**Results for Physical Shards Experiments**

The purpose of these experiments was to determine the feasibility of using physical shards as a method for segmenting a relational database according to user data. We used a very basic reddit clone to study this. In this environment, users can make posts, comment to posts, and like posts and comments.

**The setup**

We created three different prototypes of this environment:

1. Vanilla reddit – This prototype had 5 tables – one for usernames and user ids, one for comments, one for posts, and two for keeping track of comment and post likes. They were all kept in sync relationally using cascading relational constraints.
2. Partitioned Reddit with Lookups – This prototype had all the tables in vanilla reddit (except the user table itself) partitioned by user ID (we used partitioning by LIST(), more info on this Postgres feature can be found [here](https://www.postgresql.org/docs/12/ddl-partitioning.html)). The only problem with this setup initially is that any operation that has to do with posts or comments would need to have the id of the post/comment *and* the id of the user that created it (since the posts and comments tables are partitioned by userID). To avoid needing this additional piece of information, we added lookup tables that contains information on which post and which comment where made by which user.
3. Partitioned Reddit with Coalesced tables – This prototype is essentially a merger of 1 and 2 so that it could benefit from the benefits of both (i.e. faster queries from 1 and physical separation of data form 2). Tables are once again partitioned by userID, but there are also normal tables for everything. Data is first placed in the normal tables, and triggers are used to copy the new information into the partitioned counterparts of the tables. All queries are made not on the partitioned tables, but on the normal “coalesced” tables. The user table is, once again, not partitioned.

**Practical Issues**

Partitioning brought a number of problems.

First of all, the ability to use referential constraints that refer to partitioned tables isn’t supported in Postgres. That means that we couldn’t use foreign keys to ensure that, for example, comments are referencing an existing post via a postID foreign key. So, if we wanted to implement cascading updates or deletes, we would have to use triggers. This introduces overhead and isn’t very clean. This problem exists because, even though a partitioned table can be treated as a single table, it’s really a group of tables. Referential constraints are meant to between only two tables, not one table and a group of tables.

It was also seen that in the prototypes that used partitions, user creation wasn’t thread safe. Because the creation of a new user also created new partitions for many tables, concurrent table creation often lead to deadlocks. We weren’t able to find a work around for this, so for out benchmarks we created all users first on a single thread before testing the performance of concurrent reads and writes of posts, comments, etc.

Additionally, it was seen that user creation was not a constant time operation. It took significantly more time to create a user if there was already a large amount of users in the database. On one thread, with just under 3000 users in the database, only 8 users could be created per second. We predict this is because partitions are not allowed to have overlapping constraints (for example, you cannot have a two partitions into which a row that has userID = 1 can be legally inserted into). So, with every new partition added, Postgres has to check the constraints of all the other partitions to ensure that there will be no overlap.

[Implementing partitions using inheritance](https://www.postgresql.org/docs/12/ddl-partitioning.html#DDL-PARTITIONING-IMPLEMENTATION-INHERITANCE) can potentially fix this. The downside of using inheritance is that triggers must now be used to insert the correct data into the correct partitions. The referential constraint problem also persists when using inheritance.

Regardless of the method used for partitions, Postgres does not recommend using more than a few hundred partitions. Many partitions will cause considerable performance drops in the system’s query planner (though, in our experience, this slowdown was only a couple milliseconds). Handling of partitioned tables, however, can use considerable memory resources, especially since the amount of locks required increases and the number of partitions increases.

**The experiment itself**

For all 3 environments, we used pgbench to first create a few thousand users, the proceeded to make a large amount of concurrent comment writes, post writes comment likes and comment thread reads. Lastly, we made a lot of concurrent requests to retrieve all the data in the database that refers to a particular user.

We ended up only comparing the vanilla reddit against the partitioned reddit with the coalesced tables. The partitioned reddit that contained only lookup tables was essentially unusable because querying for all comments that reference to a particular post required looking through the indexes of each comment table partition. Even if hash indexes are used, this takes several seconds per query.

**Summary of results**

|  |  |  |
| --- | --- | --- |
|  | **Vanilla Reddit** | **Partitioned Reddit** |
| **Creating 3000 users** | 2887 tps | 14 tps |
| **Having 4 clients each perform 20k pgbench transactions (across 2 threads)** | 857 tps | 514 tps |
| **Having 5 clients request all user data 1500 times (across 2 threads)** | 8802 tps | 711 tps |
| **Final Size (in MB)** | 105 | 775 |

It can be seen that, really, the use of partitioned tables has no benefit. Apart from practical issues like the lack of thread safety and the referential constraint limitations, the use of physical shards is considerably slower and much more memory intensive.

