



**FOM University of Applied Sciences for Economics &
Management**
University Center Hanover

Bachelor-Thesis
in the degree programme Wirtschaftsinformatik

submitted in partial fulfillment of the requirements for the degree

Bachelor of Science (B.Sc.)

on the subject

**Implementing a Scientific Workflow Management System to
Conduct the Transition to a Different Reference Genome of a
Genetic Analysis Pipeline**

by

Benedikt Schnur 

First examiner: Prof. Dr. Stephan Kluth
Matriculation number: 538719
Submission date: February 12, 2023

Table of contents

List of Figures	IV
List of Tables	VI
Listings	VIII
1 Introduction	1
1.1 Problem	3
1.2 Goals	5
1.2.1 Professionalization of Pipeline Utilization	5
1.2.2 Usage of Cloud Services	6
2 Literature Review	7
2.1 Reference Genome	7
2.2 Genetic Sequencing	7
2.3 Impact on Information Technology	9
2.4 Scientific Workflow Management Systems	9
3 Method	13
4 Artifact Description	16
4.1 Design Search Process	16
4.2 Analyzation of Existing Shell Script	16
4.3 Initial Conversion to a Nextflow Workflow	19
4.4 BAM to FastQ Conversion	22
4.4.1 Evaluation of BAM to FastQ Tools	23
4.4.2 Integration of BAM to FastQ Tool into Nextflow Workflow	26
4.5 Separation of megSAP Steps	28
4.6 Resource Optimization of megSAP Steps	33
4.7 Resilience and Monitoring	36
5 Cost Estimation of Potential Cloud Usage	38
6 Discussion	40
7 Conclusion	46
References	47

Appendix	54
I Sum of Processed Samples	54
II Extrapolation of Whole-Genome Samples Analyzed Until Project Deadline	55
III Example Script of Pipeline Invocation with Bash	57
IV Initial Nextflow Workflow	59
V Detailed Data of BAM to FastQ Conversion Benchmarks	61
VI Nextflow Workflow with added BAM to FastQ File Conversion	63
VII Nextflow Workflow With Separated Process Steps	66
VIII Nextflow Workflow With Resource Optimization	70
IX Correspondence with Dr. Marc Sturm Discussing CPU Usage of SeqPurge	72
X Nextflow Workflow With Resilience and Monitoring	74
XI Amazon Web Services (AWS) Prize Calculation for Optimized Nextflow Workflow	76
XII Amazon Web Services (AWS) Prize Calculation for initial Nextflow Work- flow	78

List of Figures

1	Comparison of cost per genome sequencing to the sum of processed samples at the DoHG@MHH	1
2	Number of samples and bases sequenced at the DoHG@MHH	2
3	Overview of the genetic sequencing and analyzation process	5
4	The Illumina sequencing-by-synthesis approach	8
5	Design Science Research Methodology (DSRM) Process Model	15
6	Directed acyclic graph (DAG) for initial Nextflow workflow	21
7	Average CPU usage of the initial Nextflow workflow	22
8	Peak memory usage of the initial Nextflow workflow	22
9	Median duration of BAM to FastQ conversion of whole exome data by tool	24
10	Median memory usage of BAM to FastQ conversion of whole exome data by tool	25
11	Median amount of data read and written by BAM to FastQ conversion of whole exome data by tool	25
12	Duration of BAM to FastQ conversion of whole genome data by tool . . .	26
13	Directed acyclic graph (DAG) for Nextflow workflow with added BAM to FastQ file conversion	28
14	Directed acyclic graph (DAG) for Nextflow workflow after separation of megSAP steps	30
15	Average CPU usage of the Nextflow workflow after separation of steps .	31
16	Peak memory usage of the Nextflow workflow after separation of steps .	32
17	Average CPU usage of the Nextflow workflow over time after separation of steps	32
18	Memory usage of the Nextflow workflow over time after separation of steps	33
19	Comparison of pipeline runtime over all iterations	34
20	Average CPU usage of the Nextflow workflow after optimization	35
21	Peak memory usage of the Nextflow workflow after optimization	35
22	Average CPU usage of the Nextflow workflow after optimization over time	36
23	Memory usage of the Nextflow workflow after optimization over time . .	36
24	Average CPU usage of the Nextflow workflows compared	40
25	Peak memory usage of the Nextflow workflows compared	40
26	Efficiency after pipeline optimization	42
27	CPU allocation of initial and optimized workflows over time	42
28	Memory allocation of initial and optimized workflows over time	43
29	Correspondence with Dr. Marc Sturm discussing CPU usage of SeqPurge	72

29	Correspondence with Dr. Marc Sturm discussing CPU usage of SeqPurge (cont.)	73
30	Amazon Web Services (AWS) prize calculation for optimized Nextflow workflow	76
30	Amazon Web Services (AWS) prize calculation for optimized Nextflow workflow (cont.)	77
31	Amazon Web Services (AWS) prize calculation for initial Nextflow workflow	78
31	Amazon Web Services (AWS) prize calculation for initial Nextflow workflow (cont.)	79

List of Tables

1	Comparison of Nextflow with other workflow management systems	11
2	Overview of workflow managers for bioinformatics	12
3	Descriptive statistics of megSAP runtime for genome samples	19
4	Configured resource allocation after optimization	34
5	Results of Amazon Web Services (AWS) price calculator tool and selected instance types	39
6	Sum of processed samples at the DoHG@MHH per year	54
7	Reported resource usage for initial workflow	60
8	Statistics of samples used for benchmarking BAM to FastQ conversion using the command samtools flagstat	61
9	Duration of BAM to FastQ conversion of whole exome data by tool in minutes	61
10	Memory usage of BAM to FastQ conversion of whole exome data by tool	62
11	Amount of data read by BAM to FastQ conversion of whole exome data by tool	62
12	Amount of data written by BAM to FastQ conversion of whole exome data by tool	62
13	Duration of BAM to FastQ conversion of whole genome data by tool in hours	62
14	Reported resource usage for workflow with separated process steps	69
15	Reported resource usage for workflow with resource optimization	71

Acronyms

AWS	Amazon Web Services
DAG	directed acyclic graph
DoHG@MHH	Department of Human Genetics at Hanover Medical School
DSL	domain-specific language
DSR	Design Science Research
DSRM	Design Science Research Methodology
EC2	Elastic Compute Cloud
FPGA	field-programmable gate array
GUI	graphical user interface
HPC	high-performance computing
LIMS	laboratory information management system
megSAP	Medical Genetics Sequence Analysis Pipeline
MHH	Hanover Medical School
NGS	next-generation sequencing
SLURM	Simple Linux Utility for Resource Management
SNV	single nucleotide variation (A DNA sequence variation that occurs when a single nucleotide (adenine, thymine, cytosine, or guanine) in the genome sequence is altered.)
SRS	short-read sequencing
SWfMS	scientific workflow management system
UCSC	University of California, Santa Cruz
WfMS	workflow management system

Listings

1	SQL for sum of processed samples	54
2	SQL query to gather data for extrapolation	55
3	Python code to extrapolate the sum of whole-genome sequencing runs until the project's deadline	55
4	bash script to trigger the megSAP pipeline	57
5	megsap_germline.nf (Initial Version)	59
6	nextflow.config (Initial Version)	59
7	megsap_germline.nf (with added BAM to FastQ File Conversion)	63
8	nextflow.config (with added BAM to FastQ File Conversion)	64
9	megsap_germline.nf (with separated process steps)	66
10	megsap_germline.sh	69
11	nextflow.config (with resource optimization)	70
12	nextflow.config (with resilience and monitoring)	74

1 Introduction

Next-generation sequencing (NGS) has seen vast improvements in cost and speed over the past several decades, as described by Schloss [1] and Davey *et al.* [2]. Behjati and Tarpey [3] write that “using NGS an entire human genome can be sequenced within a single day. In contrast, the previous Sanger sequencing technology, used to decipher the human genome, required over a decade to deliver the final draft.” This led to the introduction and increased usage of NGS in medical diagnostics performed by the Department of Human Genetics at Hanover Medical School (DoHG@MHH), as seen in figure 1.

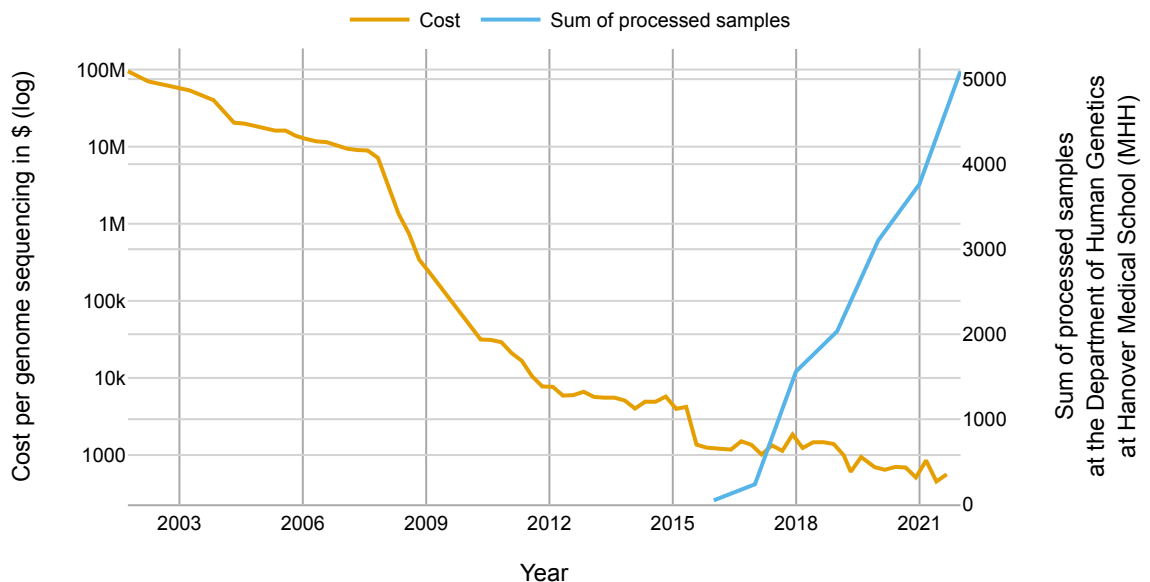


Figure 1: Comparison of cost per genome sequencing in U.S. dollar from [4] to the sum of processed samples at the Department of Human Genetics at Hanover Medical School per year (see table 6)

Initially, whole-exome sequencing was introduced at the DoHG@MHH in 2016. The exome covers nearly all the coding variation in an individual human genome and allows for a cost-effective analysis, as described in [5]. Starting in late 2021, following further cost reductions on the market, the genetic sequencing process at the DoHG@MHH switched to whole-genome sequencing as recommended in [6]. This covers all genetic data readable by the sequencing system, resulting in much more data being generated, as shown in figure 2.

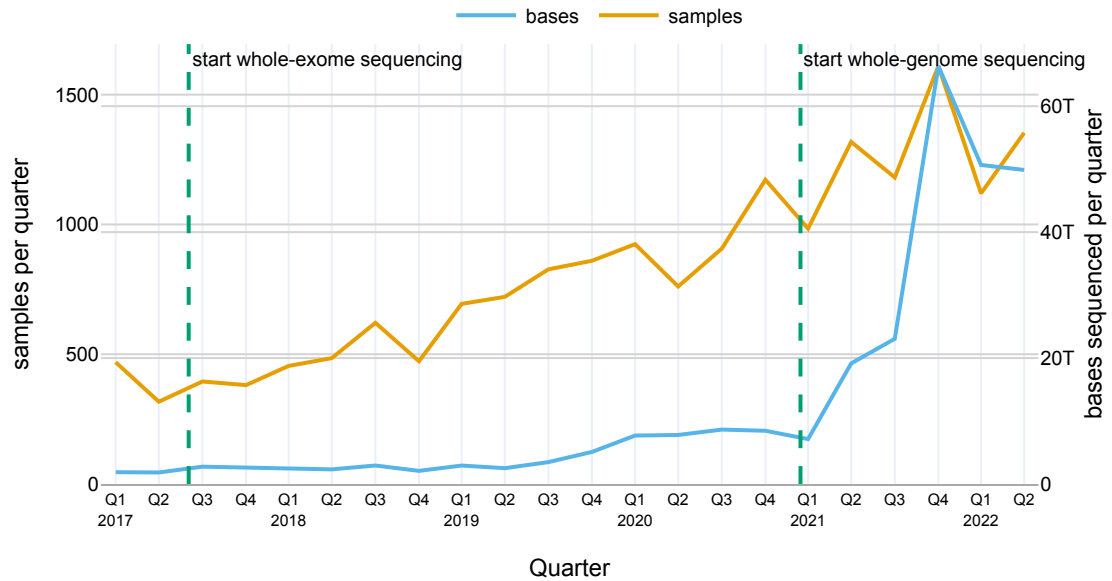


Figure 2: Number of samples and bases sequenced at the Department of Human Genetics at Hanover Medical School per year

Palladino [7, p. 15], Patel [8], and Caetano-Anolles [9] define a reference genome as a representation of a species’ complete DNA sequence used as a basis for comparisons with other genomic sequences. A reference genome serves as a standard by which variations in the genomes of individuals or populations of the same species can be characterized. There are several high-quality reference genomes available for numerous species, including humans (by the *Human Genome Project* described in [10]), mice (e.g., [11]), and many crops (see [12]). These reference genomes are the result of extensive research efforts that involve the collection and analysis of DNA from multiple individuals and the integration of data from a variety of sources.

The data analysis pipeline and associated diagnostic tools at the DoHG@MHH are currently based on the reference genome *GRCh37* (University of California, Santa Cruz (UCSC) title: *hg19*) released in 2009 (see [13]). It is used to align and sort the data fragments generated by the sequencing-by-synthesis technique used by the DoHG@MHH’s *NovaSeq 6000* [14] system by *Illumina, Inc.* into a single sequence, a process described in [15, 16, 17] and usually called *mapping*. In this context, a reference genome can provide a foundation for understanding the genetic differences that underlie health, disease, and diversity.

1.1 Problem

The DoHG@MHH uses the *Medical Genetics Sequence Analysis Pipeline (megSAP)* (<https://github.com/imgag/megSAP>) developed by the Institute of Medical Genetics and Applied Genomics at University Hospital and Faculty of Medicine Tübingen as its data analysis pipeline. In November 2021, the *megSAP* project switched to a newer reference genome, *GRCh38* (UCSC title: *hg38*) (see [18, 19]). The old version of the pipeline has been discontinued. In order not to fall behind, the DoHG@MHH wants to upgrade to the newer pipeline version as soon as possible. But switching the reference genome is not a drop-in replacement. In addition to possible compatibility issues with external databases used by biologists and medical geneticists in the diagnostic process, several local configurations and, more importantly, all self-produced reference data, needs to be adopted or reprocessed. This raises a couple of problems to be solved:

Processing capacity

The processing pipeline runs mainly on the MHH's high-performance computing (HPC) cluster. There, the DoHG@MHH has 17 nodes with a total of 656 CPU cores and 5.106 TB RAM available. The *NovaSeq 6000* handles up to 48 samples in one run. Processing these uses all that capacity for 20 h–24 h while being able to process 24 samples in parallel. No reprocessing can be done simultaneously, as diagnostic for current patients is time-sensitive and therefore has a higher priority. Investing in additional capacity in the short term is not feasible in the public service domain.

Storage capacity

At the DoHG@MHH, sequencing data is stored in the BAM file format (see [20]). Including analysis results, each sample takes approximately 100 GB in size (compressed) for whole-genome data and 10 GB for whole-exome data. About half of this are the BAM files that are backed up to tape storage after a successful run of the data analysis pipeline. All previously generated data has to be retrieved from tape storage for reprocessing. Roughly 1544 genome samples (see appendix II) are estimated to be present when the analytic pipeline is scheduled to migrate to *GRCh38* in April 2023. Thus, about $1544 \text{ samples} \times 50 \text{ GB} = 77.2 \text{ TB}$ of storage is needed for the genome data alone. Currently, all storage provided to the DoHG@MHH by the MHH (95 TB) is in use. The remaining 10 TB of space is reserved and needed for temporary file duplication while running the analytic pipeline.

Internet bandwidth

Former undocumented tests with cloud infrastructure exposed another bottleneck: The MHH has imposed a restriction of a maximum of Mbit/s of bandwidth (syn-

Introduction

chronously) for the DoHG@MHH. Uninterrupted uploading to a cloud provider would take $\frac{77.2 \text{ TB}}{500 \text{ Mbit/s}} = 14.3 \text{ d}$ to complete alone.

Processing time

As mentioned before, 48 samples are produced by one sequencing run of the *NovaSeq 6000* simultaneously (see step *sequencing* in Figure 3). These are then processed in parallel with the sequential analysis pipeline according to the steps shown in Figure 3. Reanalyzing has to start with the step *mapping*. Beginning at this step, the remaining process takes 20-24 h to complete. As described before, processing capacity is limited, so only a batch of 24 samples may be processed in parallel. With ideal conditions (no delay between pipeline runs, exclusive usage of processing resources) this would result in a total reanalyzing time of roughly $\frac{1544 \text{ samples}}{24 \text{ per batch}} \times 22 \text{ h} \approx 59 \text{ d}$ for the whole-genome data.

Architecture

In its current state, *megSAP*, which is written in *PHP*, is triggered by shell scripts. These scripts are written by researchers (biologists, biochemists and clinical geneticists) of the DoHG@MHH and follow no particular design rules. There is no monitoring system in place, the current state of a pipeline run can only be interpreted by looking at the files found in the working directory and the list of running processes. Performance reporting can only be done by manually analyzing a log file with minimal information. The pipeline is executed in MHH's HPC cluster using *Simple Linux Utility for Resource Management (SLURM)*. About 5 % of the processes fail with errors unknown to the users and have to be restarted manually (often after hours), which resolves the problem in almost all cases.

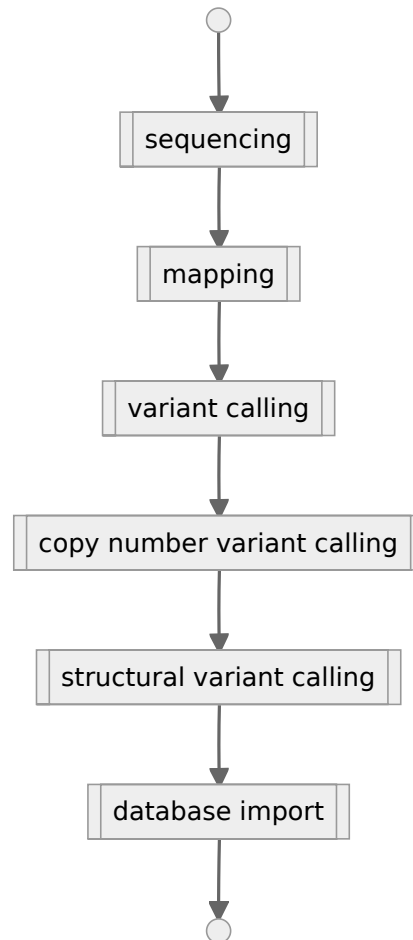


Figure 3: Overview of the genetic sequencing and analyzation process

1.2 Goals

The main objective to support the transition to a new reference genome is optimizing the analysis pipeline. Reanalysis must be performed as quickly as possible to allow the DoHG@MHH to switch to *GRCh38* as soon as possible. Two main ideas are explored, as described in the following subsections.

1.2.1 Professionalization of Pipeline Utilization

The current pipeline has been found to suffer from numerous issues related to a lack of computer science expertise among the scientists responsible for its implementation. Due to limited time dedication to this task, it has been proposed that adoption of a scientific workflow management system (SWfMS) would provide a solution to the challenges outlined previously. This approach would promote reproducibility of results, facilitate automatic retries, and enable more transparent error reporting. In support of this proposal, initial research

has identified potential SWfMS candidates, including *Nextflow* (<https://www.nextflow.io/>) and *Snakemake* (<https://snakemake.github.io/>), that will be evaluated, among others, based on their suitability. It is important to consider the accessibility and usability of the SWfMS, as the pipeline is expected to be used and maintained by biologists and medical staff in the future.

1.2.2 Usage of Cloud Services

The utilization of cloud computing for processing large data sets is a compelling proposition. Among the various cloud providers, *Amazon Web Services (AWS)* offers specialized *F1* instances through their *Elastic Compute Cloud (EC2)* service with field-programmable gate array (FPGA) support, as documented in [21]. Additionally, AWS provides official pre-installed *DRAGEN* software, shown in [22]. Nevertheless, the main challenge in implementing such a solution is the limited bandwidth, as outlined in subsection 1.1. Despite this challenge, solutions such as *AWS Snowball* (<https://aws.amazon.com/snowball/>, [23]) may offer a viable solution, and will be assessed.

However, it is important to consider the potential cost associated with the use of cloud services, as it entails additional expenses for the DoHG@MHH. Therefore, it is necessary to carefully estimate the cost of such an endeavor.

2 Literature Review

This literature review will provide an overview of the current state of NGS technology, highlight the evolving key challenges for information technology, and evaluate the existing body of knowledge on SWfMS to build a foundation for the research presented in this thesis.

2.1 Reference Genome

Initially, the switch to a newer reference genome is examined, as it is the fundamental change resulting in the problem at hand. This has been discussed by several journal articles. Schneider *et al.* [24] conclude that the current iteration of the reference genome, *GRCh38*, has seen significant improvements in its assembly statistics and contains accurate representations of important clinical regions. The addition of new sequence content not only fills gaps in previous genomic data, but also captures population genomic diversity. These improvements to the assembly make *GRCh38* an ideal substrate for annotation and a more effective mapping target. They recommend that *GRCh38* should be utilized for all types of analyses as it represents the most comprehensive and accurate depiction of the human genome to date, surpassing previous assembly versions. Guo *et al.* [25] compared 30 whole-exome sequencing samples processed each with *GRCh37* and *GRCh38*, respectively, and found that based on the comparative exome sequencing data analysis conducted between *GRCh37* and *GRCh38*, it can be concluded that *GRCh38* represents an improvement over *GRCh37*. These improvements have resulted in more accurate results for genomic analysis. Pan *et al.* [26] describe why the change to *GRCh38* should not be done by simply converting the current analyzation results to the new nomenclature, but a reanalyzation is needed, as a substantial percentage of single nucleotide variations (SNVs) failed to be converted during the process: approximately 5% when using *GRCh38* and 1% when using *GRCh37*. This observation suggests that *GRCh37*, the older version, lacks some genomic resolution compared to the newer version. After conducting a thorough comparative analysis, they recommend that *GRCh38* should be utilized in future SNV analysis as it presents a more advantageous option.

