Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Novel HPC Technologies for Rapid Analysis in Bioinformatics

Tristan Cabel[a], Gabriel Hautreux[a], Eric Boyer[a*], Simon Wong[b], Nicolas Mignerey[c], Xiangwu Lu[d], Paul Walsh[d]

[a]*Centre Informatique National de l'Enseignement Supérieur, 950, rue de Saint Priest, 34097 Montpellier Cedex 5, France.*
[b]*Irish Centre for High-End Computing, 7/F Tower Blg., Trinity Technology & Enterprise Campus, Grand Canal Quay, Dublin 2, Ireland.*
[c]*Grand Equipement National de Calcul Intensif, 12, rue de l'Eglise, 75015 Paris, France.*
[d]*NSilico Lifescience Ltd., Melbourne Building, CIT Campus, Bishopstown, Cork City, Ireland.*

Abstract

NSilico is an Irish based SME that develops software for the life sciences sector, providing bioinformatics and medical informatics systems to a range of clients. One of the major challenges that their users face is the exponential growth of high-throughput genomic sequence data and the associated computational demands to process such data in a fast and efficient manner. Genomic sequences contain gigabytes of nucleotide data that require detailed comparison with similar sequences in order to determine the nature of functional, structural and evolutionary relationships. In this regard NSilico has been working with computational experts from CINES (France) and ICHEC (Ireland) under the PRACE SHAPE programme to address a key problem that is the rapid alignment of short DNA sequences to reference genomes by deploying the Smith-Waterman algorithm on an emerging many-core technology, the Intel Xeon Phi co-processor. This white paper will discuss some of the parallelisation and optimisation strategies adopted to achieve performance improvements of the algorithm keeping in mind both existing and future versions of the hardware. The outcome has been an extremely successful collaboration between PRACE and NSilico, resulting in an implementation of the algorithm that can be readily deployed to realise significant performance gains from the next generation of many-core hardware.

1. Introduction

NSilico is a company based in Ireland and is a developer of integrated molecular diagnostics and sequence data management and analytic tools for the life sciences and healthcare industries. The company's offerings are based upon a unique and unrivalled blend of biological, computing, software development and clinical experience and expertise which enables it to provide its customers with solutions which significantly increase the efficiency and accuracy of their work. Currently, the company has two product offerings: *Simplicity*, a cloud based bioinformatics research pipeline tool; and *iCare*, for cancer care management.

Simplicity is NSilico's lead product and is one of the most comprehensive, easy-to-use, cloud-based software-as –a-service products for the automatic annotation, analysis and visualisation of high-throughput sequencing data. It is scalable and customizable to user needs and allows automated and rapid extraction and reporting of high value information that aids in the discovery of biomarkers/genetic profiles through the creation of publication standard, rich reports. Its usability and power help to dramatically reduce research time cycles. It is also highly secure and compliant with the latest standards required by regulatory bodies for complete auditing and traceability.

A significant challenge that has resulted from high-throughput sequencing technology is the production of vast amounts of genomic sequence data, including ways to store, analyse and interpret such data. Currently the typical amount of data that needs to be cleaned and processed is in the order of approximately 15GB per

* Corresponding author. *E-mail address*: boyer@cines.fr

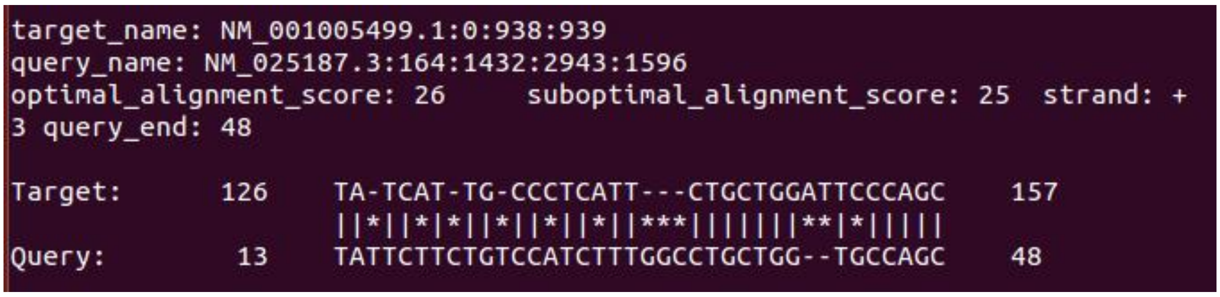
experimental run. Apart from obvious storage challenges the performance of the software algorithms that are used to process large streams of sequencing data can be an important factor in provisioning rapid and cost-effective bioinformatics solutions for customers.

