CYBERNETICA

ESTAT 2019.0232 Testing Report

Armin Daniel Kisand, Toivo Vajakas

Technical document

Version 1.0

November 24, 2021

17 pages

Doc. Y-1440-9

Disclaimer. This document was prepared by Cybernetica AS as part of a procured project under Service Contract No ESTAT 2019.0232 (Ref. Ares(2020)2309804 - 30/04/2020). The opinions expressed in this document are those of the authors. They do not purport to reflect the opinions, views or official positions of the European Commission or its members.

Contents

1 Introduction

Test descriptions and testing results of ESTAT 2019.0232 are presented in this document. Tests were conducted according to test descriptions presented below. Additional background information can be found in the Solution Analysis and Solution Architecture documents.

2 Technical Details

2.1 Test Platform Hardware

- CPU: Intel® Xeon® CPU E3-1225 v5 @ 3.30GHz
- RAM: 4x8GiB DDR4, ECC UDIMM, 2133 Mbps
- Storage: Samsung SSD 850 PRO 1TB

2.2 Intel® SGX Related Mitigation Configuration

Due to a recent security vulnerability named Load Value Injection (LVI), enclaves on older processors need to be compiled with costly software mitigations to ensure that no data can be leaked. The LVI vulnerability is fixed on a hardware level in newer processors, such that the software mitigations are no longer required. Therefor, the task enclaves do not have the LVI software mitigations applied for the most part. A minor part of the code still contains the weakened *Control-Flow* LVI software mitigations, as this code is compiled separately as part of the Sharemind HI platform. However, this has no significant impact on the performance results.

2.3 Test Scripts

All tests were implemented as shell scripts with minimal manual interaction. The used test scripts are in the source code bundle `sharemind-hi-eurostat-source-bundle` at the following places:

- Business cycle test script: `task-enclaves/test/businesscycle/client.sh`
- Load test test script: `task-enclaves/test/performance/client.sh`

3 Test Descriptions

3.1 Validating the Correctness and Unambiguity of Statistical Algorithms Description

This activity is not a traditional test but is still considered an essential part of quality assurance.

The validation includes:

1. Producing an easy-to-read prototype code from formal algorithm description in annex "Specification of Test Use-cases for Project ESTAT 2019.0232", without security requirements.
2. Validating the prototype code by statistical analysis experts (e.g. from Eurostat) with Python knowledge for conformance with intended behavior.

The validation was executed as part of the iterative process to correct all issues related to the algorithm description.

3.2 Validation of the Implementation of the Statistical Algorithm

The following steps are included:

1. Prepare dataset(s) with synthetic data.
2. Run the Solution and prototype on the same dataset and ensure that the output is mostly identical.

This test uses the same script as the load test, using parameter values that activate the execution of the python prototype on same data, and automatic comparison between the results, as described above.

3.2.1 The Considerations to Apply Approximate Comparison in Test

The python code in prototype uses *double* values[1] due to practical code readability issues, but the data files of the Solution use *float* values[2] for size optimisation purposes. There are also other possible differences – order of calculations in compiled and optimised code, precision of intermediate calculatons.

In the Solution the performance is limited by file IO so 32-bit float is preferable to ensure smaller file sizes. 64-bit double is default internal type for python so ensuring 32-bit calculations with

[1] IEEE 754 double-precision format
[2] IEEE 754 single-precision format

exact same behaviour in python would require considerable extra work and introduce some code clutter.

Therefore, the the py and C++ implementations were permitted to use logically identical but technically different floating point calculations. Therefore approximate comparison is needed to check for conformance.

3.2.2 Approximate Comparison Criterion

Comparison is done on produced csv files which contain rounded values.

An approximate matching criterion is used during the comparison of the three result report types: two (sorted) reports are considered to be equal if for any tile the difference in any field is not larger than contribution of 2 subscribers into given tile field: $e_{max} = 2 * adjustment + 1$ where $adjustment$ is calibration coefficient based on census data. 1 is added as extra margin for possible rounding.

3.3 Business Cycle Test

The business cycle test issues multiple report requests and a series of correct, missing and corrupted H files, and checks that the MNO-ND component, the MNO-VAD component, the NSI component, and the task enclaves behave correctly.

3.3.1 Test Description

The following description uses periods instead of dates to reduce visual clutter. The test scenario includes several different situations

Following data will be prepared in test initialisation phase

1. Correct H files for periods 1,2,3,8,10,20
2. Different corrupt H file and key variants for periods 4,6,17,21
3. There is no H file generated for any other period that is not indicated explicitly above

Within test, different queries will be run with different time range, the files remain the same. There are no special considerations for selecting period id values, e.g. 17 or 21.