2.2 Genetic Sequencing

In order to understand the fundamentals of genetic sequencing, the articles by Lander *et al.* [27] and Venter *et al.* [28] about the generation, assembly, and evaluation of the first whole sequence of the human genome by the *Human Genome Project* give meaningful insights. The Illumina sequencer used by the DoHG@MHH works with the NGS technology called

sequencing-by-synthesis, which was described 2008 by Mardis [29]: “Cluster strands created by bridge amplification are primed and all four fluorescently labeled, 3'-OH blocked nucleotides are added to the flow cell with DNA polymerase. The cluster strands are extended by one nucleotide. Following the incorporation step, the unused nucleotides and DNA polymerase molecules are washed away, a scan buffer is added to the flow cell, and the optics system scans each lane of the flow cell by imaging units called tiles” (illustrated in figure 4). It utilizes reversible terminator chemistry, demonstrated by Bentley *et al.* [30].

The *Genome in a Bottle Consortium*, hosted by the *National Institute of Standards and Technology (NIST)*, provides reference data and material to calibrate, benchmark and validate the genetic analysis process. The process to produce the reference data is explained by Zook *et al.* [31] and Baid *et al.* [32]. Additionally, the *Global Alliance for Genomics and Health (GA4GH)* provides guidelines how a genetic analysis pipeline can be benchmarked with this data, described by Krusche *et al.* [33].

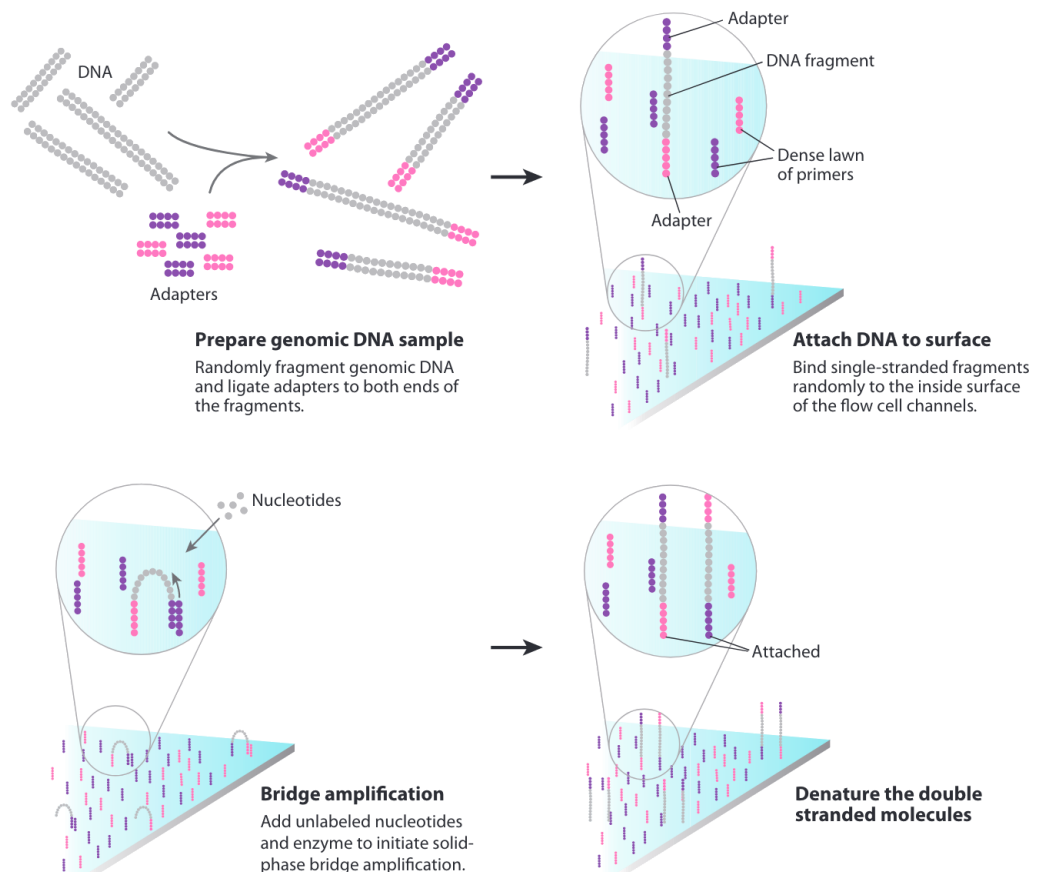


Figure 4: The Illumina sequencing-by-synthesis approach illustrated in [29]

2.3 Impact on Information Technology

Several publications highlight the need for information technology to keep up with the advancements in NGS: Mardis discusses “The Impact of Next-Generation Sequencing Technology on Genetics” [34] in 2008 and concludes that the ongoing progress in utilizing short-read sequencing (SRS) technologies in biological research necessitates the creation of novel algorithms and software capable of accommodating the unique features of these technologies. In particular, there is a need for new tools to manage the extensive data generated by SRS technologies and to efficiently perform common bioinformatics tasks (such as alignment) with a high volume of short reads. Shendure and Ji [35] predict that the focus of the challenges will transition from acquiring proficiency in the technologies to determining the most effective methods for deriving biologically relevant or clinically significant information from massive amounts of data. Voelkerding *et al.* [36] foresee that over the following years, NGS will make a successful transition into clinical diagnostics. The success of this transition will require the streamlining of processes and the ability to handle the bioinformatics challenge posed by the large amounts of sequence data output for clinical laboratories. Metzker [37] concludes, that the generation of vast amounts of NGS reads has presented numerous challenges for the existing information technology systems, including the difficulties in data transfer, storage, quality control, and computational analysis for read alignment and assembly. This has also placed pressure on laboratory information management systems for effective sample tracking and process management. Despite ongoing advancements in bioinformatics, it is essential to make further enhancements in order to keep up with the fast-paced developments in NGS technologies. There is a possibility that the costs associated with the handling and analysis of NGS data could be on par or even surpass the cost of producing the data.

2.4 Scientific Workflow Management Systems

According to Georgakopoulos *et al.* [38], “workflow management is a technology supporting the reengineering of business and information processes. It involves: 1. defining workflows, i.e., describing those aspects of a process that are relevant to controlling and coordinating the execution of its tasks (and possibly the skills of individuals or information systems required to perform each task), and 2. providing for fast (re)design and (re)implementation of the processes as business needs and information systems change”. van der Aalst and van Hee [38] define a WfMS as typically composed of three main components: a workflow model, a workflow engine, and a workflow repository. The workflow model defines the steps and tasks involved in a workflow, the workflow engine executes the workflow, and the workflow repository stores and manages the workflow information.

Execution models of SWfMSs differentiate themselves by being data-flow oriented instead of event and control-flow driven like business workflows, in accordance with Ludäscher *et al.* [40] describing their system *Kepler*. Additional SWfMSs from the early 2000s include *Taverna* as described by Oinn *et al.* [41] and *Galaxy* introduced by Giardine *et al.* [42]. Only *Galaxy* is still maintained today.

More recent forays in the realm of SWfMS are *Snakemake* (<https://snakemake.github.io/>), introduced by Köster and Rahmann [43], and *Nextflow* (<https://www.nextflow.io/>), described by Di Tommaso *et al.* [44]. *Snakemake* and *Nextflow* follow a similar concept. Both feature a domain-specific language (DSL) to describe workflows in a text-based definition, which Köster and Rahmann see as advantageous: workflows can be modified outside a graphical interface, such as on a remote server, and developers can collaborate on them utilizing source code management tools. There are differences in details, but they are nevertheless relevant for the DoHG@MHH. The HPC cluster is managed with *SLURM*, so direct support is preferred. As the use of cloud infrastructure should be evaluated, AWS support is required. Table 1 shows that *Snakemake* is lacking in these crucial areas. Di Tommaso *et al.* further note that the task sequence in *Snakemake* is determined by rules and patterns based on the input and output file names, which makes it challenging to manage multiple dynamically generated output files, leading to the need for the implementation of low-level output management procedures. However, *Nextflow* enables the use of any data structure, and its outputs are not limited to files but can also include in-memory values and data objects. Unlike *Snakemake*, which requires a directed acyclic graph (DAG) to store the task execution order, *Nextflow* uses a top to bottom processing model, which follows the natural flow of data analysis. This approach does not require pre-computing or storing a DAG, resulting in high scalability and making it suitable for large computational tasks. Compared to *Galaxy*, Di Tommaso *et al.* note that the graphical user interface (GUI), which provides robust support for non-specialists to implement *de novo* pipelines, also places a substantial development load as any pre-existing and validated third-party pipeline must be recreated and reconfigured using the GUI.

Nextflow was chosen by a group of ~25 Computational Biologists and Data Scientists at the *September 2017 Pitt-NCBI Hackathon* to create a proof-of-principle simple RNA-seq pipeline at [45]. *Snakemake* was discarded due to its inflexibility compared to *Nextflow*. *Nextflow* was ultimately selected because of its ability to utilize any programming language, handle inputs and outputs effectively, and its ease of wrapping. The authors also highlight its usefulness and versatility in their report. Larsonneur *et al.* [46] present benchmarks for several SWfMSs, and find that *Snakemake* and *Nextflow* showed close performances. Jackson *et al.* [47] use prototyping to find a suitable SWfMS to wrap their existing pipeline, *RiboViz*, and chose *Nextflow* for several reasons. The ability to execute each step within

separate subdirectories and the option to re-execute individual steps is useful for debugging purposes. Although writing in *Groovy* is required to develop workflows in *Nextflow*, the authors, who had prior experience with *Python* and *R*, found learning *Groovy* to be manageable. Additionally, the built-in support and comprehensive documentation for containers, high-performance computing systems, and cloud platforms offered by *Nextflow* appeared to be more extensive than those provided by *Snakemake*. They conclude that they can use *Nextflow* to construct *RiboViz* in a more portable and maintainable manner, enabling them to take advantage of the power of distributed computing resources to analyze large-scale datasets. Wratten *et al.* [48] compare several SWfMSs and give *Nextflow* the highest marks over several categories as shown in table 2. They also highlight the *nf-core* framework for *Nextflow* introduced by Ewels *et al.* [49] as collaboratively created best-practice analytic pipelines that are peer-reviewed by the community, which might prove useful for future endeavors at the DoHG@MHH. Ahmed *et al.* [50] benchmark *Nextflow* against *Swift/T*, *CWL* and *WDL*. They observe that *Nextflow* scales particularly well, sometimes outperforming the other tools fifty-fold. The other categories examined (modularity, robustness, reproducibility, portability, interoperability, and ease of development) show differences between the tools, with *Nextflow* fulfilling all of these satisfactorily.

Table 1: Comparison of Nextflow with other workflow management systems^a

SWfMS	Nextflow	Snakemake	Galaxy
Platform	Groovy/JVM	Python	Python
Workflow versioning	Yes	No	Yes
Automatic error failover	Yes	No	No
SLURM support	Yes	Partial	Yes
AWS support	Yes	No	Yes

^a Excerpt from [44, Table 1]

Table 2: Overview of workflow managers for bioinformatics^a

	Ease of Use	Expressiveness	Portability	Scalability	Learning Resources	Pipeline Initiatives	Sum
Galaxy	3	1	3	3	3	2	15
Nextflow	2	3	3	3	3	3	17
Snakemake	2	3	2.5	3	2	3	15.5

^a Excerpt from [48, Table 1]

3 Method

This thesis uses the Design Science Research Methodology (DSRM). DSRM itself is based on the *Design science* paradigm, described by Simon [51]. Hevner *et al.* [52] explain that design science involves the creation and evaluation of information technology artifacts aimed at addressing specific organizational issues. These artifacts can take various forms, ranging from software, formal logic, mathematical models, to informal language descriptions. The mathematical foundation of the design process enables various forms of quantitative evaluations of IT artifacts, including optimization, analytical simulations, and comparative assessments with alternative design options.

Based on the DSRM process illustrated by Peffers *et al.* in figure 5, the research entry point for this thesis will be the *Design & Development Centered Initiation*. The problem, as well as the motivation, are both already well-defined and are outlined in section 1.1: the DoHG@MHH needs to switch the genetic analysis pipeline to a newer reference genome, presenting multiple challenges to overcome. The *objectives of a solution* (see figure 5) are outlined in section 1.2: professionalization of pipeline utilization and potential usage of cloud services. Both should be accomplished by introducing a SWfMS to the genetic analysis pipeline. There is no quantitative goal to reach — all improvements are considered valuable. Nevertheless, the outcome should outweigh the investment, therefore pipeline runtime and resource usage will be evaluated. Although accessibility and usability are needed, given that the pipeline will be used by biologists and medical staff, those soft features will not be assessed.

This makes the following steps the research focus of this thesis:

Design & Development

Section 4 will document the transition from shell scripts to a SWfMS. The choice of SWfMS (*design search process*, see [54]) based on the Literature Review (section 2) will be outlined as well.

Demonstration

The translated pipeline will be run, and the results will be compared to already existing reference results to validate the correct usage as part of the *Design & Development* process.

Evaluation

Runtime and resource usage of the pipeline run by the SWfMS will be measured and compared to previous versions of the pipeline as part of the Artifact Description in section 4. The results will also be discussed in section 6.

Method

As the evaluation of the resource usage of the SWfMS will give some meaningful insights for optimization, section 4 will iterate upon these steps to reach a satisfactory outcome.

Additionally, the structure of this thesis closely follows the publication schema for a DSR study, described by Gregor and Hevner [54, Table 3]. Following their example, the closing sections, Discussion (section 6) and Conclusion (section 7), will be used to bolster the research rigorousness of this thesis.

Method

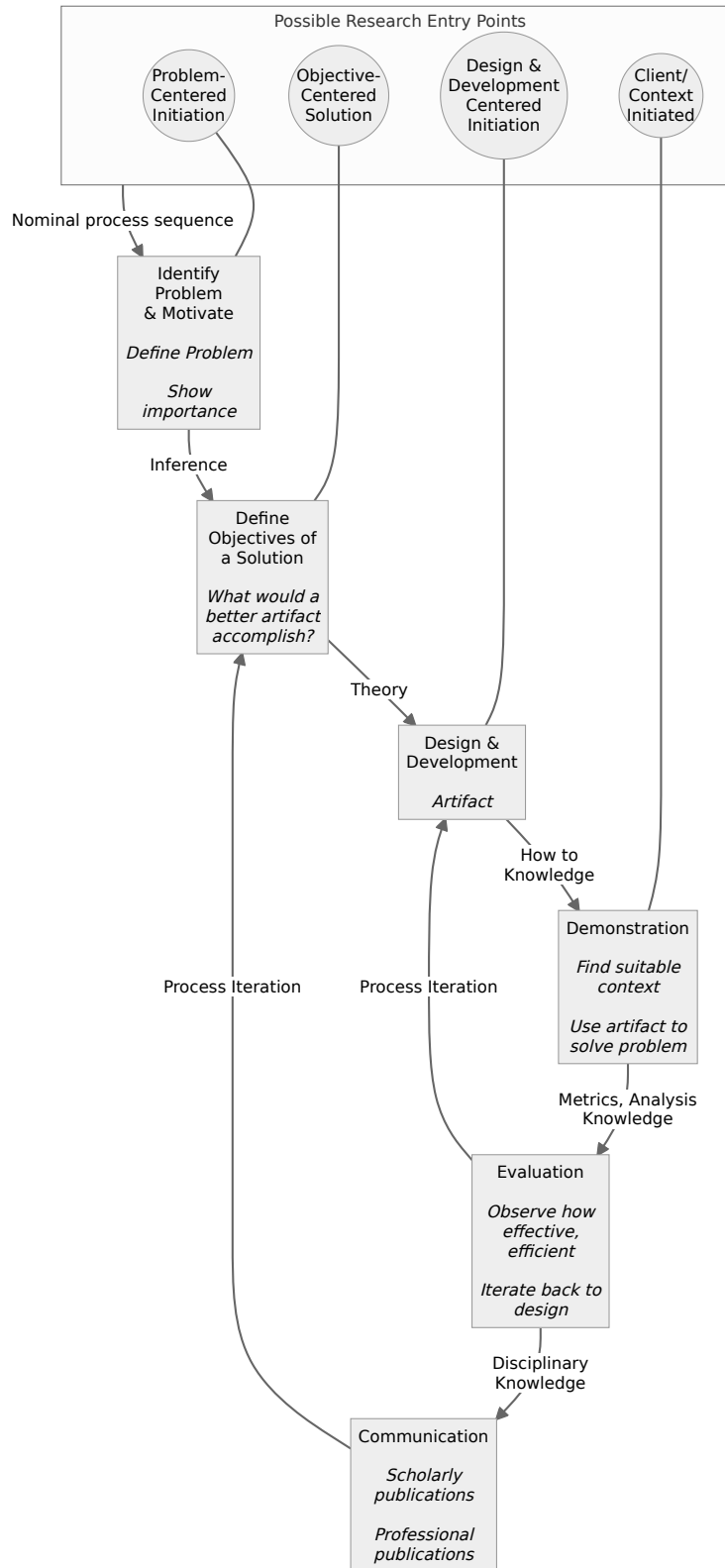


Figure 5: DSRM process model

Based on [53, Figure 1].

4 Artifact Description

The selected solution to the DoHG@MHH's problems is the implementation of a SWfMS: the pipeline usage will be professionalized and the optional usage of cloud services will be possible. The selection is based on the results of the literature review found in section 2.4. The following subsections will elaborately describe the artifact design and search process. All benchmarks are done with the sample *NAI2878* provided by the *Genome in a Bottle Consortium* if not stated otherwise. The use of this sample is recommended as described in section 2.2.

4.1 Design Search Process


The DoHG@MHH needs a SWfMS that fulfills the following needs:

- Support for *SLURM*, the job scheduler used by MHH's HPC cluster.
- Support for *Singularity*, the container format used by MHH's HPC cluster.
- Support for the optional use of cloud providers (changeable per pipeline run), at least AWS.
- Accessible and usable by the biologists and medical staff that currently operate the pipeline.

A GUI based tool like *Galaxy* seems promising regarding accessibility and usability. But as Wratten *et al.* [48] outline, expressiveness is lacking. Additionally, the already validated *megSAP* would have to be re-implemented and re-parameterized, which is not feasible. A flexible and DSL based SWfMS is needed, allowing the existing pipeline to be easily ported to the new system. *Nextflow* and *Snakemake* are popular and modern SWfMSs of this type. Based on the literature presented in section 2.4, *Nextflow* seems to have the overall edge over *Snakemake* and other tools, while satisfying all requirements. Consequently, *Nextflow* is chosen as the SWfMS to be implemented at the DoHG@MHH to reprocess the existing sequencing data.

4.2 Analyzation of Existing Shell Script

The *Nextflow* workflow will be based on the scripts currently in use at the DoHG@MHH. This workflow will be iteratively refined to improve its efficiency.

The *bash* script, found in listing 4 (appendix III) is written by Dr. rer. nat. Winfried Hofmann  of the DoHG@MHH and was created in July 2019.

Artifact Description

The script begins by defining several variables that will be used later in the script. The first three variables are the full path to the commands for the `pwd` and `mkdir` functions and the current date (lines 5–7). The `pwd` command returns the current working directory, while the `mkdir` command creates a new directory. The current date is stored in the variable `DATE` in the format *year-month-day_seconds*. The assignment of the full path of the `pwd` and `mkdir` commands within the script is not necessary, as they are not utilized in a differing environment or context. Therefore, these assignments are redundant.

The script then sets three working directories (lines 10–12):

1. The current working directory as the variable `WDIR` using the `pwd` command.
2. The working directory in the *Singularity* container as the variable `WDIR_SINGULARITY` by using the `sed` command to modify the `WDIR` path to remove the `/mnt/hgenet` prefix.
3. The hard-coded *SLURM* working directory as the variable `WDIR_SLURM`.

The script then changes the working directory to `WDIR` using the `cd` command (line 14). Since the current directory is still the same, this step is unnecessary, and the command will not result in any meaningful change to the current working directory.

Next, the script creates an array called `SAMPLEDIRARRAY` of directories having names starting with `Sample__` using the `ls` and `grep` commands (line 15). The `ntasks` variable is then set to the number of directories in the array using the `echo` command (line 16). This variable is defined in the script, but it is not used in any subsequent operations, making its definition superfluous.

The script checks if a subdirectory of `WDIR_SLURM` with the current date time combination (`DATE`) exists. If the directory does not exist, the script creates the directory using the `mkdir` command (lines 19–21). By using `mkdir`'s parameter `-p`, as described in [55], the complexity could be reduced by removing the conditional statement as an existing directory would be ignored without an error.

The script then enters a loop that iterates over each of the directories in the `SAMPLEDIRARRAY` array. For each `SAMPLEDIR`, the script uses the `sed` command to extract the sample ID from the directory name. The script then creates a bash file itself, with the sample ID as the name, and writes several `SBATCH` directives to the file via the `echo` command. `SBATCH` directives are used to specify resources and the execution environment for jobs submitted to a *SLURM* batch system with the `sbatch` command. When the script or command file is executed, the batch system allocates the specified resources and executes the job according to the specified execution environment. The used `SBATCH` commands in line 28–38 are:

#SBATCH -J jobname Specifies the name of the job.

Artifact Description

<code>#SBATCH -p partition</code>	Specifies the partition or queue in which the job should be run.
<code>#SBATCH --exclude=nodes</code>	Specifies a list of nodes on which the job should not be run.
<code>#SBATCH --no-kill</code>	Specifies that the job should not be terminated when the user logs out.
<code>#SBATCH --cpus-per-task=n</code>	Specifies the number of CPUs that should be allocated to each task of the job.
<code>#SBATCH --mem=n[G M K]</code>	Specifies the amount of memory that should be allocated to the job.
<code>#SBATCH --time=hh:mm:ss</code>	Specifies the maximum runtime of the job.
<code>#SBATCH --chdir=directory</code>	Specifies the working directory for the job.
<code>#SBATCH --error=file</code>	Specifies the file to which the standard error output of the job should be redirected.
<code>#SBATCH --output=file</code>	Specifies the file to which the standard output of the job should be redirected.

