Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya
BarcelonaTech

Assignment Report

CPDS-MIRI

Opty: Optimistic Concurrency Control

This report is the representation of the work realized in the context of the **second** Distributed Systems's class assignment
MIRI — Fall Semester

11th December 2022

Realized by:
- **Keith Joseph D'souza**

Supervising Professor:
- **Dr Jordi Guitart Fernandez**

Contents

Introduction

This assignment is part of the Concurrence, Parallelism and Distributed Systems course provided in the context of the *Master in Innovation and Research in Informatics* at the *Universitat Politècnica de Catalunya- BarcelonaTech.*

I would like to take this opportunity to gratefully acknowledge the contribution of my professor **Dr Jordi Guitart Fernandez** towards my learning process and for taking the time to assess the quality of this work.

Moreover, the work presented throughout this report deals mainly with the Optimistic Concurrency Control(Opty) method, that aims to achieve concurrency between transactional systems using the optimistic approach. It consists of three phases, namely the Read phase, the Validation phase and the Write phase. It assumes that conflicts among transactions are rare events and hence does not use a lock on data. Therefore, they are deadlock free. The client starts a transaction by creating a transaction handler. The transaction handler takes care of the read and write operations by the client and also closes the transaction at the end. The validation of the transactions to check for conflicts is done by the validator. The validator receives the read and write sets from the transaction handler. If conflicts are detected, the conflicting transactions are aborted. If not, the data is copied from local store to the real store

The structure of this document is very similar to the one proposed by my professor. It will deal with the details of all the experiments carried out, with explanations of my decisions and figures showing the results I obtained through my experiments, then it will treat the open questions before presenting some personal opinions about the content of the assignment.

Finally, the programming language concerned is **Erlang**, and the code will be available in an attached set of files sent along with this report.

Experiments

1.1 Opty.

1.1.1 Number of Clients.

Impact of changing the number of clients while keeping the number of entries, read operations per transaction, write operations per transaction and duration constant.

The table 1 consists of varying number of clients and the corresponding success rate while keeping the other parameters constant. The figure 1 depicts a part of the output and figure 2 depicts a plot of the entire output. We can see that for a single client the success rate is 100%. As the number of clients increases the success rate decreases. Also, the margin of decrease in success rate decreases as the number of clients increases. We can also see that the success rate is quite similar for different clients as shown in figure 1.

- Entries: 9, Read Operations Per Transaction: 1, Write Operations Per Transaction: 1, Duration: 3s.

Client(s)	**Success(%)**
1	100.00
2	94.15
3	88.60
4	84.28
5	79.75
6	75.55
7	71.48
8	68.03
9	64.57
10	61.46
11	59.26
12	57.23
13	54.89
14	53.47
15	51.76
16	49.99
17	48.78
18	47.30
19	46.27
20	44.56

Table 1: Observations recorded after varying the number of clients.

```
2> opty:start(1,9,1,1,3).
Starting: 1 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:246516, OK:246516, -> 100.0 %
Stopped
ok
3> opty:start(2,9,1,1,3).
Starting: 2 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:161075, OK:151665, -> 94.15800093124321 %
1: Transactions TOTAL:161246, OK:151865, -> 94.18218126341118 %
Stopped
ok
4> opty:start(3,9,1,1,3).
Starting: 3 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:119045, OK:105485, -> 88.60934940568693 %
1: Transactions TOTAL:118681, OK:105064, -> 88.52638585788795 %
2: Transactions TOTAL:119039, OK:105545, -> 88.66421928947656 %
Stopped
ok
5> opty:start(4,9,1,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:100457, OK:84633, -> 84.24798670077745 %
2: Transactions TOTAL:100369, OK:84591, -> 84.28000677500025 %
4: Transactions TOTAL:100723, OK:84932, -> 84.32234941373866 %
1: Transactions TOTAL:100414, OK:84377, -> 84.02911944549565 %
Stopped
ok
6> opty:start(5,9,1,1,3).
Starting: 5 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:85836, OK:68230, -> 79.48879258120137 %
5: Transactions TOTAL:86011, OK:68702, -> 79.87582983571869 %
4: Transactions TOTAL:85761, OK:68267, -> 79.60145054278752 %
1: Transactions TOTAL:85927, OK:68526, -> 79.74908934327976 %
2: Transactions TOTAL:85934, OK:68533, -> 79.7507389391859 %
Stopped
ok
7> opty:start(6,9,1,1,3).
Starting: 6 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:71530, OK:53875, -> 75.31804837131274 %
5: Transactions TOTAL:71524, OK:54204, -> 75.78435210558693 %
2: Transactions TOTAL:71432, OK:53920, -> 75.48437674991601 %
6: Transactions TOTAL:71478, OK:54080, -> 75.65964352667953 %
1: Transactions TOTAL:71437, OK:54339, -> 76.06562425633776 %
4: Transactions TOTAL:71417, OK:53879, -> 75.44282173712142 %
Stopped
```

Figure 1: Output when varying the number of clients.

