Artifact Submission Guide for Reproducibility

Supplementary Material for "StarfishDB: A Query Execution Engine for Relational Probabilistic Programming"

Ouael Ben Amara[*] Sami Hadouaj[†] Niccolò Meneghetti[‡]

1 Introduction

This document provides a step-by-step guide for reproducing the experiments presented in our paper, "StarfishDB: A Query Execution Engine for Relational Probabilistic Programming." Reproducing these experiments is challenging due to the system's complexity and its various dependencies. Most importantly, StarfishDB's use of just-in-time compilation, which requires significant customization of the toolchain and development environment. By following our instructions, reviewers will be able to transform all the datasets mentioned in the "Datasets" section into the plots presented in the "Experiments" section of our paper. It's important to note that this entire process will take a few days to complete. To facilitate reproducibility and troubleshooting, we have divided the process into several scripts. This modular approach allows for easier identification and resolution of potential issues at each stage of the experiment. We appreciate the reviewers' efforts in verifying our work. If you encounter any difficulties during the reproduction process, please contact the authors for assistance. We are committed to supporting your efforts and ensuring successful reproduction of our experiments. Thank you for your time and contribution to the robustness of our research.

2 Overview

This guide outlines the process for reproducing our experimental results, which involves two main steps, each using its own Docker container:

1. Step I: Running baseline experiments with Gensim
2. Step II: Running StarfishDB and Mallet experiments, and generating the csv files that will be used to generate the plots
3. Step III: Generating the plots in pdf format from the generated csv files in the previous two steps

Due to different toolchain requirements, we use separate Docker images for step I and II. This approach ensures a consistent environment for each part of the experiment.

[*]benamara@umich.edu
[†]shadouaj@umich.edu
[‡]niccolom@umich.edu

Container Responsibilities and Outputs

Each container has specific responsibilities and produces distinct outputs:

- **Gensim Container (Step I):**
  - Responsibility: Run Gensim experiments for baseline comparisons
  - Output: CSV files with plot data for each dataset and configuration
  - Directory of outputs: `gensim_experiments/benchmarks/csv/`
  - File naming convention:

    `[DATASET]_[SPLIT]_[TOPICS]topics_A[ALPHA]_B[BETA]_`
    `NI[ITERATIONS]_NT[THREADS]_RND[SEED]_gensim_plotdata.csv`

- **StarfishDB Container (Step II):**
  - Responsibilities:
    1. Run StarfishDB experiments
    2. Run Mallet experiments
    3. Generate LaTeX plots comparing all systems (Gensim, StarfishDB, and Mallet)
  - Outputs:
    1. CSV files with benchmark results for StarfishDB and Mallet
    2. PDF reports with generated plots comparing all tools (20, 50, and 100 topics)
  - Directories:
    * StarfishDB and Mallet CSV files: `benchmarks/csv/`
    * Generated plot PDFs: `report/`
  - File naming conventions:
    * StarfishDB:

      `[DATASET]_[SPLIT]_[TOPICS]topics_A[ALPHA]_B[BETA]_`
      `NI[ITERATIONS]_NT[THREADS]_RND[SEED]_gammapdb_`
      `lda-inmemory-vrexpr[P]_plotdata.csv`

    * Mallet:

      `[DATASET]_[SPLIT]_[TOPICS]topics_A[ALPHA]_B[BETA]_`
      `NI[ITERATIONS]_NT[THREADS]_RND[SEED]_mallet_plotdata.csv`

    * Plot PDFs: `[TOPICS]TopicsExpReport.pdf`

The CSV files contain detailed performance data for each system, dataset, and configuration. The StarfishDB container combines these results to generate comparative plots in the final PDF reports.

Note : Split is TEST or Train

Note on Color Coding

Throughout this document, commands are color-coded to indicate the environment in which they should be executed:

- **Blue**: Commands to be run in the main operating system (OS), that hosts the two Docker containers.
- **Green**: Commands to be run in the first Docker container, that we refer to as "Gensim Container"
- **Orange**: Commands to be run in the second Docker container, that we refer to as "StarfishDB Container"

This color coding is designed to help you easily identify which environment each command belongs to, ensuring smooth execution of the experimental setup and analysis.