The raw output of the benchmarks can be found in the appendix

**Appendix**

## Vanilla

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/create\_users\_bench.sql" --jobs 1 --transactions 3000 --client 1 --progress 5 --no-vacuum vanilla\_reddit

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/create\_users\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 1

number of threads: 1

number of transactions per client: 3000

number of transactions actually processed: 3000/3000

latency average = 0.346 ms

latency stddev = 0.328 ms

tps = 2876.667799 (including connections establishing)

tps = 2887.679615 (excluding connections establishing)

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/read\_write\_bench.sql" --jobs 2 --transactions 20000 --client 4 --progress 5 --no-vacuum vanilla\_reddit

progress: 5.0 s, 830.0 tps, lat 4.804 ms stddev 1.702

progress: 10.0 s, 874.6 tps, lat 4.571 ms stddev 0.281

progress: 15.0 s, 875.2 tps, lat 4.569 ms stddev 0.346

progress: 20.0 s, 857.8 tps, lat 4.665 ms stddev 0.591

progress: 25.0 s, 857.0 tps, lat 4.663 ms stddev 0.479

progress: 30.0 s, 866.4 tps, lat 4.618 ms stddev 0.464

progress: 35.0 s, 864.6 tps, lat 4.624 ms stddev 0.609

progress: 40.0 s, 868.6 tps, lat 4.605 ms stddev 0.318

progress: 45.0 s, 827.0 tps, lat 4.834 ms stddev 0.974

progress: 50.0 s, 867.0 tps, lat 4.613 ms stddev 0.578

progress: 55.0 s, 867.4 tps, lat 4.610 ms stddev 0.361

progress: 60.0 s, 863.2 tps, lat 4.635 ms stddev 0.589

progress: 65.0 s, 856.4 tps, lat 4.670 ms stddev 0.625

progress: 70.0 s, 857.6 tps, lat 4.661 ms stddev 0.913

progress: 75.0 s, 847.8 tps, lat 4.718 ms stddev 0.652

progress: 80.0 s, 855.8 tps, lat 4.673 ms stddev 1.361

progress: 85.0 s, 873.9 tps, lat 4.563 ms stddev 0.306

progress: 90.0 s, 860.9 tps, lat 4.659 ms stddev 1.055

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/read\_write\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 4

number of threads: 2

number of transactions per client: 20000

number of transactions actually processed: 80000/80000

latency average = 4.656 ms

latency stddev = 0.794 ms

tps = 857.675289 (including connections establishing)

tps = 857.732146 (excluding connections establishing)

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/gdpr\_rights\_bench.sql" --jobs 2 --transactions 1500 --client 5 --progress 5 --no-vacuum vanilla\_reddit

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/gdpr\_rights\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 5

number of threads: 2

number of transactions per client: 1500

number of transactions actually processed: 7500/7500

latency average = 0.555 ms

latency stddev = 0.240 ms

tps = 8802.929521 (including connections establishing)

tps = 8854.387994 (excluding connections establishing)

Andersons-MacBook-Pro:~ andersonaddo$

nspname|relname |relative size|total size|

-------|-----------------------|-------------|----------|

public |posts |40 MB |60 MB |

public |comments |20 MB |31 MB |

public |posts\_owner\_id\_hash |7648 kB |7648 kB |

public |like\_index |7224 kB |7224 kB |

public |liked\_posts |2856 kB |5296 kB |

public |posts\_pkey |5280 kB |5280 kB |

public |comments\_post\_id\_hash |4624 kB |4624 kB |

public |comments\_poster\_id\_hash|4520 kB |4520 kB |

public |liked\_posts\_pkey |2416 kB |2416 kB |

public |comments\_pkey |1768 kB |1768 kB |

public |users |200 kB |320 kB |

public |users\_pkey |88 kB |88 kB |

public |liked\_comments\_pkey |8192 bytes |8192 bytes|

public |liked\_comments |0 bytes |8192 bytes|

public |comments\_comment\_id\_seq|8192 bytes |8192 bytes|

public |users\_id\_seq |8192 bytes |8192 bytes|

public |posts\_id\_seq |8192 bytes |8192 bytes|

public |top\_10\_posts |0 bytes |0 bytes |

## Coalesce

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/create\_users\_bench.sql" --jobs 1 --transactions 3000 --client 1 --progress 5 --no-vacuum coalesce\_gdpr\_reddit