The `--exclude` command is given twice. This could be merged into one statement.

The `singularity exec` command (line 39), which is used to execute a command within a *Singularity* container, finalizes the newly created bash file. The `-B` option is followed by a list of directories to be bound from the host to the container file system, in the format `host_path:container_path`. Then the path to the *Singularity* container is specified. Finally, the command running the `analyze.php` script of *megSAP* follows, with several arguments:

`-folder $WDIR_SINGULARITY/$SAMPLEDIR`

Specifies the working directory for the pipeline based on the previously defined variables.

`-name $SAMPLEID`

Specifies the name of the sample based on the previously defined variable.

`-use_dragen`

Instructs the pipeline to use the *DRAGEN* systems of the DoHG@MHH.

`-no_abra`

Specifies to ignore the *abra* part of the pipeline.

`-system /NGS_Daten_Test/Manifest/NGSD/IDTPanelV2_GRCh38.tsv`

Specifies the path to the desired system manifest file.

`-threads 12`

Specifies the number of threads to use for the pipeline execution, in this case 12.

The loop then ends with the `sbatch` command used to submit the job to the *SLURM* batch system using the newly created script (line 40). This step is the last in the loop and concludes the pipeline script.

megSAP logs the execution time in a file for every processed sample. An analysis of 500 random samples processed in 2022 shows that the pipeline for a genome analysis usually runs for a little longer than a day (see table 3).

Table 3: Descriptive statistics of megSAP runtime for genome samples
500 random samples analyzed by the DoHG@MHH in 2022 were analyzed.

Statistic	Duration
Mean	1 d, 1 h, 56 min
Standard deviation	0 d, 6 h, 42 min
Median	1 d, 0 h, 31 min

4.3 Initial Conversion to a Nextflow Workflow

As a first step, the existing shell script is translated to a *Nextflow* workflow as accurately as possible to ensure that no errors occur based on this initial change. The definition of the workflow consists of two files:

`megsap_germline.nf`

A script written in the *Nextflow* script DSL (Version 2) (see appendix IV, listing 5).

The script header contains the shebang pointing to the *Nextflow* interpreter (using the `env` program), and the directive to use version 2 of the *Nextflow* DSL (see [56]) (line 1–2).

The script begins by setting the `sampleDir` parameter to a directory that usually contains the sample data: `${launchDir}/Sample_*` (line 4). `${launchDir}` is a variable provided by *Nextflow* and contains the directory where the `nextflow` command is run, `/Sample_*` matches any directories with names starting with `Sample_`.

The script then defines a process named `megSAP` (line 8), which takes a single input value, `sampleDirectory`. The following script block (line 11–15) defines the

Artifact Description

commands that will be executed when that process is run. First, the directory and the sample name are derived from the given sample directory using the *Nextflow* DSL. Then the *PHP* command to run *megSAP* using all relevant parameters is specified as a *bash* snippet enclosed by three double-quotes.

The workflow definition (line 18–20) introduces a channel created with the `Channel.fromPath` command. A channel is a way to represent and manage data flow within a *Nextflow* pipeline. It is a key concept which is used to define the inputs and outputs of processes, as well as to connect processes together. In this script, a channel of directory paths defined by the `sampleDir` parameter is created and then piped (see [56, Pipes]) to the `megSAP` process as input. This means that the `megSAP` process will be executed on each directory that matches the pattern defined by the `sampleDir` parameter.

`nextflow.config`

A *Nextflow* configuration file containing the definitions of the runtime environment (see appendix IV, listing 6).

The first block of code, `process {...}` (line 1–11), sets various options for the pipeline’s process. The `debug` option is set to `true`, which means that *Nextflow* will output additional information, especially the standard output, to the console for debugging purposes. The `executor` is set to `slurm`, which means that the pipeline will use the *SLURM* job scheduler to manage the execution of the pipeline’s processes. `clusterOptions` is set to a string of options passed to the *SLURM* scheduler, such as the `time`, `partition`, and `nodes to exclude`, and the `container` option is set to the path of the *Singularity* container image that will be used to run the pipeline’s processes. The `containerOptions` are additional options passed to the container to mount certain directories. The `stageInMode` is set to `'symlink'` which means that files will be symlinked into the container instead of copied. The `cache` option is set to `false`, which means that *Nextflow* will not use caching for intermediately used files when between processes. The latter is not necessary, as the *Nextflow* pipeline currently only contains one process step. The `cpus` option is set to `12` which means that the pipeline’s processes will use 12 CPU cores. The `memory` option is set to `50.GB`, which means that the pipeline’s processes will use 50 GB of memory.

The `singularity` block of the configuration (line 13–16) enables the use of *Singularity* for the pipeline, and sets the `autoMounts` option to `true`, which means that *Nextflow* will mount the file systems specified in the `containerOptions`.

The `report` (line 18–22), `timeline` (line 24–28), and `dag` (line 30–34) blocks of code enable the generation of various reports for the pipeline (see [57]) such as

Artifact Description

the HTML execution report, an HTML timeline report and a visualization of the pipeline’s DAG respectively. The `file` option is set to the desired file names of the reports and the `overwrite` option is set to true, which means that *Nextflow* will overwrite the report file if it already exists.

Finally, the `cleanup` option is set to true (line 36), which means that *Nextflow* will remove intermediate files and directories when the pipeline completes.

This split into two files leans on the design principle “separation of concerns”, as the functional workflow itself is split from the definition of the environment it is run in. This principle, generally used to describe the separation of software modules (see [58]), can be applied here as well.

Running the pipeline generates the simple DAG depicted in figure 6.

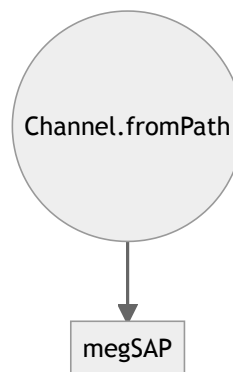


Figure 6: DAG for initial Nextflow workflow

Running the pipeline under ideal conditions, i.e., no other jobs running on the cluster, the pipeline runs for 11.5 h. Usually, this takes longer on average when multiple jobs are running (like the 48 samples processed by the sequencer), as the mapping on the DoHG@MHH’s two *DRAGEN* servers is running single threaded by design. The average CPU usage is 24.5 % of the allocated 12 CPU cores as shown in figure 7. Peak memory usage is 81.1 % of the assigned 50 GB as shown in figure 8 (see appendix IV, table 7 for the exact values reported by *Nextflow*).

Artifact Description

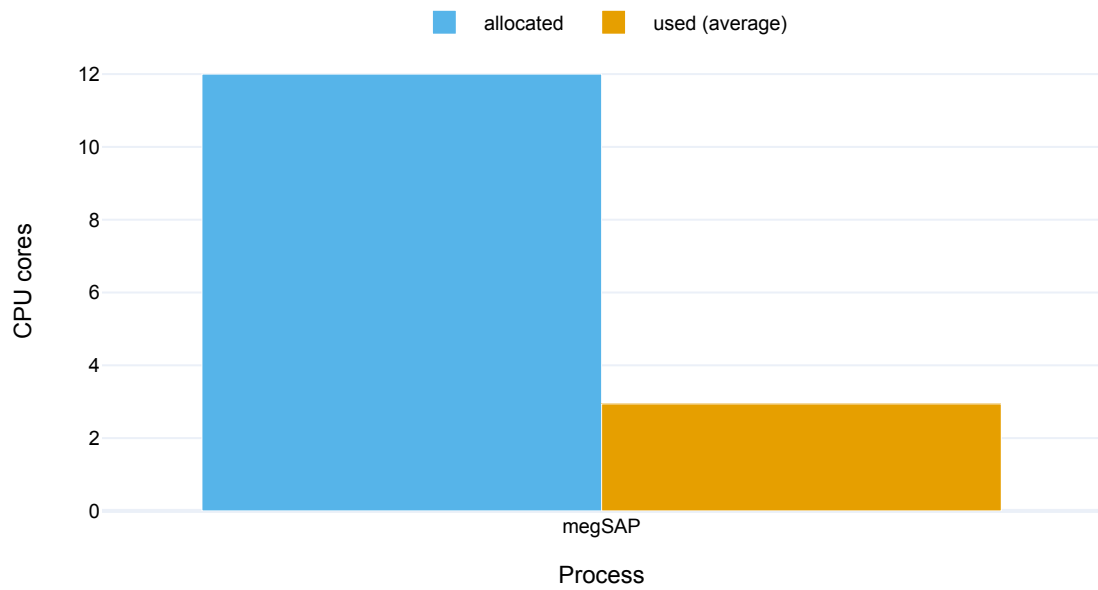


Figure 7: Average CPU usage of the initial Nextflow workflow

Detailed statistics are listed in table 7.

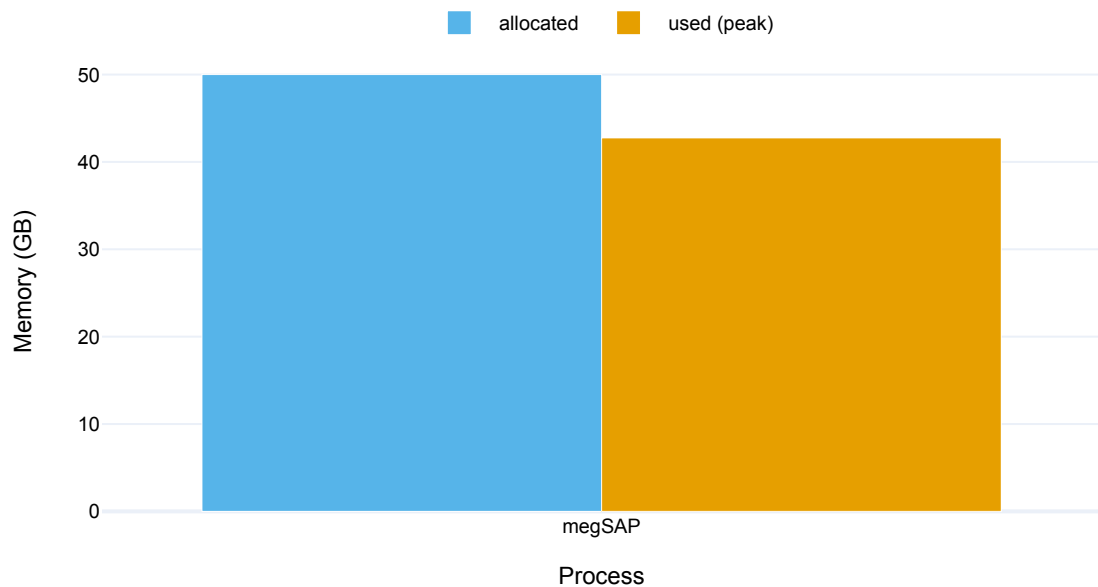


Figure 8: Peak memory usage of the initial Nextflow workflow

Detailed statistics are listed in table 7.

4.4 BAM to FastQ Conversion

The DoHG@MHH stores its archival data in the BAM file format described by Li *et al.* [20]. These are generated after the alignment to a reference genome. Hence, the files must

be converted back to an unaligned state in the FastQ file format, outlined by Cock *et al.* [59], before they can be realigned to the new reference genome. This transformation will be added to the *Nextflow* workflow as an optional processing step.

4.4.1 Evaluation of BAM to FastQ Tools

Four suitable tools have been identified by a literature research and are being evaluated:

biobambam2

Based on *biobambam* by Tischler and Leonard released in 2014. Not as actively maintained as the other tools (latest release March 17, 2021) and lacks documentation, but is nevertheless taken into consideration because of promising benchmark results documented in [60]. *biobambam2* does not support multithreading and is written in C++.

ngs-bits

Released in 2015, actively maintained (latest release July 8, 2022) by the *Institute of Medical Genetics and Applied Genomics at University Hospital and Faculty of Medicine Tübingen* [61]. *ngs-bits* does not support multithreading and is written in C++.

Picard

Released in 2009, actively maintained (latest release June 29, 2022) by the *Broad Institute of MIT and Harvard*, a non-profit biomedical and genomic research center [62]. *Picard* supports multithreading and is written in *Java*. It produces uncompressed FastQ files, which then require compression before further use.

SAMtools

Released in 2009, actively maintained (latest release September 2, 2022) by the *Genome Research Ltd.*, a non-profit British genomics and genetics research institute [63]. *SAMtools* supports multithreading and is written in *C*. BAM files need to be sorted before being converted back to the FastQ format.

All tools are assessed by processing a BAM file produced of a whole exome sequencing run analyzed by the DoHG@MHH in August 2022 with a size of 8.6 GB (see table 8 for the files statistics). This file is randomly selected and used to benchmark the performance using single- and (if possible) multithreaded (four threads) approaches. Every tool/threading combination is run ten times to ensure a representative result. To verify that the results translate to larger whole genome BAM files, each of the tools is also tested on a file from August 2022 with a size of 58 GB (see table 8 for the files statistics). Given that the results are transferable, only a single run is conducted due to time restraints. The benchmarks are

Artifact Description

run on a virtual machine equipped with 4 CPUs (Intel® Xeon® Gold 6148 processor with 2.40 GHz) and 8 GB RAM.

The main factor relevant for the valuation of the tools is the duration of the conversion. As seen in figure 9, *ngs-bits* is the fastest. Secondary factors, the amount of memory (see figure 10) and data read/written on disk (see figure 11), do favor *ngs-bits* as well (only *biobambam2* uses less memory which is negligible). Due to the already poor performance in multithreading mode, *Picard* was not tested single threaded. Multithreading capabilities do not play a relevant role in the consideration of the tool to be used: multiple single threaded conversions may be run in parallel, depending on the number of cores available on the device used and the number of files to be processed. As *ngs-bits* is already used by *megSAP*, this tool will also not introduce an additional dependency.

As expected, conversions with a genome file took longer, but with similar relative time differences between the tools (see figure 12).

Based on this results, *ngs-bits* is used for conversions of BAM to FastQ files.

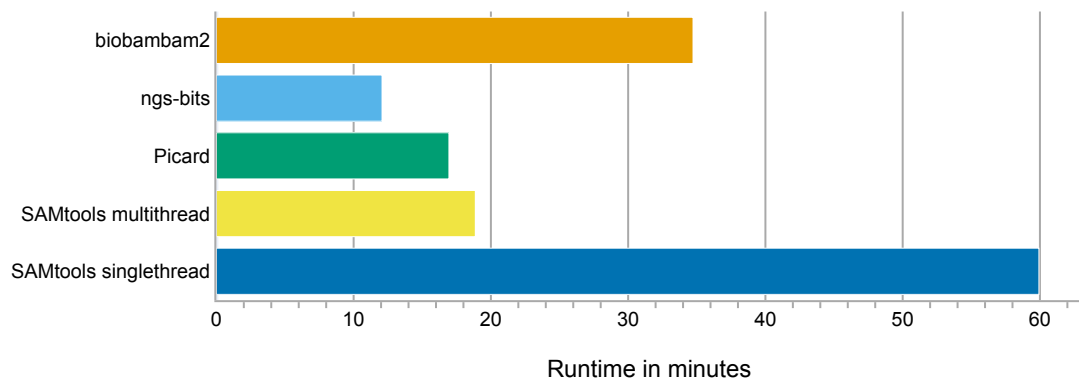


Figure 9: Median duration of BAM to FastQ conversion of whole exome data by tool

Detailed statistics are listed in table 9.

Artifact Description

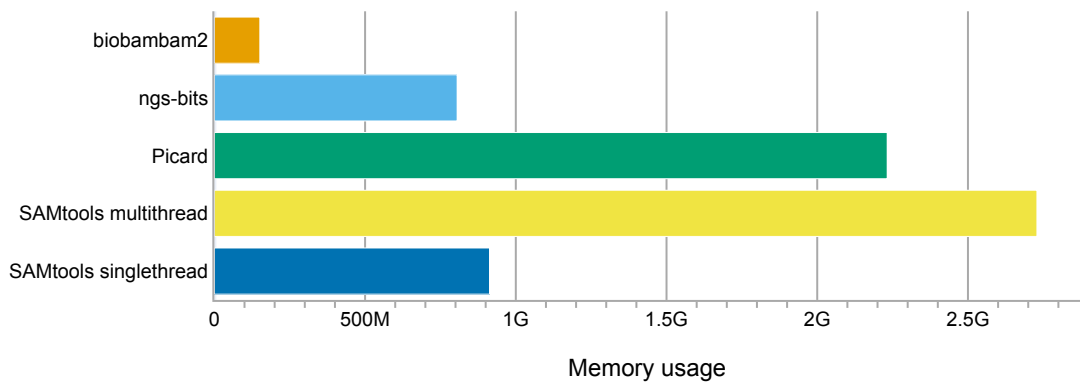


Figure 10: Median memory usage of BAM to FastQ conversion of whole exome data by tool

Detailed statistics are listed in table 10.

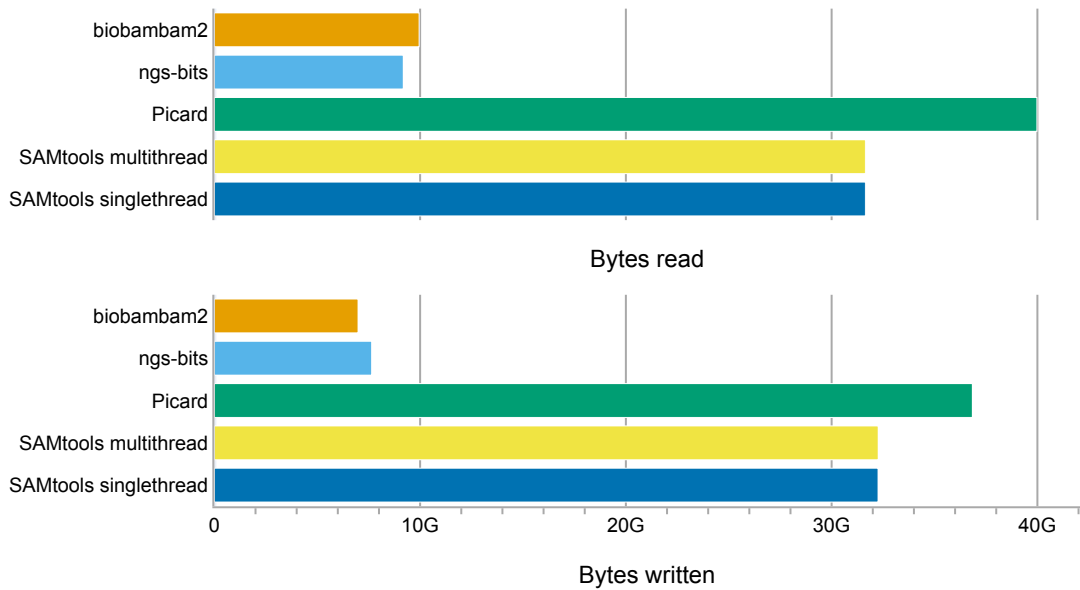


Figure 11: Median amount of data read and written by BAM to FastQ conversion of whole exome data by tool

Detailed statistics are listed in tables 11 and 12.

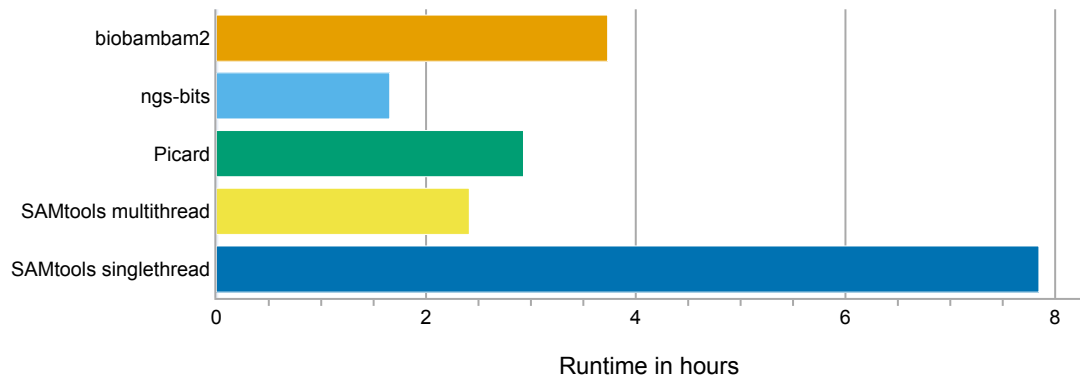


Figure 12: Duration of BAM to FastQ conversion of whole genome data by tool
Detailed statistics are listed in table 13.

4.4.2 Integration of BAM to FastQ Tool into Nextflow Workflow

To add the optional BAM to FastQ conversion to the *Nextflow* workflow, both, the workflow definition (see appendix VI, listing 7) and the configuration (listing 8), need to be modified.

A process definition, `bam2FastQ`, is added to the workflow (line 6–25). The process uses a single input variable, the `sampleDirectory` containing a BAM file as input (line 8). Two output variables are defined: The first one is a path channel for the generated FastQ files emitted as `fastq` (line 11). This ensures that *Nextflow* picks up the files and moves them to the sample directory, as later specified by the `publishDir` directive (line 14). This mechanism was not used before, as *megSAP* outputs all files to the sample directory by itself. The second output variable is the passed through `sampleDirectory` variable, which is emitted through a channel named `sampledir` (line 12). The process script extracts the sample name from the directory path (line 17), and uses the `files()` function to find the first BAM file in the `sampleDirectory` (line 18). Next, the process runs a command that calls the *ngs-bits* `BamToFastq` script (line 20–23), passing the BAM file path and the output files location and names as arguments.

To allow for BAM or FastQ file input for the pipeline, two channels are defined in the workflow part: `bam_channel` and `fastq_channel`. The `bam_channel` is created using the `Channel.fromPath()` function, which takes a path specified in the `params.sampledir` variable, and maps the input directories using a closure that filters the directories that contain at least one BAM file and no FASTQ files (line 40–44). Then the `bam2FastQ` process is applied to the `bam_channel` (line 45). The `fastq_channel` is also created using the `Channel.fromPath()` function, and maps the input directories using a closure that filters the directories that contain no BAM files and at least two FASTQ files (line 47–51). Finally, the `fastq_channel`

Artifact Description

is mixed with the `sampledir` output of the `bam2FastQ` process (the passed-through directory of the processed sample) and used as input to the `megSAP` process (line 53).

