Impact of Concurrency on the Performance of a Whole Exome Sequencing Pipeline

Daniele Dall'Olio
Universita di Bologna
Nico Curti
Universita di Bologna
Eugenio Fonzi
Istituto Scientifico Romagnolo per lo Studio e la Cura dei Tumori Srl
Claudia Sala (claudia.sala3@unibo.it)
Universita di Bologna https://orcid.org/0000-0002-4889-1047
Daniel Remondini
Universita di Bologna
Gastone C Castellani
Universita di Bologna
Enrico Giampieri
Universita di Bologna

Research article

Keywords: concurrency, parallel computing, bioinformatics, analysis pipeline, scalability, efficiency, work flow management system, snakemake

Posted Date: March 10th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-16453/v1

Version of Record: A version of this preprint was published on February 9th, 2021. See the published version at https://doi.org/10.1186/s12859-020-03780-3.

RESEARCH

Impact of Concurrency on the Performance of a Whole Exome Sequencing Pipeline

Daniele Dall'Olio[1†], Nico Curti[2†], Eugenio Fonzi[3], Claudia Sala[1*], Daniel Remondini[1], Gastone Castellani[2^] and Enrico Giampieri[2^]

*Correspondence: claudia.sala3@unibo.it
[1]Department of Physics and Astronomy, University of Bologna, 40127 Bologna (BO), Italy
Full list of author information is available at the end of the article
†Equal contributor
^Equal contributor

Abstract

Background: Current high-throughput technologies – i.e. Whole Genome Sequencing (WGS), RNA-Seq, ChIP-Seq, etc. - generate huge amounts of data and their usage gets more widespread with each passing year. Complex analysis pipelines involving several computationally-intensive steps have to be applied on an increasing number of samples. Workflow Management Systems (WMSs) allow parallelization and a more efficient usage of computational power. Nevertheless, this mostly happens by assigning the available cores to a single or few samples at a time. We refer to this approach as *naive parallel* strategy (NPS). Here, we discuss an alternative approach, which we refer to as *concurrent* execution strategy (CES), which equally distributes the available processors across every sample.

Results: Theoretically, we show that the CES results, under loose conditions, in a substantial speedup, with an ideal gain range spanning from 1 to the number of samples. Practically, we tested both strategies on a Whole Exome Sequencing (WES) pipeline applied to three publicly available matched tumour-normal sample pairs of gastrointestinal stromal tumour (GIST). The CES achieved speedups in latency up to 2-2.4.

Conclusions: Our results hint that if resources distribution is further tailored to fit specific situations, an even greater gain in performance could be achieved. For this to be feasible, a benchmarking of the tools included in the pipeline would be necessary. It is our opinion these benchmarks should be consistently performed by the tools' developers. Finally, these results suggest that concurrent strategies might also lead to energy and cost savings by making feasible the usage of low power machine clusters.

Keywords: concurrency; parallel computing; bioinformatics; analysis pipeline; scalability; efficiency; workflow management system; snakemake

Background

In this paper we examine execution strategies which a standard bioinformatics pipeline can be run with, in order to determine a time-effective strategy that can be generally suggested to the bioinformatics community. We ideally aim at a processors usage setting that achieves the best speedup of the total execution time when multiple samples should be examined at the same time. Our main focus is optimal computational power (number of processors) allocation among the steps of the pipeline, whereas we consider all others machine resources, such as memory usage and disk I/O operations, as boundaries that limits the highest possible performance that could be obtained.

A Next-Generation Sequencing (NGS) data analysis pipeline involves several computationally-intensive tasks (like read trimming, alignment, BAM post-processing, etc...). Some of these tasks are non-scalable (NS), i.e. take a fixed amount of time when given multiple resources, and others are parallelly computable (PaCo), i.e. improve when given multiple resources. Each task is typically handled by a single specific tool, which might be optimized for parallel computing. In addition, these pipelines are usually applied to several samples at once, and some steps need to be repeated for each sample while some are shared among them.

At first, pipelines were developed for a command line interpreter execution, i.e. UNIX shell. They were typically built as a sequential series of commands that needed to be reproduced for every sample. Nowadays many bioinformaticians [1, 2, 3] rely on workflow management systems (WMSs) [4, 5, 6, 7] to improve execution efficiency and scalability. WMS usually accept different programming languages within the same pipeline and can be run by a single command line. Further, WMSs are designed to independently organize all pipeline tasks before the actual execution. They trace a graph of tasks by combining several factors, including expected results, existing data and dependencies. This feature enhances scalability and it is especially useful for mutually independent tasks, which can be run simultaneously without any explicit competence of parallel computing coding. In addition, many of these systems have options to indirectly manage machine resources.

Although WMSs allow to easily manage a pipeline for multiple samples, usually each sample is processed one at a time sequentially, even if in a highly optimized way. Indeed, the most basic way to run a pipeline is to implement it with these systems and launch it on a high-performing machine by setting the number of exploitable processors and letting the WMS assigns the resources to each task. Often the default setting of the WMS is to allocate all the available processors for each step of the pipeline, unless otherwise specified. We define this parallel single sample strategy as naive parallel strategy (NPS). It is well-recognized that all PaCo tasks have a sub-linear increase of speedup (where linear is the most optimistic case, called "embarrassingly parallel"), which means they can not achieve unlimited boosting in their performances. This leads to a waste of resources since a PaCo task occupies a number of processors that it can not entirely exploit.

We observe that both the old sequential and the NPS follow a single-sample focus, while there is a compelling need to efficiently analyse large datasets, where the single-sample focus might lead to inefficiencies due to the PaCo sub-optimal parallelization. NPS, thanks to WMSs, can boost analyses in terms of automatic replicability and overall management simplification, but they have several limitations: they are not capable of predicting the scalability of PaCo tasks and they do not estimate either time or resources needed by any tasks.

