm was shown to work relatively well with some example runs for different lattices. For example, at one case, the predicted Nmd for acceptance rate 66% was around 40, while the default Nmd was 500. This gave a speedup of over 10 times, although for more realistic parameter combinations the speedup was less.

There is still some work to be done to improve this algorithm. First, the acceptance rate is being calculated as a percentage of the accepted trajectories - just like the way frequency of heads is calculated when flipping a coin. Using the resulting proportion, the confidence interval is calculated using the Wald formula for a binomial proportion.[3] Instead of this, we can use the average of the acceptance probabilities (as given by the simulator) for greater accuracy and therefore less tuning time. We could also investigate other stopping conditions, or alternative fitting parameters, such as different loss functions, or even an equation to fit that derives from physics properties.

Regarding the simulator itself, there is still room for performance improvement by perfecting its vectorization. During the beginning of my stay here at Juelich I did some interesting vectorization exercises - some of which you can find in the blog's link above. I did not have much time for a second task, to further optimize the current implementation, but I learnt a lot during the initial hands-on activity and can easily apply it in other problems in the future, as well to identify bottlenecks in performance.

References

[1] M. Creutz Simulating quarks Computers in Science Engineering, March/April 2004, p. 80 (IEEE CS and AIP, 2004).

[2] Luu, Thomas and Lähde, Timo A (2016) Quantum Monte Carlo calculations for carbon nanotubes. Physical Review B, 93(15), p.155106.

[3] Vollset, S.E., 1993. Confidence intervals for a binomial proportion. Statistics in medicine, 12(9), pp.809-824.

PRACE SoHPC Project Title

Cude colors on a phine grid

PRACE SoHPC Site

Forschungszentrum Juelich, Germany

Philippos Papaphilippou

PRACE SoHPC Authors

Philippos Papaphilippou,
Forschungszentrum Juelich, Germany

PRACE SoHPC Mentor

Dr. Dr. Stefan Krieg,
Forschungszentrum Juelich, Germany

PRACE SoHPC Contact

Philippos Papaphilippou, Forschungszentrum Juelich, Germany
Phone: +44 7729024513
E-mail: p.p.cy@ieee.org

PRACE SoHPC Software applied

scipy, gnuplot, FFMPEG, internal software

PRACE SoHPC More Information

www.philippos.info

PRACE SoHPC Acknowledgement

Apart from my SoHPC mentor Dr. Krieg, Prof. Thomas Luu of Forschungszentrum Juelich, Germany also provided insightful feedback.

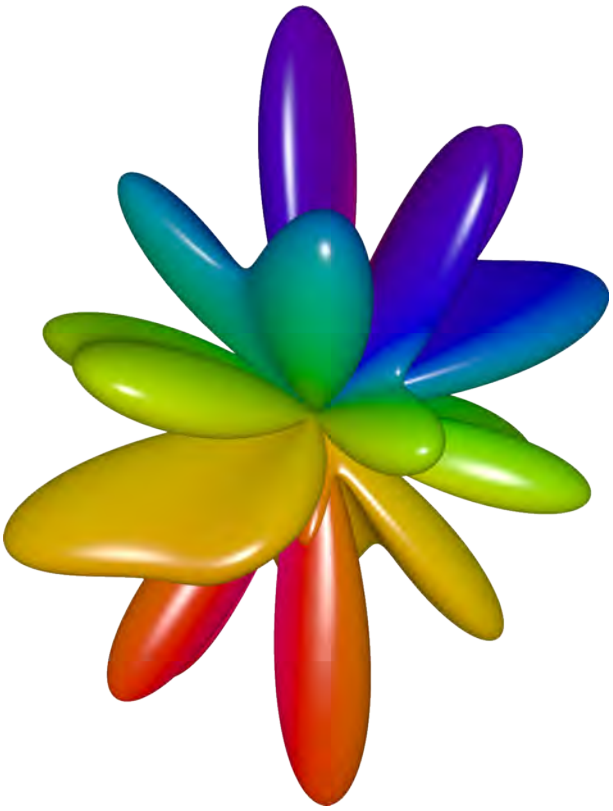
PRACE SoHPC Project ID

1716

Fast Multipole Method based solver for $1/r$ potentials using highly portable HIP framework for GPGPUs

Portable GPU code for FMM

Antti Mikkonen



Fast Multipole Method (FMM) enables fast and accurate n-body simulations. With GPGPUs it is possible to compute FMM solutions utilizing the cheapest FLOPSs available. Portability of the source code enables usage of the algorithm on all current GPGPU platforms.

Simulating the dynamics of molecules, plasma or solar systems require solving gravitational or electric forces between particles. This is called the n-body problem and its solution requires a numerical approach. With Fast Multipole Method, it is possible to reduce the computational complexity of n-body problem from quadratic to linear. Quadratic complexity means that doubling the amount of particles will increase the computation time fourfold, while with linear complexity doubling the particle amount results in twice as long computation. The importance of complexity reduction becomes evident when dealing with millions of particles - an amount which is necessary for accurate simulations.

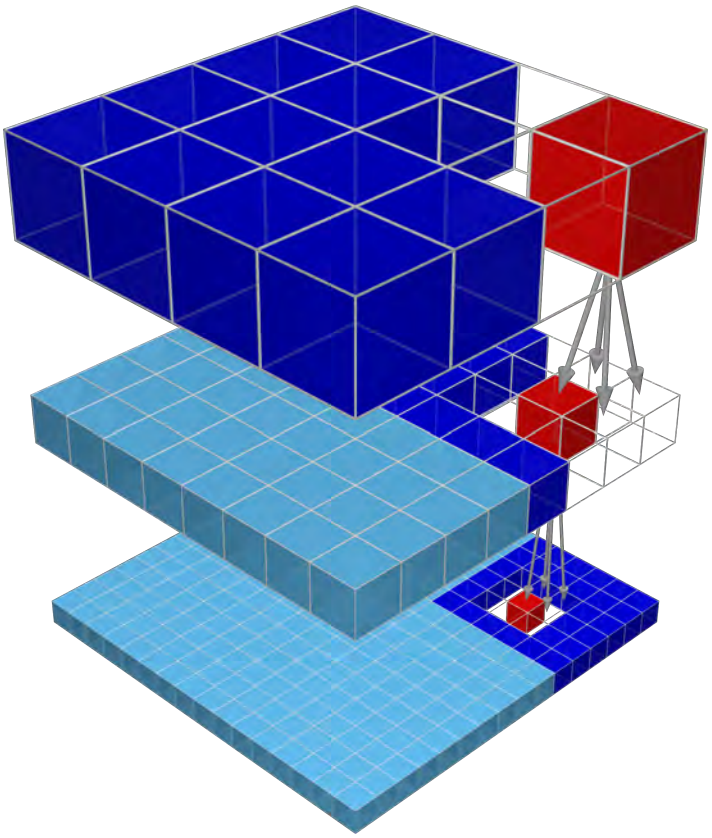
After solving the forces between particles, it is possible to iterate the dynamical simulation with one time step. Simulations usually require hundreds of thousands of time steps for the desired behaviour to occur, and a fast computation of forces is required. Thus, this becomes a high-performance computation problem. Graphical processing units (GPUs) provide an ideal approach to this problem. The problem doesn't require much memory and consists of many parts which are possible to compute using a Single Instruction Multiple Data (SIMD) approach - a quality in which GPUs excel.

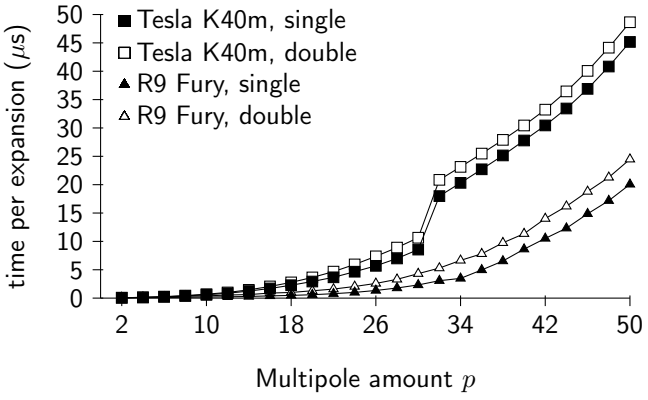
Currently there are two major GPU manufacturers, Nvidia and AMD. Their GPUs are different in architecture and the functionality they offer for GPGPU applications vary greatly - so a separate codebase is required for both Nvidia and AMD GPUs. Creating and maintaining two different source codes is time-consuming and is prone to errors. A new approach to this dilemma is AMD's open-source library Heterogeneous-Compute Interface for Portability (HIP). Utilizing the HIP framework, it is possible to decide on which hardware you want to run your code during compile-time. The primary objective of the HIP framework is to enable hardware-unaware development of GPU code - so the programmer may focus on the algorithm and not the implementation.