The used `mix` operator of the *Nextflow* DSL creates a channel containing all the elements of the original channels, in the order they were presented, as described in [64]. This allows the pipeline to start processing samples already in the FastQ file format to be processed with *megSAP* without waiting for other samples to be converted from the BAM file format.

To use *Nextflow*'s features for staging and moving files as defined by the `bam2FastQ` process, data paths have to be the same in- and outside the container, as noted in [65]. Thus, the `bind paths` definition has been appended with one-to-one mappings of the local file system in the configuration file (line 6). As presented in section 4.4.1, the `BamToFastq` command of *ngs-bits* does not need as many resources as currently defined for the whole workflow. Hence, an exclusion for the `bam2FastQ` process is introduced (line 11–14), limiting the process to 2 CPU cores and 8 GB of memory.

The new additions lead to a slightly more complex DAG, as shown in figure 13. The link from the generated FastQ files to *megSAP* has been added to the diagram manually, as *Nextflow* is not aware of the data flow. The files are simply present in the directory *megSAP* runs in.

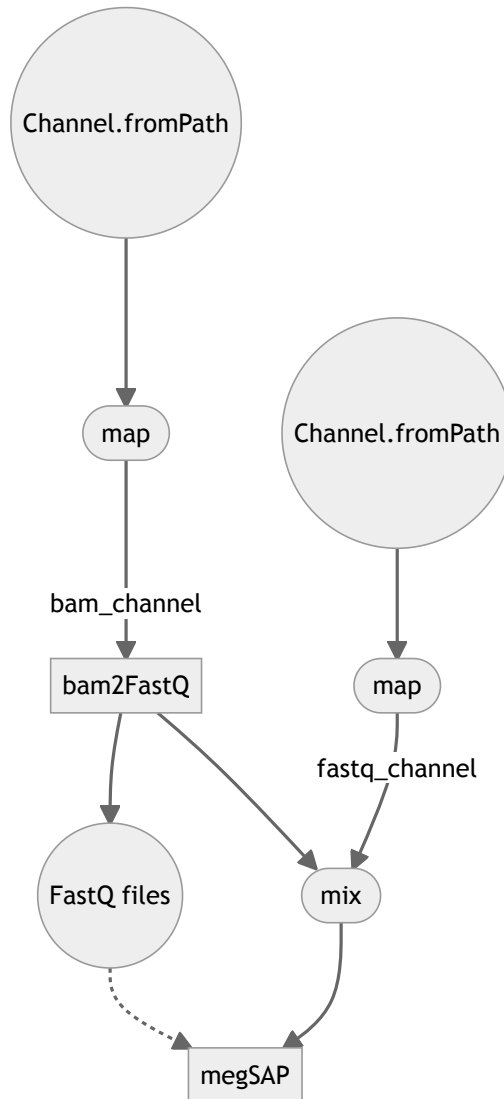


Figure 13: DAG for Nextflow workflow with added BAM to FastQ file conversion

An evaluation of the combined workflow is not conducted, as results can be derived from the previous measurements: a genome analysis will take about 1.65 h longer with BAM to FastQ conversion.

4.5 Separation of megSAP Steps

To optimize *megSAP*'s resource usage, the individual steps —mapping, variant calling, copy number variant calling, structural variant calling, and database import (see figure 3) —need to be separated. After splitting, each step can be measured on its own by *Nextflow*'s build in HTML execution report (see [57]). To accomplish this, the *Nextflow* process *megSAP* is replaced by five separate processes (listing 9, line 27–100): *megSAPma*, *megSAPvc*, *megSAPcn*,

Artifact Description

`megSAPsv`, and `megSAPdb`. These processes are nearly identical to the `megSAP` process in the last iteration, with three changes:

1. The variable `threads` containing the number of threads defined in the *Nextflow* configuration file (e.g., line 39).
2. The variable `steps` defining the steps to be run during the execution of *megSAP* (e.g., line 40). This variable is the only difference between the newly added process blocks.
3. The command to call *megSAP* has been extracted to a separate script (see listing 10) in a template folder. The `template` keyword (e.g., line 41) is used to define a reusable code fragment that can be utilized in multiple places within a workflow. Templates can be parameterized, allowing different values to be passed in each time the template is used, in this case the previous introduced parameters `containerDirectory` and `sampleName` accompanied by the new `threads` and `steps`. Templates provide a way to factor out common parts of a pipeline and simplify pipeline development, making the code more readable and maintainable. By utilizing this part of *Nextflow*'s DSL described in [56, Module templates], this common data processing operation can be shared across all processes calling *megSAP*.

Creating multiple process steps instead of implementing a reusable process with a calling parameter to set the *megSAP* step is deliberate. Resource allocation in the configuration can only be done by providing the process name or adding a label to a process. Both cannot be changed dynamically. So the goal, setting different resource constraints for each step, cannot be reached without creating multiple process blocks.

The separation shows a direct flow between the consecutive steps in the generated DAG, as seen in figure 14.

Artifact Description

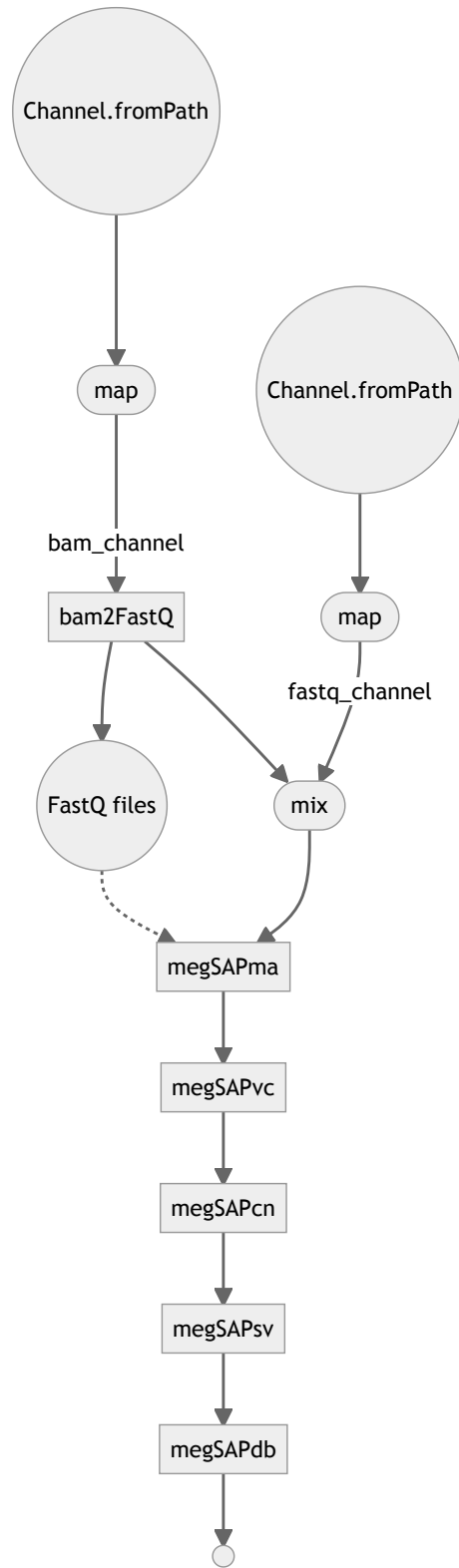


Figure 14: DAG for Nextflow workflow after separation of megSAP steps

Artifact Description

The separated workflow ran for approximately 12.9 h (see table 14). This is a little longer than the run without the split. This slight variance is expected, as other processes are running on the HPC cluster that may have an impact on file read, file write and network performance, and are not controllable. As shown previously in table 3, the standard deviation of pipeline runs in the past was 6.7 h, well above the observed 1.4 h. A runtime comparison between all iterations can be found in section 4.6, figure 19.

The average CPU utilization for each process varies substantially, with the highest utilization being 53.1 % for the `megSAPvc` process and the lowest being 1.7 % for the `megSAPdb` process as shown in figure 15. The peak memory usage also varies significantly among the processes, ranging from 2 % for the `megSAPma` process to 81.9 % for the `megSAPcn` process, as shown in figure 16. A representation of the CPU and memory allocation and usage over time can be seen in figures 17 and 18. Detailed statistics are listed in table 14.

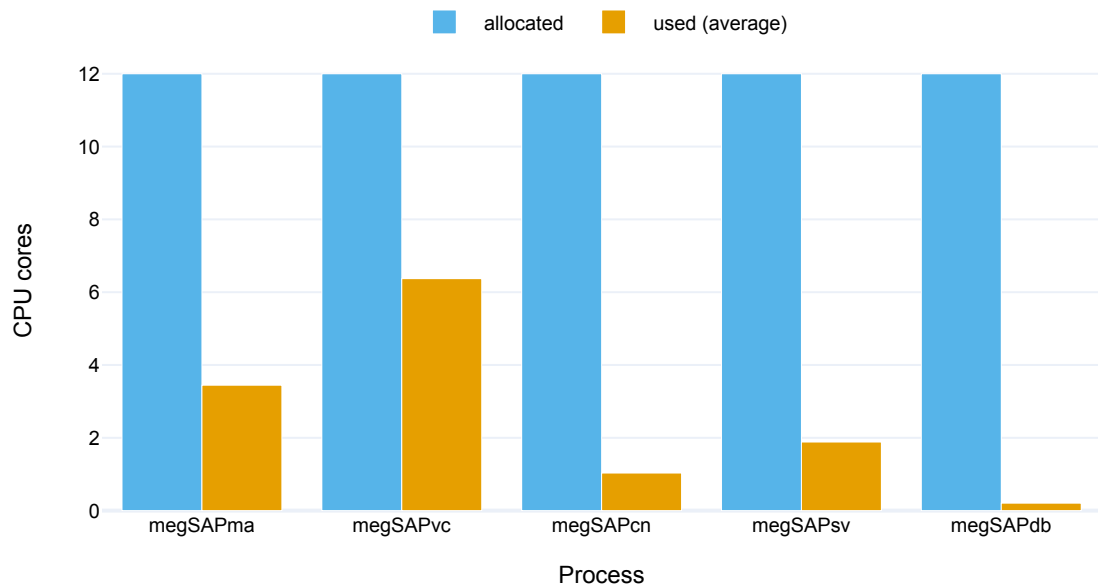


Figure 15: Average CPU usage of the Nextflow workflow after separation of steps

Artifact Description

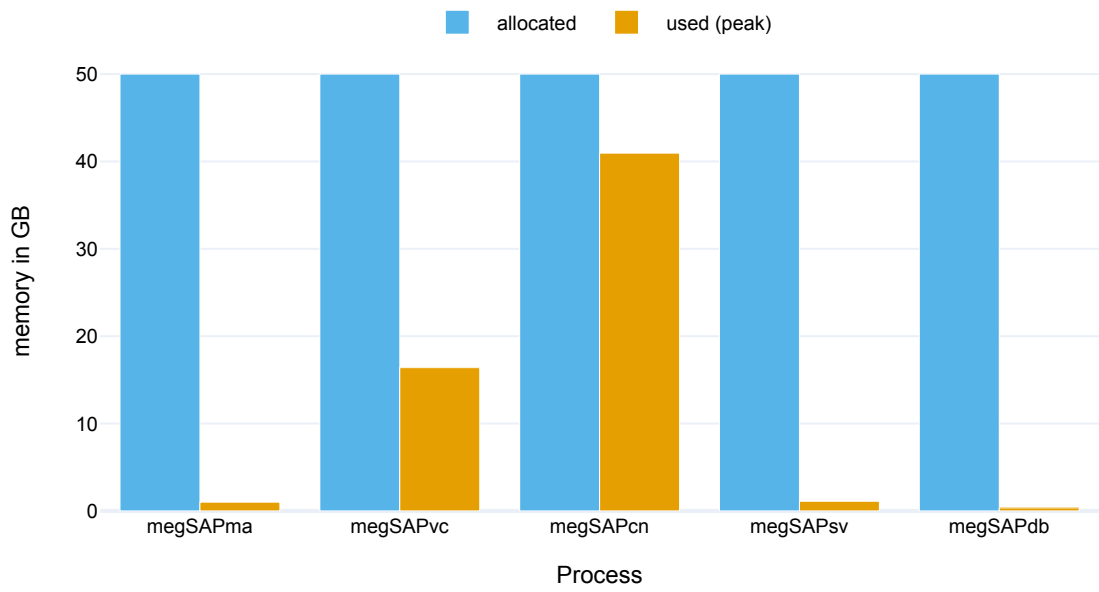


Figure 16: Peak memory usage of the Nextflow workflow after separation of steps

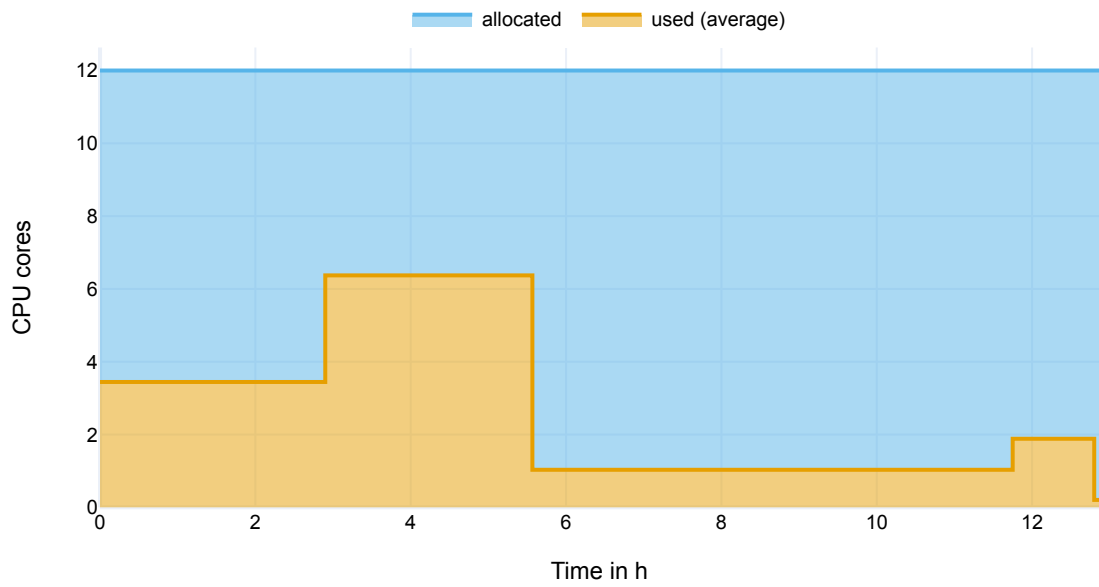


Figure 17: Average CPU usage of the Nextflow workflow over time after separation of steps

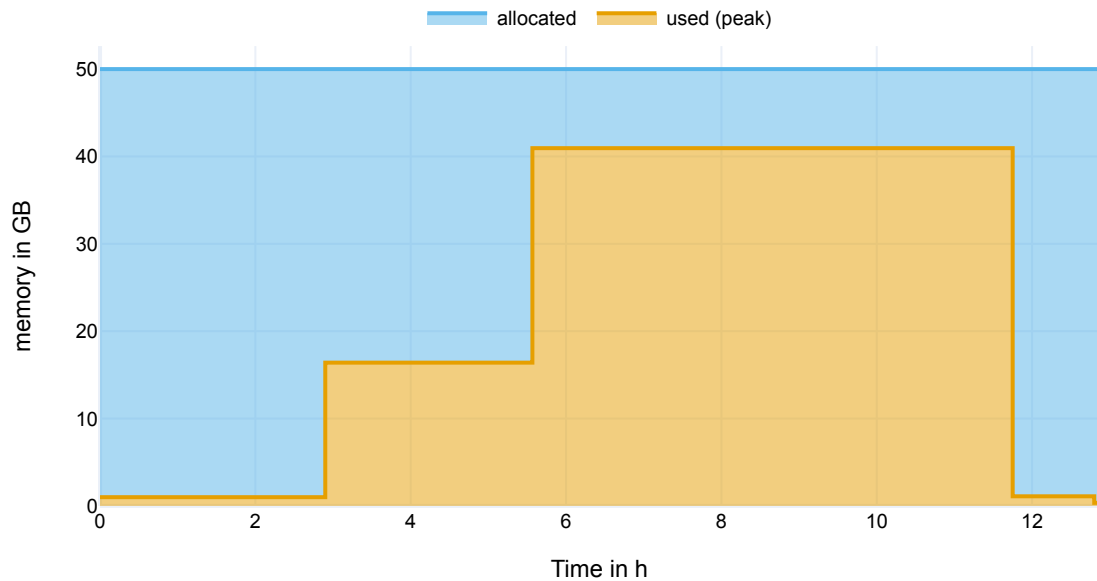


Figure 18: Memory usage of the Nextflow workflow over time after separation of steps

4.6 Resource Optimization of megSAP Steps

Based on the reported resource usage after the separation of the *megSAP* steps, each process is set up with its own resource directives within the *Nextflow* configuration. This is being realized with the `withName` process selector described in [66, Process selectors]. These allow to set specific parameters only for processes with a particular name. The values themselves are derived from the results shown previously in figure 15 and figure 16. This cannot be done as precisely as needed for the CPU requirements, as *Nextflow* measures the weighted average of CPU utilization (see [67]). Therefore, a little buffer is added to the CPU allocation. Additionally, the mapping step (`ma`) has an outstanding characteristic: within this step, *megSAP* utilizes two *Illumina DRAGEN* servers (see [68]) owned by the DoHG@MHH. These operate outside the *SLURM* cluster in their own *Sun Grid Engine* environment. Each can process one sample at a time. The external child process is not monitored by *Nextflow*, as *megSAP* itself is just idly waiting. This dilutes the CPU related measurements of this step, as it also includes the usage of *SeqPurge* to optimize the data before mapping takes place as described in [69]. That part of the mapping step is best run using 15 threads (see appendix IX) resulting in this being set as a configuration parameter. Each process step is also configured with a small buffer of 10% (and rounded up to the next even number) to factor in some fluctuation.

Artifact Description

The resulting configuration values are shown in table 4 and can be seen in the updated configuration in listing 11.

With this configuration, the pipeline ran for 13.6 h. See figure 19 for a comparison between the runtime of all pipeline configurations.

Table 4: Configured resource allocation after optimization

process	megSAPma	megSAPvc	megSAPcn	megSAPsv	megSAPdb
allocated CPUs	15	8	2	2	2
allocated memory	2 GB	20 GB	48 GB	2 GB	2 GB

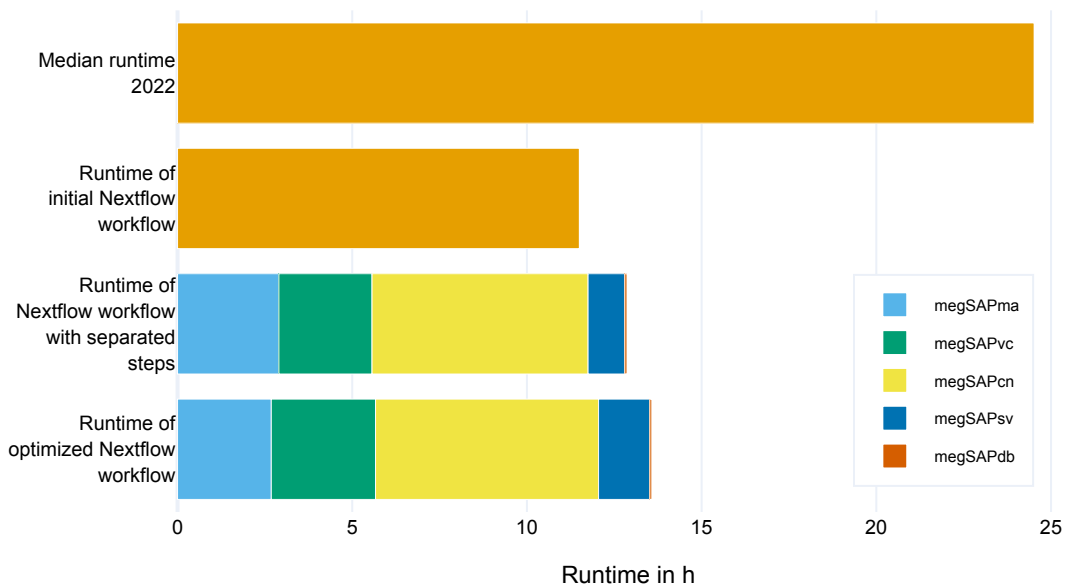


Figure 19: Comparison of pipeline runtime over all iterations

The CPU usage of the `megSAPvc` and `megSAPcn` steps is decreased, although they still do not use all allocated cores efficiently as shown in figure 20. This applies especially to the `megSAPma` step, but was expected, as described before. Memory usage is reduced in the `megSAPvc` and `megSAPcn` steps as well, with the latter not even using half its provided memory resources, as shown in figure 21. A representation of the CPU and memory allocation and usage of the optimized workflow over time can be seen in figures 22 and 23. Detailed statistics are listed in table 15.

