环境配置文档

本环境配置文档用于配置一个容器来：

- 安装pgvector-baseline
- 安装我们的改良版pgvector
- 配置ANN-benchmark测试pgvector-baseline和各种改进版本

以下的配置用于从头创建一个容器，你也可以联系 **刘宗熹** `2200013088@stu.pku.edu.cn` 来获取我们配置完成的autodl容器。

在以下的操作中，我们假设你的工作目录是 `{workspace}`

安装pgvector-baseline

回到工作目录 `cd {workspace}`

运行 `git clone https://github.com/pgvector/pgvector.git pgvevtor-baseline`

配置ANN-benchmark测试pgvector_ivfflat

环境配置

原版ann-benchmark是依靠在宿主机上为每个查询算法建立一个docker容器来测试的，但是目前我们的服务器本身就是一个docker容器。为了避免docker套docker的麻烦，且我们只需要测试pgvector_ivf，在测试不同版本的pgvector_ivf时，我们不再建立新容器，而是编译不同版本的vector.so扩展，并将其替换到postgresql中，然后运行ann-benchmark进行测试

具体配置步骤如下：

1. 拉取仓库

回到工作目录 `cd {workspace}`

运行 `git clone https://github.com/mindtravel/ann-benchmarks.git`

运行 `cd ann-benchmarks`

2. 创建并激活conda环境

运行 `conda create -n ann-benchmarks python==3.10.6`

运行 `conda activate ann-benchmarks`

3. 安装python依赖

运行 `pip install -r requirements.txt`

4. 安装各个ivf算法的依赖

测试ivf算法：运行 `./scripts/env/pgvector_ivf.sh` 安装pgvector_ivf算法环境（脚本参考 `ann_benchmarks/algorithms/pgvector_ivfflat` ）

运行测试

测试脚本在目录 `scripts/tests` 中，各个测试的内容如下：

- `compile.sh`： 编译pgvector baseline
- `pgvector_origin.sh`： 测试pgvector baseline（包含ivfflat多线程，ivfflat单线程）
- `test_all.sh`： 自动下载三个数据集 **SIFT, DEEP, TEXT**（其中TEXT是截取了一个子集）在所有数据集上运行所有测试并绘制图像到 `{workspace}/ann-benchmarks/results`

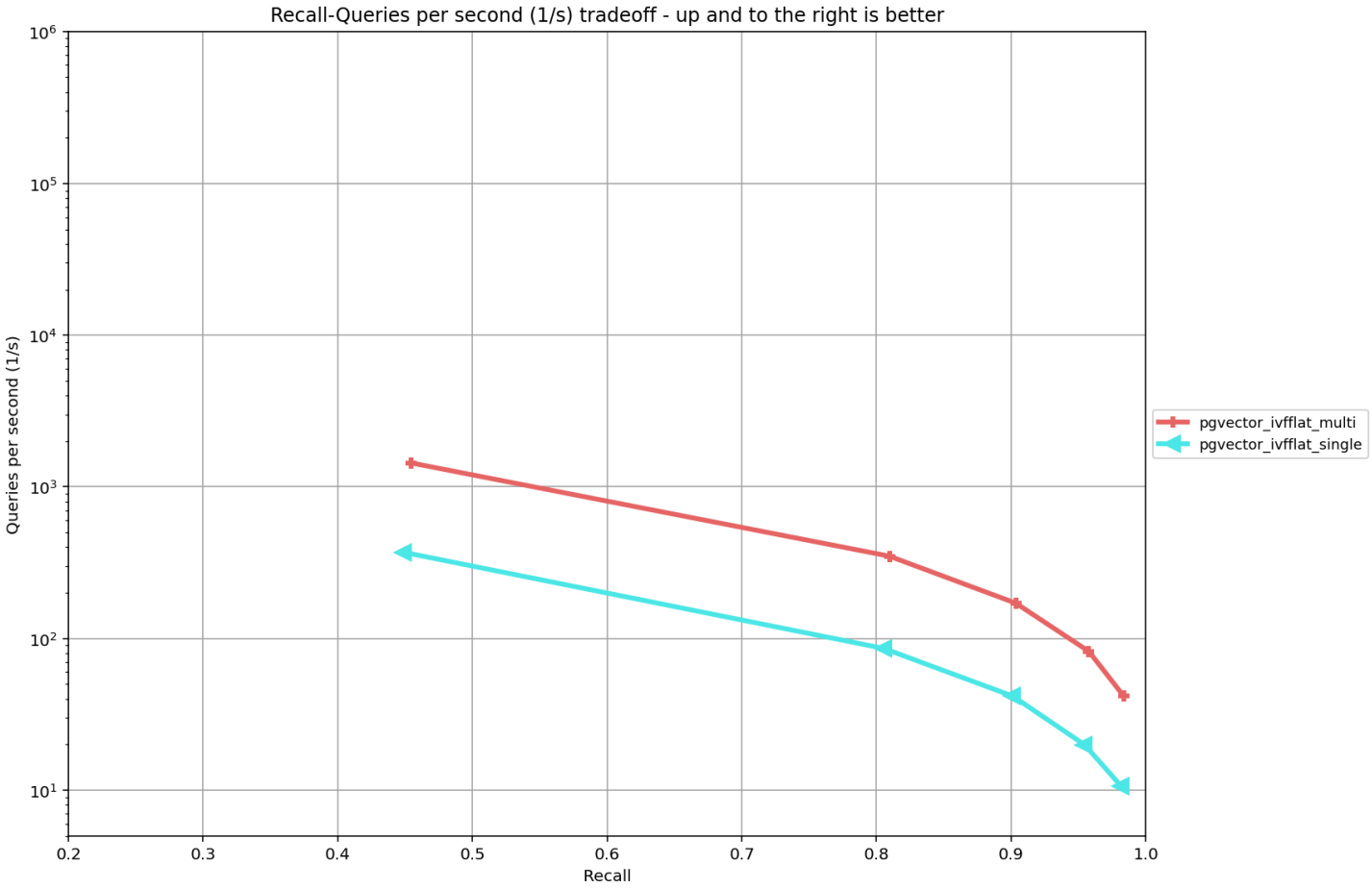
回到工作目录 `cd {workspace}` ，运行 `./scripts/tests/test_all.sh` 即可。整个脚本运行时间在一个小时左右（含下载数据集）。