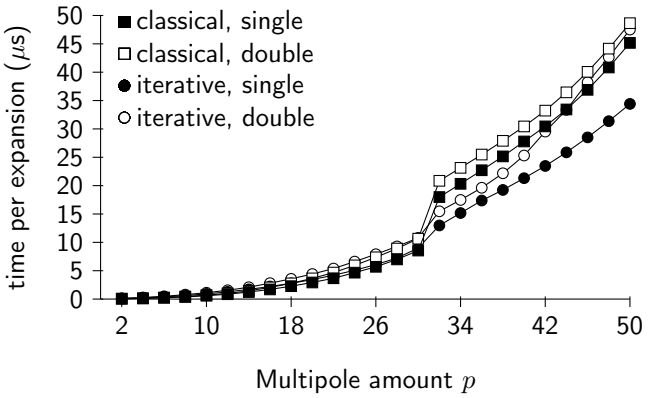
In the Fast Multipole Method, the computation area, which includes all the point masses, is divided into small subregions. Then it is possible to consider the inter-particle interactions in two classes: the near-field forces and far-field forces. A particle's near-field consists of its own region and neighbouring regions and far-field is conversely everywhere else. The speed of the FMM, results from the fact that the further away the far-field particles are from our particle of interest, the greater amount of interactions are combined into a single operation.

After the division, the far-field effects of each region is approximated using a multipole expansion of the potential function. A multipole is a linear combination of functions called spherical harmonics with complex coefficients. The featured image of the article is a visualisation of a multipole. With multipoles, you only need a handful of numbers to describe the potential function everywhere in the far-field. Since the multipole expansion is linear, you can create the combined field of several multipoles by adding up their effects. The multipoles are added up from smaller subregions to larger ones and these regions are then organised into an octree. A visualization of this tree is on the left side of following figure. This is an another source for the speedup for the FMM.

The Fast Multipole Method consists of five distinct steps. In the first four steps, the far-field forces are computed, and in the last one the near-field ones are computed. The first step is to compute the multipole expansions in the smallest subregions. In the second one, the multipoles are added up from smaller regions to bigger ones. The third step is to transform the effects of the multipoles into local effects in the same subregion size category. This third







Left: A two-dimensional Fast Multipole Method tree. Three different size levels are displayed. The near-field subregions are transparent, the far-field subregions are displayed in dark blue and non-interacting subregions in light blue. **Top right**: Computation time comparsion between Tesla K40m and Radeon R9 Fury GPUs with varying multipole counts in both single and double precision floating point numbers. **Bottom right**: Computation time comparison between the classical rotation operation and the iterative rotation operation with varying multipole counts on Tesla K40m GPU in both single and double precision floating point numbers.

step is called Multipole2Local, and it is the most computationally intesive part of the algorithm. The fourth step is to add up these local far-field effects from all size levels and this yields us the far-field potential function, from which it is easy to compute the forces. The final step is to classically compute the near-field forces for each particle pair.

I implemented this Multipole2Local (M2L) part of the algorithm on AMD's Radeon R9 Fury and Nvidia's Tesla K40m graphics cards. The implementation was done using the HIP framework. The HIP C++ library contains general GPGPU-related functions which are converted by the HIPCC compiler package to more device-specific code. This extra layer of code abstraction results in a more portable code, but with fewer features. Only features which are found on both platforms can be used in HIP. However, it is also possible to write device-specific code with appropriate preprocessor macros.

The traditional algorithm for Multipole2Local has a time complexity of $\mathcal{O}(p^4)$. This means that the runtime of the algorithm is proportional to the fourth power of the multipole amount p. As such, the algorithm gets increasingly slower and slower with more multipoles. This can partly be fixed by introducing a multipole rotation into the algorithm. Adding more functionality to an already complicated code doesn't sound too useful performance-wise. But this time it's possible to reduce the total time complexity depending on the multipole number to $\mathcal{O}(p^3)$. Thus the rotation-based M2L operation, can in theorty be at most 50 times faster than the original code.

GPUs, as high-performance computing units, may execute many threads in parallel, while having comparatively little memory. Thus a computation bottleneck may be created when thousands of threads are reading and writing data at the same time. Because of this, it might be beneficial to compute needed data on the thread itself to avoid reading slow memory. This is underlined by the fact that on a GPU, computation is cheap, but reading memory is expensive. I also implemented this method of approach on Tesla K40m.

The computation time comparison may be seen on the right side of the figure. The benchmarking was done by computing the M2L operation for 4096 multipole expansions in parallel. In the top graph, we see that both GPUs run the code quite similarly with small values of p, but the Tesla K40m experiences a sudden extra delay around $p = 32$. This is due to the fact that only 32 threads may be computed fast using SIMD methodology on an Nvidia GPU processor and this can be avoided with different implementations. On the bottom graph, it is interesting to notice that the iterative approach is significantly faster - but only when using single precision floating point numbers.

In summary, HIP provides a simple way to create portable GPU code and its learning curve is near non-existent if you're coming from Nvidia's CUDA language. However, since the framework is in development, the more complex GPU features aren't available yet.

PRACE SoHPC Project Title

Hip, hip, hooray! Get your 2-for-1 GPU deal now.

PRACE SoHPC Site

JSC, Germany

PRACE SoHPC Authors

Antti Mikkonen, University of Eastern Finland, Finland

PRACE SoHPC Mentor

Andreas Beckmann, JSC, Germany

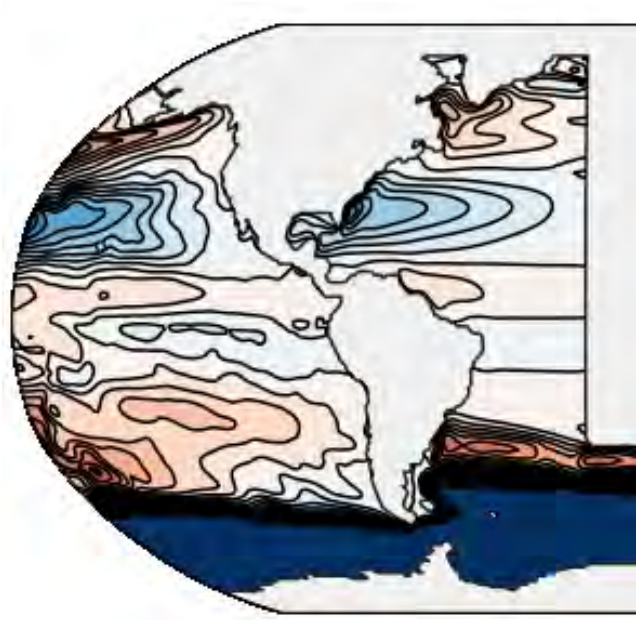
PRACE SoHPC Project ID

1717

Antti Mikkonen

Accelerating climate kernels for SoHPC

Konstantinos Koukas



Climate research depends on performing simulations over hundred-thousand-year time scales - requiring high performance computing capabilities in order to advance the field. This project manages to boost the performance of ocean simulation models by utilising all available processor cores, as well as porting a pressure solver to run on GPUs.

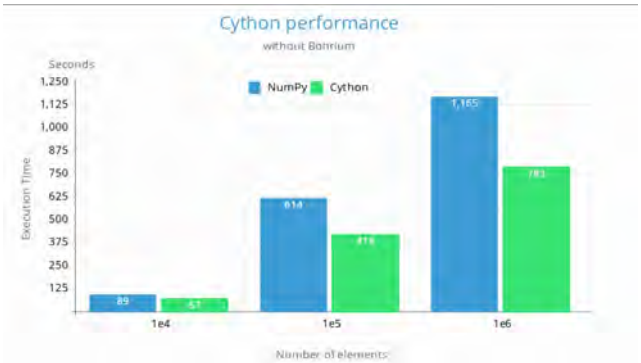
Climate research serves the critical goal of advancing our understanding and projections of climate behaviour by developing global climate models. These are used to simulate conditions over long periods of time - including factors such as the atmosphere, oceans and the sun. Evidently, a task of such complexity is computationally intensive and relies on the use of supercomputing facilities. This is why climate models estimate trends - rather than events, and their results are less detailed in comparison to similar models used for weather forecasting. Thus, climate models no longer scale as computers are getting bigger. Instead, climate research depends on extending the length of a simulation into hundreds of thousands of years - creating the need for faster computers in order to advance the field.