This SHAPE [1] project set out to examine some of the bioinformatics software packages and algorithms that are often used in the analysis of high-throughput sequencing data, and to devise appropriate strategies to parallelise and implement them. On the computational side, one area that NSilico is particularly interested in is the emerging many-core platforms that can potentially deliver large performance gains at relatively low costs for an in-house infrastructure, compared to an equivalent, conventional high performance computing infrastructure.

2. Identification of the relevant bioinformatics software package or algorithm

A number of bioinformatics packages and algorithms - mainly alignment codes such as *BLAST*, *bowtie* and *bwa* were examined for potential optimisation and parallelisation on the Intel Xeon Phi many-core architecture. Due to the relatively short duration of this project and limited human resources, the challenge was to identify a promising code where meaningful optimisation and parallelisation work can be carried out (that has not been done by others already) and studied. For some relatively large codebases (e.g. BLAST, bowtie2), these are already well optimised and fine tuned for modern CPUs; re-factoring such codes or even parts of them for the Intel Xeon Phi will likely require more time and effort than was available. In the end, the team focused on a relatively compact SIMD (Single Instruction, Multiple Data) implementation of the Smith-Waterman (SW) alignment algorithm, or more specifically the striped version of the algorithm as developed by Farrar [1], publicly available as a C/C++ library called the SSW library [3].

The Smith-Waterman algorithm is a well known dynamic programming algorithm in bioinformatics which guarantees to find the optimal alignment between any two biological sequences (provided a suitable scoring scheme). It has been parallelised on a variety of platforms, including multiple CPUs, GPUs and FPGA architectures. In high-throughput sequencing, the SW algorithm itself, or adaptations of it, are often used to align sequencing reads (i.e. the key output from sequencing machines) to reference sequences (e.g. known reference such as a bacterial or a human genome sequence). An example output from implementing the algorithm is shown in Figure 1. Further downstream analyses would then reveal different levels of DNA sequence similarity and/or variation that give rise to biological insight.



```
target_name: NM_001005499.1:0:938:939
query_name: NM_025187.3:164:1432:2943:1596
optimal_alignment_score: 26     suboptimal_alignment_score: 25  strand: +
3 query_end: 48

Target:       126    TA-TCAT-TG-CCCTCATT---CTGCTGGATTCCCAGC    157
                     ||*||*|*||*||*||*||***|||||||**|*|||||
Query:         13    TATTCTTCTGTCCATCTTTGGCCTGCTGG--TGCCAGC    48
```

Figure 1: Example output from the Smith-Waterman sequence alignment algorithm.

The initial examination of the SSW library reveals that parallelisation of the code, with the Intel Xeon Phi co-processor in mind, can be approached in two ways:

- Optimisation at the CPU core level with modern SIMD intrinsics
- OpenMP parallelisation of key loops in the algorithm

3. Compute resources

The team was successful in obtaining an allocation of compute resources on the MareNostrum system at the Barcelona Supercomputing Center, via a Preparatory Access (Type C) application. This includes compute time on the hybrid nodes of MareNostrum, each consisting of:

- 2x Intel Sandy Bridge-EP E5-2670/1600 20M 8-core at 2.6 GHz
- 2x Xeon Phi 5110P
- 8x8GB DDR3-1600 DIMMS (4GB/core)

Some of the work for this project also made use of preliminary access to a pre-production system with Intel Haswell processors.