3.3.1.1 Preconditions

1. The Sharemind HI server is started.
2. All enforcers approved the solution.
3. The MNO-ND component is started to provide the pseudonym generation in the following H file generation.
4. Valid H files with pseudonyms are generated in a staging directory for the periods 1, 2, 3, 8, 10 and 20.
5. Corrupt H files are generated in a staging directory as follows:
    - H file for period 4: Create two H files for the periods 4 and 100, and replace the H file for period 4 with the H file for period 100. Thus, the analytics enclave finds a key for period 4 but fails to depseudonymise its pseudonyms.
    - H file for period 6: Create an H file for period 101, and store it as if it were an H file for period 6. Thus, the analytics enclave cannot find a key for period 6.

- H file for period 17: Create an H file for period 17, but replace its contents with the string `invalid`. Thus, the analytics enclave finds a key for period 17 but the file content is corrupt.
- H file for period 21: Create an H file containing the string `invalid`. Thus, the analytics enclave cannot find a key for period 21.

3.3.1.2 Test Steps

The correctness of the test steps is verified through the progress reports which are generated by the NSI and VAD scripts. The generated progress reports are compared against previously recorded and validated copies. Workflow of tests is illustrated on Fig.1.

1. NSI uploads a request with the period range 0 to 5 with use case 2.
2. VAD runs the `automatic-h-file-import`, finds a new report request for the period range 0 to 5.
3. The H files for periods 1, 2 and 4 are moved to the H file import directory.
4. VAD runs the `automatic-h-file-import`, processes the periods 1, 2 and 4, skips periods 0 and 3.
5. NSI uploads a request with the period range 6 to 10 with use case 2.
6. NSI uploads a request with the period range 12 to 17 with use case 2.
7. The H files for periods 3, 6, 8, 10, 17, 20 and 21 are provided.
8. VAD runs the `automatic-h-file-import`, processes the periods 6, 8, 10 and 17 (3 is ignored). It automatically finishes the report requests with the period ranges 0 to 5 and 6 to 10, cancels the report request with the period range 12 to 17, skips periods 5, 7, 9, 11, 12, 13, 14, 15 and 16.
9. NSI uploads a request with the period range 20 to 21 with use case 2.
10. NSI uploads a request with the period range 22 to 23 with use case 2.
11. VAD runs the `automatic-h-file-import`, processes the periods 20 and 21. It automatically finishes the report requests with the period range 20 to 21.
12. VAD runs `finish-report`, canceling the report request with the period range 22 to 23.

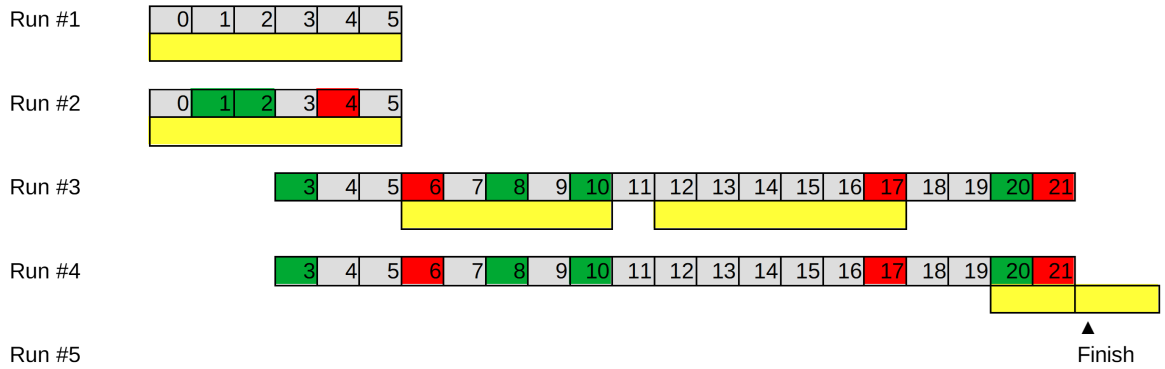

Figure 1. Illustration to test workflow. In total, 5 run commands are given. Each small rectangle represents one period, the period ID is written inside it. Each yellow elongated rectangle represents a query. Periods with a normal H file available are green color, periods with problematic H files are red, periods with a missing H file are grey. The last request does not terminate automatically, run #5 finishes it manually.