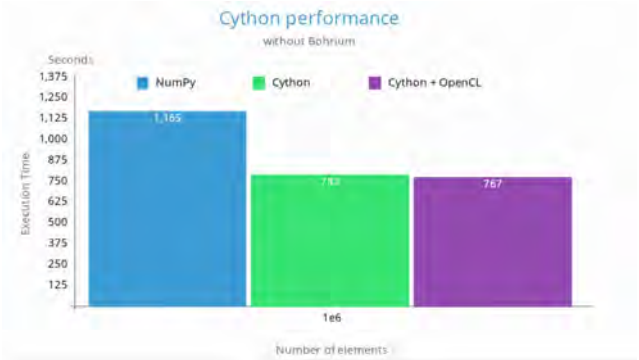
A number of ocean simulation models are implemented in the Versatile Ocean Simulation (Veros) framework. Written in pure Python, it aims to be the Swiss Army knife of ocean modelling. However, choosing a dynamically typed and interpreted language leads to heavy computational cost. Overcoming this gap is accomplished through the use of Bohrium - a framework that acts as a high performance alternative for the scientific computing library NumPy, which takes care of all the parallelism in the background. However, Bohrium carries a significant computational overhead, making custom optimisation essential.

To enable massively accelerated ocean simulations, I decided to combine the power of handmade parallel code with the automatic capabilities of Bohrium. To this end, I identified the most time-consuming parts of the framework and optimised them using Cython. Also, I took advantage of Cython's OpenMP support to enable parallel processing, and ported a pressure solver to run on GPUs - leaving the rest of the framework to be optionally optimised by Bohrium.

Profiling

Profiling showed that most time is spent in Thermodynamics and specifically, isoneutral mixing methods. These parts were optimised using the following methods.





Comparison between NumPy, Cython and Cython with the OpenCL solver.

Static Types

Cython enables the developer to combine the productivity, expressiveness and mature ecosystem of Python with the bare-metal performance of C/C++. Static type declarations allow the compiler to produce more efficient C code. In particular, low-level numerical loops in Python tend to be slow. Although NumPy operations can replace many of these cases, the best way to express many computations is through looping. Using Cython, these computational loops can have C performance. Finally, Cython allows Python to interface natively with existing C, C++ and Fortran code, as well as to disable safety checks using compiler directives. Overall, it provides a productivity and runtime speedup with minimal amount of effort. In order to take advantage of the above benefits, I added explicit types, turned certain numerical operations into loops and disabled both indexing and division checks.

Typed Memoryviews

I decided to exploit another Cython feature: Typed memoryviews. They allow efficient access to memory buffers, without any Python overhead. Indexing on memoryviews is automatically translated to a memory address. They can handle NumPy, C and Cython arrays. In my case, NumPy indexing was mostly replaced by the faster memoryview alternative.

OpenMP

Cython supports native parallelism via OpenMP. In a parallel loop, OpenMP starts a group (pool) of threads and distributes the work among them. Such a loop can only be used when the Global Interpreter Lock (GIL) is released. Luckily, the memoryview interface does not usually require the GIL. Thus, it is an ideal choice for indexing inside the parallel loop. Every computational loop was transformed into a parallel version with a constant number of threads - the number of which depends on the architecture.

ViennaCL

Veros solves a pressure equation using the Biconjugate Gradient Stabilized (BiCGSTAB) iterative method. This solver, along with other linear algebra operations are implemented in the ViennaCL library. ViennaCL is an open-source linear algebra library implemented in C++ and supports CUDA, OpenCL and OpenMP. In order to execute the solver on the GPU, I created a C++ method that uses the ViennaCL interface. Cython passes the input matrices to C++ where they are reconstructed and copied to the GPU. The solver runs on the GPU and the result vector is copied back to CPU memory. Finally, it is sent back to Cython without any copies. This procedure was implemented using both CUDA and OpenCL.

Results

As shown in the respective charts, Cython with OpenMP provided a 1.48x speedup over the original NumPy version. In the case where the pressure solver runs on the GPU, only a minor improvement is observed. Bohrium was not used in the benchmarks because of its large overhead for the examined setup sizes.

Conclusion

Clearly, handmade optimisation accelerated ocean simulations compared to pure NumPy, when the number of elements is below a certain threshold and Bohrium is not beneficial. However, even in larger cases, parts that Bohrium fails to speed up could benefit from such a custom tuning process. You can watch a visual explanation of the project in its presentation video.

PRACE SoHPC Project Title
Accelerating climate kernels

PRACE SoHPC Site
Niels Bohr Institute, Denmark

PRACE SoHPC Authors
Konstantinos Koukas, Department of Informatics and Telecommunications of the University of Athens, Greece


Konstantinos Koukas

PRACE SoHPC Mentor
Mads Ruben Burgdorff Kristensen, Niels Bohr Institute, Denmark

PRACE SoHPC Contact
Brian Vinter, Niels Bohr Institute
Phone: +45 35 32 14 21
E-mail: vinter@nbi.ku.dk

PRACE SoHPC Software applied
Veros, Cython, ViennaCL

PRACE SoHPC More Information
http://veros.readthedocs.io/
http://cython.org/
http://viennacl.sourceforge.net/

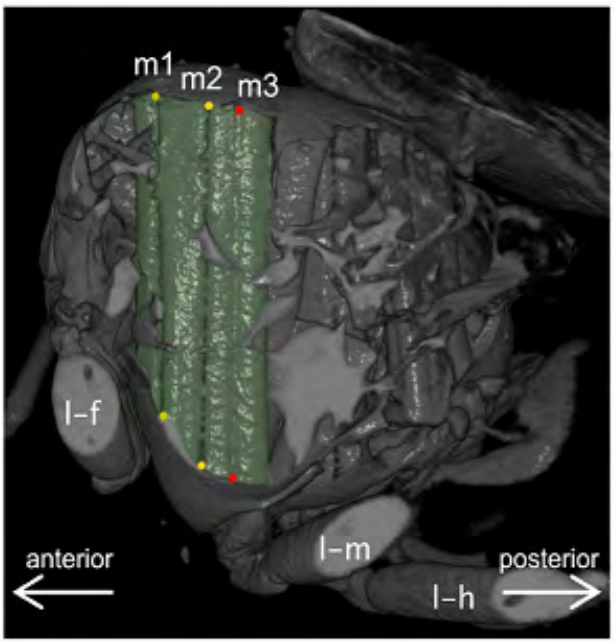
PRACE SoHPC Acknowledgement
PRACE

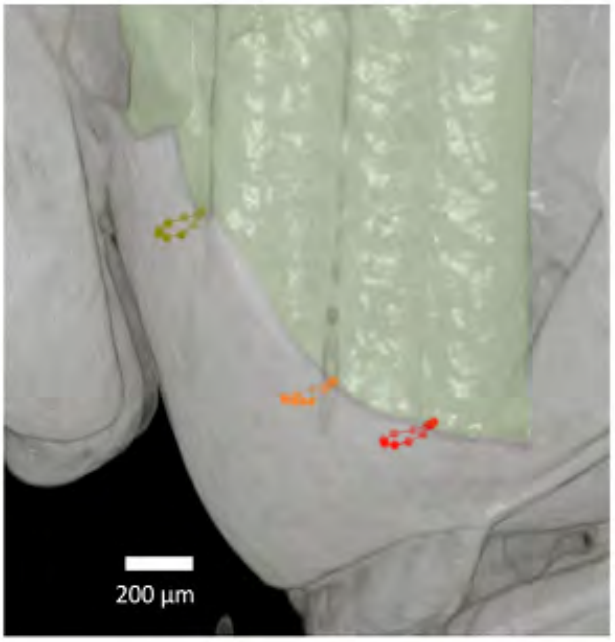
PRACE SoHPC Project ID
1718

Generalising object tracking algorithms to time sequences of 3D voxel data

Tracing in 4D data for SoHPC

Alessandro Marzo





Europe has several synchrotron facilities which can produce enormous datasets - especially the medical beam lines. They collect 3D volumes at a frequency that produce 3D movies. The analysis of this data typically involves tracing one or more objects in 3D over time. Existing algorithms are typically implemented in Matlab and do not scale to the new data rates. Thus, the aim of the project is to obtain a single algorithm implemented in a High Performance version.