运行完脚本后，命令行中会显示表格形式的测试结果，之后要查看这个表格结果，不需要重新运行 `test_all.sh` ，只需要运行 `python plot.py --x-scale linear --y-scale log --batch`

测试结果

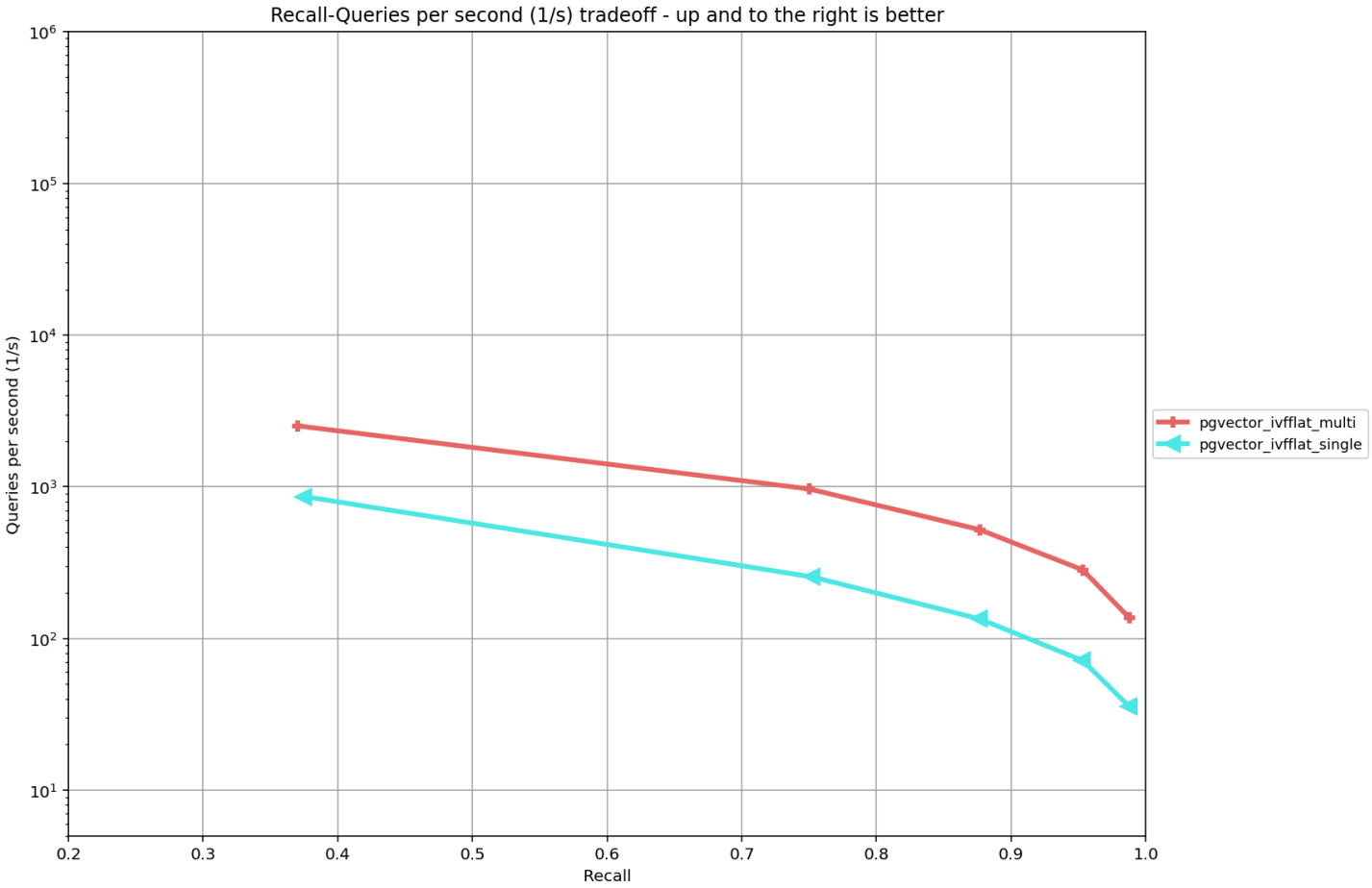
deep-image-96-angular-batch

```
writing output to results/deep-image-96-angular-batch.png
res:                                                    experience name       recall         qps         dpv
  0:                      PGVector ivfflat Ours (lists=3160, probes=1, workers=20)        0.455    1436.304     696.231
  1:                      PGVector ivfflat Ours (lists=3160, probes=5, workers=20)        0.810     348.776    2867.168
  2:                     PGVector ivfflat Ours (lists=3160, probes=10, workers=20)        0.904     170.727    5857.304
  3:                     PGVector ivfflat Ours (lists=3160, probes=20, workers=20)        0.958      82.195   12166.150
  4:                     PGVector ivfflat Ours (lists=3160, probes=40, workers=20)        0.984      41.573   24054.119
  5:                       PGVector ivfflat Origin (lists=3160, probes=1)        0.449     369.262    2708.102
  6:                       PGVector ivfflat Origin (lists=3160, probes=5)        0.805      85.909   11640.187
