USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

GraphAce: Secure Two-Party Graph Analysis Achieving Communication Efficiency

Jiping Yu, *Tsinghua University and Ant Group;* Kun Chen, *Ant Group;* Yunyi Chen and Xiaoyu Fan, *Tsinghua University and Ant Group;* Xiaowei Zhu and Cheng Hong, *Ant Group;* Wenguang Chen, *Tsinghua University and Ant Group*

https://www.usenix.org/conference/usenixsecurity25/presentation/yu-jiping

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix of "GraphAce: Secure Two-Party Graph Analysis Achieving Communication Efficiency"

Jiping Yu [1,2,*], Kun Chen [2], Yunyi Chen [1,2,*], Xiaoyu Fan [1,2,*], Xiaowei Zhu [2], Cheng Hong [2], and Wenguang Chen [1,2]

[1] Tsinghua University
[2] Ant Group

A Artifact Appendix

A.1 Abstract

The paper proposes GraphAce, an efficient secure two-party graph analysis framework. For each iteration, it achieves low complexities of $\Theta(|V|)$ communication, breaking the $\Omega(|V|+|E|)$ lower bound of previous secure solutions, and $\Theta(|V|+|E|)$ computation, which is the same as insecure methods. Evaluations show that GraphAce exceeds previous methods by up to tens of thousands of times in speed and saves up to 99.99% communication, depending on the application and the network. This artifact is anticipated to satisfy: (Available) the source code of GraphAce and other materials mentioned in the "Open Science" Section are made available; (Functional) the artifact contains the necessary instructions and scripts for the software to be executed successfully; (Reproduced) the artifact can reproduce the main claims of the paper, i.e. GraphAce outperforms the existing work GraphSC in terms of communication and execution time, for various graphs, applications, and network configurations.

One-stop quick start (for Ubuntu systems):

```
sudo apt install docker.io xz-utils
wget -O graphace.tar.xz "https://zenodo.org/\
records/15009743/files/graphace.tar.xz"
cat graphace.tar.xz | xz -d | tar x
cd graphace
sudo docker build -t graphace .
sudo docker run \
  --cap-add SYS_NICE --cap-add NET_ADMIN \
  -v ./results:/app/results -it graphace \
  python3 run.py --preset toy
```

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

*Work is done during the research internship at Ant Group.

A.2.2 How to access

The artifact is available at https://zenodo.org/records/15009743. (Note that this is an updated version of https://zenodo.org/records/14720419.)

A.2.3 Hardware dependencies

The artifact requires a machine that meets these requirements:

- **CPU**: x86-64. For more precise replication of the authors' results, high-performance server CPUs are recommended (the authors employed two Intel Xeon 8380 CPUs, providing a total of 80 cores or 160 threads). However, PC or laptop CPUs are generally sufficient to confirm the artifact's operational functionality.
- **RAM**: varies with the experiment. The most extensive experiment requires 768 GB of memory. However, 128 GB is sufficient to validate the major claims. The minimal RAM necessary to conduct a meaningful comparative experiment is 32 GB.
- **Disk**: at least 32 GB free space.
- **OS**: Linux (distribution is not restricted).
- **Network**: an Internet connection.

A.2.4 Software dependencies

The artifact is available in `.tar.xz` format, necessitating decompression capabilities of the evaluator. Docker is required to evaluate this artifact. For example, on Ubuntu systems, the required dependencies can be installed using the command `sudo apt install docker.io xz-utils`.

Docker handles all specific dependencies related to compilation, thus freeing the evaluator from these concerns.

A.2.5 Benchmarks

The artifact operates independently of external data sets. It relies solely on synthetic data that the program generates according to the specified data size.

A.3 Set-up

A.3.1 Installation

For artifact installation, run the commands listed below, assuming that `graphace.tar.xz` has been downloaded to the present working directory.

```
cat graphace.tar.xz | xz -d | tar x
cd graphace
sudo docker build -t graphace .
```

This is expected to finish in 20 minutes. Should the installation encounter issues owing to an external factor (such as an unstable Internet connection preventing the download of a dependency), it may be beneficial to clear the Docker build cache and attempt the process again.