The following sections will guide you through setting up the required environments, running the experiments, and generating the final results. Please follow the instructions carefully, paying attention to the color-coded commands to ensure they are executed in the correct environment. The sections "Required Libraries and Dependencies" and "Datasets" provide an overview of the software components used in our experiments and details about the datasets used for benchmarking, respectively.

3 Setting up the Machine

We tested our experiment on an AWS instance and on our server. The technical properties of both computers are reported in the Appendix B.

1. Set up an EC2 instance on AWS with the following specifications:
   - 24 CPUs
   - 192 GB of RAM
   - 600 GB of storage
   - REDHAT 9
   - Instance type: z1d.6xlarge

   Note: Add the 600 GB of storage in the storage section when creating the AWS machine. Store the .pem key to use it to connect to the AWS machine.

2. Connect to your Amazon instance using the generated .pem key.
3. Install Docker:

```
sudo dnf install podman-docker

```

4. Install Git:

```
sudo yum install git

```

5. Clone the repository:

```
git clone https://github.com/nmeneghetti2/starfishdb_sigmod2024_ari.git
cd starfishdb_sigmod2024_ari


```

For a detailed explanation of the project's structure and a comprehensive overview of its various directories, please refer to Appendix C. This appendix provides insights into the organizational layout of the StarfishDB project.

4 Part I: Running Gensim Experiments

Our experimental process consists of two main steps, each utilizing its own Docker container due to different toolchain requirements:

1. Step I: Running baseline experiments with Gensim (covered in this section)
2. Step II: Running StarfishDB and Mallet experiments (covered in the next section)

The Gensim container is responsible for running single-threaded experiments on NYTIMES and PUBMED datasets. All Gensim experiments are conducted in the `gensim_experiments` directory. Follow these steps to run the Gensim experiments:

1. Navigate to the scripts directory:

```
cd gensim_experiments/scripts/

```

2. Build the Gensim Docker image:

```
sudo ./build_gensim_lda_docker_img.sh

```

   Note: This step should take approximately 10 minutes.

3. Run the script create_gensim_lda_docker_cont.sh to create the docker container. This will create a container with a name following this pattern: `gensim_lda_container_<username><hash>`. If you are using the AWS instance, the container will be named: `gensim_lda_container_ec2-usere82282fd`. In all cases, the script will print for you the command to be used in step 5.

```
    sudo ./create_gensim_lda_docker_cont.sh

```

4. Install and run tmux to prevent disconnection issues:

```
sudo yum install tmux
tmux

```

5. Within the tmux session, execute the Docker container, replacing `<container_name>` with the name returned in step 3:

```
sudo docker exec -it gensim_lda_container_ec2-usere82282fd bash

```

6. Once inside the container, navigate to the scripts directory:

```
cd scripts

```

7. Run the data preparation script:

```
./get_uci_datasets.sh

```

   Note: This step will take around 1 hour. The tmux session will prevent interruptions due to disconnections.

8. Run the Gensim benchmark:

```
./run_gensim_benchmark.sh

```

Note: This runs experiments for KOS, NYTIMES, and PUBMED datasets. The process takes several hours to finish:

- KOS: 2 minutes
- NYTIMES: around 7 hours (2 hours for training + 5 hours for converting into Mallet format)
- PUBMED: up to 48 hours (most time spent on indexing)

9. When the experiments are complete, exit the container:

```
exit

```

10. Exit the tmux session:

```
exit

```

You are now back in your original terminal session on the host machine, still in the `gensim_experiments/scripts/` directory.

To return to the project root directory, run:

```
cd ../..
```

You are now in the root directory of the StarfishDB project.

Results will be available in the `/app/benchmarks/csv` directory inside the container. An example filename for plot data is `KOS_test_20topics_A0.2_B0.1_NI20_NT1_RND123_gensim_plotdata.csv` for the KOS dataset.

Logs will be stored under: `/logs/gensim_benchmark_YYYYMMDD_HHMMSS.log` inside the container.

5 Part II: Running StarfishDB Experiments and Generating Plots

In the previous step, we used the first container to perform the Gensim experiments. In this second step, we will create a second container to run the StarfishDB and Mallet experiments and then create the required plots.

Overview of Experiments:

We have 2 main sets of experiments:

- **The single-threaded experiments**: will run LDA using starfishDB and mallet on NYTIMES and PUBMED using 20 topics. (approx 15 hours)
- **The multithreaded experiments** in which we have two sets:
  - **50 topics**: This will run starfishDB and mallet experiments on NYTIMES and PUBMED datasets using 1, 2, 4, and 8 threads. The LDA model has 50 topics in this set of experiments.