- 2x Xeon® E5-2697 v3 -2.6 GHz 14 Core (C0 Pre-Production Processors)
- 128GB DDR4 (8x 16GB) at 2133MHz

4. Optimisation at the CPU core level with modern SIMD intrinsics

The initial codebase of the SSW library was written with SSE2 (Streaming SIMD Extensions 2) intrinsics, which are vectorisation routines dating back to 2001 introduced by Intel with the launch of the Pentium 4 CPU architecture. SSE2 enables SIMD integer instructions to be performed on a variety of data types, from 8-bit integers to 64-bit floating point numbers, using 128-bit registers. However, SSE2 has since been superceded by more advanced versions of vectorisation extensions that are supported by more modern hardware. In 2011, the first processors were shipped with support for AVX (Advanced Vector Extensions) intrinsics, where the width of the SIMD register file was increased from 128 bits to 256 bits. However, 8-bit and 16-bit integer SSE instructions did not expand to operate on the wider 256-bit AVX registers until the release of AVX2, which is only supported by some of the more recent CPU architectures from Intel (e.g. Haswell). The next set of SIMD extensions, AVX-512, expands the width of the register file even further to 512 bits.

The current generation of the Intel Xeon Phi co-processor, codenamed Knights Corner, does not support traditional SSE or AVX but a custom set of intrinsics for its vector units with wide, 512-bit registers. The next generation of the co-processor, codenamed Knights Landing will have more 512-bit registers and will support AVX-512 SIMD extensions.

Hence in order keep the SSW code maintainable and readable, all the intrinsic calls were replaced by call to pre-processor keywords. With this method, one can easily choose at compile time to revert to the original code or to compile the library optimised with AVX, AVX2, either for the CPU and/or for the Intel Xeon Phi co-processors.

The main issue we first encountered was that the Sandy Bridge processors we used are compatible with AVX but not AVX2. 8-bit unsigned integer data elements were used but operations on this type of data are only available on AVX2, not on AVX. Therefore, only the operations on 256-bit data elements were optimised with AVX but for operations such as additions, subtractions or element-wise maximum, we revert to SSE2 intrinsics.

In AVX2, only one intrinsic is required: `__m256i _mm256_adds_epu8 (__m256i a, __m256i b)`

The preparation work done with AVX has also allowed us to see how to increase the number of elements in a segment and to identify some pitfalls in AVX2. As an example, at some point we have to shift our vector 1 byte to the left and in SSE2 this is done by the function `__m128i _mm_slli_si128 (__m128i a, int imm)` which does the following operation:

```
dst[127:0] := a[127:0] << (imm*8)
```

In AVX2, there is a similar operation `__m256i _mm256_slli_si256 (__m256i a, const int imm)` but when we look into what it's really doing, we can observe that the operation is:

```
dst[127:0] := a[127:0] << (imm*8)
dst[255:128] := a[255:128] << (imm*8)
```

As we can see, it is slightly different than the behaviour expected because the bits 127 and below will not be reported in the upper part of the vector. Hence, we used instead the following function of 4 instructions:

```
//shift by one byte to the left
//4 instructions

inline __m256i bslli_si256(__m256i arg)
{
    __m256i mask = _mm256_permute2x128_si256(arg, arg, _MM_SHUFFLE(0,0,3,0) );
    return _mm256_alignr_epi8(arg, mask, 16-1);
}
```

As part of this work, we obtained preliminary access from Intel to a Haswell processor in order to carry out some tests. The system contains dual E5-2697 v3 (C0 – Pre-Production Processors) with 128GB DDR4 2133MHz Memory. Our tests shows that the computation part was 22% faster using AVX2 compared to SSE2.

Adaptation for the Intel Xeon Phi co-processor

Intel Xeon Phi co-processors work on segments of 512 bits. However, it faces similar problem as with AVX previously: operations are only supported on 32-bit or 64-bit elements. The initial thought would be to concatenate four 8-bit elements with some bit-shifting but that leads to problems for subtractions and element-wise maximum. Hence our approach was to use 32-bit elements instead of 8-bit elements for the "Knights Corner" generation of co-processor.