This test covers the following situations:

1. Single report request is waiting for execution. After that, the request is discovered and executed by the runtime script.
2. Two report requests are waiting for execution. After that, the requests are discovered and executed by the runtime script.
3. Skipped files at the beginning and end of a request
4. Old files will be ignored (H file for period 3).
5. Errors in the middle and at the end of a report request are handled correctly.
6. Automatically finishing a report when deemed necessary (either canceling the report or issuing the report generation)
7. Manually finishing a request.

The correct behavior is checked automatically inside the test script.

3.4 Load Tests

The load tests were designed to measure the performance characteristics of the Solution. The load tests rely on assumptions given in the Solution Architecture delivery document in subchapter “Performance Requirements” for the selection of test parameters.

The following tests were performed:

- Test the performance at full load specified.
- Test the effects of the subscriber count.
- Test the effects of the tile count per subscriber, other parameters as in the dimensioning table.

For a shorter testing cycle the tests can implement a full report period of 90 days. The volume of daily calculations and aggregated intermediate results shall match the volumes predicted for maximum specified period, so the results characterise the behaviour of the Solution at the maximum specified parameters.

The load tests use the simulated data produced by the data generator. The data generator is described in the Synthetic Test Data Generation document. The data generator parameters must be tuned so that the data volume is similar to values described in subchapter “Performance Requirements” in the Solution Architecture deliverable. The relation between the data generator parameters and the volume of data in later processing stages is not straightforward and needs some experimentation. One has to select data generation parameters so that following indicators are close to the specified values.

- average number of tiles per subscriber in S at the end of the 90-day report period (specified value 200)
- number of Y records per subscriber (specified value 10).

In the current version of the Solution the parameters of the data generator and the parameters of the analytics enclave affect the tile count per subscriber and the Y count per subscriber only indirectly. Hence the real tile count per subscriber in daily H data files and in aggregated S data files will be approximate.

3.4.1 Full Load

In this test the subscriber count is the specified maximum of 50 000 000 subscribers. The report duration is 90 days. The average daily tile count per user is 11.

3.4.2 Performance for Reduced Subscriber Counts

The maximum number of unique subscribers in the dimensioning table was 50 000 000. The performance is tested for reduced user count of 20 000 000. Other parameters are fixed as in the full load test.

3.4.3 Performance for Reduced Tile Counts per Subscriber

The average daily tile count per user is varied as 6 and 3. Other parameters are fixed as in the full load test.

3.4.4 Performance for Reduced Report Period

The full 90 day calculations already contain the information for shorter periods. Thus, separate tests were not necessary.

3.4.5 List of All Planned Load Tests

What is tested	Subscriber count	Tiles	Period duration days
Full load	50 000 000	11	90
Reduced subscriber count	20 000 000	11	90
Reduced subscriber count	20 000 000	6	90
Reduced tile count	50 000 000	6	90
Reduced tile count	50 000 000	3	90

3.4.6 Load Test Technical Details

3.4.6.1 Preconditions

1. The Sharemind HI server is started.
2. All enforcers approved the Solution.
3. Run `atop -w` in the background.

3.4.6.2 Test Steps

1. Repeat the following steps until all H files for the data range of the report request have been supplied:
    1. Generate at most k new H files in parallel for the next dates. k was chosen to be 4, and it is limited by the amount of available CPU cores, RAM and disk space.
    2. VAD runs the `automatic-h-file-import` to process the newly generated H files.
2. Stop the background `atop` job.
3. Download all application logs, which contain information about the sizes of the data files and the in-enclave execution times.

4 Results

4.1 Validating the Correctness and Unambiguity of Statistical Algorithms Description

The prototype was prepared by Cybernetica AS, reviewed and approved by Eurostat.

Result: success.

4.2 Validation of the Implementation of the Statistical Algorithm

The algorithm was run on emulated data. Following parameter values were used: 10 000 subscribers, an average daily tile count per user of 11, and running 90 days. Other parameters were default values. In this configuration, following reports were produced

- fingerprint report: 24080 records
- functional urban fingerprint_report: 8091 records
- top anchor distribution report: 6178 records

The enclave output and the output of the python prototype were exactly identical.

By changing the `day_quantisation_threshold` parameter value from 10 to 0.25, the report results contain more records. For given synthetic data, `day_quantisation_threshold` parameter value 10 was reasonable in a sense that produces expected amount of report records. Value 0.25 is overly sensitive and produces too many records, therefore not reasonable for practical work.