- **100 topics**: This will run starfishDB and mallet experiments on NYTIMES and PUBMED datasets using 2, 4, and 8 threads. The LDA model has 100 topics in this set of experiments.

The result of each experiment corresponds to a CSV file containing the execution time and perplexity reported at each iteration. Those files will be created in `benchmarks/csv`

Reminder: The commands in blue need to be run on the host machine

1. Open a terminal on the host machine and navigate to the project directory:

```
cd starfishdb_sigmod2024_ari/

```

2. run the script build_devenv_docker_img.sh, that will create a second docker image, to run StarfishDB and Mallet. (Approx 20 minutes)

```
sudo ./scripts/build_devenv_docker_img.sh

```

3. Run the script create_devenv_docker_cont.sh to create the docker container. This will create a container with a name following this pattern: `starfishdb_dev_env_container_<username><hash>`. If you are using the AWS instance, the container will be named: `starfishdb_dev_env_container_ec2 - user863a3113`

```
sudo ./scripts/create_devenv_docker_cont.sh

```

4. Run tmux to prevent disconnection issues:

```
tmux

```

5. Log into the container:

```
sudo docker exec -it starfishdb_dev_env_container_ec2-user863a3113 bash

```

Note: The exact container name may vary. Replace with the appropriate name if different.

6. Run the `main_script.sh`. This script will download all the required dependencies, compile them, compile StarfishDB, and run all the experiments using Mallet and StarfishDB:

```
cd scripts
source main_script.sh

```

Note: This will take around 4 days to complete.

7. Exit the container

```
        exit

```

8. Exit the temux session

```
        exit

```

Important:
IMPORTANT: Each command executed in the main_script.sh script will save its execution trace to a log file located in the log folder gammapdb_arrow/logs.

6 Details of main_script.sh

This section can be skipped if the previous section finished exectution without any problems.
In case of a network error or script failure, you can re-execute the content of `main_script.sh` step by step.
Before proceeding, ensure that the data directory is empty. If not, remove its content.

1. Download and compile dependencies:

```
source scripts/get_deps.sh

```

   This step takes approximately 2.5 hours.

2. Download and preprocess data:

```
source get_uci_datasets.sh

```

   This step takes approximately 1 hour.

3. Run single-threaded experiments:

```
./scripts/run_lda_benchmarks.sh

```

   This step takes approximately 15 hours.

4. Run multi-threaded experiments with 50 topics:

```
./scripts/run_lda_benchmarksP50.sh

```

5. Run multi-threaded experiments with 100 topics:

```
./scripts/run_lda_benchmarksP100.sh

```

7 Generating Plots

To generate plots for the experiments:

1. On the host machine (not inside the container) and run:

```
        cd starfishdb_sigmod2024_ari/
        cp gensim_experiments/benchmarks/csv/* benchmarks/csv/

```

2. Go back to the the starfishDB container:

```
        sudo docker exec -it gensim_lda_container_ec2-usere82282fd bash

```

3. Run the script responsible for generating the plots from the csv files obtained by running all the experiments:

```
        source scripts/make_plots.sh

```

IMPORTANT: The plots will be in the the "report" folder. Each script will generate a pdf with its plots for a total of 3 pdf file: 20TopicsExpReport.pdf, 50TopicsExpReport.pdf and 100TopicsExpReport.pdf

7.1 Details of make_plots.sh

This section can be skipped if the previous section finished execution without any problems.

This section explains the contents of the make_plots.sh script. The instructions provided here are for informational purposes only - you do not need to actually run these commands. The goal is to ensure the content can be executed in a modular fashion, in case an unexpected error occurs.

1. Navigate to the reports directory:

```
cd report

```

2. Generate plots for single-threaded experiments (20 topics):

```
source create20TopicsReport.sh

```

3. Generate plots for multi-threaded experiments (50 topics):

```
source create50TopicsReport.sh

```

4. Generate plots for multi-threaded experiments (100 topics):

```
source create100TopicsReport.sh

```

8 Congrats!

If you made until here, Congrats!

9 Required Libraries and Dependencies

The provided script (`get_deps.sh`) will automatically handle the installation of all necessary dependencies. This section provides an overview of the main components and libraries used in our experiments. For a complete, exhaustive list of all dependencies, please refer to Appendix A at the end of this document.