  7:                      PGVector ivfflat Origin (lists=3160, probes=10)        0.901      41.666   24000.467
  8:                      PGVector ivfflat Origin (lists=3160, probes=20)        0.955      19.844   50392.800
  9:                      PGVector ivfflat Origin (lists=3160, probes=40)        0.982      10.580   94517.038
```



sift-128-euclidean-batch

res:	experience name	recall	qps	dpv
0:	PGVector ivfflat Ours (lists=1000, probes=1, workers=20)	0.370	2523.140	396.332
1:	PGVector ivfflat Ours (lists=1000, probes=5, workers=20)	0.750	968.480	1032.546
2:	PGVector ivfflat Ours (lists=1000, probes=10, workers=20)	0.877	521.822	1916.363
3:	PGVector ivfflat Ours (lists=1000, probes=20, workers=20)	0.953	283.248	3530.476
4:	PGVector ivfflat Ours (lists=1000, probes=40, workers=20)	0.988	136.440	7329.230
5:	PGVector ivfflat Origin (lists=1000, probes=1)	0.375	862.653	1159.215
6:	PGVector ivfflat Origin (lists=1000, probes=5)	0.752	254.346	3931.657
7:	PGVector ivfflat Origin (lists=1000, probes=10)	0.877	134.641	7427.181
8:	PGVector ivfflat Origin (lists=1000, probes=20)	0.953	71.445	13996.759
9:	PGVector ivfflat Origin (lists=1000, probes=40)	0.988	35.517	28155.626