Artifact Description

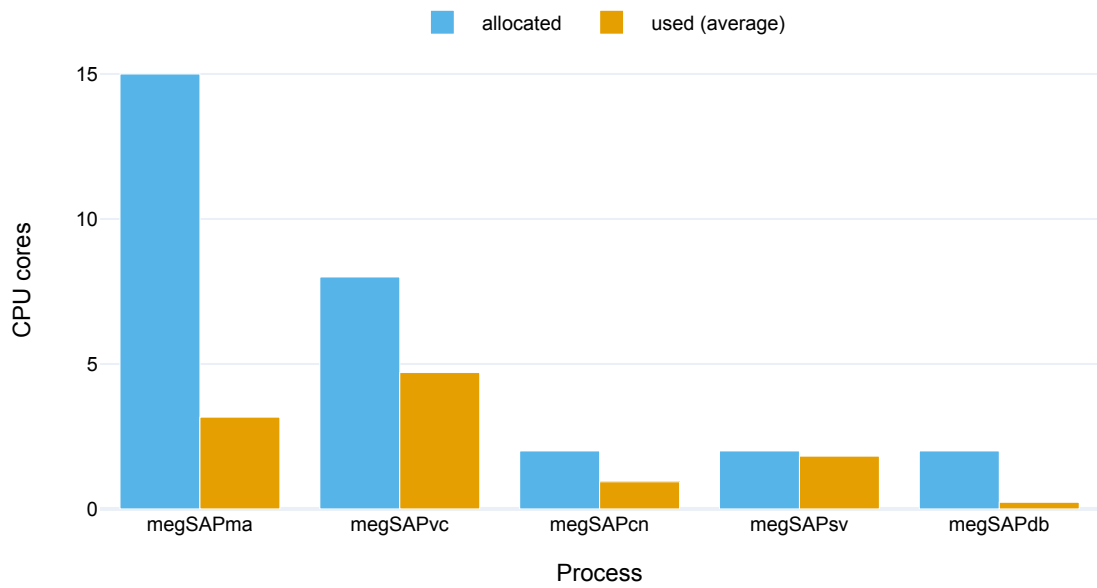


Figure 20: Average CPU usage of the Nextflow workflow after optimization

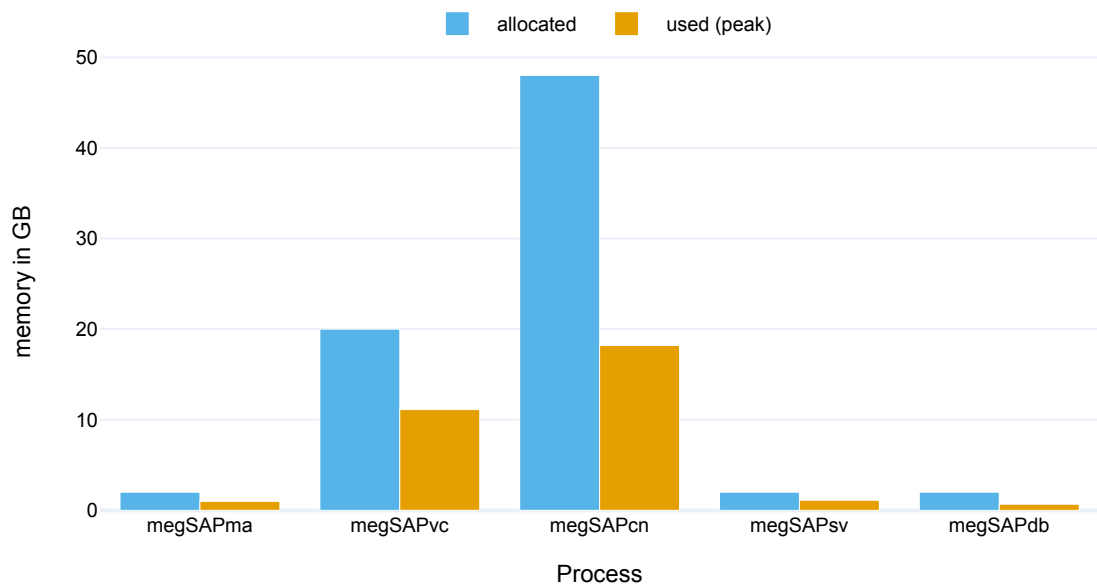


Figure 21: Peak memory usage of the Nextflow workflow after optimization

Artifact Description

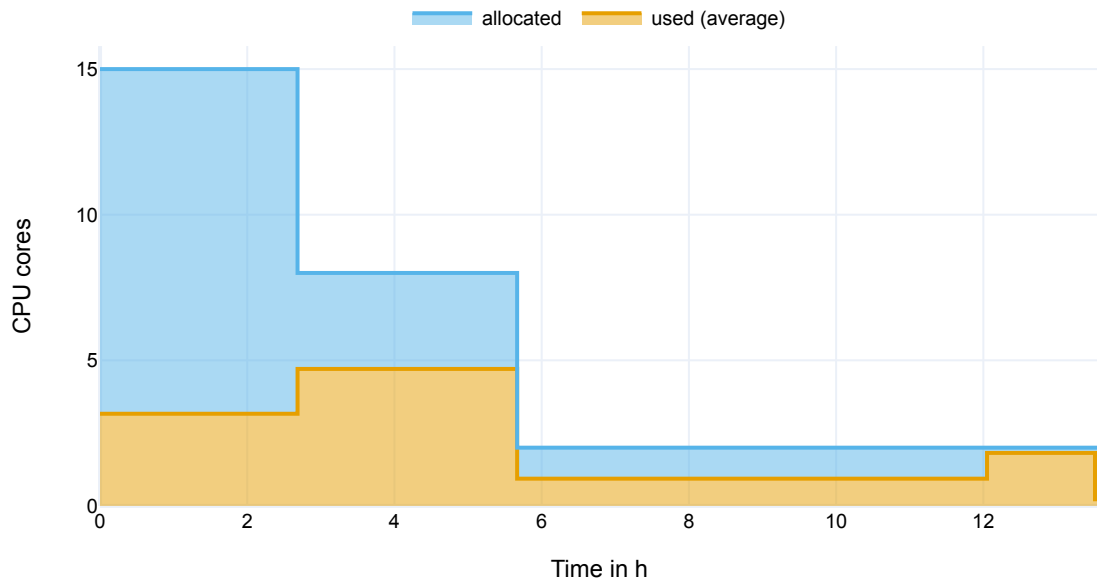


Figure 22: Average CPU usage of the Nextflow workflow after optimization over time

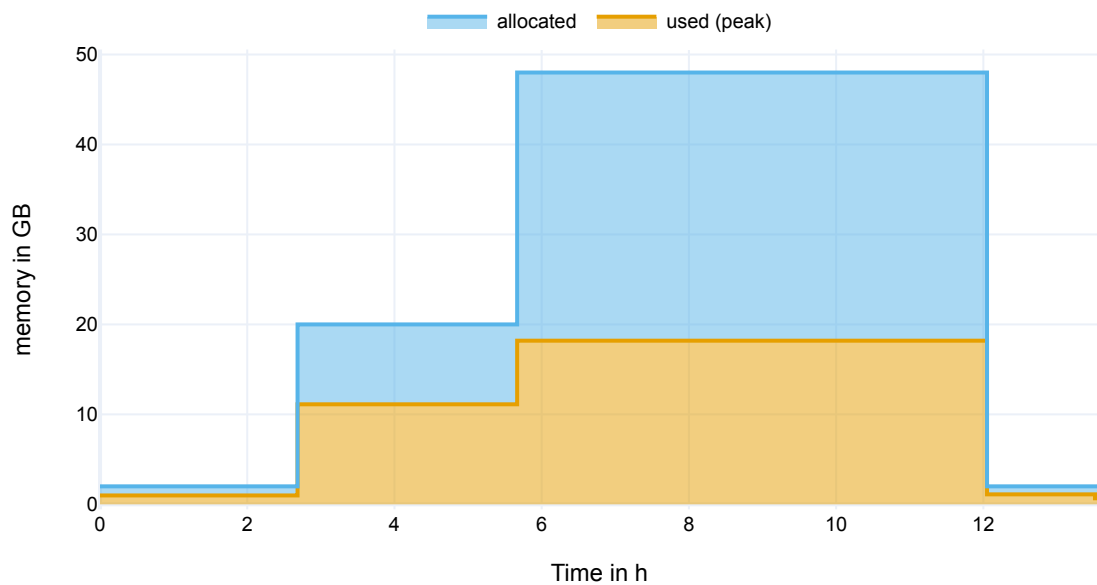


Figure 23: Memory usage of the Nextflow workflow after optimization over time

4.7 Resilience and Monitoring

To make the pipeline less prone to errors, two of *Nextflows* directives for error handling are introduced to the process configuration (see appendix X, listing 12):

Artifact Description

`maxRetries`

This option sets the maximum number of times a task will be re-executed in case of failure (see [70, Directives/maxRetries]). This is set to 2 (line 12).

`errorStrategy`

This directive provides the capability to specify how an error should be handled by the process (see [70, Directives/errorStrategy]). By default, if an error status is returned by the executed script, the process immediately stops, leading to termination of the entire pipeline. This `errorStrategy` is called `terminate`. Other options are:

`finish` Activates a controlled shutdown of the pipeline in response to an error event, while allowing the execution of any submitted tasks to complete.

`ignore` Ignores any execution errors during the process.

`retry` Restarts a process that returns an error condition.

To retry a process step after an error, the `retry` option is chosen. This option falls back to the `terminate` option if the maximum defined number of retries is reached. If multiple samples are analyzed in one execution of the *Nextflow* workflow, this is not desired, as other samples might process without any errors. Thus, a dynamic approach of setting the `errorStrategy` is employed, setting the option to `ignore` in the event of a failed second execution attempt (line 13).

Connecting *Nextflow* to an email server allows for automatic notifications for pipeline completion, whether successful or with errors (see [66, Scope mail]). This is set up in line 63–67.

An evaluation of these changes is not needed, as those neither affect resource consumption nor runtime.

5 Cost Estimation of Potential Cloud Usage

Utilization of cloud computing for the processing of the amount of data to be reanalyzed is a promising approach and a possible solution to the DoHG@MHHs problems as outlined in section 1.2. Among cloud providers, AWS offers specialized *F1* instances through their *EC2* service with FPGA support and pre-installed, official *DRAGEN* software. Thus, AWS was chosen for this approach.

Costs for cloud computing should be calculated in advance to ensure that the DoHG@MHH fully understands the expenses involved and can make an informed decision about whether the benefits of using cloud computing are worth the costs. This includes considering the cost of any additional services or solutions that may be required to address any limitations or challenges posed by cloud computing, such as limited bandwidth.

By using the benchmarking results given by the last iteration of the *Nextflow* implementation, a precise selection of the compute instance is possible regarding the required resource needs of each processing step.

AWS provides a price calculator tool (see [71]) designed to estimate the cost of using AWS services. The calculator considers various factors such as the types of instances used, the number of instances, storage requirements, and data transfer costs to generate a comprehensive estimate of the total cost of using AWS services. The tool provides a detailed breakdown of costs by service, region, and usage.

The price calculation for one sample is done with the following assumptions:

- A sample takes up 100 GB of space (an approximation, *NA12878* takes up 107 GB for example). This consists of two parts: the uploaded data to be analyzed (either in BAM or FastQ file format), and the analyzation results produced by the pipeline (including the BAM file mapped to *GRCh38*). Both consume an equal size of space.
- The lowest possible *EC2 Intel* architecture-based instance configuration fitting to the results obtained in section 4.6 is used.
- The exact usage time measured in section 4.6 is used as well.
- Calculations are done for *AWS*' Frankfurt region because of legal and performance reasons.
- The step `megSAPdb` is left out. This has to be done locally at the DoHG@MHH to use the data in its database. Its resource consumption is negligible, as shown previously.
- Data storage and transfer is calculated based on one sample. Pricing for more storage and data transfer may be cheaper in bulk.

Cost Estimation of Potential Cloud Usage

Price calculations for the use of the specialized *DRAGEN* equipped *F1* instances of *EC2* are not possible with the price calculator tool, but can be done via the product page ([22]). Analyzing *megSAP*'s log files show that the *DRAGEN* part of the mapping step takes approximately one hour. Using the *f1.2xlarge* instance type, this will amount to 12.08 \$. The calculated cost for all other services, including an overview of the selected *EC2* instance types, is detailed in table 5 and amounts to 14.25 \$, bringing the total cost for one sample to 26.33 \$ (if retained on the *S3* storage for a maximum of one month).

Given the number of samples calculated in section 1.1, the total cost to reanalyze in the cloud would sum up to $1544 \text{ samples} \times 26.33 \$ = 40\,653.52 \$$.

To handle the upload of all data currently archived at the DoHG@MHH, an *AWS Snowball* edge storage device may be used instead of straining the relatively slow upload capacity provided. The approximately 77.2 TB (see section 1.1) of data would fit on one device. Assuming the data transfer at the DoHG@MHH can be completed in 10 d, which is feasible, the cost would be 300 \$ (see [72]), with an additional 30 \$ for each additional day.

This removes half of the data transfer costs (only the upload, not the download of the analyzed data), bringing the total down to $40\,653.52 \$ - \left(\frac{4.50 \$}{2} \times 1544\right) + 300 \$ = 37\,479.52 \$$.

Table 5: Results of AWS price calculator tool and selected instance types

Details can be found in appendix XI.

Description/Process	EC2 instance type	CPUs	Memory in GB	Cost in \$
Storage				2.45
Data Transfer				4.50
bam2fastq	t3.large	2	8	0.16
megSAPma	c6i.4xlarge	16	32	2.08
megSAPvc	t3.2xlarge	8	32	1.15
megSAPcn	r5.2xlarge	8	64	3.88
megSAPsv	t3.small	2	2	0.04

6 Discussion

The separation of pipeline steps and their individual resource allocation allows for a more efficient usage of the HPC cluster. Previously unnecessarily reserved threads and memory can now be used to process more samples in parallel. More resources can be utilized in parallel, as shown in section 4.6 and visualized side-by-side in figure 24 and figure 25.

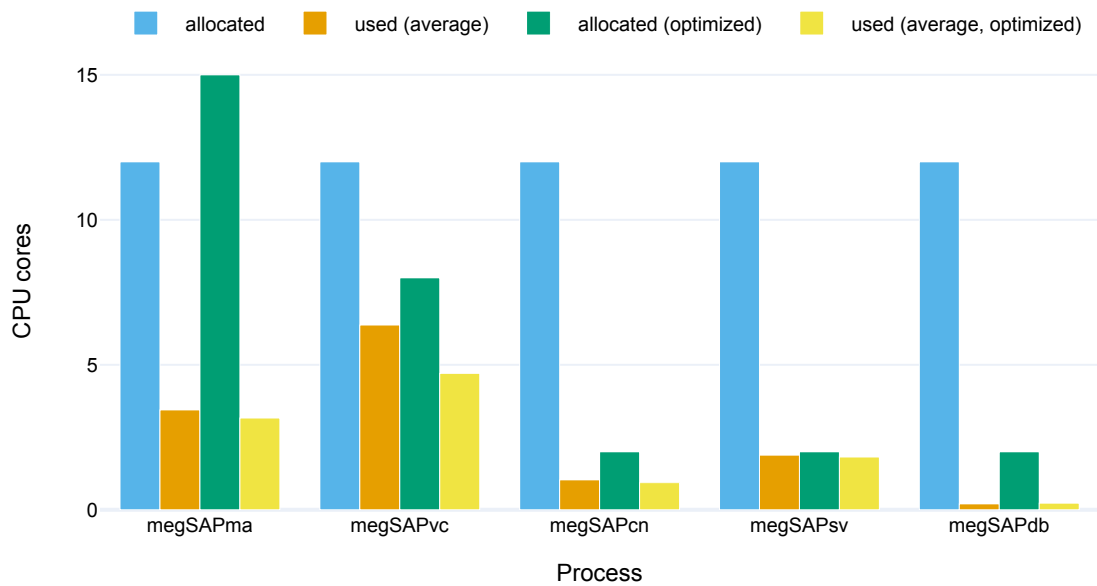


Figure 24: Average CPU usage of the Nextflow workflows compared

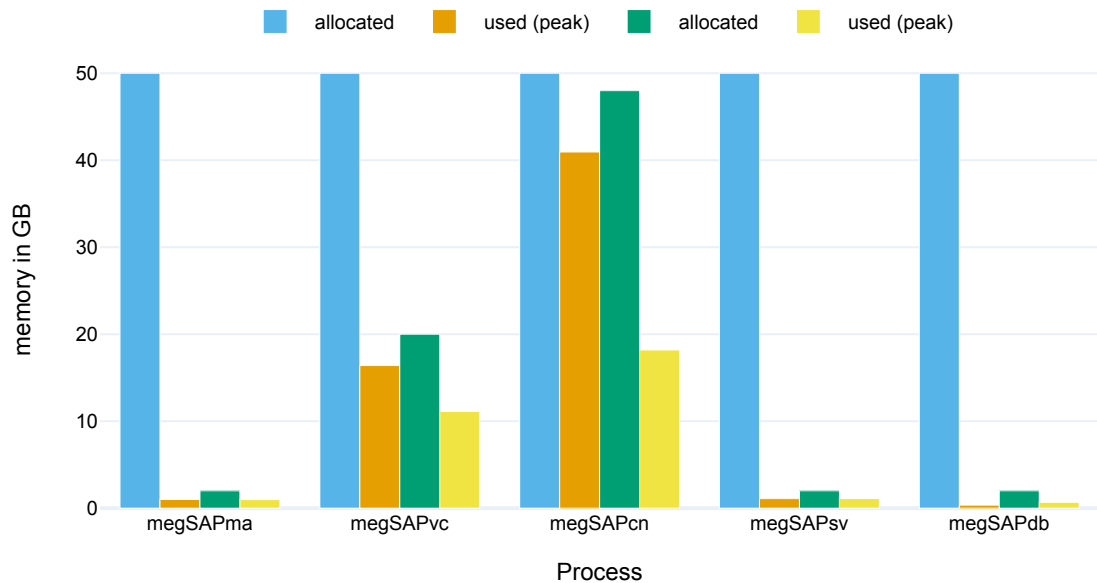


Figure 25: Peak memory usage of the Nextflow workflows compared

Discussion

To quantify the gains, the measuring unit *CPU hours* respectively *memory hours* is calculated for the reserved resources. As seen in equations (1) and (2), this resolves to 138.0 CPU h and 575 GB h for the initial workflow described in section 4.3.

$$12 \text{ CPU} \times 11.5 \text{ h} = 138.0 \text{ CPU h} \quad (1)$$

$$50 \text{ GB} \times 11.5 \text{ h} = 575 \text{ GB h} \quad (2)$$

After optimization, the result is 79.563 CPU h for reserved CPU resources and 374.501 GB h for memory as calculated in equations (3) and (4)

$$\begin{aligned} \text{megSAPma: } & 15 \text{ CPU} \times 2.683 \text{ h} = 40.245 \text{ CPU h} \\ \text{megSAPvc: } & 8 \text{ CPU} \times 2.938 \text{ h} = 23.504 \text{ CPU h} \\ \text{megSAPcn: } & 2 \text{ CPU} \times 6.383 \text{ h} = 12.766 \text{ CPU h} \\ \text{megSAPsv: } & 2 \text{ CPU} \times 1.467 \text{ h} = 2.934 \text{ CPU h} \\ \text{megSAPdb: } & 2 \text{ CPU} \times 0.057 \text{ h} = 0.114 \text{ CPU h} \\ \hline & \text{Optimized workflow} = 79.563 \text{ CPU h} \end{aligned} \quad (3)$$

$$\begin{aligned} \text{megSAPma: } & 2 \text{ GB} \times 2.683 \text{ h} = 5.366 \text{ GB h} \\ \text{megSAPvc: } & 20 \text{ GB} \times 2.938 \text{ h} = 59.76 \text{ GB h} \\ \text{megSAPcn: } & 48 \text{ GB} \times 6.383 \text{ h} = 306.384 \text{ GB h} \\ \text{megSAPsv: } & 2 \text{ GB} \times 1.467 \text{ h} = 2.934 \text{ GB h} \\ \text{megSAPdb: } & 1 \text{ GB} \times 0.057 \text{ h} = 0.057 \text{ GB h} \\ \hline & \text{Optimized workflow} = 374.501 \text{ GB h} \end{aligned} \quad (4)$$

This means that CPU resources are used 42.35 % more efficient than before and memory is used 34.87 % more economically after the optimizations as also shown in figure 26.

A comparison of initial and optimized CPU and memory allocation over time is shown in figures 27 and 28

Discussion

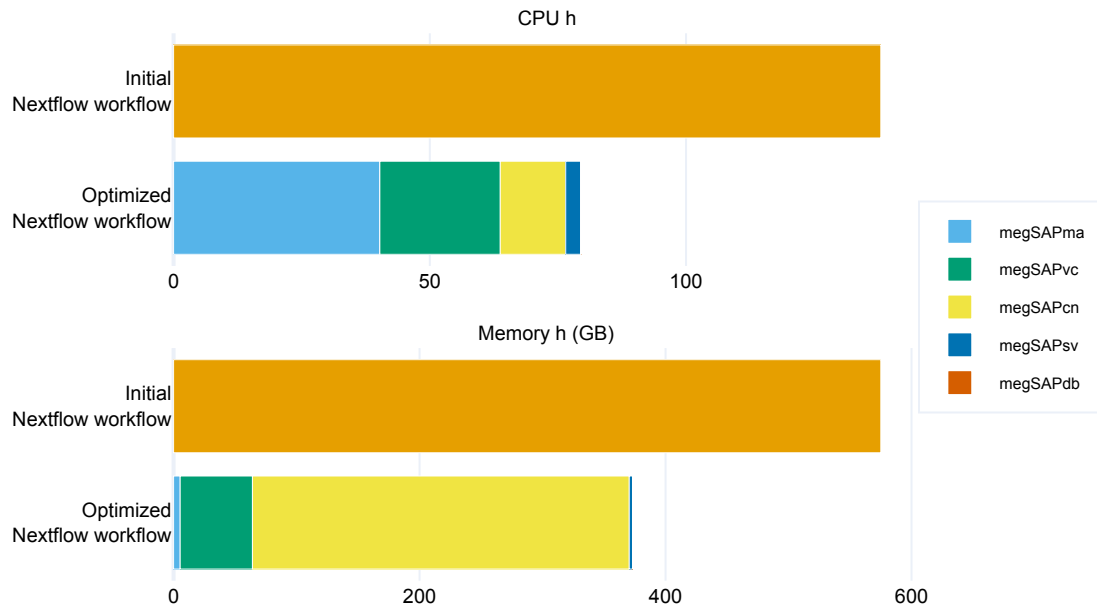


Figure 26: Efficiency after pipeline optimization

Resource consumption of the megSAPdb process is too low to be rendered.