With this paper, we question the usage of NPS and suggest a new execution strategy that we refer to as concurrent execution strategy (CES). To optimally use all processors, CES divides them equally among all samples' PaCo tasks. In doing so, multiple samples' tasks can concur simultaneously and, though individually running slower, they can be completed at the same time all together, resulting in a lower total execution time. In particular, CES is based on the idea of equally splitting the number of processors over the number of samples. This strategy is not guaranteed to be the most efficient one in all occasions, but it is the simplest representation of a concurrent organization.

To empirically assess the efficacy of the CES compared with NPS we tested these two strategies on a WES pipeline that we have developed in our lab.

Results

Application of Amdahl's law to identical processes

We represent a general pipeline execution in a mathematical framework. There are two levels that are worth to examine: the single task level and the pipeline level. These levels are similar and closely related, since the latter is the combination of all tasks. For the sake of simplicity, we decide to describe the mathematical framework of the former level.

The single task level describes the behaviour of those pipeline's tasks that are PaCo. We assume these kind of tasks to be always composed by a NS component and an embarrassingly parallel (EP) component (Amdahl's law hypothesis). The former does not change its execution time regardless of the number of processors allocated to the task. On the contrary, the latter will have a speedup in the execution time linear to the number of allocated processors. Then, we define the total time T needed by a single-processor to process a single sample as:

$$T = F + P \ , \tag{1}$$

where F and P represent execution times respectively for the NS component and EP component. We can also write the former equation, defining $\gamma = \frac{P}{T}$, as:

$$T = (1-\gamma)T + \gamma T \ , \tag{2}$$

where $F = (1-\gamma)T$ and $P = \gamma T$. The value of γ ranges from 0 to 1 and it stands for proportion of time of the EP task.

Now, Amdahl's law [8] states that potential speedup in latency using K processors at fixed workload can be formulated as:

$$s(K, \gamma) = \frac{1}{1 - \gamma + \frac{\gamma}{K}} \ . \tag{3}$$

That is, a single-sample execution can be sped up with a factor equals to $s(K;\gamma)$ by supplying the process with K processors.

Now, we analyse the performances of both the NPS and CES when we need to run a PaCo task for N samples with K available processors. We assume for simplicity K is a N multiple. It is obvious to conclude that both strategies are better than a sequential execution.

With simple considerations we can describe execution time for both strategies respectively with

$$T_{NPS} \approx N\left(F + \frac{P}{K}\right) \tag{4}$$

and

$$T_{CES} \approx \left(F + \frac{P}{\left(\frac{K}{N}\right)} \right) \; . \tag{5}$$

Using NPS the total time T_{NPS} is composed of the sequential execution of the N samples each one at their best possible speed. Using CES all the processes of each sample are executed at the same time and thus the total time T_{CES} would be the maximum execution time of each individual sample. All samples assumed equal, this becomes the execution time of a single sample using the allocated number $\frac{K}{N}$ of processors.

The same equations can be expressed in terms of γ as:

$$\begin{aligned} T_{NPS} &\approx \frac{N\,T}{K} \left[\gamma\,(1-K) + K\right] \\ T_{CES} &\approx \frac{T}{K} \left[\gamma\,(N-K) + K\right] \; . \end{aligned} \tag{6}$$

In analogy with Amdahl's law, we can derive the speedup between the two strategies:

$$s(K,\gamma) = \frac{N\left[\gamma\,(1-K) + K\right]}{\gamma\,(N-K) + K} \; . \tag{7}$$

With $N > 1$, speedup ranges from 1 ($\gamma = 1$) to N ($\gamma = 0$) with any K processors. Consequently, this result suggests that at the single task level the CES is always more efficient than the NPS. This result is based on the assumption that the EP component has a linear and unbounded potential speedup. This assumption is not always true and many bioinformatics tools reach a plateau after a linear-wise increase, before one would expect from Amdahl's law. Therefore the effective speedup of CES over NPS can in general be even higher than the expected one.

Furthermore we can assume that a bioinformatics pipeline can be represented by Equation 1 as well, since it is a combination of several NS and PaCo components. Therefore, we can extend the previous result and claim that the CES does perform faster than the NPS.

Performance comparison on real data

We ran our pipeline on WES data of three public human paired-matched samples, one blood sample and one matching tumour. We compared the execution time and memory occupation running a variants calling pipeline on the subjects either with the NPS and with the CES.

We compare our results in terms of speedup with those expected by Equation 7. In Figure 1 we represent several theoretical speedup behaviours given different γ values as continuous lines. Since we can estimate the improvable proportion of sequential execution time of our pipeline ($\gamma_{expected}$), Figure 1 also shows the expected speedup for this value (as dashdotted lines), which is about 0.689.

Comparison between the NPS and the CES can be aided by the analysis of their tasks and processors subdivision. Figure 2 shows both subdivisions along time for a

3-samples case that has been run with 12 processors. Figure 3 shows the corresponding memory occupation. Figures for other cases (2-samples and different number of processors) are reported in Additional files 3-6.

Many pipeline's tasks are PaCo and Figure 4 shows the speedup behaviour of some of these tasks. Only curves that show a significant increase in speedup are reported.

Discussion

The results reported above clearly show the CES to be faster than the NPS. Indeed, Figure 1 outlines that our actual speedups (represented as dashed lines) have values greater than 1 regardless of the number of processors and the number of samples.

These results are consistent with Equation 7, which expected the CES to be the most time-effective approach under proper conditions. Nevertheless, some insights need to be highlighted, since stepping from a single task level to the pipeline level is not completely straightforward.