A.3.2 Basic Test

To perform a basic test, execute the following:

```
sudo docker run \
  --cap-add SYS_NICE --cap-add NET_ADMIN \
  -v ./results:/app/results -it graphace \
  python3 run.py \
    --framework graphace --graph scale14 \
    --application pr --network 1DC
```

This command executes the PageRank application using GraphAce on the scale-14 graph over a 1-DC network. The completion is expected in one minute. Upon success, it provides information on execution time and the amount of communication, indicating the artifact's successful installation.

Should you be interested, the command is thoroughly explained as follows.

- The `--cap-add` parameters are essential for the artifact's proper functionality. Specifically, `SYS_NICE` permits the regulation of NUMA accesses within the Docker container, and `NET_ADMIN` facilitates the manipulation of bandwidth and network latency. These networking effects are confined to the container and do not impact programs external to the Docker environment.
- The `-v` parameter ensures that running logs are accessible in the `results` directory outside the container.
- The last four parameters are configurable, which is especially useful if the evaluator is interested in a specific experiment but does not wish to conduct a whole experiment preset (E1 to E5) below. In detail,
  * `--framework` specifies the framework to run. It can be `graphace` or `graphsc`.
  * `--graph` specifies the input graph. GraphSC supports `scale14`, `scale17`, and `scale20`, while GraphAce supports `scale{14,17,20,24,27}` and `scale{14,17,20}{II,III,IV,V}` (for Table 3 and Figure 8). There is no `scale20I`, for example, because it is actually the same as `scale20`.
  * `--application` specifies the application to run. It can be `pr`, `spmv`, `sslp`, `wcc`, or `mssp`.
  * `--network` specifies the network configuration. It can be `1DC`, `interDC`, or `interContinent`.

A.4 Evaluation workflow

A.4.1 Major Claims

The artifact is expected to support the major claims presented in the paper, as outlined below.

(C1): GraphAce decreases the amount of communication per iteration compared to GraphSC, in multiple applications and varying graph scales. This is demonstrated by any of the experiments (E1) to (E5), whose results are illustrated in Figure 6 of Section 6.2.

(C2): GraphAce decreases iteration time in most scenarios compared to GraphSC, in multiple applications, varying graph scales, and different network conditions. This is demonstrated by any of the experiments (E1) to (E5), whose results are illustrated in Figure 7 of Section 6.2.

(C3): GraphAce consistently provides communication and time savings compared to GraphSC for graphs with the same scale but varying average degrees, as observed in the two representative applications, PR and MSSP. This is demonstrated by any of the experiments (E1) to (E5), whose results are illustrated in Figure 8 of Section 6.3.

A.4.2 Experiments

Rather than presenting a separate experiment for each figure, we present five "presets," each of which can at least partially support all three major claims. The rationale behind this is the substantial time required to fully reproduce each subfigure; the quickest takes 6 hours, whereas the slowest takes up to 2 days, and we have over twenty subfigures. Therefore, we offer five experimental presets with completion times ranging from 10 minutes to 10 days. This allows the evaluator to select an appropriate preset based on their compute-time budget, to derive the most meaningful results within the constraints of limited execution time.

Experiment preset	Compute time[1]	Min. RAM	Support (C1)?	Support (C2)&(C3)?
(E1) Toy	10 min.	32G	Limited	Limited
(E2) Small	2.5 hours	32G	Partial	Partial
(E3) Medium	15 hours	128G	Yes	Partial
(E4) Large	75 hours	128G	Yes	Yes
(E5) XLarge	10 days	768G	Yes, better speedup ratio	

Table 1: Experiment presets

[1]The compute time is estimated on the basis of the hardware used by the authors, specifically two Intel Xeon 8380 CPUs with a total of 80 cores or 160 threads. If the evaluator uses a lower performance machine, it is expected that additional computation time will be required for the experiments.

Table 1 provides data on the five presets available. Larger presets necessitate increased computational time and RAM but can yield more thorough results (e.g., encompassing more applications and network configurations) and an improved ratio, as GraphAce displays enhanced complexity and benefits more from larger inputs. Therefore, selecting the largest feasible preset is advised to more accurately replicate the results presented in the paper. Should you need to stop execution prematurely, rest assured that an intermediate CSV file containing all results up to that moment is generated at the end of each run, enabling the evaluator to review partial findings even if the script does not run to completion.