progress: 5.0 s, 47.4 tps, lat 21.039 ms stddev 4.575

progress: 10.0 s, 33.6 tps, lat 29.773 ms stddev 7.356

progress: 15.0 s, 27.4 tps, lat 36.266 ms stddev 26.466

progress: 20.0 s, 26.4 tps, lat 38.133 ms stddev 3.442

progress: 25.0 s, 23.2 tps, lat 43.104 ms stddev 4.293

progress: 30.0 s, 21.2 tps, lat 46.979 ms stddev 4.501

progress: 35.0 s, 20.0 tps, lat 50.135 ms stddev 4.556

progress: 40.0 s, 19.2 tps, lat 52.179 ms stddev 3.674

progress: 45.0 s, 17.8 tps, lat 55.754 ms stddev 3.965

progress: 50.0 s, 16.8 tps, lat 59.275 ms stddev 3.883

progress: 55.0 s, 16.4 tps, lat 61.713 ms stddev 3.838

progress: 60.0 s, 15.2 tps, lat 65.017 ms stddev 3.083

progress: 65.0 s, 14.6 tps, lat 69.118 ms stddev 4.372

progress: 70.0 s, 13.8 tps, lat 72.636 ms stddev 3.939

progress: 75.0 s, 13.4 tps, lat 74.436 ms stddev 3.715

progress: 80.0 s, 13.0 tps, lat 77.119 ms stddev 3.492

progress: 85.0 s, 12.6 tps, lat 78.175 ms stddev 3.575

progress: 90.0 s, 12.6 tps, lat 79.670 ms stddev 2.339

progress: 95.0 s, 12.2 tps, lat 82.277 ms stddev 2.978

progress: 100.0 s, 11.8 tps, lat 84.983 ms stddev 3.466

progress: 105.0 s, 11.8 tps, lat 84.710 ms stddev 3.793

progress: 110.0 s, 11.2 tps, lat 88.993 ms stddev 3.792

progress: 115.0 s, 11.2 tps, lat 89.667 ms stddev 3.420

progress: 120.0 s, 10.8 tps, lat 92.363 ms stddev 3.989

progress: 125.0 s, 10.6 tps, lat 93.596 ms stddev 2.968

progress: 130.0 s, 10.6 tps, lat 95.182 ms stddev 3.019

progress: 135.0 s, 10.2 tps, lat 97.283 ms stddev 3.844

progress: 140.0 s, 10.2 tps, lat 97.486 ms stddev 2.228

progress: 145.0 s, 10.4 tps, lat 96.967 ms stddev 3.291

progress: 150.0 s, 10.2 tps, lat 98.127 ms stddev 2.258

progress: 155.0 s, 10.0 tps, lat 99.842 ms stddev 2.742

progress: 160.0 s, 9.8 tps, lat 103.101 ms stddev 9.141

progress: 165.0 s, 9.6 tps, lat 103.073 ms stddev 3.535

progress: 170.0 s, 9.6 tps, lat 104.157 ms stddev 2.336

progress: 175.0 s, 9.4 tps, lat 105.478 ms stddev 4.806

progress: 180.0 s, 9.4 tps, lat 107.069 ms stddev 2.038

progress: 185.0 s, 9.2 tps, lat 109.110 ms stddev 2.384

progress: 190.0 s, 9.0 tps, lat 110.986 ms stddev 2.614

progress: 195.0 s, 8.8 tps, lat 112.582 ms stddev 4.509

progress: 200.0 s, 8.8 tps, lat 113.858 ms stddev 3.273

progress: 205.0 s, 8.8 tps, lat 115.170 ms stddev 2.321

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/create\_users\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 1

number of threads: 1

number of transactions per client: 3000

number of transactions actually processed: 3000/3000

latency average = 68.671 ms

latency stddev = 29.449 ms

tps = 14.561551 (including connections establishing)

tps = 14.561831 (excluding connections establishing)

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/read\_write\_bench.sql" --jobs 2 --transactions 20000 --client 4 --progress 5 --no-vacuum coalesce\_gdpr\_reddit