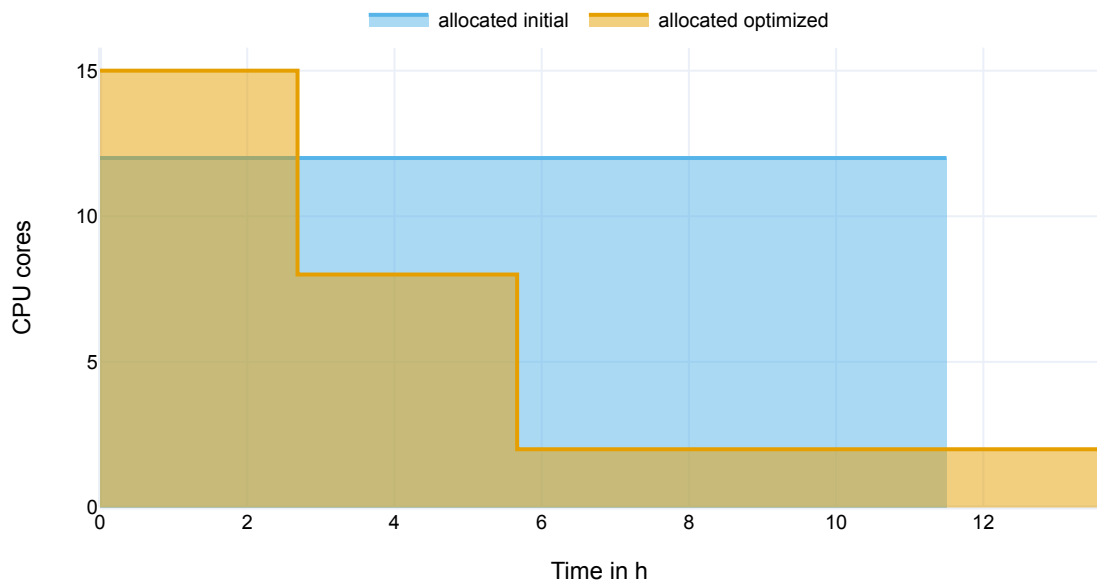


Figure 27: CPU allocation of initial and optimized workflows over time

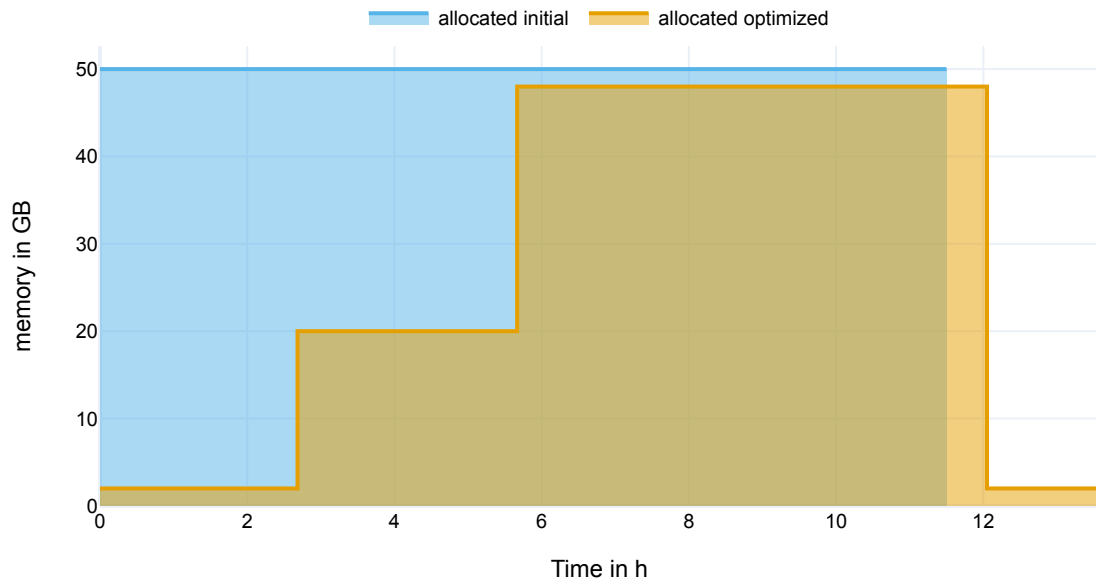


Figure 28: Memory allocation of initial and optimized workflows over time

This can also be transferred to the estimated cost of using *AWS* to compute the pipeline in the cloud. With the optimized pipeline, this amounts to 26.33 \$ as shown in section 5. Using a *m5.4xlarge* instance type to run the pipeline in its previous state (see section 4.3) for 11.5 h would raise this to 29.77 \$ (see appendix XII for details), saving 11.56%. This amount is lower than the realizable retrenchments for the HPC cluster, as *AWS* instance types do not exactly match to the required resources and the overhead produces more cost than necessary.

These resource adjustments should be monitored and improved upon in the future. Slight changes might show improvements on execution time or allow for less resources to be allocated to some steps. A first example might be the `megSAPcn` step. Its memory consumption seems to be affected by the number of threads, as it dropped more than half after reducing the number of available CPUs.

Two different approaches can be taken to optimize the resource usage:

1. Using resources to process a sample as fast as possible. This may be needed to answer a particular time critical diagnostic question.
2. Using resources to process multiple samples simultaneously as efficient as possible. Routine diagnostic profits from an average optimal processing time calculated over multiple samples.

As resource usage and runtime do not correlate linearly, but probably follow more of a sigmoid curve, these two approaches differ. For this thesis, the latter approach has been

taken, as its goal is to process numerous samples retroactively. But the DoHG@MHH sometimes has different needs regarding the speed of diagnostics, e.g., for the *Baby Lion* study (see [73]), which employs rapid whole-genome sequencing as described by Saunders *et al.* [74], the pipeline should run as fast as possible. Handling both approaches may be put into practice for the *Nextflow* pipeline by using different configuration profiles (see [66, Config profiles]) for different needs.

Since the CPU utilization is only calculated as a weighted average, thread allocation is an explorative task. Additionally, *megSAP* combines too many tools into one step to make informed decisions on thread usage. This is particularly visible in the mapping step, as outlined in section 4.5: The resource-intensive task *SeqPurge* is combined with the, at least from a cluster perspective, idling task that waits for the *DRAGEN* mapping to be done. To facilitate more optimization in the future, *megSAP* has to be separated into more individual steps. At best, each tool has its own process in *Nextflow* in the future —this would render all the current *PHP* scripts of *megSAP* obsolete and move all logic to *Nextflow* - a vision worth considering.

Otherwise, the memory utilization is measured by peak usage and the optimization for each step, as presented in section 4.6, can approach this optimum. These optimizations would also benefit from a fragmentation of the *megSAP* steps, as changes could be more precise and not only accommodate for the extremum usage of a whole pipeline step, but for each tool used —in the same manner as the CPU usage optimizations.

One pertinent measurement was not included for this thesis: disk utilization. The HPC cluster does not have tiered storage. All data resides on the same network attached storage. Therefore, no actions could have been derived by any findings. But as the operations performed by the pipeline are using large files to begin with, it might be a promising endeavor to employ faster scratch devices during the pipeline run.

The implementation of *Nextflow* also introduces a remedy for previous problems:

Resilience

By leveraging *Nextflow*'s built-in `maxRetries` directive (see section 4.7) it is possible to cure not reproducible pipeline errors. By simply restarting the relevant step with more resources, these errors can be automatically fixed.

Nevertheless, it might be expedient to check for certain error-codes to react accordingly. For example, an out-of-memory error could be fixed systematically instead of just blindly trying the failed step again with more resources.

Reporting

The email reporting features of *Nextflow* are a first step towards better monitoring

Discussion

of the pipeline. This also applies to the detailed reports generated by each workflow run.

The created *Nextflow* workflow is hosted on the MHH's internal *GitLab* servers. Using *Git* as a version control system allows for tracking changes made to the workflow over time, so it can easily be reverted to previous versions if needed. Hosting the workflows in *GitLab* also provides a backup and helps to ensure that the workflow is not lost in case of any data loss. *GitLab* also makes it easier to share the workflow and to reuse it in different projects.

Two issues outlined in section 1.1 were not fixed by the initial introduction of *Nextflow*: the workflow still has to be triggered by a shell command and no accessible monitoring is possible during the pipeline run. Both of these impediments might be fixable by harnessing the features offered by *Nextflow Tower* (see [75]). It promises enhanced monitoring, logging, and observability for workflows and streamlined deployment of pipelines by providing a easy to use web-based interface. Incorporating *Nextflow Tower* into the newly built *Nextflow* workflow will be the next project at the DoHG@MHH.

7 Conclusion

In conclusion, this bachelor thesis provides a comprehensive insight on the significance of implementing a scientific workflow management system to support the transition to a newer reference genome, in this case *GRCh38*. By leveraging the included resource usage reports of the chosen solution *Nextflow*, it is possible to improve the efficiency of the genetic analysis pipeline greatly, with a reduction in resource usage by over a third.

The adoption of *Nextflow* represents the beginning of a journey for the DoHG@MHH to use their genetic analysis pipeline more professional. This work lays the foundation for further advancements, such as the separation of *megSAP* steps and the addition of *Nextflow Tower*. These suggestions, if implemented, would provide the DoHG@MHH with even greater benefits in terms of efficiency, accessibility, and usability.

One of the main challenges that the DoHG@MHH faces is limited processing capacity, which has been remedied by the implementation of *Nextflow*. In addition, the cost of using cloud infrastructure has also been presented as a possible solution in the future. The use of cloud infrastructure would provide the DoHG@MHH with unlimited processing capacity, and therefore, the ability to perform genetic analyses with as much parallelism as needed. Especially for the task at hand, to reanalyze all old samples with a new reference genome without disturbing regular diagnostics, the cost of such an endeavor might be worth it.

The results of this thesis demonstrate that the implementation of a SWfMS into the genetic analysis pipeline has many benefits, not only in terms of efficiency and accuracy, but also in terms of future scalability. This transition to *GRCh38* has been made possible due to the use of *Nextflow*, and the results will be transferred to the routine medical diagnostics pipeline, as *Nextflow* will be adopted there. The migration will be seamless, with the current scripts being replaced by a script that starts the new *Nextflow* workflow.

Finally, this thesis provides a clear and concise overview of the implementation of a SWfMS to support the transition to a new reference genome. The results demonstrate that the use of *Nextflow* is an effective solution, not only for the DoHG@MHH, but also for any laboratory that needs to transition to a new reference genome. The benefits of using a SWfMS, such as *Nextflow*, are numerous and include improved efficiency, scalability, and cost savings. This thesis serves as a blueprint for the successful implementation of a SWfMS in a genetic analysis pipeline and provides a roadmap for future improvements and advancements.

References

- [1] J. A. Schloss, “How to get genomes at one ten-thousandth the cost,” *Nature Biotechnology*, vol. 26, no. 10, pp. 1113–1115, 2008-10, ISSN: 1546-1696. DOI: 10.1038/nbt1008-1113.
- [2] J. W. Davey, P. A. Hohenlohe, P. D. Etter, J. Q. Boone, J. M. Catchen, and M. L. Blaxter, “Genome-wide genetic marker discovery and genotyping using next-generation sequencing,” *Nature Reviews Genetics*, vol. 12, no. 7, pp. 499–510, 2011-07, ISSN: 1471-0064. DOI: 10.1038/nrg3012.
- [3] S. Behjati and P. S. Tarpey, “What is next generation sequencing?” *Archives of Disease in Childhood - Education and Practice*, vol. 98, no. 6, pp. 236–238, 2013-08. DOI: 10.1136/archdischild-2013-304340.
- [4] K. A. Wetterstrand. “DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP).” (2021), [Online]. Available: <https://www.genome.gov/sequencingcostsdata> (visited on 2022-05-11).
- [5] M. J. Bamshad *et al.*, “Exome Sequencing As a Tool for Mendelian Disease Gene Discovery,” *Nature Reviews Genetics*, vol. 12, no. 11, pp. 745–755, 2011-09. DOI: 10.1038/nrg3031.
- [6] C. G. van El *et al.*, “Whole-genome sequencing in health care,” *European Journal of Human Genetics*, vol. 21, no. 6, pp. 580–584, 2013-05. DOI: 10.1038/ejhg.2013.46.
- [7] M. A. Palladino, *Understanding the Human Genome Project*. Benjamin Cummings, 2002, ISBN: 978-0-8053-6774-4.
- [8] K. Patel. “Get to Know Your Reference Genome (GRCh37 vs GRCh38).” (2018-04), [Online]. Available: <https://bitesizebio.com/38335/get-to-know-your-reference-genome-grch37-vs-grch38/> (visited on 2022-06-22).
- [9] D. Caetano-Anolles. “Reference genome.” (2022-09), [Online]. Available: <https://gatk.broadinstitute.org/hc/en-us/articles/360035891071-Reference-genome> (visited on 2022-10-12).
- [10] F. S. Collins, M. Morgan, and A. Patrinos, “The Human Genome Project: Lessons from Large-Scale Biology,” *Science*, vol. 300, no. 5617, pp. 286–290, 2003-04-11. DOI: 10.1126/science.1084564.
- [11] J. Lilue *et al.*, “Sixteen diverse laboratory mouse reference genomes define strain-specific haplotypes and novel functional loci,” *Nature Genetics*, vol. 50, no. 11, pp. 1574–1583, 11 2018-11, ISSN: 1546-1718. DOI: 10.1038/s41588-018-0223-8.

References

- [12] P. L. Morrell, E. S. Buckler, and J. Ross-Ibarra, “Crop genomics: Advances and applications,” *Nature Reviews Genetics*, vol. 13, no. 2, pp. 85–96, 2012-02, ISSN: 1471-0064. DOI: 10.1038/nrg3097.
- [13] National Library of Medicine. “GRCh37.” (2009), [Online]. Available: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13/ (visited on 2022-05-18).
- [14] Illumina, Inc. “NovaSeq 6000 System.” (2022), [Online]. Available: <https://www.illumina.com/systems/sequencing-platforms/novaseq.html> (visited on 2022-10-11).
- [15] R. A. Holt and S. J. M. Jones, “The new paradigm of flow cell sequencing,” *Genome Research*, vol. 18, no. 6, pp. 839–846, 2008-06, ISSN: 1088-9051, 1549-5469. DOI: 10.1101/gr.073262.107.
- [16] H. Li, J. Ruan, and R. Durbin, “Mapping short DNA sequencing reads and calling variants using mapping quality scores,” *Genome Research*, vol. 18, no. 11, pp. 1851–1858, 2008-11, ISSN: 1088-9051, 1549-5469. DOI: 10.1101/gr.078212.108.
- [17] C. W. Fuller *et al.*, “The challenges of sequencing by synthesis,” *Nature Biotechnology*, vol. 27, no. 11, pp. 1013–1023, 2009-11, ISSN: 1546-1696. DOI: 10.1038/nbt.1585.
- [18] National Library of Medicine. “GRCh38.” (2013), [Online]. Available: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/ (visited on 2022-05-18).
- [19] M. Sturm. “GitHub: Merging GRCh38 branch into master.” (2021), [Online]. Available: <https://github.com/imgag/megSAP/pull/102> (visited on 2022-05-17).
- [20] H. Li *et al.*, “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009-08, ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btp352.
- [21] Amazon Web Services. “Amazon EC2 F1-Instances.” (2022), [Online]. Available: <https://aws.amazon.com/de/ec2/instance-types/f1/> (visited on 2022-05-17).
- [22] Amazon Web Services. “AWS Marketplace: DRAGEN Complete Suite Pricing.” (2023), [Online]. Available: <https://aws.amazon.com/marketplace/pp/prodview-ypz2tpzy6f5xq> (visited on 2023-02-05).
- [23] Amazon Web Services. “AWS Snowball Edge Device Specifications.” (2022), [Online]. Available: <https://docs.aws.amazon.com/snowball/latest/developer-guide/sbe-specifications.html> (visited on 2022-10-25).
- [24] V. A. Schneider *et al.*, “Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly,” *Genome Research*, vol. 27, no. 5, pp. 849–864, 2017-05, ISSN: 1088-9051, 1549-5469. DOI: 10.1101/gr.213611.116.

References

- [25] Y. Guo, Y. Dai, H. Yu, S. Zhao, D. C. Samuels, and Y. Shyr, “Improvements and impacts of GRCh38 human reference on high throughput sequencing data analysis,” *Genomics*, vol. 109, no. 2, pp. 83–90, 2017-03, ISSN: 0888-7543. DOI: 10.1016/j.ygeno.2017.01.005.
- [26] B. Pan *et al.*, “Similarities and differences between variants called with human reference genome HG19 or HG38,” *BMC Bioinformatics*, vol. 20, no. 2, p. 101, 2019-03, ISSN: 1471-2105. DOI: 10.1186/s12859-019-2620-0.
- [27] E. S. Lander *et al.*, “Initial sequencing and analysis of the human genome,” *Nature*, vol. 409, no. 6822, pp. 860–921, 2001-02, ISSN: 1476-4687. DOI: 10.1038/35057062.
- [28] J. C. Venter *et al.*, “The Sequence of the Human Genome,” *Science*, vol. 291, no. 5507, pp. 1304–1351, 2001-02. DOI: 10.1126/science.1058040.
- [29] E. R. Mardis, “Next-generation DNA sequencing methods,” *Annual Review of Genomics and Human Genetics*, vol. 9, pp. 387–402, 2008, ISSN: 1527-8204. DOI: 10.1146/annurev.genom.9.081307.164359.
- [30] D. R. Bentley *et al.*, “Accurate whole human genome sequencing using reversible terminator chemistry,” *Nature*, vol. 456, no. 7218, pp. 53–59, 2008-11, ISSN: 1476-4687. DOI: 10.1038/nature07517.
- [31] J. M. Zook *et al.*, “Extensive sequencing of seven human genomes to characterize benchmark reference materials,” *Scientific Data*, vol. 3, no. 1, p. 160 025, 2016-06, ISSN: 2052-4463. DOI: 10.1038/sdata.2016.25.
- [32] G. Baid *et al.*, “An Extensive Sequence Dataset of Gold-Standard Samples for Benchmarking and Development,” p. 2020.12.11.422022, 2020-12. DOI: 10.1101/2020.12.11.422022.
- [33] P. Krusche *et al.*, “Best practices for benchmarking germline small-variant calls in human genomes,” *Nature Biotechnology*, vol. 37, no. 5, pp. 555–560, 2019-05, ISSN: 1546-1696. DOI: 10.1038/s41587-019-0054-x.
- [34] E. R. Mardis, “The impact of next-generation sequencing technology on genetics,” *Trends in genetics: TIG*, vol. 24, no. 3, pp. 133–141, 2008-03, ISSN: 0168-9525. DOI: 10.1016/j.tig.2007.12.007.
- [35] J. Shendure and H. Ji, “Next-generation DNA sequencing,” *Nature Biotechnology*, vol. 26, no. 10, pp. 1135–1145, 2008-10, ISSN: 1546-1696. DOI: 10.1038/nbt1486.
- [36] K. V. Voelkerding, S. A. Dames, and J. D. Durtschi, “Next-Generation Sequencing: From Basic Research to Diagnostics,” *Clinical Chemistry*, vol. 55, no. 4, pp. 641–658, 2009-04. DOI: 10.1373/clinchem.2008.112789.

References

- [37] M. L. Metzker, “Sequencing technologies — the next generation,” *Nature Reviews Genetics*, vol. 11, no. 1, pp. 31–46, 2010-01, ISSN: 1471-0064. DOI: 10.1038/nrg2626.
- [38] D. Georgakopoulos, M. Hornick, and A. Sheth, “An overview of workflow management: From process modeling to workflow automation infrastructure,” *Distributed and Parallel Databases*, vol. 3, no. 2, pp. 119–153, 1995-04, ISSN: 1573-7578. DOI: 10.1007/BF01277643.
- [39] W. van der Aalst and K. van Hee, *Workflow Management: Models, Methods, and Systems*. MIT Press, 2004, ISBN: 978-0-262-72046-5.
- [40] B. Ludäscher *et al.*, “Scientific workflow management and the Kepler system,” *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006, ISSN: 1532-0634. DOI: 10.1002/cpe.994.
- [41] T. Oinn *et al.*, “Taverna: A tool for the composition and enactment of bioinformatics workflows,” *Bioinformatics (Oxford, England)*, vol. 20, no. 17, pp. 3045–3054, 2004-11, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth361.
- [42] B. Giardine *et al.*, “Galaxy: A platform for interactive large-scale genome analysis,” *Genome Research*, vol. 15, no. 10, pp. 1451–1455, 2005-10, ISSN: 1088-9051. DOI: 10.1101/gr.4086505.
- [43] J. Köster and S. Rahmann, “Snakemake—a scalable bioinformatics workflow engine,” *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, 2012-10, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts480.
- [44] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, “Nextflow enables reproducible computational workflows,” *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, 2017-04, ISSN: 1087-0156, 1546-1696. DOI: 10.1038/nbt.3820.
- [45] A. Poholek, A. Sun, Z. Fang, N. Rittenhouse, R. Sethi, and R. Mallela. “Workflow Management Strategy Discussion with a Group of 25 Computational Biologists and Data Scientists.” (2017), [Online]. Available: <https://github.com/NCBI-Hackathons/SPeW#workflow-management-strategy-discussion-with-a-group-of-25-computational-biologists-and-data-scientists> (visited on 2022-05-17).
- [46] E. Larssonneur, J. Mercier, N. Wiart, E. L. Floch, O. Delhomme, and V. Meyer, “Evaluating Workflow Management Systems: A Bioinformatics Use Case,” in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2018-12. DOI: 10.1109/bibm.2018.8621141.