The problem of object tracking is well studied for two dimensional videos, and the academic literature offers lots of different algorithms for many different situations. The same cannot be said for three dimensional data. Existing implementations of 3D object tracking algorithms are scarce and typically implemented in programming languages that do not perform well as the data grows. This is what's happening as medical and scientific research needs to study phenomena in more detail. The aim of my project is to overcome this problem by realising an optimised and parallelised implementation of a 3D object tracking algorithm.

What is 4D data?

By 4D data we mean 3D movies, but not the kind you can watch in a cinema. 3D films in cinemas are not really 3D, but they use stereoscopic photography to give an illusion of depth. This means that the movie is recorded from two different horizontal positions to get a stereoscopic image pair. These two two-dimensional images are then presented separately to the left and right eye of the viewer to give the perception of 3D depth when they are combined in the brain.

3D movies mentioned in this article, maintain full information about depth. Each 3D volume is represented using voxels. Voxels are the 3D version of pixels in a 2D image. If you think of pixels as pieces of a cardboard puzzle, voxels would be like LEGO. Because of this, voxels require more disk space and are more computationally intensive to analyse and process. Each frame can take up to 15 GB and just two frames are equal to an entire Blu-ray disc! This is why we need high performance computing.

Starting the project

Fortunately for me, as I started studying the various 2D algorithm versions, I noticed that the math used could be easily generalised to three dimensions without modifying most of the steps of the algorithms. This was quite a promising start. The next step was using existing

implementations of 2D algorithms as a starting point towards generalising to 3D volumes. This required looking into the OpenCV library.

As I started diving into the OpenCV implementation code, I quickly realised that it was too optimised for me to make something useful out of it. Optimisation often introduces complexity in the code which in return makes it harder to understand. Let's remember that the OpenCV project began 18 years ago and the main contributors were a group of Russian optimisation experts who worked at Intel. And they also had 18 years to optimise it even more!

So I needed to start from scratch, reinvent the wheel and make my own implementations of the algorithms.

The approach

Python is the most straightforward choice to complete the project in a short time frame. Python is a modern programming language that enables fast prototyping of algorithms because it is easy to learn and use. Furthermore, it offers a lot of specialised packages for scientific computing and visualisation.

But this was not the only reason to choose Python. We can also use Bohrium - a framework that acts as a high performance alternative for the scientific computing library numpy. Bohrium takes care of all the parallelism in the background, so we can concentrate on writing nice, readable code, that will be capable of running on multicore CPU and GPU without modifications.

In the end, these advantages allowed me to implement and compare the accuracy and performance of two different object tracking algorithms: Medianflow and Kernelized Correlation Filters (KCF).

Medianflow

Medianflow[1] is the first algorithm I chose to generalise to 4D data. It is mainly based on one simple idea: the algorithm should be able to track the object regardless of the direction of time-flow. We first identify objects using a grid of points and then we evaluate them with a measure called the Forward-Backward error, so we can only use the best points for tracking. The concept is well explained by Fig 1.

Point 1 is visible in both images and the tracker is able to track it correctly. Tracking this point forward and backward results in identical trajectories.

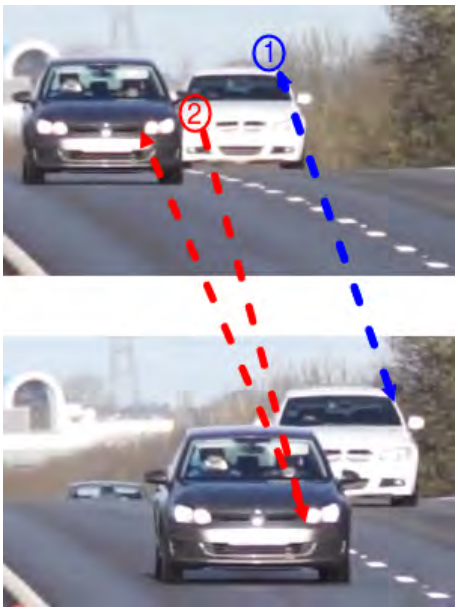


Figure 1: Forward-backward consistency assumption.

On the other hand, Point 2 is occluded in the second image and the tracker localises a different point. Tracking this point backward, the tracker finds a different location than the original one. In the algorithm, the discrepancies between these forward and backward trajectories are measured. If they differ significantly, the forward trajectory is considered as incorrect. This Forward-Backward error penalises these inconsistent trajectories and enables the algorithm to reliably detect tracking failures and select reliable trajectories in video sequences.

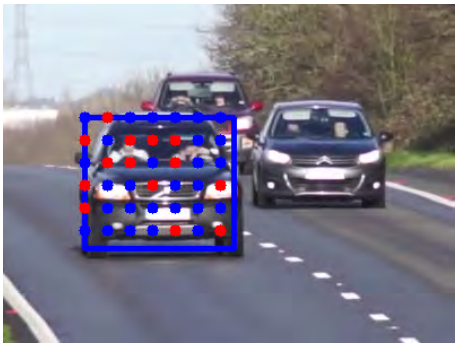


Figure 2: Red points are correctly tracked and the ones chosen to update the bounding box for the next frame.

The points are tracked using an old-style computer vision algorithm such as the Lucas-Kanade feature tracker.[2] This is based on sparse optical flow estimation. The concept of optical flow is quite old and it extends outside the field of Computer Vision - but in this article you can think of optical flow as the displacement of one point between two frames. This algorithm works by solving the optical flow equation 1 using the least square principle.

$$\nabla I^T \cdot \vec{V} = -I_t \qquad (1)$$

As you may have noticed, this equation is valid for both two and three dimensions, so we can generalise this algorithm to 3D volumes without any setbacks.

Kernelized Correlation FIlters

The second algorithm I chose to implement is KCF.[3] While MedianFlow is based on old computer vision techniques, KCF is inspired by new statistical machine learning methods. It works by building a discriminative linear classifier (which some people would call artificial intelligence) tasked with distinguishing between the target and the surrounding environment. This method is called 'Tracking by detection'. The classifier is typically trained with translated and scaled sample patches to learn a representation of the target. It then learns to predict the presence or absence of the target in an image patch. This can be very computationally expensive:

- In the training phase, the classifier is trained online with samples collected during tracking. Unfortunately, the potentially large number of samples becomes a computational burden, which directly conflicts with real-time tracking requirements. On the other hand, limiting the samples may sacrifice performance and accuracy.
- In the detection phase, similar to other algorithms, the classifier is tested on many candidate patches to find the most likely location. This is also very computationally expensive and we encounter the same problems as before.

KCF solves this by using some nice mathematical properties in both training and detection. The first mathematical tool that KCF employs is the Fourier transform - taking advantage of the convolution theorem. This states that the convolution of two patches is equivalent to an element-wise product in the Fourier domain. So, formulating the objective in the Fourier domain can allow us to specify the desired output of a