It is important to note that while SSE2 intrinsics operate on 16 x 8-bit elements on 128-bit registers, even with the wider 512-bit registers on the Knights Corner we were still using intrinsics that operate on 16 x 32-bit elements, i.e. both cases involved operations on 16 elements at a time. Furthermore extra code and instructions (to verify positive values) were required on the Knights Corner as it does not support unsigned 8-bit integers but signed 32-bit elements. Therefore we do not expect any performance gains due to intrinsics with the current Knights Corner generation of the Xeon Phi.

For the next generation "Knights Landing" Xeon Phi co-processor, operations on AVX2 will be supported. Consequently, we may take the same approach done with AVX which is to use AVX-512 vectors to call AVX2 routines on 256 bit elements. We have already shown promising results on CPUs (see above) that using AVX2 instead of SSE2 gave rise to significant performance gains.

5. OpenMP parallelisation on the key loops of the algorithm

First approach: parallelise reference sequence matching

The first approach was to parallelise the outer loop of the code, which scans the reference sequence and compares it with the query. The idea is simple: if we parallelise this loop, it just means that the reference sequence will be split up for comparison with query sequences.

The only critical point is that at the edges of each slice, padding needs to be added to ensure no matching hits are missed. This padding is proportional to the size of the query and in order to achieve good performance with the parallelised version, the assumption was that the query sequences were much shorter than the reference.

The preliminary results using the example dataset provided with the code (reference: 10M.fa; query: 54mer_hap1_1.100.fa) were quite convincing since the scalability of the code is quite good up to the 8 cores of each Sandy Bridge CPU. This was expected since each node of the BSC MareNostrum system has dual sockets with two 8-core Sandy bridge processors without hyperthreading. We also used the environment variable KMP_AFFINITY=compact to place threads on neighbouring cores. Hence, as observed in Figure 2, best performances are seen when using 8 cores (a full socket); scaling beyond additional cores incurred significant performance penalties as it involves time-consuming data transfer between sockets.

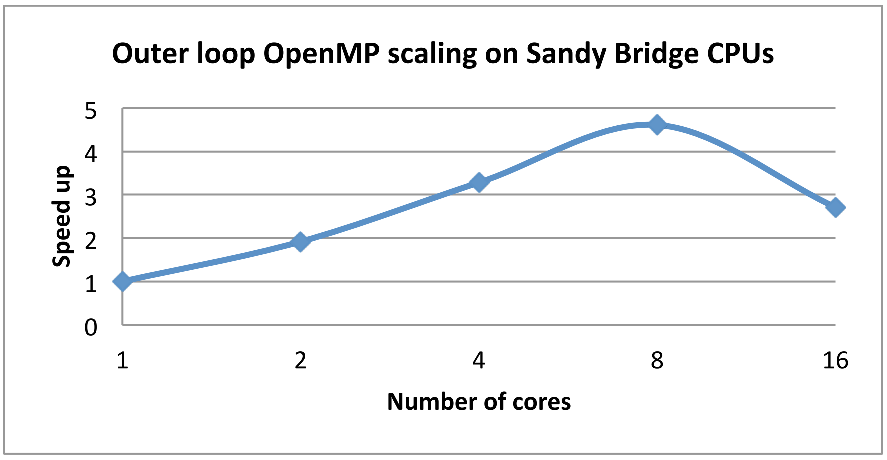


Figure 2: Scalability of the code where the outer loop is parallelised with OpenMP, i.e. split up reference sequence

Second approach: couples version

NSilico subsequently provided a real test dataset that is representative of what is being used for their every-day business, referred to from here as the "human ref" dataset. This dataset is composed of a reference database of thousands fragments of variable size. An important aspect of this reference database is that the fragments are not as long, and in some cases one cannot make the assumption that the query sequences are shorter than the reference as before.