References

- [47] M. Jackson, K. Kavoussanakis, and E. W. J. Wallace, “Using prototyping to choose a bioinformatics workflow management system,” *PLOS Computational Biology*, vol. 17, no. 2, F. Ouellette, Ed., e1008622, 2021-02. DOI: 10.1371/journal.pcbi.1008622.
- [48] L. Wratten, A. Wilm, and J. Göke, “Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers,” *Nature methods*, 2021. DOI: 10.1038/s41592-021-01254-9.
- [49] P. A. Ewels *et al.*, “The nf-Core framework for community-curated bioinformatics pipelines,” *Nature Biotechnology*, 2020. DOI: 10.1038/s41587-020-0439-x.
- [50] A. E. Ahmed *et al.*, “Design considerations for workflow management systems use in production genomics research and the clinic,” *Scientific Reports*, vol. 11, no. 1, 2021-11. DOI: 10.1038/s41598-021-99288-8.
- [51] H. A. Simon, “The Science of Design: Creating the Artificial,” *Design Issues*, vol. 4, no. 1/2, p. 67, 1988. DOI: 10.2307/1511391.
- [52] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, vol. 28, no. 1, p. 75, 2004. DOI: 10.2307/25148625.
- [53] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007-12. DOI: 10.2753/mis0742-1222240302.
- [54] S. Gregor and A. R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, 2013, ISSN: 02767783. DOI: 10.25300/MISQ/2013/37.2.01.
- [55] IEEE and The Open Group. “The Open Group Base Specifications Issue 7 - mkdir.” (2018), [Online]. Available: <https://pubs.opengroup.org/onlinepubs/9699919799/utilities/mkdir.html> (visited on 2023-01-09).
- [56] Seqera Labs. “DSL 2 — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/dsl2.html> (visited on 2023-01-16).
- [57] Seqera Labs. “Tracing & visualisation — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/tracing.html> (visited on 2023-01-17).
- [58] D. L. Parnas, “On the criteria to be used in decomposing systems into modules,” *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972-12, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/361598.361623.

References

- [59] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, “The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants,” *Nucleic Acids Research*, vol. 38, no. 6, pp. 1767–1771, 2009-12. DOI: 10.1093/nar/gkp1137.
- [60] G. Tischler and S. Leonard, “Biobambam: Tools for read pair collation based algorithms on BAM files,” *Source Code for Biology and Medicine*, vol. 9, p. 13, 2014-06. DOI: 10.1186/1751-0473-9-13.
- [61] M. Sturm, C. Schroeder, J. Matthes, and S. Ossowski. “Ngs-bits Short-read sequencing tools for diagnostics.” (2018), [Online]. Available: https://github.com/imgag/ngs-bits/blob/master/doc/data/poster_ECCB2018.pdf (visited on 2022-09-16).
- [62] Broad Institute, *Picard toolkit*, 2019. [Online]. Available: <https://broadinstitute.github.io/picard/> (visited on 2022-09-16).
- [63] P. Danecek *et al.*, “Twelve years of SAMtools and BCFtools,” *GigaScience*, vol. 10, no. 2, giab008, 2021-02, ISSN: 2047-217X. DOI: 10.1093/gigascience/giab008.
- [64] Seqera Labs. “Operators#mix — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/operator.html#mix> (visited on 2023-01-19).
- [65] Seqera Labs. “Containers#Singularity — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/container.html#container-singularity> (visited on 2023-01-19).
- [66] Seqera Labs. “Configuration — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/config.html#> (visited on 2023-02-02).
- [67] Seqera Labs. “Metrics — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/metrics.html> (visited on 2023-01-31).
- [68] Illumina, Inc. “Illumina DRAGEN Bio-IT Platform.” (2022), [Online]. Available: <https://emea.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html> (visited on 2022-05-16).
- [69] M. Sturm, C. Schroeder, and P. Bauer, “SeqPurge: Highly-sensitive adapter trimming for paired-end NGS data,” *BMC bioinformatics*, vol. 17, p. 208, 2016-05-10, ISSN: 1471-2105. DOI: 10.1186/s12859-016-1069-7. pmid: 27161244.
- [70] Seqera Labs. “Processes — Nextflow 22.10.4 documentation.” (2022), [Online]. Available: <https://www.nextflow.io/docs/latest/process.html> (visited on 2023-02-05).

References

- [71] Amazon Web Services. “AWS Pricing Calculator.” (2023), [Online]. Available: <https://docs.aws.amazon.com/pricing-calculator/latest/userguide/what-is-pricing-calculator.html> (visited on 2023-02-01).
- [72] Amazon Web Services. “AWS Snowball Pricing | Amazon Web Services.” (), [Online]. Available: <https://aws.amazon.com/snowball/pricing/> (visited on 2023-02-05).
- [73] Medizinische Hochschule Hannover. “Rapid diagnosis instead of a long odyssey.” (2022-04-25), [Online]. Available: <https://www.mhh.de/en/presse/mhh-insight/news-detailed-view/rapid-diagnosis-instead-of-a-long-odyssey> (visited on 2023-02-02).
- [74] C. J. Saunders *et al.*, “Rapid Whole-Genome Sequencing for Genetic Disease Diagnosis in Neonatal Intensive Care Units,” *Science Translational Medicine*, vol. 4, no. 154, 154ra135–154ra135, 2012-10-03. DOI: 10.1126/scitranslmed.3004041.
- [75] Seqera Labs. “Nextflow Tower - Introduction.” (2022), [Online]. Available: <https://help.tower.nf/22.3/> (visited on 2023-02-06).
- [76] NumPy Developers. “Numpy Polynomial.fit.” (2022), [Online]. Available: <https://numpy.org/doc/stable/reference/generated/numpy.polynomial.polynomial.Polynomial.fit.html> (visited on 2022-05-18).
- [77] Genome Research Limited. “Samtools-flagstat manual page.” (2022-08), [Online]. Available: <http://www.htslib.org/doc/samtools-flagstat.html> (visited on 2022-10-06).

Appendix

I Sum of Processed Samples

The sum of processed samples has been generated by a query to the database (see Listing 1) of the DoHG@MHH's laboratory information management system (LIMS).

Table 6: Sum of processed samples at the DoHG@MHH per year

Year	Sum of processed samples
2016	37
2017	292
2018	466
2019	661
2020	1460
2021	1794

Listing 1: SQL for sum of processed samples

```
1: FROM
2:   av_analytik AS AN
3:   INNER JOIN av_status_stamm AS ANS ON ANS.guid_status_stamm = AN.
   guid_status_stamm
4:   AND ANS.status_bezeichnung NOT LIKE 'ungültig'
5: WHERE
6:   AN.guid_analytik_stamm IN (
7:     -- HGE_Exome
8:     '134b1aa2-cedb-400d-8f5d-588a3be669f9',
9:     -- HGE_Genom
10:    '0547c8a1-e48c-4050-9ee5-307ac032c666',
11:    -- HGE_rapidGenom
12:    '0ab57b84-4cf6-4383-927c-7143b81e37ba',
13:    -- FGEN_Exom
14:    '7be0cede-ecd2-4932-b53e-63133bf658f6',
15:    -- FGEN_Genom
16:    'fff951e3-9f5a-40a8-bb6f-888cdc24de7a'
17:  )
18: GROUP BY YEAR(AN.datum)
19: ORDER BY YEAR(AN.datum) ASC
```

II Extrapolation of Whole-Genome Samples Analyzed Until Project Deadline

Data for the first four months of the year 2022 has been generated by a query to the database (Listing 2) and extrapolated by calculating the least squares fit (see [76]) with a python script (Listing 3).

Listing 2: SQL query to gather data for extrapolation

```

1: SELECT AN.datum AS 'Date'
2: FROM av_analytik AS AN
3:     INNER JOIN av_status_stamm AS ANS ON ANS.guid_status_stamm = AN.
   guid_status_stamm
4:     AND ANS.status_bezeichnung NOT LIKE 'ungültig'
5: WHERE AN.guid_analytik_stamm IN (
6:     -- HGE_Genom
7:     '0547c8a1-e48c-4050-9ee5-307ac032c666',
8:     -- HGE_rapidGenom
9:     '0ab57b84-4cf6-4383-927c-7143b81e37ba',
10:    -- FGEN_Genom
11:    'fff951e3-9f5a-40a8-bb6f-888cdc24de7a'
12: )
13: AND AN.datum BETWEEN '01.01.2022' AND '31.12.2022'
14: ORDER BY AN.datum ASC

```

Listing 3: Python code to extrapolate the sum of whole-genome sequencing runs until the project's deadline

```

1: import datetime
2:
3: import numpy as np
4: import pandas as pd
5: from numpy.polynomial import Polynomial
6:
7: # read csv from sql result
8: df_whole_genomes = pd.read_csv(
9:     "./whole-genome-sequencings.csv",
10:    parse_dates=["Date"],
11: )
12:
13: # group by week and count rows
14: df_whole_genomes = (
15:     df_whole_genomes.groupby(pd.Grouper(freq="W-MON", key="Date"))
16:     .size()
17:     .reset_index(level=0)
18: )

```

Appendix

```
19: df_whole_genomes = df_whole_genomes.rename(columns={0: "Count"})
20:
21: # convert dates to numpy float for polynomial fit
22: whole_genome_numeric_dates = np.array(
23:     df_whole_genomes["Date"].dt.isocalendar().week.values, dtype="float64
24:     "
25: )
26: # do polynomial fitting for current data
27: whole_genomes_polyfit = Polynomial.fit(
28:     whole_genome_numeric_dates, df_whole_genomes["Count"], 1
29: )
30:
31: # add extrapolated data for missing weeks to project deadline
32: currentdate = df_whole_genomes["Date"].max() + datetime.timedelta(days=7)
33: while currentdate < datetime.datetime(2023, 3, 31):
34:     currentweek = currentdate.isocalendar().week
35:     additional_row = pd.DataFrame(
36:         [[currentdate, whole_genomes_polyfit(currentweek)]],
37:         columns=["Date", "Count"],
38:         index=[currentweek - 1],
39:     )
40:     df_whole_genomes = pd.concat([df_whole_genomes, additional_row])
41:     currentdate += datetime.timedelta(days=7)
42:
43: # get sum of all weekly data
44: print(df_whole_genomes["Count"].sum())
```

III Example Script of Pipeline Invocation with Bash

Listing 4: bash script to trigger the megSAP pipeline

```

1: #!/bin/bash
2: #
3: # Script for analyzing DNA samples (analyze from megSAP on singularity
  container), Winfried Hofmann, Department of Human Genetics, 7/2019
4: #
5: PWD=$(which pwd)
6: MKDIR=$(which mkdir)
7: DATE=$(date "+%Y-%m-%d_%s")
8: #
9: # working directory within the container
10: WDIR=$(PWD)
11: WDIR_SINGULARITY=$(echo "$WDIR" | sed 's!/mnt/hgenet!!g!')
12: WDIR_SLURM=/mnt/hgenet/NGS_Daten_nfs/slurm
13: #
14: cd $WDIR
15: SAMPLEDIRARRAY=$(ls | grep Sample_)
16: NTASKS=$(echo ${#SAMPLEDIRARRAY[@]})
17: #
18: # subdirectory with the current date is created when not existent
19: if [ ! -d "$WDIR_SLURM/$DATE" ]; then
20:     $MKDIR "$WDIR_SLURM/$DATE";
21: fi
22: #
23: # for each sample, a bash file is created
24: for SAMPLEDIR in ${SAMPLEDIRARRAY[*]};do
25:     SAMPLEID=$(echo "$SAMPLEDIR" | sed -e 's/.*Sample_//g')
26:     echo "$SAMPLEID"
27:     echo "#!/bin/bash
28: #SBATCH -J $SAMPLEID
29: #SBATCH -p lowprio
30: #SBATCH --exclude=hpc-rc12,hpc-rc13,hpc-bc15-[01,03,06,14]
31: #SBATCH --no-kill
32: #SBATCH --cpus-per-task=12
33: #SBATCH --mem=50G
34: #SBATCH --time=2-00:00:00
35: #SBATCH --chdir=$WDIR_SLURM/$DATE
36: #SBATCH --error=$WDIR_SLURM/$DATE/$SAMPLEID.error.log
37: #SBATCH --output=$WDIR_SLURM/$DATE/$SAMPLEID.log
38: #SBATCH --exclude=hpc[05,06],hpc-rc[12,13]
39: singularity exec -B /mnt/hgenet/NGS_Daten_nfs/singularity_etc/resolv.conf
   :/etc/resolv.conf,/mnt/hgenet/NGS_Daten_nfs/singularity_etc/hosts:/etc/
   hosts,/mnt/hgenet/NGS_tmp:/tmp,/mnt/hgenet/NGS_Daten_nfs:/NGS_Daten_nfs,/

```

Appendix

```
mnt/hgenet/NGS_Daten4:/NGS_Daten4,/mnt/hgenet/NGS_Daten_Test:/
NGS_Daten_Test /mnt/hgenet/NGS_Daten_Test/User/hofmannw/singularity/test/
megSAP-2022_08-64.sif php /megSAP/src/Pipelines/analyze.php -folder
$WDIR_SINGULARITY/$SAMPLEDIR -name $SAMPLEID -use_dragen -no_abra -system
/NGS_Daten_Test/Manifest/NGSD/IDTPanelV2_GRCh38.tsv -threads 12" > "
$WDIR_SLURM/$DATE/$SAMPLEID.sh"
40: sbatch "$WDIR_SLURM/$DATE/$SAMPLEID.sh"
41: done
```

IV Initial Nextflow Workflow

Listing 5: megsap_germline.nf (Initial Version)

```

1: #!/usr/bin/env nextflow
2: nextflow.enable.dsl=2
3:
4: params.sampleDir = "${launchDir}/Sample_*"
5:
6: process megSAP {
7:     input:
8:     val sampleDirectory
9:
10:    script:
11:    containerDirectory=sampleDirectory.toString().replaceFirst(/\mnt\/
    hgenet/, '')
12:    sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
13:    """
14:    php /megSAP/src/Pipelines/analyze.php -folder $containerDirectory -
    name $sampleName -use_dragen -no_abra -system /NGS_Daten_Test/Manifest/
    NGS_D/WGS_lotus_GRCh38.ini -threads 12
15:    """
16: }
17:
18: workflow {
19:     Channel.fromPath(params.sampleDir, type: 'dir') | megSAP
20: }

```

Listing 6: nextflow.config (Initial Version)

```

1: process {
2:     debug = true
3:     executor = 'slurm'
4:     clusterOptions = '--time=2-00:00:00 -p lowprio --exclude=hpc[05,06],
    hpc-rc[07,12,13],hpc-bc15-[01,03,06,14] --no-kill'
5:     container = 'file:///mnt/hgenet/NGS_Daten_Test/User/hofmannw/
    singularity/test/megSAP-2022_08-152.sif'
6:     containerOptions = '-B /mnt/hgenet/NGS_Daten_nfs/singularity_etc/
    resolv.conf:/etc/resolv.conf,/mnt/hgenet/NGS_Daten_nfs/singularity_etc/
    hosts:/etc/hosts,/mnt/hgenet/NGS_tmp:/tmp,/mnt/hgenet/NGS_Daten_nfs:/
    NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4:/NGS_Daten4,/mnt/hgenet/
    NGS_Daten_Test:/NGS_Daten_Test'
7:     stageInMode = 'symlink'
8:     cache = false
9:     cpus = { 12 }

```

Appendix

```
10:     memory = { 50.GB }
11: }
12:
13: singularity {
14:     enabled = true
15:     autoMounts = true
16: }
17:
18: report {
19:     enabled = true
20:     file = 'nextflow_report.html'
21:     overwrite = true
22: }
23:
24: timeline {
25:     enabled = true
26:     file = 'nextflow_timeline.html'
27:     overwrite = true
28: }
29:
30: dag {
31:     enabled = true
32:     file = 'nextflow_dag.mmd'
33:     overwrite = true
34: }
35:
36: cleanup = true
```

Table 7: Reported resource usage for initial workflow

<u>process</u>	<u>megSAP</u>
allocated cpus	12
%cpu	294.2
allocated memory	50 GB
peak_vmem	42.752 GB
duration	11 h 36 min
read_bytes	421.380 GB
write_bytes	303.939 GB

V Detailed Data of BAM to FastQ Conversion Benchmarks

Table 8: Statistics of samples used for benchmarking BAM to FastQ conversion using the command `samtools flagstat` (see [77])

	whole exome sample reads	whole genome sample reads
in total (QC-passed reads + QC-failed reads)	94 905 155	945 531 162
primary	93 901 428	934 455 908
secondary	0	0
supplementary	1 003 727	11 075 254
duplicates	3 912 559	88 969 100
primary duplicates	3 866 188	88 113 730
mapped	94 564 593	943 533 900
primary mapped	93 560 866	932 458 646
paired in sequencing	93 901 428	934 455 908
read1	46 950 714	467 227 954
read2	46 950 714	467 227 954
properly paired	89 632 844	873 201 928
with itself and mate mapped	93 324 280	931 711 492
singletons	236 586	747 154
with mate mapped to a different chr	3 227 688	23 593 106
with mate mapped to a different chr (mapQ>=5)	2 383 985	17 533 528

Table 9: Duration of BAM to FastQ conversion of whole exome data by tool in minutes

type	mean	std	min	25%	50%	75%	max
biobambam2	34.81	0.36	34.40	34.49	34.73	35.17	35.24
ngs-bits	12.09	0.09	11.99	12.03	12.07	12.11	12.31
Picard	17.04	0.26	16.79	16.91	16.94	17.04	17.66
SAMtools multithread	18.78	0.16	18.56	18.63	18.87	18.91	18.98
SAMtools singlethread	59.69	0.64	58.91	59.03	59.93	60.09	60.59

Appendix

Table 10: Memory usage of BAM to FastQ conversion of whole exome data by tool

tool	mean	std	min	25%	50%	75%	max
biobambam2	143 MB	111 kB	143 MB	143 MB	143 MB	143 MB	143 MB
ngs-bits	768 MB	1 MB	766 MB	766 MB	768 MB	769 MB	769 MB
Picard	2 GB	23 MB	2 GB	2 GB	2 GB	2 GB	2 GB
SAMtools multithread	2 GB	1 MB	2 GB	2 GB	2 GB	2 GB	2 GB
SAMtools singlethread	870 MB	602 kB	870 MB	870 MB	870 MB	870 MB	872 MB

Table 11: Amount of data read by BAM to FastQ conversion of whole exome data by tool

tool	mean	std	min	25%	50%	75%	max
biobambam2	9 GB	24 MB	9 GB	9 GB	9 GB	9 GB	9 GB
ngs-bits	8 GB	42 kB	8 GB	8 GB	8 GB	8 GB	8 GB
Picard	37 GB	248 kB	37 GB	37 GB	37 GB	37 GB	37 GB
SAMtools multithread	29 GB	43 kB	29 GB	29 GB	29 GB	29 GB	29 GB
SAMtools singlethread	29 GB	50 kB	29 GB	29 GB	29 GB	29 GB	29 GB

Table 12: Amount of data written by BAM to FastQ conversion of whole exome data by tool

tool	mean	std	min	25%	50%	75%	max
biobambam2	6 GB	0 B	6 GB	6 GB	6 GB	6 GB	6 GB
ngs-bits	7 GB	0 B	7 GB	7 GB	7 GB	7 GB	7 GB
Picard	34 GB	265 kB	34 GB	34 GB	34 GB	34 GB	34 GB
SAMtools multithread	30 GB	6 kB	30 GB	30 GB	30 GB	30 GB	30 GB
SAMtools singlethread	30 GB	0 B	30 GB	30 GB	30 GB	30 GB	30 GB

Table 13: Duration of BAM to FastQ conversion of whole genome data by tool in hours

type	duration in h
biobambam2	3.73
ngs-bits	1.65
Picard	2.93
SAMtools multithread	2.41
SAMtools singlethread	7.85

VI Nextflow Workflow with added BAM to FastQ File Conversion

Listing 7: megsap_germline.nf (with added BAM to FastQ File Conversion)

```

1: #!/usr/bin/env nextflow
2: nextflow.enable.dsl=2
3:
4: params.sampleDir = "${launchDir}/Sample_*"
5:
6: process bam2FastQ {
7:     input:
8:     val sampleDirectory
9:
10:    output:
11:    path "${sampleName}_R?_001.fastq.gz", emit: fastqs
12:    val "${sampleDirectory.toString()}", emit: sampleDir
13:
14:    publishDir "${sampleDirectory}", mode: 'move'
15:
16:    script:
17:    sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
18:    bamFile=files("${sampleDirectory.toString()}/*.bam")[0].toString()
19:    """
20:    /megSAP/data/tools/ngs-bits/bin/BamToFastq \
21:        -in $bamFile \
22:        -out1 \${PWD}/${sampleName}_R1_001.fastq.gz \
23:        -out2 \${PWD}/${sampleName}_R2_001.fastq.gz
24:    """
25:    }
26:
27: process megSAP {
28:     input:
29:     val sampleDirectory
30:
31:     script:
32:     containerDirectory=sampleDirectory.toString().replaceFirst(/\/mnt\/
hgenet/, '')
33:     sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
34:     """
35:     php /megSAP/src/Pipelines/analyze.php -folder $containerDirectory -
    name $sampleName -use_dragen -no_abra -system /NGS_Daten_Test/Manifest/
    NGSD/WGS_lotus_GRCh38.ini -threads 12
36:     """
37: }

```

Appendix

```
38:
39: workflow {
40:     bam_channel = Channel.fromPath(params.sampleDir, type: 'dir').map( {
41:         it ->
42:             if (files("${it}/*.bam").size() >= 1 && files("${it}/*_R?_00?.
43:                 fastq.gz").size() < 2) {
44:                 it
45:             }
46:         })
47:     bam2FastQ(bam_channel)
48:
49:     fastq_channel = Channel.fromPath(params.sampleDir, type: 'dir').map(
50:     { it ->
51:         if (files("${it}/*.bam").size() < 1 && files("${it}/*_R?_00?.
52:             fastq.gz").size() >= 2) {
53:             it
54:         }
55:     })
56:     fastq_channel.mix(bam2FastQ.out.sampleDir) | megSAP
57: }
```

Listing 8: nextflow.config (with added BAM to FastQ File Conversion)

```
1: process {
2:     debug = true
3:     executor = 'slurm'
4:     clusterOptions = '--time=2-00:00:00 -p lowprio --exclude=hpc[05,06],
5:         hpc-rc[07,12,13],hpc-bc15-[01,03,06,14] --no-kill'
6:     container = 'file:///mnt/hgenet/NGS_Daten_Test/User/hofmannw/
7:         singularity/test/megSAP-2022_08-152.sif'
8:     containerOptions = '-B /mnt/hgenet/NGS_Daten_nfs/singularity_etc/
9:         resolv.conf:/etc/resolv.conf,/mnt/hgenet/NGS_Daten_nfs/singularity_etc/
10:         hosts:/etc/hosts,/mnt/hgenet/NGS_tmp:/tmp,/mnt/hgenet/NGS_Daten_nfs:/
11:         NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4:/NGS_Daten4,/mnt/hgenet/
12:         NGS_Daten_Test:/NGS_Daten_Test,/mnt/hgenet/NGS_tmp,/mnt/hgenet/
13:         NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4,/mnt/hgenet/NGS_Daten_Test'
14:     stageInMode = 'symlink'
15:     cache = false
16:     cpus = { 12 }
17:     memory = { 50.GB }
18:     withName: 'bam2FastQ' {
19:         cpus = 2
20:         memory = 8.GB
21:     }
22: }
```