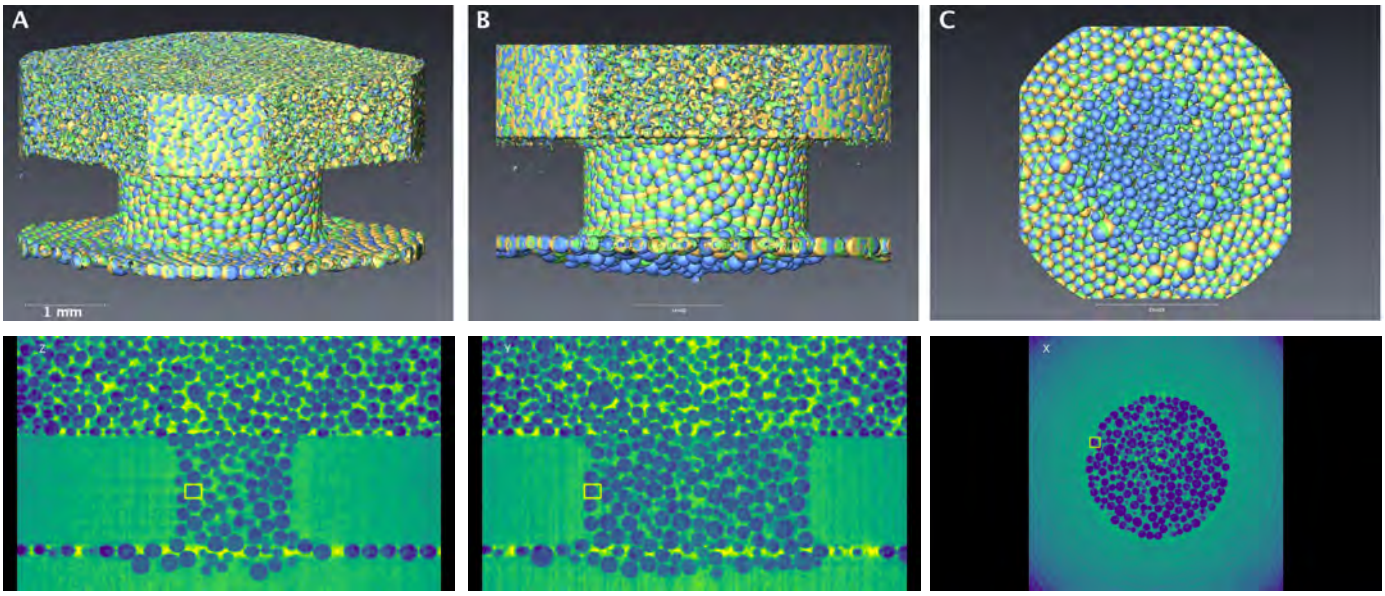


Figure 3: Top: Tomographic reconstruction of a system that is a constriction through which a liquid foam is flowing. Bottom: Tracking results visualisation shown for the last frame of the volume sequence. The algorithm worked since it is still tracking the bubble in the last frame.

linear classifier for several translated image patches at once.

This isn't the only benefit of the Fourier transform because interestingly, as we add more samples, the problem acquires a circulant structure. A circulant block matrix is a matrix that contains all possible translations of the image along its dimension. These matrices contain a lot of redundant data, but only the first row is actually needed to generate the matrix. These matrices also have a really amazing mathematical property as they are all made diagonal by the Discrete Fourier Transform (DFT) - regardless of the generating image. So, by working in the Fourier domain, we can avoid the expensive computation of matrix multiplications because all these matrices are diagonal and all operations can be done element-wise on their diagonal elements.

Using these nice properties, we can reduce the computational cost from $O(n^3)$ to nearly linear $O(n \log n)$, bounded by the DFT. The $O(n^3)$ complexity comes from the Ridge regression formula 2 that we need to solve to learn a representation of the target:

$$w = (X^T X + \lambda I)^{-1} X^T y \qquad (2)$$

In other words, the goal of training is to find a function $f(z) = w^T z$ that minimises the square error over samples (which are the rows of the circulant block matrix) and their regression targets y_i. This means finding the parameters w.

There is still one last step we can do to improve the algorithm. We can use the "kernel trick" to allow more powerful non-linear regression functions $f(z)$. This means moving the problem to a different set of variables (the dual space) where the optimisation problem is still linear. On the downside, this usually means that evaluating $f(z)$ typically grows in complexity with the number of samples. But we can also use all previously discussed properties with the kernelised version of Ridge Regression and obtain non-linear filters that are as fast as linear correlation filters - both to train and evaluate.

Results and Conclusions

The algorithms are tested and the results are shown for the data in Fig 3. The reason I implemented two algorithms is because while the 3D generalisation of Medianflow worked with a good tracking accuracy, its execution time was problematic. Selecting a more advanced and modern tracking algorithm like KCF improved the performance - achieving a $\times 100$ speedup of the program with the same tracking accuracy. This didn't leave me a lot of time to dedicate to the parallelisation of the algorithm. Using Bohrium with Medianflow had positive results, achieving a $\times 8$ speedup in certain parts of the algorithm. But further work is needed with KCF.

Nevertheless, this area of computer vision, the field which deals with these kinds of problems, is still at its beginning. Only in the last few years has this problem even come up and the computation required by the generalised algorithms is only now possible thanks to modern hardware capabilities. So it was very exciting for me to work on this challenging project!

References

[1] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 2756–2759. IEEE, 2010.

[2] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674-679.

[3] Henriques, João F., et al. "High-speed tracking with kernelized correlation filters." IEEE Transactions on Pattern Analysis and Machine Intelligence 37.3 (2015): 583-596.

PRACE SoHPC Project Title
Tracing in 4D data

PRACE SoHPC Site
Niels Bohr Institute, Denmark

PRACE SoHPC Author/Contact
Alessandro Marzo, University of Bologna, Italy

PRACE SoHPC Mentor
Brian Vinter, UCPH, Denmark


Alessandro Marzo

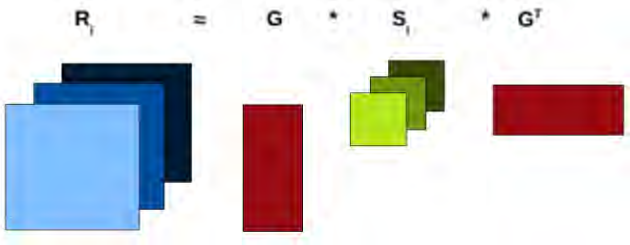
PRACE SoHPC Software applied
Bohrium

PRACE SoHPC More Information
bohrium.readthedocs.io

PRACE SoHPC Acknowledgement
I would like to thank my project mentors Brian Vinter and Mads R. B. Kristensen for all the advice and support they gave me. I would also like to thank all the staff at NBI and PRACE.

PRACE SoHPC Project ID
1719

Matrix Trifactorization



Jan Packhäuser

The aim of the Multitype Symmetric Non-negative Matrix Tri-Factorization (MSNNMTF) is to extract information contained in multitype relational data sets, e.g. networks, that include both intertype and intratype relationships. The large-scaled real data need to be arranged by efficient methods. Therefore, this report explains a parallelised implementation of MSNNMTF based on two approaches using either a fixed point method or a projected gradient method. Applications of the MSNNMTF can be found in lots of fields, such as data science or biology.

In order to evaluate a multiple network alignment one has to simultaneously decompose each network matrix R_i into the product of three matrix factors of lower dimensions. This can be seen as a minimisation problem in the following way:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} ||R_i - GSG^T||^2 \\ \text{s.t.} \quad & G \geq 0, \quad S = S^T, \end{aligned}$$

where $R_i \in \mathbf{R}^{n\ x\ n}$ are squared network matrices, $S_i \in \mathbf{R}^{k\ x\ k}$ are also squared matrices containing specific information about the network i and $G \in \mathbf{R}^{n\ x\ k}$ is shared across all single decompositions and contains common information of all networks. Throughout the whole report $||\cdot||$ denotes the Frobenius norm.

In Figure 1, the idea of the approach is illustrated.

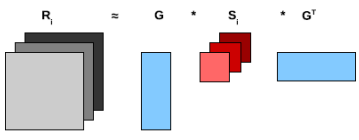


Figure 1: Idea of MSNNMTF

To solve this optimisation problem, we first derive the Karush-Kuhn-Tucker (KKT) conditions (see figure 2), and apply iterative methods on the first order condition. These iterative methods will be the fixed point method on the one hand and the projected gradient method on the other hand. The following four equations are the so called KKT-conditions:

$$\sum_{i=1}^{N} \left(-2(R_i^T G S_i + R_i G S_i^T) + 2(G S_i G^T G S_i^T + G S_i^T G^T G S_i)\right) - \beta = 0 \quad (1)$$

$$G^T R_i G - G^T G S_i G^T G = 0 \quad (2)$$

$$\beta, G \geq 0 \quad (3)$$

$$\langle \beta, G \rangle = 0 \quad (4)$$

Figure 2: KKT-conditions

where $\beta \in \mathbf{R}^{n\ x\ k}$ is the dual variable with regard to constraint $G \geq 0$. We assume that the matrices R_i are symmetric (these are not very restrictive assumptions since the underlaying networks are undirected), therefore S_i are symmetric, too. The KKT conditions 1 to 2 imply that the Lagrangian function is set to zero and conditions 3 to 4 are called complementary conditions.

Which iterative methods could be applied?