First, Equations 4 and 5 still consider an optimistic linear speedup for all PaCo steps. Since bioinformatics tools are known to follow a speedup lower than the one expected from Amdahl's law (see Figure 4) with respect to supplied processors, the CES over NPS speedup could be expected to reach greater values.

Second, both equations are obtained assuming the NS component to be either a single step or a series of sequential steps. None of these assumptions is true at pipeline level because all samples' NS components are mutually independent. This fact implies that samples' NS components can generally be run as soon as necessary resources are available, resulting in a machine usage optimization and eventually in an execution time reduction for both strategies.

Third, we designed Equations 4 and 5 that are true when all N samples share the same execution time T, which is a very rough approximation since this quantity depends necessarily on the input data size. If T across samples is roughly uniform then the previous approximation is reliable. Otherwise, speedup in latency decreases progressively compared to the behaviour expected by Equation 7. The worst case occurs when there is a single sample in cohort whose execution time is much greater than the others.

Hence, the strictest condition requires all samples to be roughly of the same size, but this is typically true for samples acquired with the same experimental design. Nonetheless, these curves do not accurately fit the theoretical curves that are expected for our $\gamma_{expected}$=0.689 (see Figure 1).

In general, we notice that our observed speedup behaviours increase faster than expected by Equation 7. This can be explained by the sub-Amdahl speedup increase in the bioinformatics PaCo tasks. The sub-Amdahl speedup increase, showed in Figure 4, expresses the loss of performance along the number of supplied processors. Therefore, it is more efficient to run samples with processors evenly distributed, than to run them with all processors assigned to each sample one after the other. This aspect becomes progressively more evident when many processors are involved in the execution: we observe from all speedups shown in Figure 4 that there are almost no differences in speedup by using 8 and 16 processors, which suggests a waste of resources.

Another factor influences the lack of fitness between actual and theoretical increase in speedup. As stated above, our pipeline NS components are not a set of sequential steps, which means they are actually run as soon as the necessary resources are available. The NPS is then penalized by Equation 4 because it actually does not need to repeat all NS tasks N times and the fixed time component (F) is less than what assumed. Nonetheless the NPS is affected by a slowdown compared to the CES, since the PaCo tasks allocate all processors ending up being executed sequentially. In contrast, the CES is slightly penalized by Equation 5, given that the NS component is actually less than what assumed.

Therefore, we expect a decrease in speedup when the organization of the NS components is similar between the NPS and the CES (to see an example of this effect observe the first 2 hours of execution with both strategies in Figure 2). This happens when the NPS slowdown compared with CES does not hinder significantly the whole execution. Otherwise, we expect an increase in speedup, since the CES always manages to optimize available resources, whereas the NPS tends to focus a part of them only on few tasks. These expectations are confirmed by the results shown in Figure 1, where the speedup is lower than expected when using few processors (4 and 6 respectively with 2 and 3 samples) and it increases when the number of processors is increased, approaching the expected values.

The fact that with both cases (2 and 3 samples) we observe speedups that approach the expected ones suggests that our approximation of $\gamma_{expected}$ is fairly accurate. At the same time, we also observe two well different behaviours respectively for the 2-samples case and the 3-samples one. In the former, the observed speedups are lower than expected but they increase until exceeding it, whereas in the latter the observed speedup approaches the expected one but it is always below it. This difference can be explained by looking at the memory usage of both cases (see Figure 3). In the 3-samples case we notice a memory overloading problem in a middle step of the pipeline. This causes a decrease in resources optimization, since we can not overcome machine's boundaries, as can be seen in Figure 3 (between 4th and 6th hours of execution). We do not have the same limit with the 2-samples case, where we run our pipeline as if the available memory is unbounded. We can then assume that the actual speedup in the 3-samples case would exceed the theoretical one as the 2-samples case speedups do, but this does not happen due to technical limitations.

It is noteworthy that alongside a reduction in latency the CES also optimizes memory usage. As can be seen in Figure 3 the NPS underutilizes the available memory, while the CES has less unused resources throughout its execution.

Conclusions

Nowadays, many researchers working in the bioinformatics field design and implement pipelines for the processing and analyses of biological data. Despite the large number of these pipelines, we could not find in literature reliable guidelines for time-efficient executions. In this paper we highlight three strategies which a bioinformatics pipeline can be run with: *sequential* strategy, NPS and CES. Our objective was to determine the best strategy among these three approaches in a general case of N samples and at most K available processors. As explained above, each strategy consists respectively in:

– running one sample at the time by assigning K processors to every sample (*sequential* strategy), this is the standard approach for shell executions;
– running all samples simultaneously by allowing each sample to potentially allocate all K processors (NPS), this is the standard approach for WMSs;
– running all samples simultaneously by equally splitting K processors across samples (CES).

We expressed mathematically that for equal-size samples the CES is always the most efficient strategy (when there are no other technical limitations) and we did measure it with actual executions on a WES pipeline developed in our lab.

Although we examined a simple case of CES, we obtained performance gains in terms of speedup up to 2-2.4 with respect to the NPS. This performance gain is achieved by a strategy based on the idea of limiting the individual resources assigned to each PaCo task and to let them concur. This resources fragmentation is expected to be faster than the NPS even with the best conditions for the latter. An additional advantage of the CES is that, by limiting the processors of the PaCo tasks, one is likely to fall into the speedup linear increase interval of such tasks. In doing so, concurrent executions limit the waste of resources by uniformly distributing many tasks over the available resources.

With this paper we described an alternative approach to the NPS that might be followed to boost performances. We noticed that more complex *concurrent* schemes may yield faster executions; though this would require to measure all computable task behaviours in order to find an optimal distribution. Since most of such tasks use tools developed by specialized companies, these may aid bioinformaticians by providing documentation about speedups as a function of number of processors and input data size, alongside memory usage and disk access.