Appendix

```
16:
17: singularity {
18:     enabled = true
19:     autoMounts = true
20: }
21:
22: report {
23:     enabled = true
24:     file = 'nextflow_report.html'
25:     overwrite = true
26: }
27:
28: timeline {
29:     enabled = true
30:     file = 'nextflow_timeline.html'
31:     overwrite = true
32: }
33:
34: dag {
35:     enabled = true
36:     file = 'nextflow_dag.mmd'
37:     overwrite = true
38: }
39:
40: cleanup = true
```

VII Nextflow Workflow With Separated Process Steps

Listing 9: megsap_germline.nf (with separated process steps)

```

1: #!/usr/bin/env nextflow
2: nextflow.enable.dsl=2
3:
4: params.sampleDir = "${launchDir}/Sample_*"
5:
6: process bam2FastQ {
7:     input:
8:     val sampleDirectory
9:
10:    output:
11:    path "${sampleName}_R?_001.fastq.gz", emit: fastqs
12:    val "${sampleDirectory.toString()}", emit: sampleDir
13:
14:    publishDir "${sampleDirectory}", mode: 'move'
15:
16:    script:
17:    sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
18:    bamFile=files("${sampleDirectory.toString()}/*.bam")[0].toString()
19:    """
20:    /megSAP/data/tools/ngs-bits/bin/BamToFastq \
21:        -in $bamFile \
22:        -out1 \${PWD}/${sampleName}_R1_001.fastq.gz \
23:        -out2 \${PWD}/${sampleName}_R2_001.fastq.gz
24:    """
25:    }
26:
27: process megSAPma {
28:     input:
29:     val sampleDirectory
30:
31:     output:
32:     val "$sampleDirectory"
33:
34:     script:
35:     containerDirectory=sampleDirectory.toString().replaceFirst(/\mnt\/
    hgenet/, '')
36:     sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
37:     threads=task.cpus
38:     steps='ma'
39:     template 'megasap_germline.sh'

```

Appendix

```
40: }
41:
42: process megSAPvc {
43:     input:
44:         val sampleDirectory
45:
46:     output:
47:         val "$sampleDirectory"
48:
49:     script:
50:         containerDirectory=sampleDirectory.toString().replaceFirst(/\mnt\/
hgenet/, '')
51:         sampleName=sampleDirectory.toString().tokenize('/').last().
replaceFirst(/.*Sample_/, '')
52:         threads=task.cpus
53:         steps='vc'
54:         template 'megsap_germline.sh'
55: }
56:
57: process megSAPcn {
58:     input:
59:         val sampleDirectory
60:
61:     output:
62:         val "$sampleDirectory"
63:
64:     script:
65:         containerDirectory=sampleDirectory.toString().replaceFirst(/\mnt\/
hgenet/, '')
66:         sampleName=sampleDirectory.toString().tokenize('/').last().
replaceFirst(/.*Sample_/, '')
67:         threads=task.cpus
68:         steps='cn'
69:         template 'megsap_germline.sh'
70: }
71:
72: process megSAPsv {
73:     input:
74:         val sampleDirectory
75:
76:     output:
77:         val "$sampleDirectory"
78:
79:     script:
```

Appendix

```
80:     containerDirectory=sampleDirectory.toString().replaceFirst(/\/mnt\/
    hgenet/, '')
81:     sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
82:     threads=task.cpus
83:     steps='sv'
84:     template 'megsap_germline.sh'
85: }
86:
87: process megSAPdb {
88:     input:
89:     val sampleDirectory
90:
91:     output:
92:     val "$sampleDirectory"
93:
94:     script:
95:     containerDirectory=sampleDirectory.toString().replaceFirst(/\/mnt\/
    hgenet/, '')
96:     sampleName=sampleDirectory.toString().tokenize('/').last().
    replaceFirst(/.*Sample_/, '')
97:     threads=task.cpus
98:     steps='db'
99:     template 'megsap_germline.sh'
100: }
101:
102: workflow {
103:     bam_channel = Channel.fromPath(params.sampledir, type: 'dir').map( {
    it ->
104:         if (files("${it}/*.bam").size() >= 1 && files("${it}/*_R?_00?.
    fastq.gz").size() < 2) {
105:             it
106:         }
107:     })
108:     bam2FastQ(bam_channel)
109:
110:     fastq_channel = Channel.fromPath(params.sampledir, type: 'dir').map(
    { it ->
111:         if (files("${it}/*.bam").size() < 1 && files("${it}/*_R?_00?.
    fastq.gz").size() >= 2) {
112:             it
113:         }
114:     })
115:
```

Appendix

```
116:     fastq_channel.mix(bam2FastQ.out.sampledir) | megSAPma | megSAPvc |
      megSAPcn | megSAPsv | megSAPdb
117: }
```

Listing 10: megsap_germline.sh

```
1: #!/bin/bash
2: php /megSAP/src/Pipelines/analyze.php -folder $containerDirectory -name
   $sampleName -use_dragen -no_abra -system /NGS_Daten_Test/Manifest/NGSD/
   WGS_lotus_GRCh38.ini -threads $threads -steps $steps
```

Table 14: Reported resource usage for workflow with separated process steps

process	megSAPma	megSAPvc	megSAPcn	megSAPsv	megSAPdb
allocated cpus	12	12	12	12	12
%cpu	344.8	636.8	103.0	188.1	20.9
allocated mem- ory	50 GB	50 GB	50 GB	50 GB	50.000 GB
peak_vmem	1.313 GB	17.595 GB	43.140 GB	1.319 GB	456.023 MB
duration	2 h 54 min	2 h 40 min	6 h 11 min	1 h 3 min	3 min 51 s

VIII Nextflow Workflow With Resource Optimization

Listing 11: nextflow.config (with resource optimization)

```

1: process {
2:     debug = true
3:     executor = 'slurm'
4:     clusterOptions = '--time=2-00:00:00 -p lowprio --exclude=hpc[05,06],
hpc-rc[07,12,13],hpc-bc15-[01,03,06,14] --no-kill'
5:     container = 'file:///mnt/hgenet/NGS_Daten_Test/User/hofmannw/
singularity/test/megSAP-2022_08-152.sif'
6:     containerOptions = '-B /mnt/hgenet/NGS_Daten_nfs/singularity_etc/
resolv.conf:/etc/resolv.conf,/mnt/hgenet/NGS_Daten_nfs/singularity_etc/
hosts:/etc/hosts,/mnt/hgenet/NGS_tmp:/tmp,/mnt/hgenet/NGS_Daten_nfs:/
NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4:/NGS_Daten4,/mnt/hgenet/
NGS_Daten_Test:/NGS_Daten_Test,/mnt/hgenet/NGS_tmp,/mnt/hgenet/
NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4,/mnt/hgenet/NGS_Daten_Test'
7:     stageInMode = 'symlink'
8:     scratch = '/mnt/hgenet/NGS_tmp/nextflow'
9:     cache = false
10:    cpus = { 12 }
11:    memory = { 50.GB }
12:    withName: 'bam2FastQ' {
13:        cpus = 2
14:        memory = 8.GB
15:    }
16:    withName: 'megSAPma' {
17:        cpus = 15
18:        memory = 2.GB
19:    }
20:    withName: 'megSAPvc' {
21:        cpus = 8
22:        memory = 20.GB
23:    }
24:    withName: 'megSAPcn' {
25:        cpus = 2
26:        memory = 48.GB
27:    }
28:    withName: 'megSAPsv' {
29:        cpus = 2
30:        memory = 2.GB
31:    }
32:    withName: 'megSAPdb' {
33:        cpus = 2
34:        memory = 1.GB
35:    }

```

Appendix

```
36: }
37:
38: singularity {
39:     enabled = true
40:     autoMounts = true
41: }
42:
43: report {
44:     enabled = true
45:     file = 'nextflow_report.html'
46:     overwrite = true
47: }
48:
49: timeline {
50:     enabled = true
51:     file = 'nextflow_timeline.html'
52:     overwrite = true
53: }
54:
55: dag {
56:     enabled = true
57:     file = 'nextflow_dag.mmd'
58:     overwrite = true
59: }
60:
61: cleanup = true
```

Table 15: Reported resource usage for workflow with resource optimization

process	megSAPma	megSAPvc	megSAPcn	megSAPsv	megSAPdb
allocated cpus	15	8	2	2	2
%cpu	316.6	470.4	93.6	181.9	22.8
allocated mem- ory	2 GB	20 GB	48 GB	2 GB	1 GB
peak_vmem	1.251 GB	14.999 GB	18.786 GB	1.210 GB	455.805 MB
duration	2 h 41 min	2 h 59 min	6 h 23 min	1 h 28 min	3 min 26 s

IX Correspondence with Dr. Marc Sturm Discussing CPU Usage of SeqPurge

Schnur, Benedikt

Von: Dr. Marc Sturm <Marc.Sturm@med.uni-tuebingen.de>
Gesendet: Mittwoch, 4. Januar 2023 12:24
An: Schnur, Benedikt
Cc: Hofmann, Winfried Dr.; Schmidt, Gunnar Dr.
Betreff: AW: megSAP SeqPurge Threads

Hi Benedikt,

ja, ich habe es bei n-2 gelassen, weil die IO-Threads nicht immer 100% brauchen.

Mit dem aktuellen master von megSAP wären es bei 15 Threads 3 für IO und 12 für die Prozessierung. Mehr als 12 Threads macht es eher langsamer als schneller, daher ist das das neue Maximum.

Viele Grüße,
Marc

Von: Schnur.Benedikt@mh-hannover.de <Schnur.Benedikt@mh-hannover.de>
Gesendet: Mittwoch, 4. Januar 2023 10:12
An: Dr. Marc Sturm
Cc: Hofmann.Winfried@mh-hannover.de; Schmidt.Gunnar@mh-hannover.de
Betreff: AW: megSAP SeqPurge Threads

Hallo Marc,

danke für die Info. Habe ich in der SeqPurge Beschreibung jetzt auch nachgelesen. Etwas un-intuitiv. ☹️

Damit wäre ich dann bei 9 Threads für den darunterliegenden analyze.php Aufruf (zumindest aktuell) und dann später mal 13, richtig?

Liebe Grüße
Benedikt

--

Benedikt Schnur
IT-Projektmanager
Institut für Humangenetik
Medizinische Hochschule Hannover (MHH)
OE 6300, Carl-Neuberg-Str. 1, 30625 Hannover, Deutschland
Tel.: +49 511 532-32583
schnur.benedikt@mh-hannover.de
<https://www.mhh.de/humangenetik>

Von: Dr. Marc Sturm <Marc.Sturm@med.uni-tuebingen.de>
Gesendet: Dienstag, 3. Januar 2023 18:39
An: Schnur, Benedikt <Schnur.Benedikt@mh-hannover.de>
Cc: Hofmann, Winfried Dr. <Hofmann.Winfried@mh-hannover.de>; Schmidt, Gunnar Dr. <Schmidt.Gunnar@mh-hannover.de>
Betreff: AW: megSAP SeqPurge Threads

Hi Benedikt,

1

Figure 29: Correspondence with Dr. Marc Sturm discussing CPU usage of SeqPurge

Appendix

weil 2 Threads für Read/Write benutzt werden, also bleiben n-2 für die eigentlich Prozessierung der Reads. Mittlerweile sind es sogar 3 Threads (1xR, 2xW), weil das Schreiben der zwei Outputdateien auch parallel läuft.

Aber die obere Grenze muss ich noch anheben - 6 ist da zu niedrig. Es skaliert jetzt glaube bis ca 10 linear.

Viele Grüße,
Marc

Von: Schnur.Benedikt@mh-hannover.de <Schnur.Benedikt@mh-hannover.de>
Gesendet: Dienstag, 3. Januar 2023 15:01
An: Dr. Marc Sturm
Cc: Hofmann.Winfried@mh-hannover.de; Schmidt.Gunnar@mh-hannover.de
Betreff: megSAP SeqPurge Threads

Hallo Marc,

kurze Verständnisfrage weil es mir eben im Log aufgefallen ist und ich dann im Code nachgeforscht habe: Warum nutzt `seqpurge` nicht den an `analyze.php` übergebenen Wert für `threads` sondern 2 weniger (<https://github.com/imgag/megSAP/blob/master/src/Pipelines/mapping.php#L118>)?

Liebe Grüße und frohes neues Jahr
Benedikt

--

Benedikt Schnur
IT-Projektmanager
Institut für Humangenetik
Medizinische Hochschule Hannover (MHH)
OE 6300, Carl-Neuberg-Str. 1, 30625 Hannover, Deutschland
[Tel.:+49 511 532-32583](tel:+49511532-32583)
schnur.benedikt@mh-hannover.de
<https://www.mhh.de/humangenetik>

Figure 29: Correspondence with Dr. Marc Sturm discussing CPU usage of SeqPurge (cont.)

X Nextflow Workflow With Resilience and Monitoring**Listing 12: nextflow.config (with resilience and monitoring)**

```

1: process {
2:     debug = true
3:     executor = 'slurm'
4:     clusterOptions = '--time=2-00:00:00 -p lowprio --exclude=hpc[05,06],
hpc-rc[07,12,13],hpc-bc15-[01,03,06,14] --no-kill'
5:     container = 'file:///mnt/hgenet/NGS_Daten_Test/User/hofmannw/
singularity/test/megSAP-2022_08-152.sif'
6:     containerOptions = '-B /mnt/hgenet/NGS_Daten_nfs/singularity_etc/
resolv.conf:/etc/resolv.conf,/mnt/hgenet/NGS_Daten_nfs/singularity_etc/
hosts:/etc/hosts,/mnt/hgenet/NGS_tmp:/tmp,/mnt/hgenet/NGS_Daten_nfs:/
NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4:/NGS_Daten4,/mnt/hgenet/
NGS_Daten_Test:/NGS_Daten_Test,/mnt/hgenet/NGS_tmp,/mnt/hgenet/
NGS_Daten_nfs,/mnt/hgenet/NGS_Daten4,/mnt/hgenet/NGS_Daten_Test'
7:     stageInMode = 'symlink'
8:     scratch = '/mnt/hgenet/NGS_tmp/nextflow'
9:     cache = false
10:    cpus = { 12 }
11:    memory = { 50.GB }
12:    maxRetries = 2
13:    errorStrategy = { task.attempt < 2 ? 'retry' : 'ignore' }
14:    withName: 'bam2FastQ' {
15:        cpus = 2
16:        memory = { 8.GB * task.attempt }
17:    }
18:    withName: 'megSAPma' {
19:        cpus = 15
20:        memory = { 2.GB * task.attempt }
21:    }
22:    withName: 'megSAPvc' {
23:        cpus = 8
24:        memory = { 20.GB * task.attempt }
25:    }
26:    withName: 'megSAPcn' {
27:        cpus = 2
28:        memory = { 48.GB * task.attempt }
29:    }
30:    withName: 'megSAPsv' {
31:        cpus = 2
32:        memory = { 2.GB * task.attempt }
33:    }
34:    withName: 'megSAPdb' {
35:        cpus = 2

```

Appendix

```
36:     memory = { 1.GB * task.attempt }
37:   }
38: }
39:
40: singularity {
41:   enabled = true
42:   autoMounts = true
43: }
44:
45: report {
46:   enabled = true
47:   file = 'nextflow_report.html'
48:   overwrite = true
49: }
50:
51: timeline {
52:   enabled = true
53:   file = 'nextflow_timeline.html'
54:   overwrite = true
55: }
56:
57: dag {
58:   enabled = true
59:   file = 'nextflow_dag.mmd'
60:   overwrite = true
61: }
62:
63: mail {
64:   from = 'Gen-IT@mh-hannover.de'
65:   smtp.host = 'exchreceive.mh-hannover.local'
66:   smtp.port = 25
67: }
68:
69: cleanup = true
```

XI AWS Prize Calculation for Optimized Nextflow Workflow



Figure 30: AWS prize calculation for optimized Nextflow workflow

Appendix

Amazon EC2	No group applied	EU (Frankfurt)	0.00 USD	3.88 USD
Description: megSAPcn Config summary: Tenancy (Shared Instances), Operating system (Linux), DT Inbound: Not selected (0 tb_month), DT Outbound: Not selected (0 tb_month), DT Intra-Region: (0 tb_month), Workload (Monthly, Baseline: 0, Peak: 1, Duration of peak: 0 Day 6 Hr 23 Min), Enable monitoring (disabled), Advance EC2 instance (r5.2xlarge), Pricing strategy (On-Demand), Snapshot Frequency (No snapshot storage)				
Amazon EC2	No group applied	EU (Frankfurt)	0.00 USD	0.04 USD
Description: megSAPsv Config summary: Tenancy (Shared Instances), Operating system (Linux), DT Inbound: Not selected (0 tb_month), DT Outbound: Not selected (0 tb_month), DT Intra-Region: (0 tb_month), Workload (Monthly, Baseline: 0, Peak: 1, Duration of peak: 0 Day 1 Hr 28 Min), Enable monitoring (disabled), Advance EC2 instance (t3.small), Pricing strategy (On-Demand), Snapshot Frequency (No snapshot storage)				
Amazon EC2	No group applied	EU (Frankfurt)	0.00 USD	0.16 USD
Description: bam2Fast Config summary: Tenancy (Shared Instances), Operating system (Linux), DT Inbound: Not selected (0 tb_month), DT Outbound: Not selected (0 tb_month), DT Intra-Region: (0 tb_month), Workload (Monthly, Baseline: 0, Peak: 1, Duration of peak: 0 Day 1 Hr 39 Min), Snapshot Frequency (No snapshot storage), Enable monitoring (disabled), Advance EC2 instance (t3.large), Pricing strategy (On-Demand)				


Acknowledgement
AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#) .

Figure 30: AWS prize calculation for optimized Nextflow workflow (cont.)

XII AWS Prize Calculation for initial Nextflow Workflow

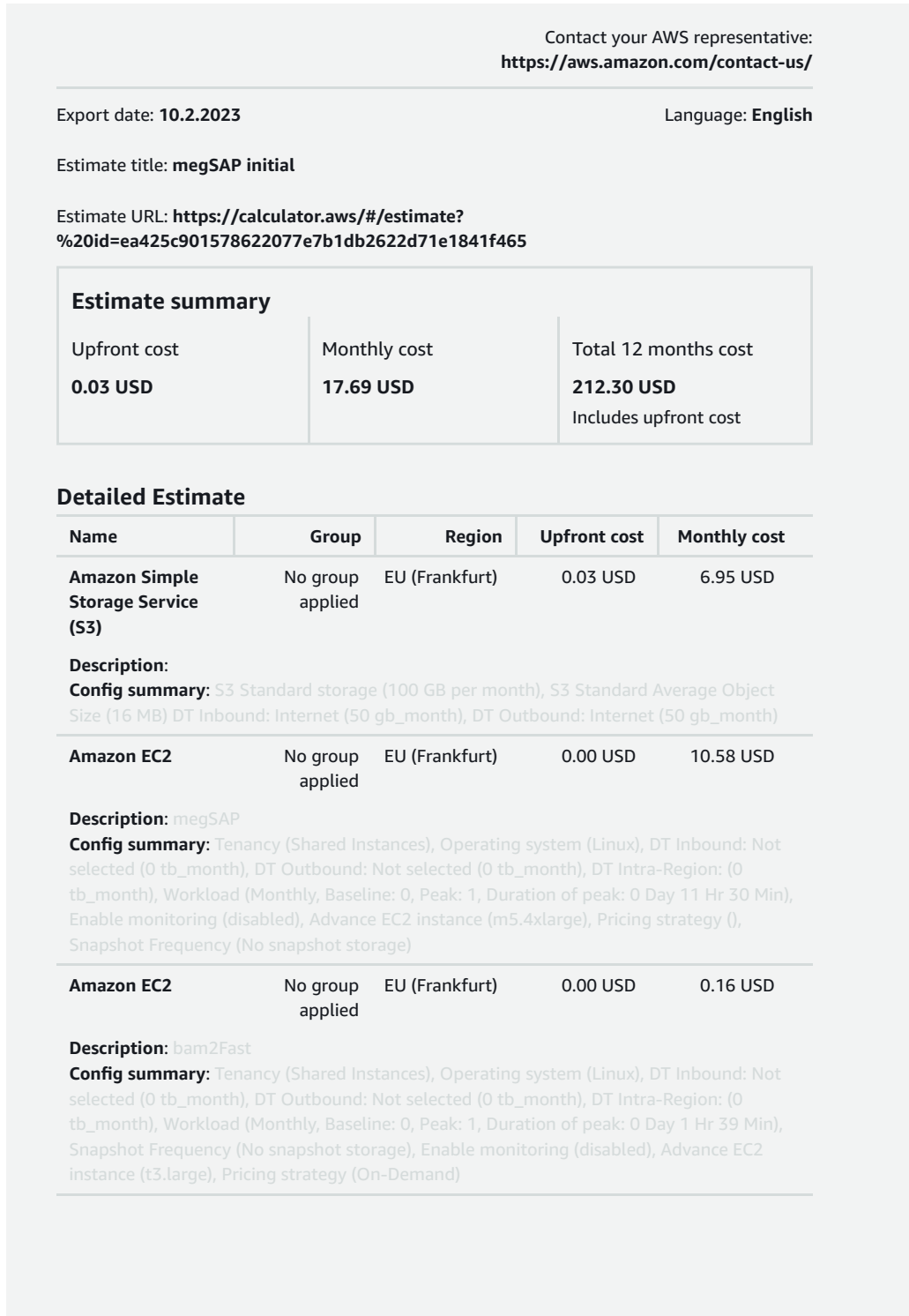


Figure 31: AWS prize calculation for initial Nextflow workflow

Appendix

Acknowledgement


AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#) .

Figure 31: AWS prize calculation for initial Nextflow workflow (cont.)