Fixed point method

Similar to,[1] one can derive the following update rules for the fixed point method.

$$S_i = (G^T G)^{-1}(G^T R_i G)(G^T G)^{-1}$$

$$G_{ij} = G_{ij}\sqrt{\frac{\sum_i \left((R_i G S_i)_{ij}^{+} + (G(S_i G^T G S_i)^{-})_{ij}\right)}{\sum_i \left((R_i G S_i)_{ij}^{-} + (G(S_i G^T G S_i)^{+})_{ij}\right)}}$$

This results in the algorithm you can see in Figure 3.

Projected gradient method

An alternative way to solve () is by alternating G and $\{S_i\}$. More precisely, we solve the optimisation problem for fixed G in variables $\{S_i\}$ and then the other way around. This well-known nonlinear optimisation approach is called Black coordinate descent, see.[2] The optimisation problem to compute $\{S_i\}$ for fixed G is an unconstrained convex problem, and due to convexity, we compute the

Fixed Point Method to solve ()

INPUT: $R_i, k, G, MAXITER$
count $= 0$
while count $<$ MAXITER
for $i = 1, \ldots, N$
Update S_i
end (for)
Compute factor that is needed to update G
Update G
count $++$
end (while)

Figure 3: Pseudocode to solve MSNNMTF with fixed point method

optimal solution using the following formula:

$$S_i = (G^T G)^{-1}(G^T R_i G)(G^T G)^{-1}$$

The formula arises when setting the gradient of the objective function to 0:

Instead, we have a non-convex constrained optimisation problem for fixed $\{S_i\}$ with the constraint $G \geq 0$. This can be approximately solved using a projected gradient method as suggested in.[3] Note, that the gradient of the objective function is

$$\nabla_G F(G) = 4 \sum_{i=1}^{N} (-R_i G S_i + G S_i G^T G S_i).$$

Hence, we obtain the following algorithm (see Figure 4), to solve ().
Computing the step size α, one has to consider $f(\alpha) = F(G + \alpha \cdot \nabla_G F(G))$ and determine $\alpha \in [-1, 0.5]$ yielding the smallest value of f. Calculating the optimal α is expensive. Therefore, we follow the approach to discretise the interval $[-1, 0.5]$ with a certain acuteness and take the best choice of α out of this discrete interval. This results in an imprecise output, but can be computed much faster and can even be parallelised.

Implementation

The goal of this report is to present a C++ parallel version of an existing MATLAB code, which was developed during my Summer of HPC project. The steps that define the complexity of the algorithm require solving two systems of linear equations in a row for each iteration, and each level i in both methods. It is also pretty expensive to compute the update factor for the fixed point method where many matrix multiplications are required. For the projected gradient method, we have to compute the gradient first before we can update our G.
These steps can be easily parallelised - splitting up parts of the whole computation and either building the update factors or the gradient within the for-loop that computes S_i. So, we accumulate everything in the for-loop that can be calculated there and since these calculations are independent, we will assimilate them concurrently using OpenMP. After this basic step, the main potential of parallelisation is already exploited for the fixed point method. However, for the Projected gradient method the computation of the different discrete interval α can still be parallelised. In a second step we can think about how we could improve the code further. Many matrix multiplications are present which can be parallelised.

The code is written in C++ using external linear algebra libraries. The Eigen-library is utilised throughout the whole code. Not only to handle both sparse matrices and dense matrices but also to solve systems of linear equations.

A few challenges

Some time was spent to detect the right linear algebra library. We decided to use the Eigen-library because it is quite easy and because it is uses efficient libraries as backends itself - such as BLAS and LAPACK. We know, that to gain performance then it is best to avoid libraries that are so called user-friendly, but one must take into account that code also needs to be legible for people who are not familiar with programming.

First improvements

We experienced first improvements once we used the Eigen library correctly. This means that we used the correct matrix formats throughout the code but essentially we had to resolve aliasing performance problems which are assumed by the Eigen's library at matrix multiplications by default. Specifically, aliasing means that the Eigen's library evaluates the product in a temporary matrix which is assigned to the actual matrix afterwards. This is necessary for a few cases, but mostly we can abolish this effort. This yields a considerable performance improvement when processing huge matrices.

More improvements

We used available Intel compiler licences and this benefited us quite a lot to use it and all compiler flags that come along with it. We could even achieve more performance utilising the Intel-MKL while we combined it with the Eigen-library. This works particularly when you enable BLAS and LAPACK as backends for the matrix multiplications and decompositions. We also linked it with OpenBLAS which is an open-source optimized version of BLAS. The Eigen-library automatically detects that it can use the optimised version instead of its customary version. From now on, all matrix multiplications that mainly occur inside for-loops were parallelised as well.

RESULTS

Comparison of RSE

In this section, we investigate the difference in convergency for both the FPM and PGM. For this purpose, we generate random sparse matrices R_i with a density of 5% and compute the final RSE after 1000 iterations for different inner dimensions k. In table 1 we present numerical results that we obtained.

Projected Gradient Method to solve ()

INPUT: $R_i, k, G, MAXITER$
count $= 0$
while count $<$ MAXITER
for $i = 1, \ldots, N$
Update S_i
end (for)
Compute step size α and gradient $\nabla_G F(G)$
Update G with $G = G + \alpha \cdot \nabla_G F(G)$
G = max(G,0)
count $++$
end (while)

Figure 4: Pseudocode to solve MSNNMTF with projected gradient method

n	k	RSE FPM	RSE PGM
1000	5	0.945899	0.945645
1000	10	0.943787	0.940385
1000	20	0.939502	0.929871
1000	50	0.926201	0.897076
1000	100	0.894461	0.841111
1000	500	0.610505	0.400760
1000	999	0.001531	0.001531

Table 1: The table compares FPM and PGM. It shows the value of RSE after 1000 iterations for a fixed dimension n of 1000 and for different dimensions k.

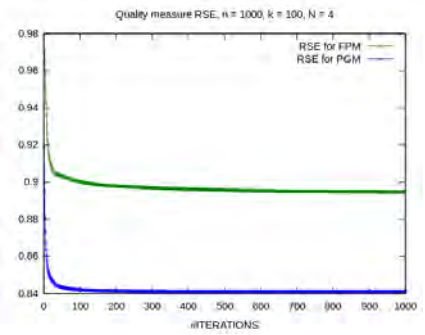


Figure 5: This figure compares the value of RSE for both methods FPM and PGM at each iteration.

Figure 5 shows that the fixed point method converges much slower than the projected gradient method.

Benchmark comparison

In this section we provide two different benchmarks where we want to investigate the impact of the different dimensions. Firstly, we fix $k = 20$, $N = 4$ and run the code for different dimensions n while we measure the execution time. Secondly, we fix $n = 2000$, $N = 4$ and run the code for different dimensions k. The results are visualised in the following figure in a double logarithmic manner. In both cases the maximum number of iterations was 500.

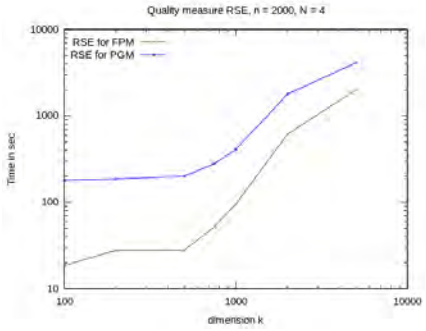


Figure 6: This figure shows the execution time for both methods FPM and PGM for different dimensions k.

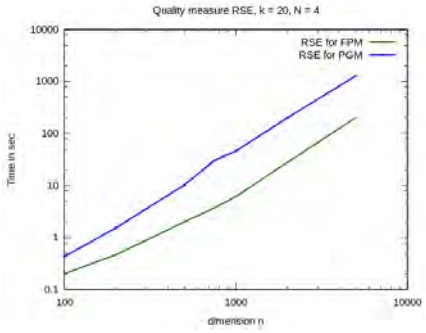


Figure 7: This figure shows the execution time for both methods FPM and PGM for different dimensions n.

Of course, we also want to compare the execution time between the C++ code and the MATLAB code.