These findings have implications for the feasibility of low-power clusters to be used in bioinformatics analysis, as the division of resources has not to necessarily occur on the same machine but on multiple ones. Since CES limits each task to relatively few resources, one can potentially distribute all tasks across multiple machines with lower requirements in terms of resources, such as low-power machines. We did not thoroughly analyse this possibility, but in some preliminary analysis we observed no significant difference to occur between using a cluster of low-power machines and a single high performing machine server [9]. The main benefits of this alternative are the noteworthy saving on electrical costs and the chance to acquire a greater amount of resources with multiple low power machines for the same cost of a single traditional server. The feasibility of running bioinformatics pipelines on low-power machine clusters with CES is yet to be fully explored, but might potentially offer the chance to time-effective executions alongside relevant savings.

Methods

WES Pipeline

We decided to exploit a WMS called Snakemake [4], which adapts the GNU Make concept reimplemented leveraging Python language. We chose this system due to its ability of handling intermediate files, its rich options set and its user friendliness. Furthermore, Snakemake is able to use the Conda environment management system [10] both for creation and activation of several environments at runtime.

Our pipeline follows the typical steps for somatic variants calling from WES data and it has been validated with specific version of tools, most of which are provided automatically by Conda (see Supplementary Table 1, Additional file 2). After preliminary trimming (Adapter-Removal) [11, 12], reads are mapped to the selected reference human genome with BWA MEM [13]; resulting BAM are post-processed with tools from SAMtools, Picard [14, 15] and GATK (sorting, indexing, marking of duplicates, indel realignment, base quality score recalibration) [16]. Indels and SNVs are then independently called with MuTect [17] and VarScan2 [18], annotated with ANNOVAR [19] and filtered with in-house scripts. Additionally, this pipeline is also able to call mitochondrial variants: trimmed reads are independently aligned to a reference exome and those that aligned off-target are collected (Picard) and re-aligned to the mitochondrial genome. From this point, BAM post-processing, variant calling, annotation and filtering are executed as for the nuclear genome variants. Further, our pipeline benchmarks everyone of its steps thanks to Snakemake, which relies on *psutil* to retrieve information about each task, such as execution time and memory usage.

Input data

WES FASTQ files were downloaded from NCBI's Sequence Read Archive (SRA) [20]. They derive from three pairs of matched tumour-normal GIST samples (SRR1299130 - SRR1299131, SRR1299134 - SRR1299135, SRR1299140 - SRR1299141), whose genomic DNA was enriched by exome capture with SureSelect Human All Exon 50Mb kit (Agilent Technologies) and sequenced on the Illumina HiSeq 2000 platform in 100-bp paired-end reads. Digital size ranges from 4.1GB to 5.7GB.

Tested executions

We tested the following processors configurations by using Snakemake's *cores* option. At first we sequentially run our pipeline on all samples by first supplying 2, then 4 and 8 processors.

Afterwards we organized our three samples in the three possible pairs and we independently run each of them using both the NPS and the CES. We tested three processors settings: 4, 8 and 16 processors. Both strategies exploited the same number of available processors but, as previously explained, the latter strategy split this number equally over samples, whereas the former did not.

Lastly we compared the two strategies by running our pipeline on all three samples within the same execution. We still set up three processors configurations (6, 12, 24) and we independently used both strategies. We decided to complete the three pairs and triplet runs with several processors settings in order to accurately explore the mathematical domain of Equation 7.

Machine server technical specifications

We performed all our computations on a standard high-performance machine server equipped with two $Xeon\ E5-2620v4$ CPUs, $2\,TB$ of storage (HDD) and $128\,GB$ of memory (RAM). In detail, the mounted $Xeon\ E5-2620v4$ CPU consists of a Broadwell-EP microarchitecture with $2.10(3.00)\,GHz$ frequency, 8 cores and $20\,MB$ of cache.

Declarations

Availability of data and materials
The WES FASTQ files supporting the conclusions of this article are available in the National Center for Biotechnology Information (NCBI) repository [20].

Competing interests
The authors declare that they have no competing interests.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Author's contributions
DD and NC performed the work and wrote the manuscript; EG, GC and CS ideated the project; EF developed the pipeline; EG, EF, GC, DR and CS helped with the analysis; EG, GC, CS and EF helped in writing the paper.

Funding
The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme "HARMONY" Grant n. 116026, "VEO" Grant n. 874735, and from ETN-ITN "ImforFuture" Grant n. 721815.

Acknowledgements
We would like to thank the "EDIMES" lab (University of Bologna), the INFN Gruppo V "AIM - Artificial Intelligence in Medicine" and the INFN "TTLab - Laboratorio di Trasferimento Tecnologico dell'INFN in Emilia Romagna" for their scientific support.

Author details
[1]Department of Physics and Astronomy, University of Bologna, 40127 Bologna (BO), Italy. [2]Department of Experimental, Diagnostic and Specialty Medicine, University of Bologna, 40138 Bologna (BO), Italy. [3]Istituto Scientifico Romagnolo per lo Studio e la Cura dei Tumori (IRST) IRCCS, 47014 Meldola, Italy.