The following key libraries and dependencies will be downloaded and installed by the provided scripts:

1. ClangJit (LLVM project with C++ JIT support)
   - Version: Commit 91084ef018240bbb8e24235ff5cd8c355a9c1a1e
   - Source: `https://github.com/hfinkel/llvm-project-cxxjit.git`
2. Apache Arrow
   - Version: 9.0.0
   - Source: `https://archive.apache.org/dist/arrow/arrow-9.0.0/apache-arrow-9.0.0.tar.gz`
3. Mallet (MAchine Learning for LanguagE Toolkit)
   - Version: v202108
   - Source: `https://github.com/mimno/Mallet.git`
4. TeX Live (for generating reports)
   - Version: 2024
   - Source: CTAN mirror

Both Docker containers (Gensim and StarfishDB) are based on CentOS 7.9.2009 and include essential development tools, Python 3, and various libraries necessary for the experiments. The host machine requires Docker, Python 3, and Git.

For a detailed list of all dependencies, including those for the host machine and both Docker containers, please refer to the Appendix A.

This section outlines the software components and libraries used in our experiments. The provided scripts automate all installation and configuration processes, ensuring a reproducible environment. The sections above will guide the reviewers on how to run the scripts.

10 Datasets

The experiments will use the following datasets. They will be downloaded and preprocessed using the `get_uci_datasets.sh script`. all downloaded from the UCI Machine Learning Repository:

1. KOS
   - Source: UCI dataset repository
   - Description: Blog entries from the Dailykos weblog
2. NYTimes
   - Original source: ldc.upenn.edu (Linguistic Data Consortium)
   - Description: A collection of New York Times news articles

- Statistics:
  - Documents (D): 300,000
  - Vocabulary size (W): 102,660
  - Total words (N): approximately 100,000,000

3. PubMed

- Original source: www.pubmed.gov
- Description: A collection of PubMed abstracts
- Statistics:
  - Documents (D): 8,200,000
  - Vocabulary size (W): 141,043
  - Total words (N): approximately 730,000,000

Appendix A: List of Dependencies

This appendix provides a list of all dependencies used in our experiments.

.1 Host Machine Requirements

- Docker
- Python 3
- Git
- Tmux or Screen

.2 Gensim Container (CentOS 7.9.2009)

- Development Tools group
- Python 3.6 (via SCL rh-python36)
- Git
- Sudo
- Wget
- Nano
- CMake3
- Ninja-build
- Screen
- Tmux
- Devtoolset-11
- Devtoolset-7
- Java 11 OpenJDK
- Ant
- Snappy
- GSL and GSL-devel
- Bzip2
- BC (Basic Calculator)

Python packages:

- NumPy
- Pandas

- Gensim 3.8.3
- SciPy
- Tqdm
- Matplotlib
- Smart_open 5.2.1
- Mpmath
- hmmlearn

.3 StarfishDB Container (CentOS 7.9.2009)

- Development Tools group
- CMake3
- Ninja-build
- Screen
- Tmux
- Devtoolset-11
- Devtoolset-7
- Git
- Python 3
- Sudo
- Java 11 OpenJDK
- Ant
- Nano
- Snappy
- GSL and GSL-devel
- Bzip2
- Wget
- BC (Basic Calculator)

Python packages:

- Mpmath
- NumPy
- Pandas

Additional components:

- TeX Live 2024

Appendix B: Machines configuration

This appendix provides an exhaustive technical specification of the machines that we used to run the experiments.

AWS Machine

Operating System Information

```
Linux ip-172-31-10-123.us-east-2.compute.internal
5.14.0-427.20.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Thu May 23 16:37:13
EDT 2024 x86_64 x86_64 x86_64 GNU/Linux

LSB Version:    n/a
Distributor ID: RedHatEnterprise
Description:    Red Hat Enterprise Linux 9.4 (Plow)
Release:        9.4
Codename:       n/a
```