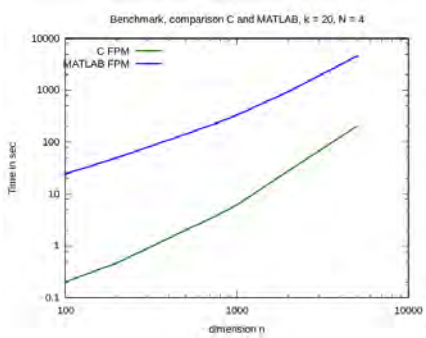


Figure 8: This figure shows the execution time for the method FPM and compares the C++- with the MATLAB-implementation.

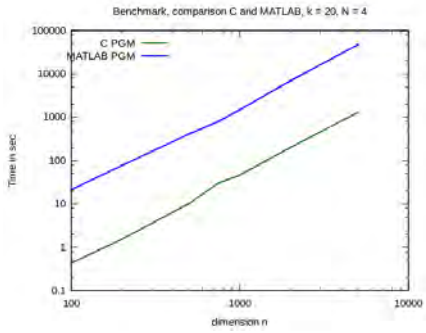


Figure 9: This figure shows the execution time for the method PGM and compares the C++- with the MATLAB-implementation.

Conclusions

The approach to solve the Matrix-Trifactorization with either the fixed point method or the projected gradient method work out quite well. However, we have to point out that the FPM converges much slower and sometimes it does not even come as close to the solution as the PGM. For reasons of accuracy, we highly recommend the usage of PGM instead of the FPM.

Future research

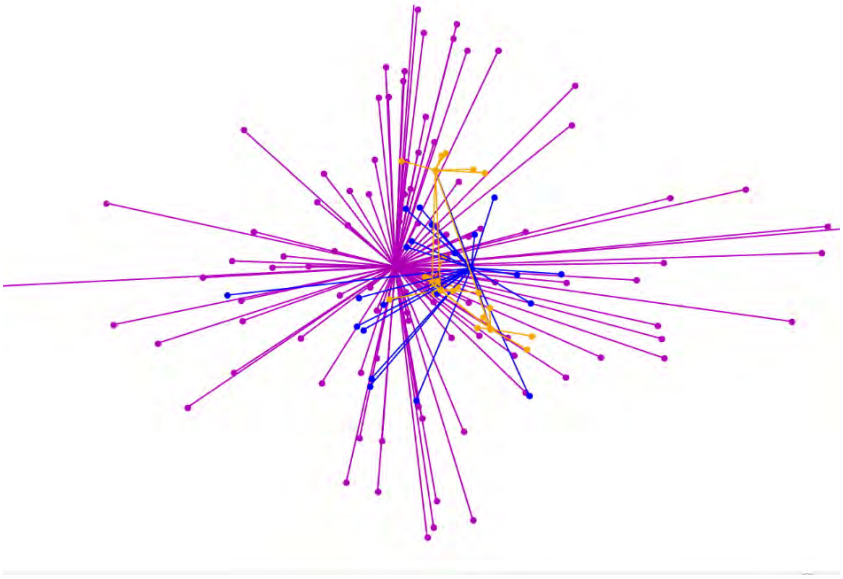


Figure 10: Multilayered networks

On the other side, we have to state that the FPM iterates much faster than the PGM as one can see in Figure 7. In fact, we can read a linear dependency out of the figure since the execution time is plotted in a double logarithmic manner and the shape of the graphs are linear. More precisely, the difference in the y-axis intercept reveals the factor of speed gain whereas the gradient of both lines point out the degree of the polynomials. Therefore, FPM iterates about 5 to 6 times faster than PGM for an increasing dimension n (see Figure 7). Overall, the question of how one can benefit from FPM arises. In practice, it is a well-known approach in many applications to use fixed point methods as a preprocessor in order to find better starting values.

Apart from that, we investigate, based on the execution time, how much faster the C++ implementation is compared to the MATLAB-implementation for both methods. In figures 8 and 9 we can see that there is a huge difference. Generally, the MATLAB-implementation is about 20 times slower for FPM and about 35 times slower for PGM. However, these results also depend on the hardware. At LECAD we make use of 24 cores of Intel Xeon E5-2680V3 processors with a clock rate of 2,5 GHz.

Of course, it would be interesting to process the extracted information further to the point of multilayered network matrices (see Figure 10).

Different methods with more potential

In the future, it would be interesting to work on even more approaches to solve the optimisation problem that occurs in . There are methods that are more eligible to be parallelised. Here, e.g., we did not see the potential of using more than one node. So, it could be possible to implement hybrid implementations for a few methods.

More constraints

Also, one could add more constraints to the optimisation problem that occurs in . We actually only took non-negativity constraints for G into account but in practice it is also often important to force non-negativity for the matrices S_i, where the network related information is saved.

More characteristics

In further investigations one could also consider orthogonality constraints on the matrix G.

Starting values

Since we have already noticed that the behaviour of convergency and hence, also the results, depend on the chosen starting values for the matrix G it makes sense to improve the choice of starting points. Until now, we fill the starting point matrix G purely random as we make use of random number generators following an uniform distribution. A more advanced approach could, for instance, be to generate the starting point matrix based on the solution of the singular value decomposition of the average network matrix.

References

[20] Figueredo, A. J. and Wolf, P. S. A. (2009).

[1] Fei Wang, Tao Li, and Changshui Zhang. Semi-supervised clustering via matrix factorization. In SDM, pages 1–12. SIAM,2008.

[2] Dimitri P. Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, MA, 1999.

[3] Chih-Jen Lin. Projected gradient methods for non-negative matrix factorization. Neural computation, 19(10):2756–2779, 2007.

[4] Janez Povh. Unpublished paper on nonnegative matrix factorization. 2017.

PRACE SoHPCProject Title

A parallel algorithm for non-negative matrix trifactorization

PRACE SoHPCSite

University of Ljubljana, Faculty of mechanical engineering, Laboratory LECAD, Slovenia

Jan Packhäuser

PRACE SoHPCAuthors

Jan Packhäuser, University of Ulm, Germany

PRACE SoHPCMentor

Janez Povh, LECAD, Faculty of mechanical engineering, University of Ljubljana, Slovenia

PRACE SoHPCContact

Dr. Leon Kos, University of Ljubljana
E-mail: leon.kos@lecad.fs.uni-lj.si

PRACE SoHPCSoftware applied

C++, MATLAB, OpenBLAS, OpenMP, Eigen

PRACE SoHPCMore Information

HPC Prelog

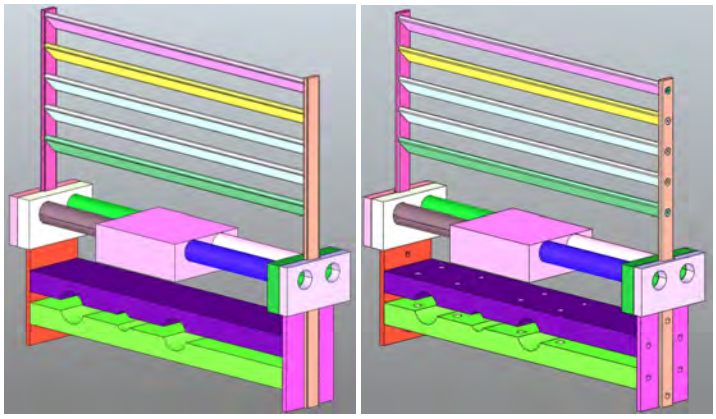
PRACE SoHPCAcknowledgement

I would like to thank PRACE and the organisational team of summer of HPC for this great summer experience. Also, I want to express my gratitude to my mentor Prof. Dr. Janez Povh and the summer of HPC coordinator Dr. Leon Kos for giving me the opportunity to learn and discover.

PRACE SoHPCProject ID

1720

CAD Data Extraction for CFD



Paras Kumar

Geometry cleanup or defeaturing (i.e.,removal of unnecessary feature details like fillets, chamfers, holes etc.) is an inevitable yet quite cumbersome sub-task of the CAD-CFD process chain. This is usually carried out manually and is thus time consuming. The project aims at developing a PythonOCC based utility to automate the process of CAD data extraction.

Studying the flow of fluids can help unravel the mystery behind various physical phenomena. Examples include flow of air, water and blood around an aeroplane, a ship and our blood vessels respectively. On account of their mathematical and computational complexity, problems of fluid mechanics almost invariably require the use of numerical techniques for their solution. The field of study dealing with numerical solutions to fluid flow problems is **C**omputational **F**luid **D**ynamics.

What is CAD?