As in the first approach, the level of overhead increases as more threads are used to divide the reference sequence. This was not a bottleneck for the Xeon Phi before but it became one because multiple fragments of variable size are being used. Hence a new approach was taken to continue the parallelisation work for the Intel Xeon Phi.

This new approach we used is very simple: instead of parallelising the algorithm itself, we decided to parallelise individual computations on the query/reference fragment couples, thereby increasing the intensity of the computation. Moreover, since the amount of memory required was quite small, we decided to pre-load all the required data in memory before the computations.

Human ref results on the Xeon Phi

Thanks to the 240 nominal cores on Xeon Phi, we expect to compute up to 240 couples at a time. This method is very efficient when you don't have information about the size of the reference fragments and when a lot of comparisons have to be performed.

Shown below in Figure X are the human ref performance results suing the native mode of the Xeon Phi. A speed-up of 2.12 (relative to the 60-thread version) was achieved using 240 threads on the co-processor. The speed-up is convenient because most of the code does not achieve any speed-up if more than 120 threads are used (i.e. 2 threads per core using 60 cores). The speed-up between 1 to 60 threads (i.e. one thread per core) is not presented but it is almost linear; one can see that the speed-up from 30 to 60 threads is close to two.

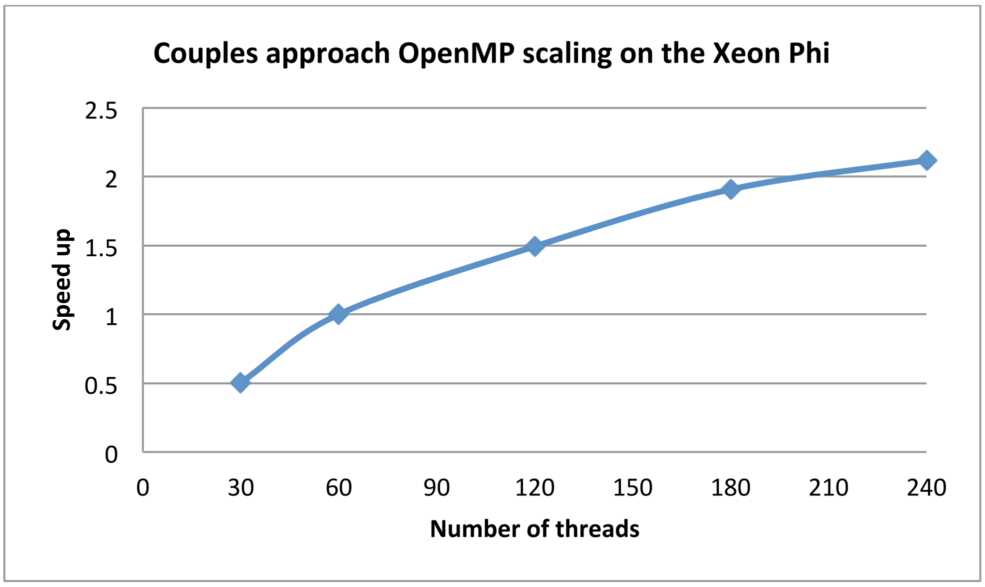


Figure 3: Scalability of the couples approach OpenMP parallelisation on the Intel Xeon Phi

Human ref results on Sandy Bridge CPUs

Shown below in Figure X are human ref performance results on Sandy Bridge CPUs. Scalability is good up to 8 threads after which the performance drops owing to expensive communication costs between CPU sockets.