CPU Information

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Address sizes:                   46 bits physical, 48 bits virtual
Byte Order:                      Little Endian
CPU(s):                          24
On-line CPU(s) list:             0-23
Vendor ID:                       GenuineIntel
Model name:                      Intel(R) Xeon(R) Platinum 8151 CPU @ 3.40GHz
CPU family:                      6
Model:                           85
Thread(s) per core:              2
Core(s) per socket:              12
Socket(s):                       1
Stepping:                        4
BogoMIPS:                        6799.99
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic
                                 sep mtrr pge mca cmov pat pse36 clflush
                                 mmx fxsr sse sse2 ss ht syscall nx pdpe1gb
                                 rdtscp lm constant_tsc arch_perfmon
                                 rep_good nopl xtopology nonstop_tsc cpuid
                                 aperfmperf tsc_known_freq pni pclmulqdq
                                 monitor ssse3 fma cx16 pcid sse4_1 sse4_2
                                 x2apic movbe popcnt tsc_deadline_timer aes
                                 xsave avx f16c rdrand hypervisor lahf_lm
                                 abm 3dnowprefetch pti fsgsbase tsc_adjust
                                 bmi1 avx2 smep bmi2 erms invpcid mpx
                                 avx512f avx512dq rdseed adx smap
```

```
                                  clflushopt clwb avx512cd avx512bw avx512vl
                                  xsaveopt xsavec xgetbv1 xsaves ida arat
                                  pku ospke
Hypervisor vendor:                KVM
Virtualization type:              full
L1d cache:                        384 KiB (12 instances)
L1i cache:                        384 KiB (12 instances)
L2 cache:                         12 MiB (12 instances)
L3 cache:                         24.8 MiB (1 instance)
NUMA node(s):                     1
NUMA node0 CPU(s):                0-23
Vulnerability Gather data sampling: Unknown: Dependent on hypervisor status
Vulnerability Itlb multihit:      KVM: Mitigation: VMX unsupported
Vulnerability L1tf:               Mitigation; PTE Inversion
Vulnerability Mds:                Vulnerable: Clear CPU buffers attempted,
                                  no microcode; SMT Host state unknown
Vulnerability Meltdown:           Mitigation; PTI
Vulnerability Mmio stale data:    Vulnerable: Clear CPU buffers attempted,
                                  no microcode; SMT Host state unknown
Vulnerability Retbleed:           Vulnerable
Vulnerability Spec rstack overflow: Not affected
Vulnerability Spec store bypass:  Vulnerable
Vulnerability Spectre v1:         Mitigation; usercopy/swapgs barriers and
                                  __user pointer sanitization
Vulnerability Spectre v2:         Mitigation; Retpolines, STIBP disabled,
                                  RSB filling, PBRSB-eIBRS Not affected
Vulnerability Srbds:              Not affected
Vulnerability Tsx async abort:    Not affected
```

Memory Information

```
              total        used        free      shared  buff/cache   available
Mem:          186Gi       1.9Gi       185Gi       9.0Mi       609Mi       184Gi
Swap:            0B          0B          0B
```

Disk Usage

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M     0  4.0M   0% /dev
tmpfs            94G   84K   94G   1% /dev/shm
tmpfs            38G  9.1M   38G   1% /run
/dev/nvme1n1p4  599G  125G  475G  21% /
/dev/loop2       64M   64M     0 100% /var/lib/snapd/snap/core20/2379
/dev/loop1       39M   39M     0 100% /var/lib/snapd/snap/snapd/21759
/dev/loop0       26M   26M     0 100% /var/lib/snapd/snap/latexml/83
/dev/nvme1n1p3  960M  168M  793M  18% /boot
/dev/nvme1n1p2  200M  7.1M  193M   4% /boot/efi
tmpfs            19G  4.0K   19G   1% /run/user/1000
```

Network Information

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
   default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group
   default qlen 1000
    link/ether 02:62:94:9c:57:2d brd ff:ff:ff:ff:ff:ff
    altname enp0s5
    altname ens5
    inet 172.31.10.123/20 brd 172.31.15.255 scope global dynamic noprefixroute
    eth0
       valid_lft 2621sec preferred_lft 2621sec
    inet6 fe80::62:94ff:fe9c:572d/64 scope link
       valid_lft forever preferred_lft forever
3: podman0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
   group default qlen 1000
    link/ether 8e:97:1f:56:3e:cc brd ff:ff:ff:ff:ff:ff
    inet 10.88.0.1/16 brd 10.88.255.255 scope global podman0
       valid_lft forever preferred_lft forever
    inet6 fe80::8c97:1fff:fe56:3ecc/64 scope link
       valid_lft forever preferred_lft forever
4: veth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
   podman0 state UP group default qlen 1000
    link/ether 1a:ce:12:1d:e3:44 brd ff:ff:ff:ff:ff:ff link-netns
    netns-8a488824-1097-dda6-c7d8-87947b581de2
    inet6 fe80::18ce:12ff:fe1d:e344/64 scope link
       valid_lft forever preferred_lft forever
```

Nemesis Machine

Operating System Information