progress: 5.0 s, 390.0 tps, lat 10.100 ms stddev 15.098

progress: 10.0 s, 456.6 tps, lat 8.847 ms stddev 8.302

progress: 15.0 s, 481.8 tps, lat 8.326 ms stddev 4.644

progress: 20.0 s, 434.0 tps, lat 9.216 ms stddev 18.729

progress: 25.0 s, 567.8 tps, lat 7.043 ms stddev 1.139

progress: 30.0 s, 543.2 tps, lat 7.364 ms stddev 2.972

progress: 35.0 s, 554.4 tps, lat 7.211 ms stddev 2.228

progress: 40.0 s, 558.8 tps, lat 7.160 ms stddev 1.365

progress: 45.0 s, 644.8 tps, lat 6.201 ms stddev 0.634

progress: 50.0 s, 502.2 tps, lat 7.962 ms stddev 13.043

progress: 55.0 s, 556.0 tps, lat 7.197 ms stddev 7.836

progress: 60.0 s, 562.0 tps, lat 7.116 ms stddev 3.600

progress: 65.0 s, 565.4 tps, lat 7.074 ms stddev 5.578

progress: 70.0 s, 631.2 tps, lat 6.334 ms stddev 1.821

progress: 75.0 s, 630.6 tps, lat 6.340 ms stddev 2.332

progress: 80.0 s, 457.2 tps, lat 8.755 ms stddev 11.586

progress: 85.0 s, 564.0 tps, lat 7.087 ms stddev 2.903

progress: 90.0 s, 486.2 tps, lat 8.228 ms stddev 3.317

progress: 95.0 s, 494.4 tps, lat 8.090 ms stddev 1.246

progress: 100.0 s, 561.6 tps, lat 7.123 ms stddev 1.704

progress: 105.0 s, 586.6 tps, lat 6.814 ms stddev 4.155

progress: 110.0 s, 559.0 tps, lat 7.157 ms stddev 6.160

progress: 115.0 s, 589.2 tps, lat 6.787 ms stddev 12.095

progress: 120.0 s, 631.2 tps, lat 6.337 ms stddev 2.085

progress: 125.0 s, 492.8 tps, lat 8.113 ms stddev 10.159

progress: 130.0 s, 470.4 tps, lat 8.508 ms stddev 5.040

progress: 135.0 s, 403.2 tps, lat 9.910 ms stddev 17.609

progress: 140.0 s, 420.4 tps, lat 9.520 ms stddev 16.760

progress: 145.0 s, 423.0 tps, lat 9.442 ms stddev 11.150

progress: 150.0 s, 444.6 tps, lat 8.988 ms stddev 6.609

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/read\_write\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 4

number of threads: 2

number of transactions per client: 20000

number of transactions actually processed: 80000/80000

latency average = 7.709 ms

latency stddev = 8.242 ms

tps = 514.127292 (including connections establishing)

tps = 514.140729 (excluding connections establishing)

Andersons-MacBook-Pro:~ andersonaddo$ pgbench --file "/Users/andersonaddo/Desktop/GDPR Malte Work/gdpr\_rights\_bench.sql" --jobs 2 --transactions 1500 --client 5 --progress 5 --no-vacuum coalesce\_gdpr\_reddit

progress: 5.0 s, 5.0 tps, lat 59.650 ms stddev 106.337

progress: 10.0 s, 1186.0 tps, lat 8.167 ms stddev 205.545

transaction type: /Users/andersonaddo/Desktop/GDPR Malte Work/gdpr\_rights\_bench.sql

scaling factor: 1

query mode: simple

number of clients: 5

number of threads: 2

number of transactions per client: 1500

number of transactions actually processed: 7500/7500

latency average = 7.003 ms

latency stddev = 182.917 ms

tps = 711.972536 (including connections establishing)

tps = 712.274087 (excluding connections establishing)

nspname|relname |relative size|total size|

-------|--------------------|-------------|----------|

public |posts |42 MB |53 MB |

public |comments |21 MB |27 MB |

public |like\_index |6328 kB |6328 kB |

public |liked\_posts |2952 kB |5496 kB |

public |posts\_pkey |5448 kB |5448 kB |

public |comments\_post\_id\_idx|4616 kB |4616 kB |

public |liked\_posts\_pkey |2520 kB |2520 kB |

public |comments\_pkey |1824 kB |1824 kB |

public |users |200 kB |320 kB |

shards |posts\_394 |184 kB |232 kB |

shards |posts\_1596 |184 kB |232 kB |

shards |posts\_136 |176 kB |224 kB |

shards |posts\_174 |72 kB |120 kB |

shards |posts\_1935 |72 kB |120 kB |

shards |posts\_2730 |72 kB |120 kB |

shards |posts\_2122 |64 kB |112 kB |

shards |posts\_2234 |64 kB |112 kB |

shards |posts\_2512 |64 kB |112 kB |

shards |posts\_2293 |64 kB |112 kB |

shards |posts\_1846 |56 kB |104 kB |

shards |posts\_1732 |56 kB |104 kB |

shards |posts\_1897 |56 kB |104 kB |

shards |posts\_2417 |56 kB |104 kB |

shards |posts\_637 |56 kB |104 kB |

shards |posts\_357 |56 kB |104 kB |

shards |posts\_1767 |48 kB |96 kB |

shards |posts\_2984 |48 kB |96 kB |

shards |posts\_999 |48 kB |96 kB |

shards |posts\_2717 |48 kB |96 kB |

shards |posts\_1604 |48 kB |96 kB |

shards |posts\_1704 |48 kB |96 kB |

(This goes on for all the thousands of partitions, though they start to reduce in size and stabilize at around 88kb per partition)