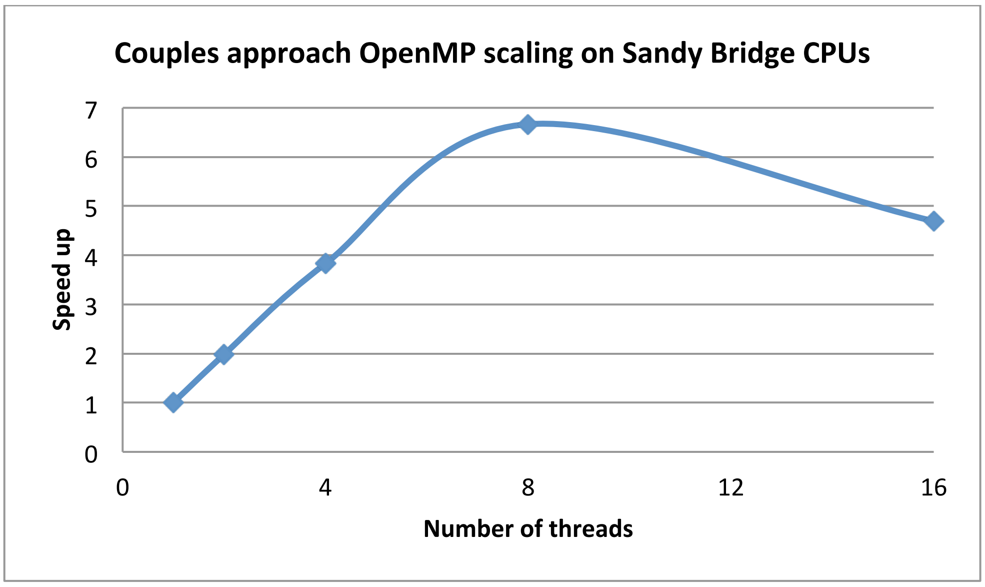


Figure 4: Scalability of the couples approach OpenMP parallelisation on Sandy Bridge CPUs

However, performance of the code on Sandy Bridge is five times faster than that on the Intel Xeon Phi. This corresponds to initial expectations at the beginning of the project: we do not expect to see real performance gains right now but our code is written so that we can readily benefit from next generation hardware. Furthermore, prior to our optimisation work, the Xeon Phi code was more than 25 times slower than the Sandy Bridge.

We expect to see real rewards in the next generation of the Xeon Phi co-processor. While the current generation hardware is a co-processor with 60 cores (each one of those based on the original P54C Pentium core), the next generation "Knights Landing" hardware will be a major revision with up to 72 out-of-order Silvermont (Atom) cores. It means that instead of relying on 32-bit elements, AVX2 will be supported that enables operations on 8-

bit or 16-bit elements. Moreover, instead of being a co-processor, Knights Landing will be available as standalone CPUs that obviates expensive data transfers on the PCIe bus.

6. Conclusions

In conclusion, the SHAPE project has been a very successful collaboration between NSilico and the PRACE partners involved. NSilico has benefited from domain expertise from PRACE in first identifying a bioinformatics codebase with real potential to be deployed on cutting-edge many-core hardware. It has also since gained invaluable insights into the optimisation and parallelisation work involved in porting the code to the Intel Xeon Phi. Next steps are already being discussed on testing and deployment of the code with the release of the next generation "Knights Landing" hardware, as well as potential incorporation into NSilico's in-house bioinformatics pipelines.

We have adopted two approaches to optimise and parallelise the SSW library, first using modern SIMD intrinsics and the second using OpenMP. Significant lessons have been learnt in implementing the right intrinsics for the target architecture, e.g. this can be a time-consuming development process. On top of that the OpenMP parallelisation work has led to a code that shows good parallel performance results for the CPU and promising results for Xeon Phi hardware. While the resulting SSW library achieves expectedly limited performance gains on the current generation of the Xeon Phi, it has been re-factored in a way to readily take advantage of the next generation of hardware such as Knights Landing.

References

[1] The PRACE SHAPE programme web site:
http://www.prace-ri.eu/shape

[2] Farrar, M. (2007). Striped Smith-Waterman speeds database searches six times over other SIMD implementations. Bioinformatics 23:156-161.

[3] Zhao M., Lee W.-P., Garrison E.P. and Marth G.T. (2013). SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications. PlosONE 8:e82138.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-312763. This project was one of 10 projects in the SHAPE (SME HPC Adoption Programme in Europe) Pilot in 2014 [1]. Paul Walsh is a research fellow on FP7 Marie Curie Actions IAPP Project ClouDx-I under grant agreement no. 324365.