```
Linux nemesis.engin.umd.umich.edu 4.18.0-553.16.1.el8_10.x86_64 #1 SMP Thu
Aug 1 04:16:12 EDT 2024 x86_64 x86_64 x86_64 GNU/Linux

LSB Version:    :core-4.1-amd64:core-4.1-noarch
Distributor ID: RedHatEnterprise
Description:    Red Hat Enterprise Linux release 8.10 (Ootpa)
Release:        8.10
Codename:       Ootpa
```

CPU Information

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              28
On-line CPU(s) list: 0-27
Thread(s) per core:  2
Core(s) per socket:  14
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
BIOS Vendor ID:      Intel(R) Corporation
CPU family:          6
Model:               85
Model name:          Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz
BIOS Model name:     Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz
Stepping:            4
CPU MHz:             2200.000
CPU max MHz:         3200.0000
CPU min MHz:         1000.0000
BogoMIPS:            4400.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            1024K
L3 cache:            19712K
NUMA node0 CPU(s):   0-27
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
                     cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht
                     tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
                     arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
                     cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx
                     smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1
                     sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave
                     avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault
```

```
epb cat_l3 cdp_l3 invpcid_single pti intel_ppin ssbd mba
ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid
ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms
invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx
smap clflushopt clwb intel_pt avx512cd avx512bw avx512vl
xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc
cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts hwp
hwp_act_window hwp_pkg_req pku ospke md_clear flush_l1d
arch_capabilities
```

Memory Information

```
              total        used        free      shared  buff/cache   available
Mem:          219Gi       9.2Gi       110Gi       7.0Mi        99Gi       208Gi
Swap:         8.0Gi       291Mi       7.7Gi
```

Disk Usage

```
Filesystem                 Size  Used Avail Use% Mounted on
devtmpfs                   110G     0  110G   0% /dev
tmpfs                      110G  336K  110G   1% /dev/shm
tmpfs                      110G  1.8M  110G   1% /run
tmpfs                      110G     0  110G   0% /sys/fs/cgroup
/dev/mapper/rhel-root       32G  6.1G   26G  20% /
/dev/sda2                 1006M  363M  644M  37% /boot
/dev/sda1                  200M  9.2M  191M   5% /boot/efi
/dev/mapper/rhel-scratch    16G   11G  5.1G  69% /scratch
/dev/mapper/rhel-tmp       4.0G   62M  4.0G   2% /tmp
/dev/mapper/rhel-var       100G   14G   87G  14% /var
/dev/mapper/rhel-home      1.3T  461G  872G  35% /home
tmpfs                       22G  380K   22G   1% /run/user/114296514
tmpfs                       22G  336K   22G   1% /run/user/114252542
overlay                    100G   14G   87G  14% /var/lib/containers/storage/
                                                 overlay/f39239e5ac083dd5af8c81
                                                 613c683bbc00bd8ddf1b0cfd86a060
                                                 1c4033c6cdac/merged
overlay                    100G   14G   87G  14% /var/lib/containers/storage/
                                                 overlay/b8ad4df7ea957e1fa62246
                                                 251db8fd138e883a03c07a79fb2ff3
                                                 0f20c0dacad6/merged
shm                         63M     0   63M   0% /var/lib/containers/storage/
                                                 overlay-containers/0924d7625ad
                                                 f9739f47274a6bc9c7b28fbed6e829
                                                 0afb5c9e2e06ade39141acb/
                                                 userdata/shm
overlay                    100G   14G   87G  14% /var/lib/containers/storage/
                                                 overlay/4b4a30944bd1858918d1f0
                                                 a4dbacc710835362f21772d0700a08
```

```
                                                  89042d575106/merged
shm                        63M     0   63M   0% /var/lib/containers/storage/
                                                  overlay-containers/6e514f78b07
                                                  f5dcc432c9752603795759b05667c3
                                                  c6a0ddf4c6f7736798d4a95/
                                                  userdata/shm
overlay                   100G   14G   87G  14% /var/lib/containers/storage/
                                                  overlay/695e59d4a12dc44834353
                                                  91ae8d81c2e60400248c167668b1c7
                                                  b3d1520cbc810/merged
```