With `day_quantisation_threshold` equal to 0.25, the fingerprint report had 2 (out of 32137) tiles where some calculated field value did not match. The value difference was 10 in all these cases. $adjustment$ value was 9.5527. The difference value 10 indicated that the difference was caused by 1 subscriber processed differently, rounded in final report csv to integer value. $10 < e_{max} = 2 * 9.5527 + 1 = 20.1054$ so this file passed the test successfully. The functional urban fingerprint report and top anchor distribution report were exactly identical in the enclave output and the output of the python prototype.

So with practical parameter values the test did not find any differences for 10 000 users. Overly sensitive parameter was run just to demonstrate that mismatch can happen and still was within the approximate matching criterion limits.

Result: success.

4.3 Business Cycle Test

The test was run as described in the description. The behaviour of the task enclaves, as observed through the progress reports compiled by the client applications for the VAD and the NSI, matched the expectations. The results were automatically checked by the test script.

Result: success.

4.4 Load Tests

4.4.1 Description of Test Inputs and Calculation Times

The data for test were generated using the data generator. The tile count per user was controlled by parameter TILES_PER_SUBPERIOD values 5, 3, and 1 which resulted in generated data with approximately 11, 6, and 3 average daily tiles per subscriber. USER_COUNT was specified 1 000 000, DUPLICATE_COUNT was 19, 49, or 99 to get 20 million, 50 million, or 100 million subscriber dataset. The data generator code is not optimised and the duplicates were used to reduce resource needs during data generation. We checked that starting from 1 million unique users, the performance figures were not affected by duplicate mechanism. We compared performance characteristics on two datasets: 100 000 unique users and 9 duplicates vs 1 000 000 unique users.

Belgium census dataset with 32 140 tiles total was used for generating the tiles. Count of unique tiles has no effect on speed of daily calculations.

Other parameters for data generator were default values in python except FILL_SUBPERIOD_FROM_NIGHT_TILE_PROBABILITY = 0.3, FILL_SUBPERIOD_FROM_DAY_TILE_PROBABILITY = 0.15

The performed tests slightly differ from plan due to changed focus and time constraints.

Parameter	Unit	Data1	Data2	Data3	Data4	Data5
Report period	days	90	90	90	90	90
unique subscriber	10^6 subscribers	100	50	50	50	20
tiles per subscriber daily	tiles/day	11.6	11.3	6.9	2.8	11.3
size of daily H file	Gbytes	30	15	9	4	6
tiles in S at end of period per subscriber	tiles/ subscriber	235	236	153	86	230
Maximum size of intermediate results	Gbytes	1125	566	367	206	221
Longest daily processing	hours	5.3	2.6	1.7	0.9	1.0
Total processing time	hours	308	151	97	53	62

A larger tile count resulted in a larger input data H and larger intermediate data structure S. The emprical cumulative distribution function (ECDF) of data size per subscriber for input data H, intermediate data S, and intermediate data Y are illustrated on figures Fig. 2, 3, 4. These ECDF illustrations are provided for input dataset with 11 tiles per subscriber.