References

1. Schmied, C., *et al.*: An automated workflow for parallel processing of large multiview SPIM recordings. Bioinformatics (2016). doi:10.1093/bioinformatics/btv706
2. Piro, V.C., *et al.*: MetaMeta: integrating metagenome analysis tools to improve taxonomic profiling. Microbiome (2017). doi:10.1186/s40168-017-0318-y
3. Cornwell, M.I., *et al.*: VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. BMC Bioinformatics (2018). doi:10.1186/s12859-018-2139-9
4. Köster, J., Rahmann, S.: Snakemake-a scalable bioinformatics workflow engine. Bioinformatics (2012). doi:10.1093/bioinformatics/bts480
5. Jafar Taghiyar, M., *et al.*: Kronos: A workflow assembler for genome analytics and informatics. GigaScience (2017). doi:10.1093/gigascience/gix042
6. Kluge, M., *et al.*: Watchdog - a workflow management system for the distributed analysis of large-scale experimental data. BMC Bioinformatics (2018). doi:10.1186/s12859-018-2107-4
7. Kotliar, M., *et al.*: CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. GigaScience (2019). doi:10.1093/gigascience/giz084
8. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS Conference Proceedings - 1967 Spring Joint Computer Conference, AFIPS 1967 (1967). doi:10.1145/1465482.1465560
9. Curti, N., *et al.*: Cross-Environment Comparison of a Bioinformatics Pipeline: Perspectives for Hybrid Computations. In: Euro-Par 2018: Parallel Processing Workshops, pp. 638–649. Springer, Cham (2019). doi:10.1007/978-3-030-10549-5
10. Anaconda Software Distribution. https://anaconda.com/
11. FASTQC A Quality Control Tool for High Throughput Sequence Data. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
12. Lindgreen, S.: AdapterRemoval: Easy cleaning of next-generation sequencing reads. BMC Research Notes (2012). doi:10.1186/1756-0500-5-337
13. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics (2009). doi:10.1093/bioinformatics/btp324
14. Li, H., *et al.*: The Sequence Alignment/Map format and SAMtools. Bioinformatics (2009). doi:10.1093/bioinformatics/btp352
15. Picard. http://broadinstitute.github.io/picard/
16. McKenna, Aaron, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, M.D., DePristo, A., M., McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., DePristo, M.A.: The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Research (2010). doi:10.1101/gr.107524.110.20

17. Cibulskis, K., Lawrence, M.S., Carter, S.L., Sivachenko, A., Jaffe, D., Sougnez, C., Gabriel, S., Meyerson, M., Lander, E.S., Getz, G.: Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. Nature Biotechnology (2013). doi:10.1038/nbt.2514
18. Koboldt, D.C., Zhang, Q., Larson, D.E., Shen, D., McLellan, M.D., Lin, L., Miller, C.A., Mardis, E.R., Ding, L., Wilson, R.K.: VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome Research (2012). doi:10.1101/gr.129684.111
19. Wang, K., Li, M., Hakonarson, H.: ANNOVAR: Functional annotation of genetic variants from high-throughput sequencing data. Nucleic Acids Research (2010). doi:10.1093/nar/gkq603
20. Bethesda (MD): National Library of Medicine (US), N.C.f.B.I.: National Center for Biotechnology Information (NCBI)[Internet]. https://www.ncbi.nlm.nih.gov/

Supplementary information

Supplementary Figure 1 (Additional file 1). Comprehensive legend for Figures 2 and 3.
Supplementary Table 1 (Additional file 2). Description of the NGS pipeline used for this paper.
Supplementary Figure 2 (Additional file 3). Representation of all pipelines executions (6, 12 and 24 processors) on all 3 samples based on processors usage along time for both NPS and strategies (Figure 2 as well).
Supplementary Figure 3 (Additional file 4). Representation of all pipelines executions (6, 12 and 24 processors) on all 3 samples based on memory usage along time for both NPS and strategies (Figure 3 as well).
Supplementary Figure 4 (Additional file 5). Representation of all pipelines executions (4, 8 and 16 processors) on 2 samples based on processors usage along time for both NPS and strategies (all possible 2-samples combinations are reported).
Supplementary Figure 5 (Additional file 6). Representation of all pipelines executions (4, 8 and 16 processors) on 2 samples based on memory usage along time for both NPS and strategies (all possible 2-samples combinations are reported).

Figures

Figure 1: **Speedup comparison.**
Expected speedup for CES compared to the NPS for 2 and 3 samples as a function of the available number of processors and PaCo proportion γ. The magenta dashed line represents the measured speedup of the real pipeline tested. The green dashed-dotted line represents the expected performance gain based on the estimated proportion of sequential execution time of our pipeline. In the 2-samples figure there are three observed speedups associated to all the combinations of two samples out of the three available. In the 3-samples figure there is a single observed speedup associated as there is only one possibile triplet out of the three available samples.

Figure 2: **Pipelines execution comparison (processors allocation).**
Representation of a pipeline execution on all 3 samples based on processors usage along time for both NPS and CES. Each rectangle represents a task, its base and height are respectively equal to the allocated processors and to the time elapsed. The hatch type indicates the paired-matched sample that a task works on. The hatch color specifies the data type taken as input by a task. The color identifies the task type. See Supplementary Figure 1, Additional File 1 for comprehensive legend. See figure 3 for its corresponding memory usage.

Figure 3: **Pipelines execution comparison (memory usage).**
Representation of a pipeline execution on all 3 samples based on memory usage along time for both NPS and CES. Each rectangle represents a task, its base and height are respectively equal to the allocated amount of memory and to the time elapsed. The hatch type indicates the paired-matched sample that a task works on. The hatch color specifies the data type taken as input by a task. The color identifies the task type. See Supplementary Figure 1, Additional File 1 for comprehensive legend. See figure 2 for its corresponding processors usage.

Figure 4: **Single tasks' speedup comparison.**
Performance gain of several PaCo tasks with respect to the number of processors exploited and the comparison with the Amdahl's law predictions. The points represent the actual measured speedup. The dashed-lines shows the best Amdahl's fit with the associated γ. On the left plot we report tasks behaving according to Amdahl's law. On the right plot we report tasks with significant departure from Amdahl's law predictions, with a speedup for high number of processors lower than expected. Only PaCo tasks with a significant speedup behaviour are reported.