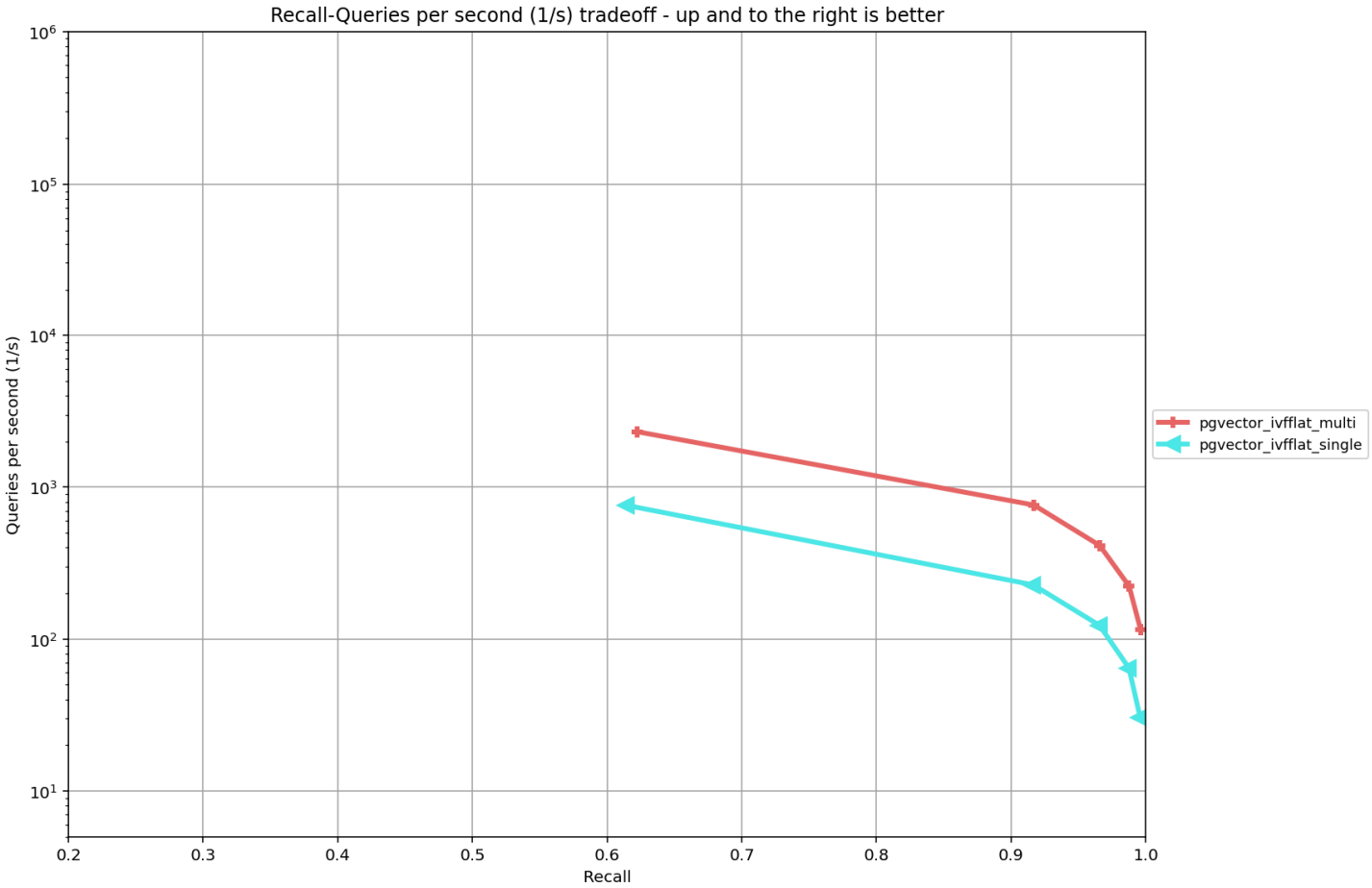


TEXT1M-200-angular-batch

```
Writing output to results/TEXT1M-200-angular-batch.png
```

res:

	experience name	recall	qps	dpv
0:	PGVector ivfflat Ours (lists=990, probes=1, workers=20)	0.622	2318.516	431.310
1:	PGVector ivfflat Ours (lists=990, probes=5, workers=20)	0.917	762.872	1310.837
2:	PGVector ivfflat Ours (lists=990, probes=10, workers=20)	0.966	413.391	2419.017
3:	PGVector ivfflat Ours (lists=990, probes=20, workers=20)	0.988	223.174	4480.809
4:	PGVector ivfflat Ours (lists=990, probes=40, workers=20)	0.996	115.699	8643.095
5:	PGVector ivfflat Origin (lists=990, probes=1)	0.614	759.670	1316.362
6:	PGVector ivfflat Origin (lists=990, probes=5)	0.916	227.270	4400.061
7:	PGVector ivfflat Origin (lists=990, probes=10)	0.966	123.152	8120.045
8:	PGVector ivfflat Origin (lists=990, probes=20)	0.987	64.500	15503.957
9:	PGVector ivfflat Origin (lists=990, probes=40)	0.996	30.317	32984.787


Recall-Queries per second (1/s) tradeoff - up and to the right is better
Queries per second (1/s)
Recall
pgvector_ivfflat_multi
pgvector_ivfflat_single