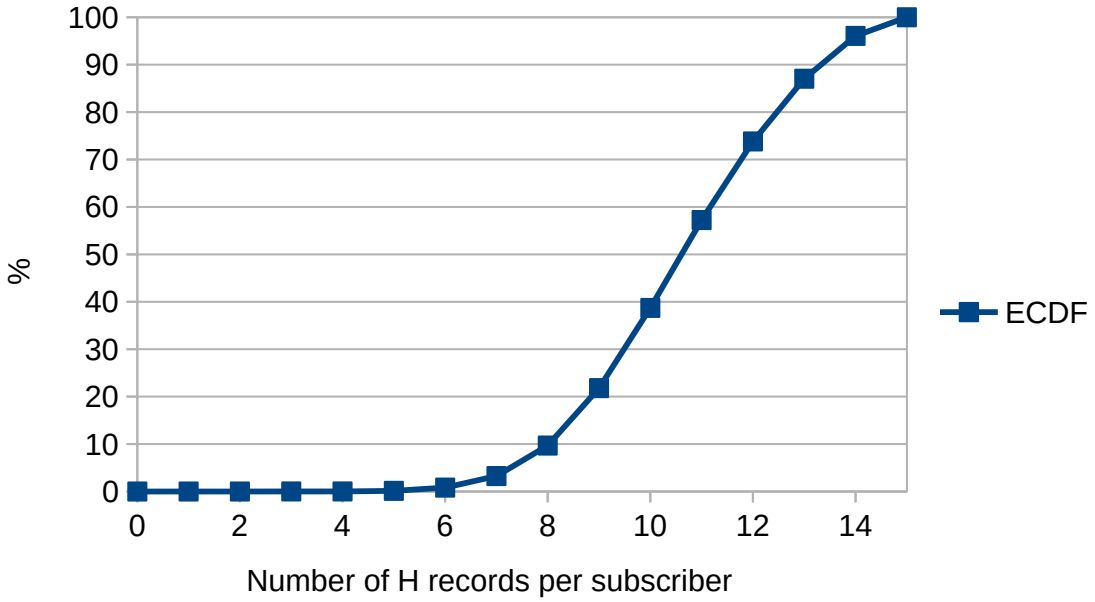

Figure 2. ECDF of tile counts for each subscriber in H file, for dataset 11 tiles/subscriber. Vertical axis - probability that a subscriber had a given tile count or less. Horisontal axis - count of unique tiles per subscriber

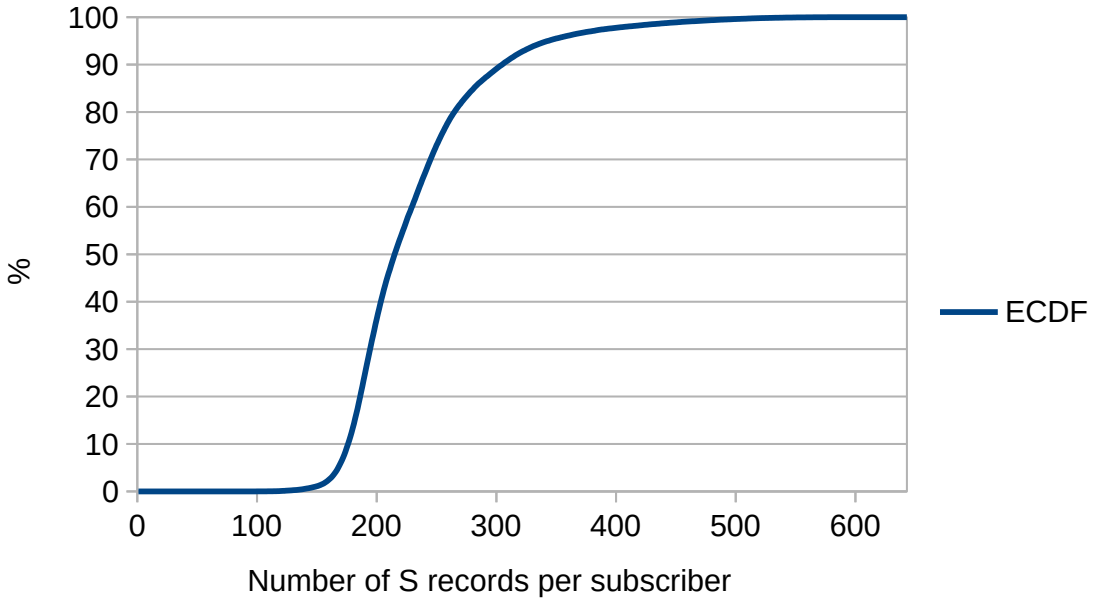

Figure 3. ECDF of tile counts for each subscriber in S file at end of aggregation, for dataset 11 tiles/subscriber. Vertical axis - probability that a subscriber had a given tile count or less. Horisontal axis - count of unique tiles per subscriber

4.4.2 Execution Time of Daily Calculation

Figure 5 presents the execution times of daily data import calculations in the Solution. The size of intermediate data S increased with each period. The size of S is the dominating factor behind the increase in the execution time. The calculations of the last period take less time: during the last period, the internal S data is consumed on the fly for the final reports and no S data is written