Network Information

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
   default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
   default qlen 1000
    link/ether 20:67:7c:ef:fa:7c brd ff:ff:ff:ff:ff:ff
    altname enp100s0f0
    inet 141.215.12.21/24 brd 141.215.12.255 scope global noprefixroute eno1
       valid_lft forever preferred_lft forever
3: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
   group default qlen 1000
    link/ether 20:67:7c:ef:fa:7d brd ff:ff:ff:ff:ff:ff
    altname enp100s0f1
4: eno3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
   group default qlen 1000
    link/ether 20:67:7c:ef:fa:7e brd ff:ff:ff:ff:ff:ff
    altname enp100s0f2
5: eno4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
   group default qlen 1000
    link/ether 20:67:7c:ef:fa:7f brd ff:ff:ff:ff:ff:ff
    altname enp100s0f3
6: cni-podman0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
   state UP group default qlen 1000
    link/ether da:c7:f7:a9:98:5f brd ff:ff:ff:ff:ff:ff
    inet 10.88.0.1/16 brd 10.88.255.255 scope global cni-podman0
       valid_lft forever preferred_lft forever
    inet6 fe80::d8c7:f7ff:fea9:985f/64 scope link
       valid_lft forever preferred_lft forever
81: veth33729651@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue master cni-podman0 state UP group default
    link/ether 5a:4b:f6:72:d6:c5 brd ff:ff:ff:ff:ff:ff link-netns
    netns-8324ccdd-1f10-4fe4-755a-2800bca5b267
```

```
    inet6 fe80::584b:f6ff:fe72:d6c5/64 scope link
       valid_lft forever preferred_lft forever
110: veth958113e5@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
     noqueue master cni-podman0 state UP group default
    link/ether 9e:2d:ad:51:df:a8 brd ff:ff:ff:ff:ff:ff link-netns
    netns-ab210d78-69df-bfb1-c7d8-517c97df6e84
    inet6 fe80::9c2d:adff:fe51:dfa8/64 scope link
       valid_lft forever preferred_lft forever
```

Appendix C: Project Structure

This appendix provides an overview of the StarfishDB project structure, detailing the main components and their purposes.

.1 Root Directory

The root directory contains several important files:

- `CMakeLists.txt`: Main CMake configuration file
- `README.md`: Project readme file
- Various configuration and license files (`LICENSE`, `NOTICE.txt`)

.2 Key Directories

.2.1 src/

This directory contains the StarfishDB source code, including:

- Main application file: `gammapdb_arrow.cpp`
- Header files for various components (`BDExpr.h`, `VrdExpr.h`, etc.)
- Subdirectories for specific modules:
  - `hmm/`: Hidden Markov Model related files
  - `lda/`: Latent Dirichlet Allocation related files

.2.2 scripts/

Contains scripts for running different parts of the project and Docker-related files:

- Various shell scripts for building, running benchmarks, and managing dependencies
- `dockerfiles/`: Contains Dockerfiles for different environments

.2.3 report/

Contains scripts and LaTeX files for generating reports and plots:

- `create*TopicsReport.sh`: Scripts for creating different topic reports
- `*TopicsExpReport.tex`: LaTeX files for experiment reports

.2.4 benchmarks/

- The `csv` sub-directory contains the csv files that result from the exection of each experiment.

.2.5 gensim_experiments/

Contains files related to Gensim experiments:

- Similar structure to the root directory, with its own scripts, benchmarks, and configuration files

.2.6 Other Important Directories

- `extras/`: Additional components and utilities
- `external/`: External dependencies or libraries
- `libs/`: Directory for storing compiled libraries
- `logs/`: Directory for storing operation logs
- `patches/`: Contains modifications and adaptations for Mallet and ClangJIT

.3 Key Files

.3.1 In scripts/

- `main_script.sh`: Main script for running the entire experiment pipeline
- `get_deps.sh`: Script for downloading and installing dependencies
- `run_*_benchmarks.sh`: Scripts for running various benchmarks

.3.2 In src/

- `gammapdb_arrow.cpp`: Main application file
- Other header files and cpp files that contain the actual implementation of starfishDB