Figures

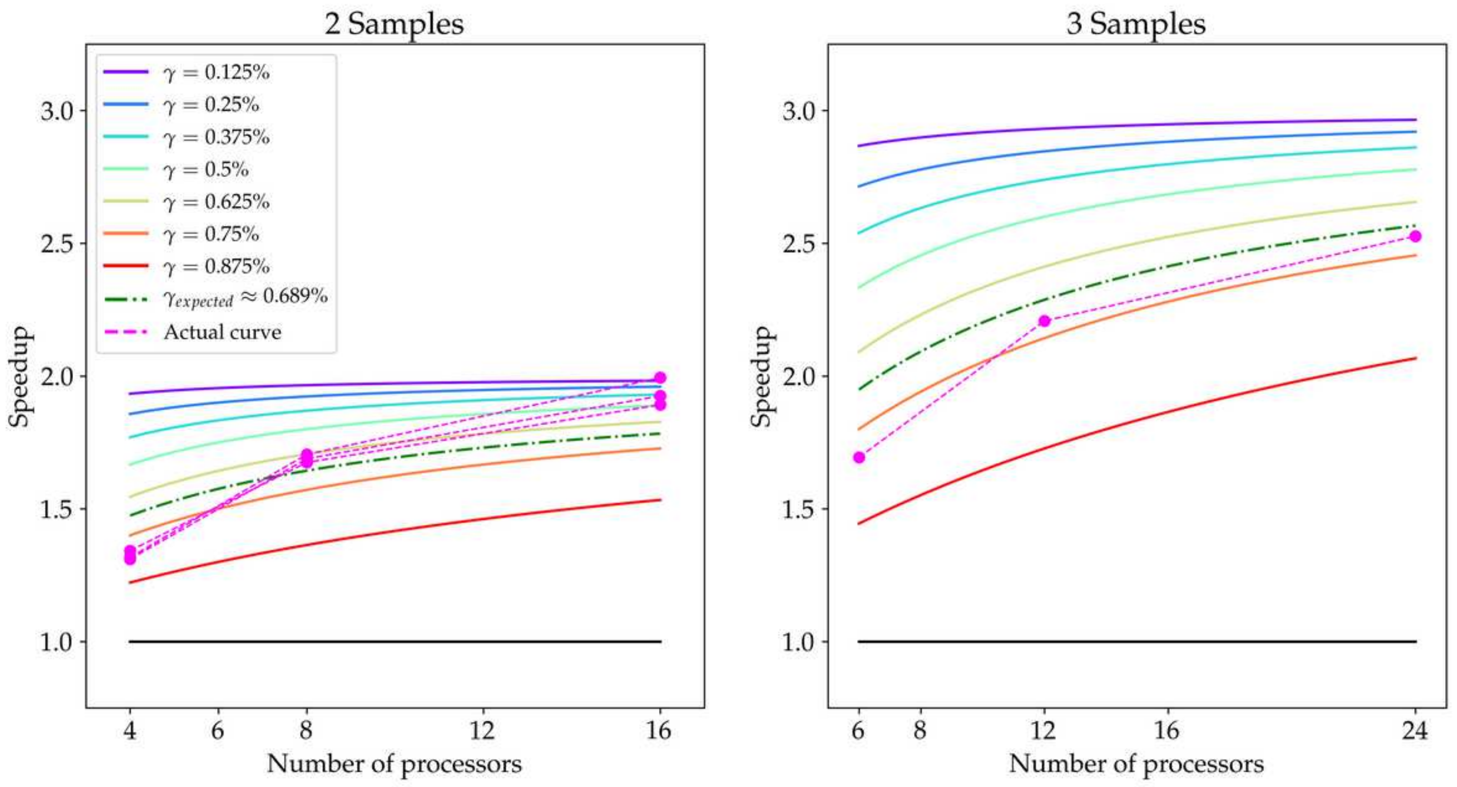

Figure 1

Speedup comparison. Expected speedup for CES compared to the NPS for 2 and 3 samples as a function of the available number of processors and PaCo proportion . The magenta dashed line represents the measured speedup of the real pipeline tested. The green dashed-dotted line represents the expected performance gain based on the estimated proportion of sequential execution time of our pipeline. In the 2-samples gure there are three observed speedups associated to all the combinations of two samples out of the three available. In the 3-samples gure there is a single observed speedup associated as there is only one possibile triplet out of the three available samples.