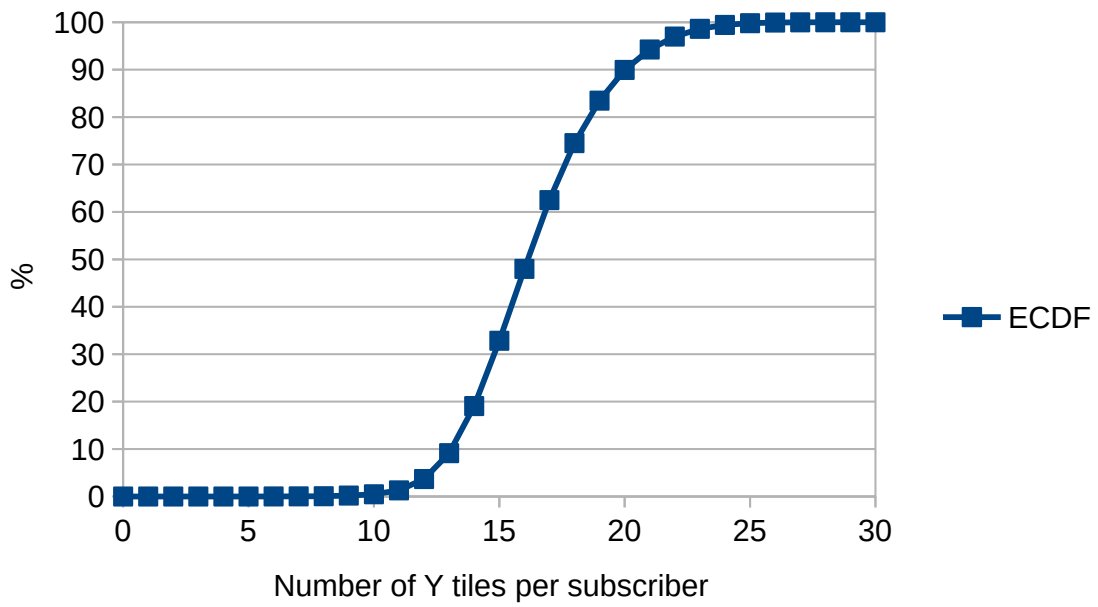

Figure 4. ECDF of tile counts for each subscriber in Y file, for dataset 11 tiles/subscriber. Vertical axis - probability that a subscriber had a given tile count or less. Horisontal axis - count of tiles per subscriber

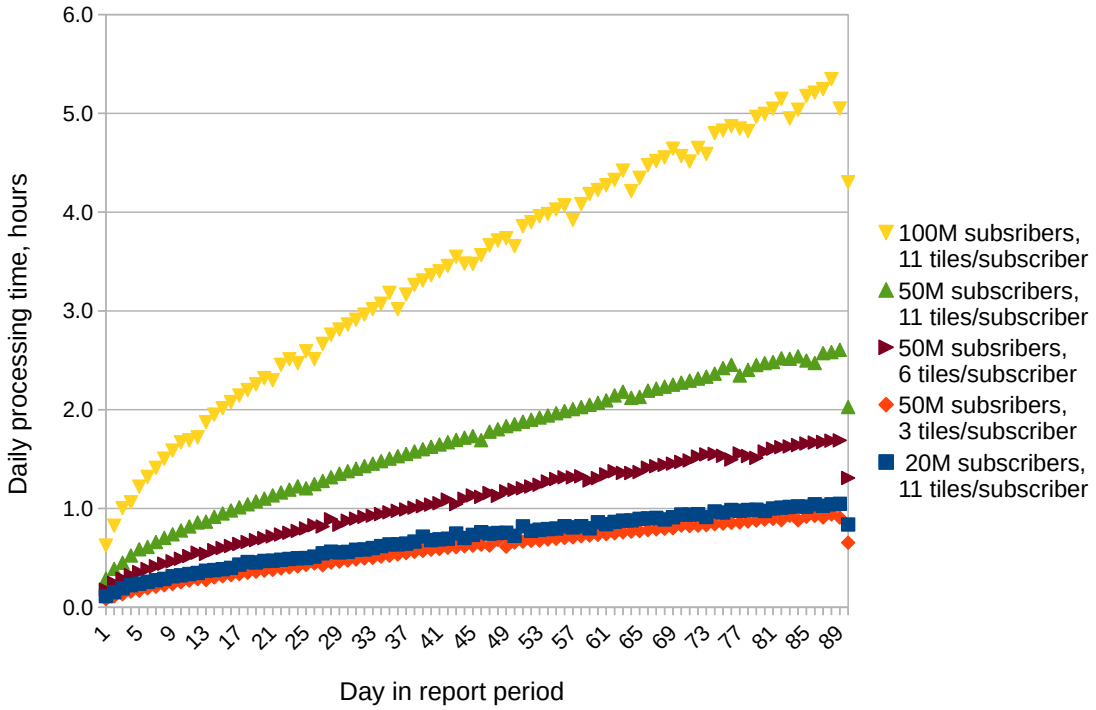

Figure 5. Execution times for each day during the 90 day test period, for various combinations of emulated subscriber counts and average tile count per subscriber. Last period time includes also the final report calculations. Vertical axis: execution time to import and aggregate the H file of a given day (hours); horizontal axis: day number in report date range 1..90.

back to disk. The time to read and write encrypted data is significant, thus the execution time is reduced when this work can be omitted. The execution time of the last period includes also final