In the presets "Toy" and "Small", the resulting data for Figure 8 refer to graphs of scale 14, rather than scale 20 in the paper. Analogously, the "Medium" and "Large" presets correspond to a scale of 17. *The speedup ratio might be less advantageous than reported in the paper if the utilized CPU exhibits lower performance than the one used by the author, or if only smaller presets are executed.*

(E1): [Toy preset] [**10 compute-minutes + 32 GB RAM**]: This preset aims to provide a quick glimpse of the results. It is expected to verify (C1) to (C3), but only on a single application (PR), a single network setting (1-DC), and limited graph sizes (up to scale 20 for GraphAce and scale 14 for GraphSC).
How to: Execute the command and wait for the results.
Preparation: No specific preparation is needed other than to complete the basic test.
Execution: Execute the following command:

```
sudo docker run \
  --cap-add SYS_NICE --cap-add NET_ADMIN \
  -v ./results:/app/results -it graphace \
  python3 run.py --preset toy
```

Results: The results will appear in the `results` directory as a CSV file, which can indicate that (part of C1) GraphAce decreases the communication amount for PR, (part of C2) GraphAce shortens the execution time for PR 1-DC, and (part of C3) this benefit remains consistent across graphs with varying average degrees.

(E2): [Small preset] [**2.5 compute-hours + 32 GB RAM**]: This preset aims to provide a slightly more comprehensive view of the results. It is expected to verify (C1) to (C3), but only on two representative applications (PR and MSSP), two network settings (1-DC and Inter-DC), and limited graph sizes (up to scale 20 for GraphAce PR and scale 14 for the others).
Execution: `sudo docker run \`
`  --cap-add SYS_NICE --cap-add NET_ADMIN \`
`  -v ./results:/app/results -it graphace \`
`  python3 run.py --preset small`
Results: The results will appear in the `results` directory as a CSV file, which can indicate that (part of C1) GraphAce decreases the communication amount for PR and MSSP, (part of C2) GraphAce shortens the execution time for PR/MSSP 1-DC/Inter-DC, and (part of C3) this benefit remains consistent across graphs with varying average degrees.
It should be noted that GraphAce could potentially operate at a slower speed compared to GraphSC for MSSP 1-DC, particularly for the constrained graph scale 14, as mentioned in the paper.

(E3): [Medium preset] [**15 compute-hours + 128 GB RAM**]: This preset is expected to verify (C1) to (C3) for all the five applications, but only on two network settings (1-DC and Inter-DC), and limited graph sizes (Scale 24 for GraphAce PHE, Scale 17 for GraphAce FHE and GraphSC 1-DC, and Scale 14 for GraphSC Inter-DC).
Execution: `sudo docker run \`
`  --cap-add SYS_NICE --cap-add NET_ADMIN \`
`  -v ./results:/app/results -it graphace \`
`  python3 run.py --preset medium`
Results: The results will appear in the `results` directory as a CSV file, which can indicate that (C1) GraphAce decreases the communication amount, (part of C2) GraphAce shortens the execution time for 1-DC/Inter-DC, and (part of C3) this benefit remains consistent across graphs with varying average degrees.

(E4): [Large preset] [**75 compute-hours + 128 GB RAM**]: This preset is expected to fully verify all major claims (C1) to (C3).
Execution: `sudo docker run \`
`  --cap-add SYS_NICE --cap-add NET_ADMIN \`
`  -v ./results:/app/results -it graphace \`
`  python3 run.py --preset large`
Results: The results will appear in the `results` directory as a CSV file, which fully supports (C1) to (C3).

(E5): [XLarge preset] [**10 compute-days + 768 GB RAM**]: This preset is expected to fully verify all major claims (C1) to (C3). In comparison to (E4), it is anticipated that this will deliver a greater communication saving ratio or an increase in execution time speedup, due to the larger graph scale and GraphAce's complexity improvement.
Execution: `sudo docker run \`
`  --cap-add SYS_NICE --cap-add NET_ADMIN \`
`  -v ./results:/app/results -it graphace \`
`  python3 run.py --preset xlarge`
Results: The results will appear in the `results` directory as a CSV file, which fully supports (C1) to (C3).

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.