Computer **A**ided **D**esign is the technical discipline involving the application of computers or supercomputers to the design of products - ranging from your cell phone to the wings of an aircraft. This is much more viable, not only economically but also in terms of time and effort - when compared to the traditional approach of testing with physical prototypes. The CAD model also referred to as CAD Data or simply CAD, contains the geometric details (amongst others) of the product and is created using CAD modelling software such as SOLIDWORKS or OpenCASCADE.

CAD-CFD Process Chain

CAD data serves as starting point for various computer based design validation techniques - known otherwise as "simulation techniques including CFD". The next step is *CAD data extraction*, which involves removal of unwanted features. This prepares the CAD data for further processing. This is followed by mesh generation and setting up the simulation case by specifying the initial and boundary conditions, system properties, solvers and control parameters to be used for result output. All this, is collectively termed as *pre-processing* and transforms the problem into an equation system which is then solved to get the results. The flow diagram in Figure 1 summarises the different steps involved in a CFD analysis.

A **SMALL** Problem

As mentioned above, CAD data supplied by the designer almost invariably contains small details which are unnecessary for simulation. These include:

1. Small parts such as nuts, bolts, screws
2. Intricate feature details such Holes, Fillets, Chamfers

Including this data in numerical simulations could lead to much complex or even incorrect numerical models.

Geometry cleanup or defeaturing, refers to the removal of these unwanted intricate details and is one of the trickiest sub tasks in the process-chain. If handled manually, this could be quite cumbersome and time consuming.

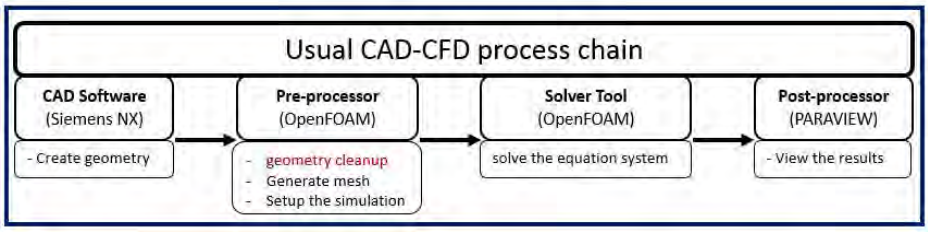


Figure 1: Usual CAD-CFD Process Chain

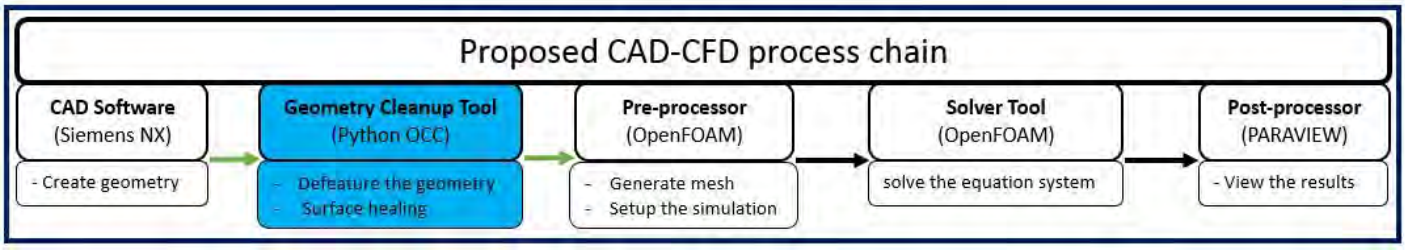


Figure 2: Modified CAD-CFD Process Chain, with Geometry Cleanup Utility included

Proposed Solution

To ease and automate the process of defeaturing, a utility (piece of software) based on the Python programming language has been developed. For dealing with geometry, PythonOCC i.e., a Python based implementation of the OpenCASCADE library has been employed. The insertion of this utility into the CAD-CFD process chain results in the modified process chain as depicted in Figure 2.

The Geometry Cleanup Utility

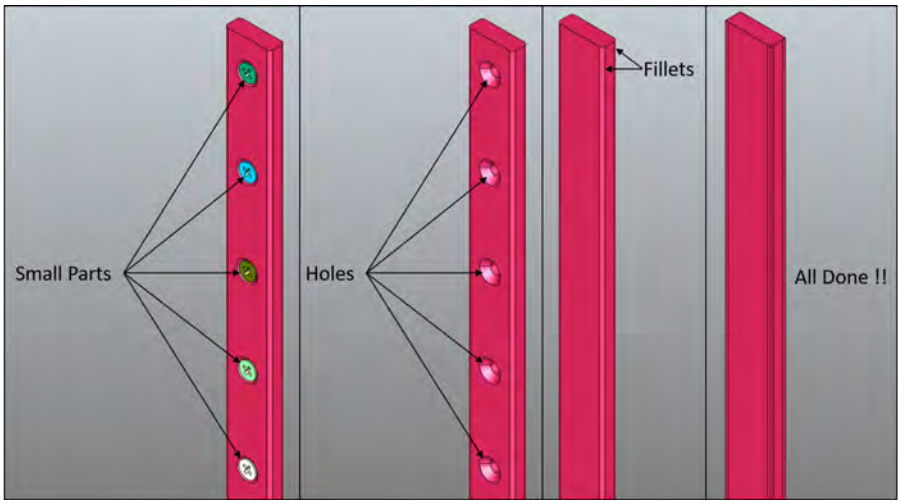


Figure 3: Defeaturing schematic description of a plate as done within the Utility

The utility in its present state provides the user with the following functionalities:

1. Removal of small parts with volume below a specified threshold
2. Removal of holes with radii less than a specified value
3. Removal of fillets (edge fillets only) with radii less than a specified value

The process is explained in a schematic manner for the example of a plate as depicted in Figure 3.

Classification of Fillets

Fillet is a rounding of the sharp edges of a component. For the purpose of defeaturing, a classification based on the rounding done at the end of the edge forming the fillet has been employed.

The classes currently handled by the utility include:

1. Isolated Edge Fillet - IEF
2. Single Connected Edge Fillet - SCEF
3. Double connected Edge Fillet - DCEF

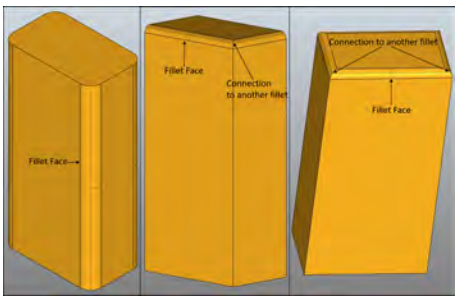


Figure 4: Different types of fillets (IEF, SCEF, DCEF)

Different types of fillets along with the corresponding defeatured models are shown in Figure 4 and Figure 5.

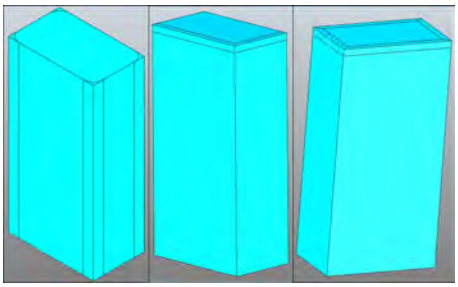


Figure 5: Defeatured model for different types of fillets (IEF, SCEF, DCEF)

References

[1] PythonOCC Documentation https://github.com/tpaviot/pythonocc

[2] Opencascade Reference Manual https://www.opencascade.com/doc/occt-6.9.0/refman/html/index.html

PRACE SoHPC Project Title

CAD Data Extraction for CFD Simulation

PRACE SoHPC Site

University of Ljubljana, Slovenia

PRACE SoHPC Authors

Paras Kumar, [University of Erlangen-Nuremberg,] Germany

PRACE SoHPC Mentor

Dr. Marijo Telenta, LECAD Lab, Univeristy of Ljubljana, Slovenia

Paras Kumar

PRACE SoHPC Contact

Paras Kumar, University of Erlangen-Nuremberg
Phone: +49 176 37797745
E-mail: paras.kumar@fau.de

PRACE SoHPC Software applied

PythonOCC, Pycharm

PRACE SoHPC More Information

www.pythonocc.org, www.jetbrains.com/pycharm/,

PRACE SoHPC Acknowledgement

Heartfelt thanks to Dr. Marijo Telenta and Dr. Leon Kos (site coordinator) for their continuous support and guidance during the project tenure.

PRACE SoHPC Project ID

1721

www.summerofhpc.prace-ri.eu