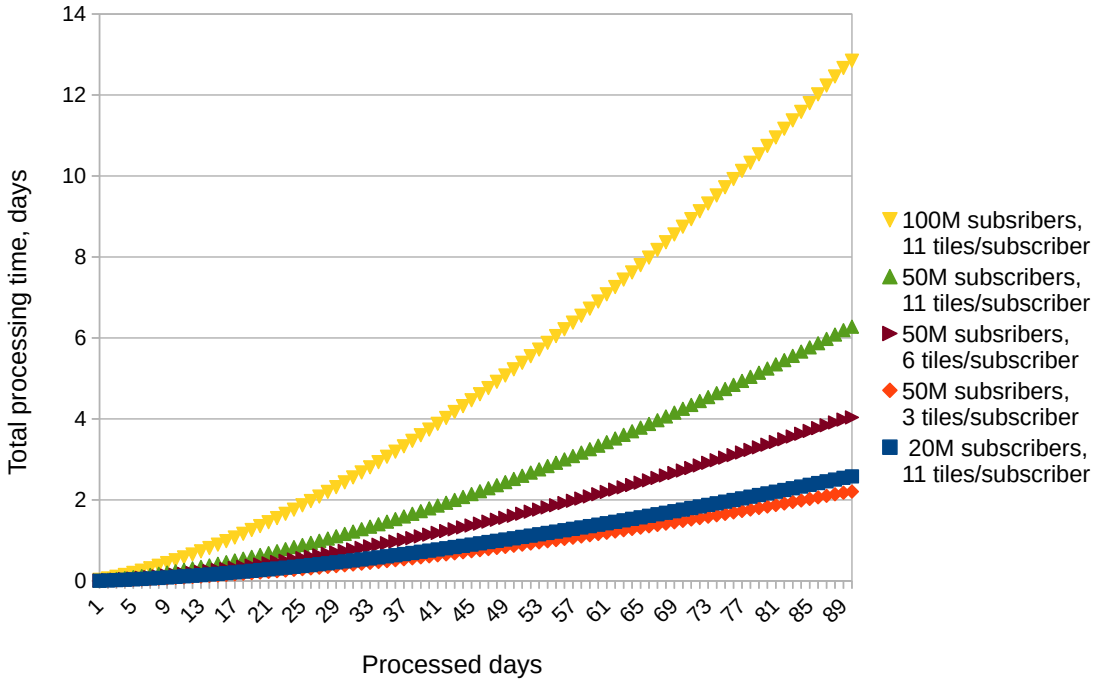

Figure 6. Cumulative execution times over the 90 day test period, for various combinations of emulated subscriber counts and average tile count per subscriber. Vertical axis: cumulative execution time (days); horisontal axis: day number in report date range 1..90.