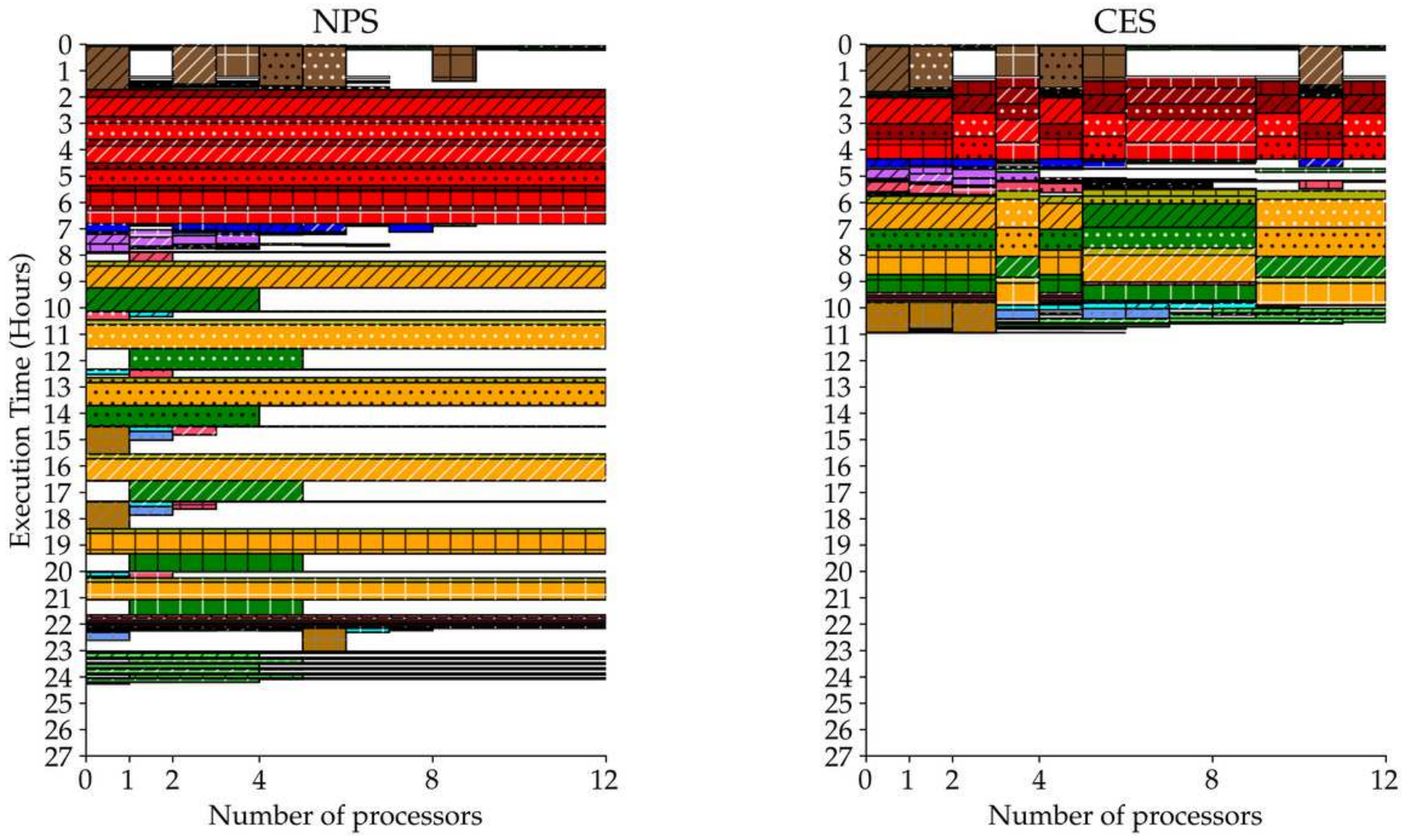

Figure 2

Pipelines execution comparison (processors allocation). Representation of a pipeline execution on all 3 samples based on processors usage along time for both NPS and CES. Each rectangle represents a task, its base and height are respectively equal to the allocated processors and to the time elapsed. The hatch type indicates the paired-matched sample that a task works on. The hatch color species the data type taken as input by a task. The color identies the task type. See Supplementary Figure 1, Additional File 1 for comprehensive legend. See gure 3 for its corresponding memory usage.