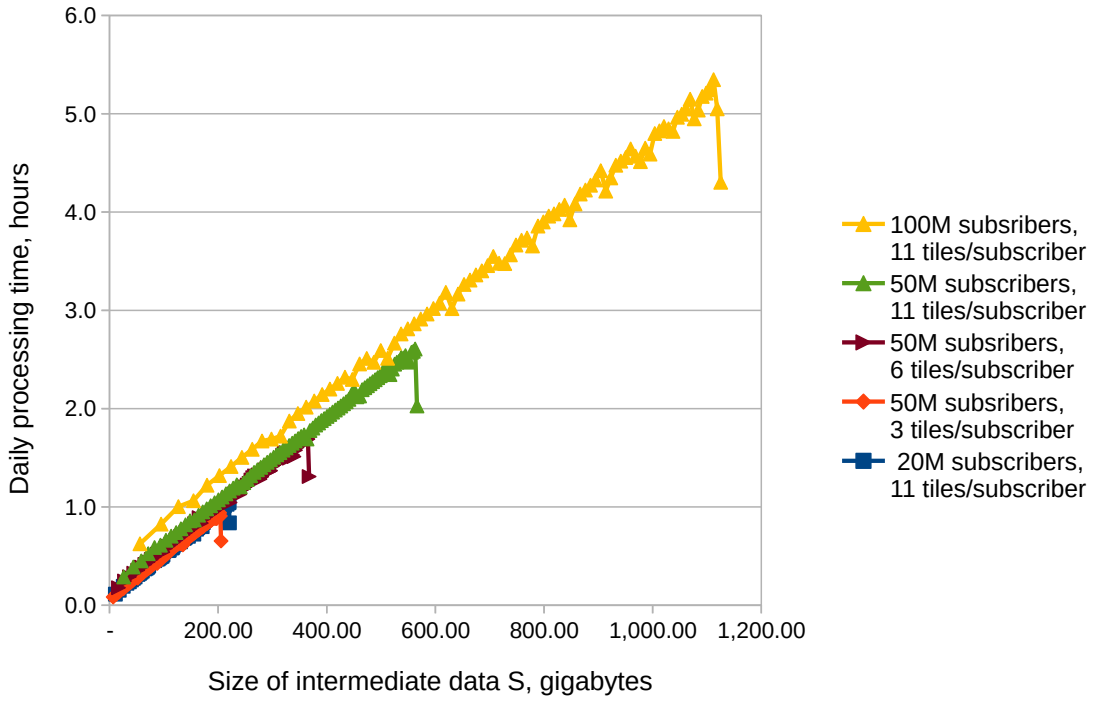

Figure 7. Relationship between data import time and intermediate data file S size (GBytes), for various combinations of emulated subscriber counts and average tile count per subscriber. Vertical axis: execution time for import of 1 day H data (hours); horisontal axis: size of intermediate results S in GBytes.

report calculations, but these take significantly less time than H import and aggregation.

Figure 7 shows the dependency between the execution time and the size of the aggregated intermediate data S. In different runs the execution times are very similar for the same S size. The figure illustrates the fact that intermediate file reads and writes are the largest contributors to the execution time.

The practically linear relation visible on Figure 7 indicates that the input datasets used in load tests did not push the Solution to scalability limits.

4.4.3 Total Execution Time of Calculations

In the main usecase the calculations are done daily as the data becomes available and total times are not of no direct concern. Total times shall be considered when recalculating all past data at once.

Figure 6 shows the cumulative execution time for different reporting period durations (in days), separately for different combinations of subscriber count and average daily tile count per subscriber. Total recalculation of all 90 days data would take slightly less than 7 days for 50 million users. For reduced tile count or user count the computation times would be lower accordingly.

4.4.4 Discussion of the Results

The most significant factor for performance was the size of the intermediate S data. The performance was mostly linear relative to the size of S. The linear relation indicates that the implementation scales well within size limits of the tested data.

The size of S depends on two factors – the subscriber count and the average unique tile count per subscriber. The subscriber count is an externally controlled parameter. The number of stored unique tiles can be reduced in the system but it affects statistical accuracy.

4.4.5 Discussion of Performance Improvement Ideas

There are several known ideas for further improvement of performance. Some relatively straightforward improvements are described below for consideration in future work. One shall also bear in mind that the usecases implemented are only proof-of-concept example usecases. Real algorithms could be significantly different and the most useful optimisations might also differ.

4.4.5.1 Modify Statistical Calculations Algorithm

In the Solution Architecture deliverable the maximum data volume is specified as 200 tiles/person on average. In this PoC we did not implement any mechanisms to keep this parameter within the specified range. In practical deployments it would be desirable to build some safeguards into the algorithm to limit the data size, e.g. detect the subscribers with abnormally large tile counts and discard the tiles with the lowest weights. Effects on statistical accuracy need to be evaluated also, preferably on real data.

4.4.5.2 Replace IO Library

In the current implementation the encrypted S file is created through the Intel® Protected File System Library (PFSL) which is part of the Intel® SGX SDK. This library supports a rich set

of features including random I/O. These features introduce a significant performance overhead. The implementation however reads intermediate results S files sequentially and writes new S files sequentially. Internal experiments showed that more than half of the execution time is spent on reading and writing the intermediate results S.

In the Solution, the PFSL is used as it provides a readily available interface for file storage, and a custom replacement would introduce more code to audit. Hence for this proof of concept less code (PFSL) was favored over higher performance (custom file crypto).

In the current implementation more than 50% of time was spent on reading and writing intermediate data S. We estimate that ca 25-35% performance improvement can be achieved by replacing the IO library.

4.4.5.3 Parallel Processing

The analytics enclave of the Solution is single threaded. All solution code runs on one server node. Introducing multithreading might improve the performance.

Multithreading increases the complexity of the code, and any synchronisation errors are especially dangerous and exploitable in Intel® SGX enclaves[3]. Hence for the proof of concept, multithreading was ruled out. A later version might introduce multithreading, but due to the peculiarities of Intel® SGX[4], any prediction of possible performance improvements is rather difficult.

The sample usecase algorithms are easily parallelised; so it is possible to increase the throughput of the Solution by means of distributed solution running on several server nodes.

[3] https://link.springer.com/chapter/10.1007/978-3-319-45744-4_22

[4] https://arxiv.org/pdf/2010.13216.pdf