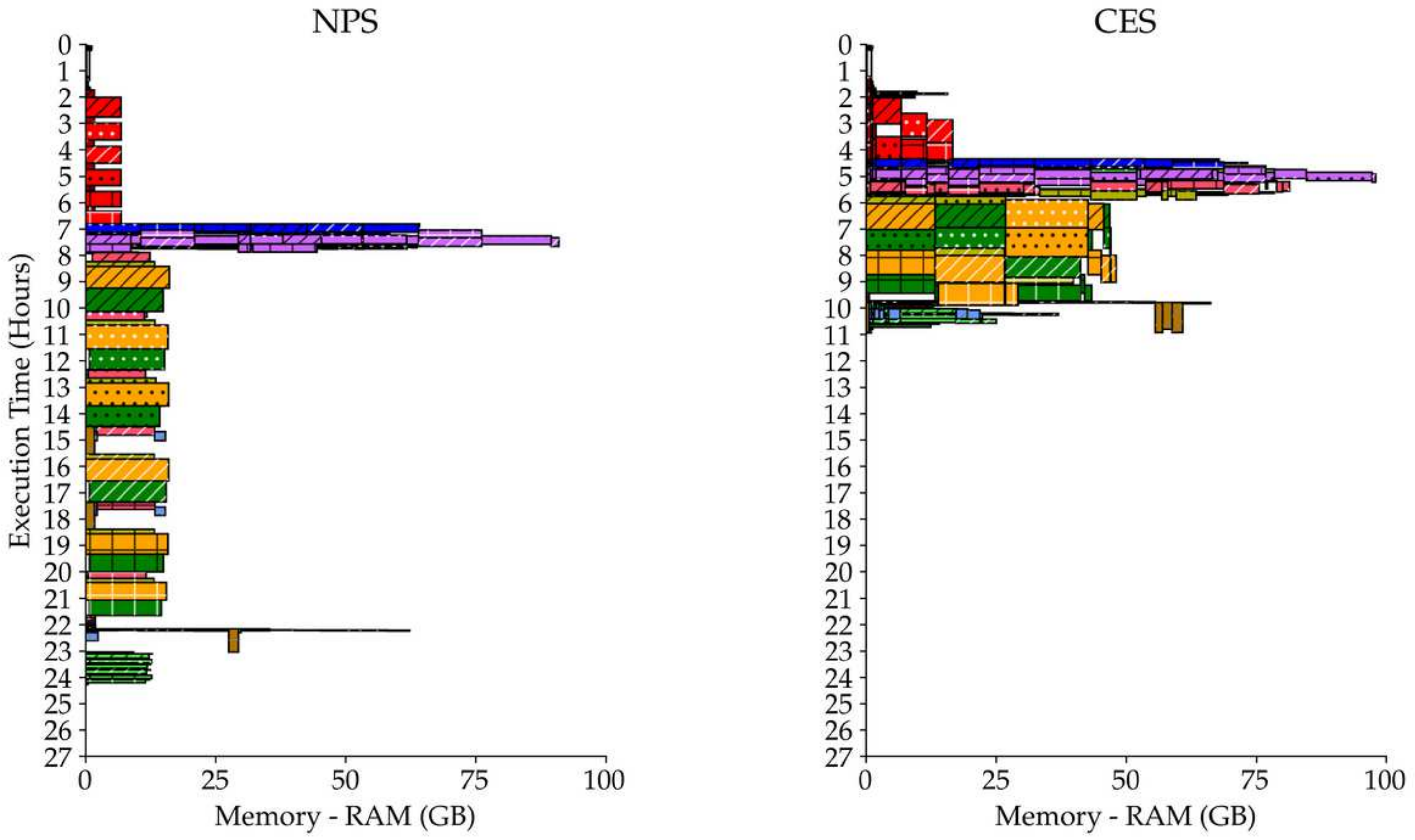

Figure 3

Pipelines execution comparison (memory usage). Representation of a pipeline execution on all 3 samples based on memory usage along time for both NPS and CES. Each rectangle represents a task, its base and height are respectively equal to the allocated amount of memory and to the time elapsed. The hatch type indicates the paired-matched sample that a task works on. The hatch color species the data type taken as input by a task. The color identies the task type. See Supplementary Figure 1, Additional File 1 for comprehensive legend. See gure 2 for its corresponding processors usage.

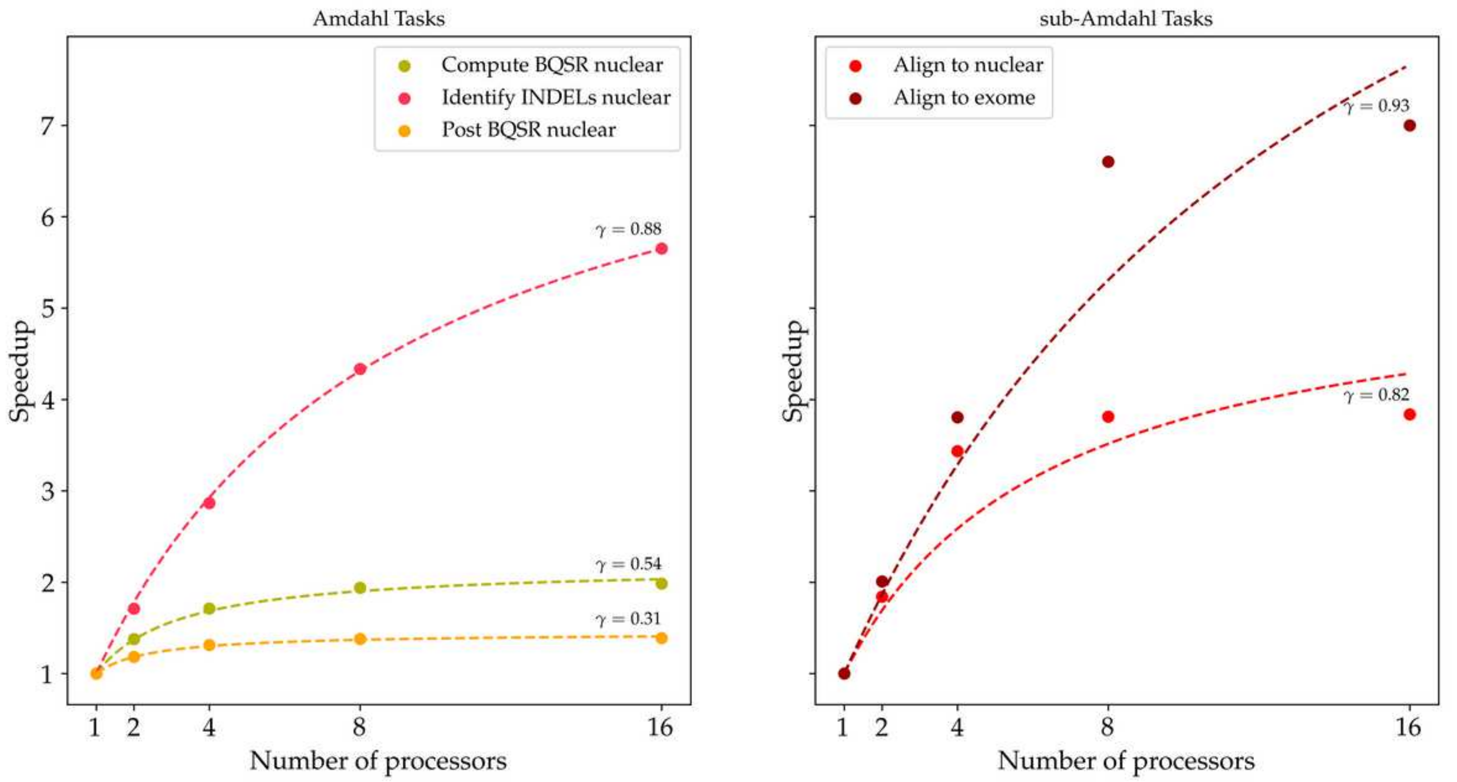

Figure 4

Single tasks' speedup comparison. Performance gain of several PaCo tasks with respect to the number of processors exploited and the comparison with the Amdahl's law predictions. The points represent the actual measured speedup. The dashed-lines shows the best Amdahl's t with the associated . On the left plot we report tasks behaving according to Amdahl's law. On the right plot we report tasks with signicant departure from Amdahl's law predictions, with a speedup for high number of processors lower than expected. Only PaCo tasks with a signicant speedup behaviour are reported.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- bmcarticle.bib
- Additionalfile5.pdf
- Additionalfile1.png
- Additionalfile3.pdf
- Additionalfile6.pdf
- Additionalfile4.pdf
- Additionalfile2.ods