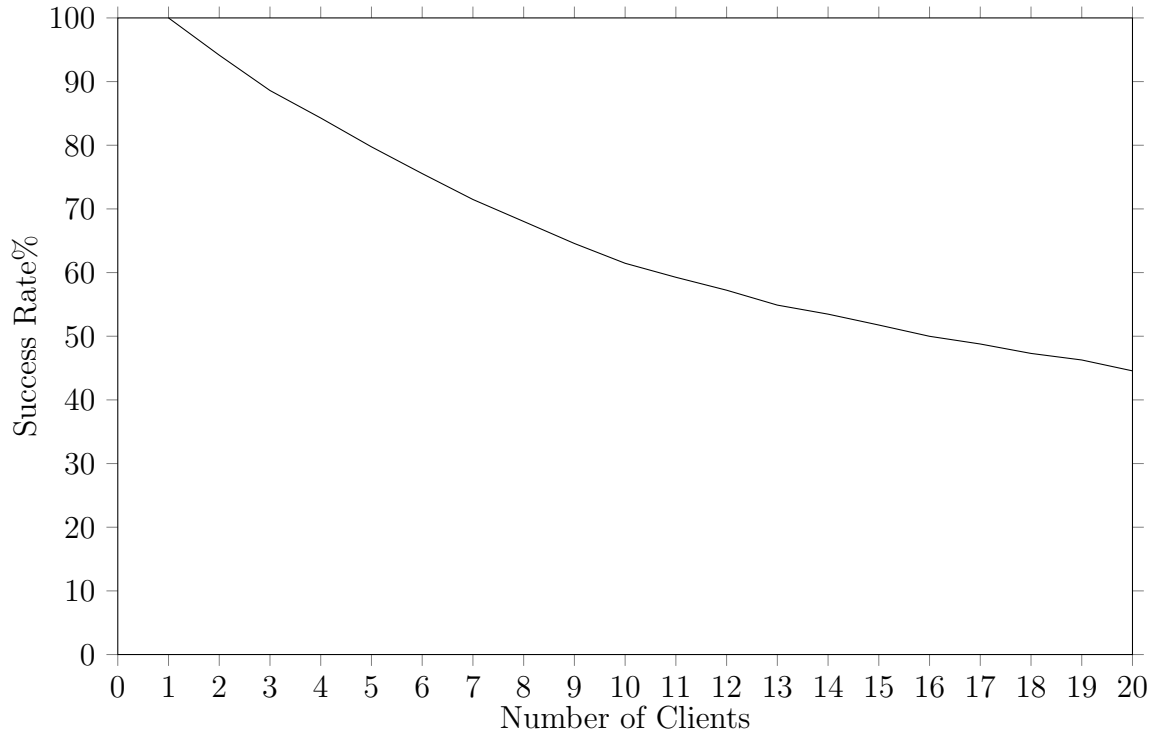

Figure 2: Impact of number of clients on success rate.

1.1.2 Number of Entries.

Impact of changing the number of entries while keeping the number of clients, read operations per transaction, write operations per transaction and duration constant.

The table 2 consists of varying number of entries and the corresponding success rate while keeping the other parameters constant. The figure 3 depicts a part of the output and figure 4 depicts a plot of the entire output. We can see that for 1 entry the success rate is around 60.75% whereas, for 20 entries it is around 91.75%, that is, as the number of entries increases the success rate also increases. We can also see that the success rate is quite similar for different clients as seen in figure 3.

- Clients: 4, Read Opertaions Per Transaction: 1, Write Operations Per Transaction: 1, Duration: 3s.

entries	**Success(%)**
1	60.75
2	63.62
3	68.95
4	72.95
5	76.51
6	79.02
7	80.98
8	82.78
9	84.19
10	85.48
11	86.47

entries	Success(%)
12	87.41
13	88.17
14	88.80
15	89.35
16	90.10
17	90.55
18	91.00
19	91.36
20	91.75

Table 2: Observations recorded after varying the number of entries.

```
25> opty:start(4,1,1,1,3).
Starting: 4 CLIENTS, 1 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:118658, OK:72210, -> 60.85556810328844 %
3: Transactions TOTAL:118525, OK:71856, -> 60.62518456021936 %
4: Transactions TOTAL:118699, OK:72411, -> 61.00388377324156 %
1: Transactions TOTAL:118629, OK:72179, -> 60.844312942029354 %
Stopped
ok
26> opty:start(4,2,1,1,3).
Starting: 4 CLIENTS, 2 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:105809, OK:67348, -> 63.65054012418603 %
4: Transactions TOTAL:105610, OK:67540, -> 63.952277246472875 %
3: Transactions TOTAL:105507, OK:67569, -> 64.04219625238136 %
1: Transactions TOTAL:105692, OK:67178, -> 63.56015592476252 %
Stopped
ok
27> opty:start(4,3,1,1,3).
Starting: 4 CLIENTS, 3 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:103105, OK:71085, -> 68.94428010280782 %
4: Transactions TOTAL:103265, OK:71246, -> 68.99336658112622 %
2: Transactions TOTAL:102937, OK:70999, -> 68.97325548636545 %
3: Transactions TOTAL:103192, OK:71118, -> 68.91813318861928 %
Stopped
ok
28> opty:start(4,4,1,1,3).
Starting: 4 CLIENTS, 4 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:102463, OK:74755, -> 72.95804339127295 %
2: Transactions TOTAL:102458, OK:74665, -> 72.87376290772805 %
4: Transactions TOTAL:102555, OK:75129, -> 73.25727658329677 %
1: Transactions TOTAL:102320, OK:74902, -> 73.20367474589523 %
Stopped
ok
29> opty:start(4,5,1,1,3).
Starting: 4 CLIENTS, 5 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:100600, OK:76893, -> 76.43439363817097 %
4: Transactions TOTAL:100782, OK:77109, -> 76.51068643210097 %
2: Transactions TOTAL:100535, OK:76888, -> 76.47883821554683 %
1: Transactions TOTAL:100486, OK:76569, -> 76.19867444221086 %
Stopped
ok
```

Figure 3: Output when varying the number of entries.

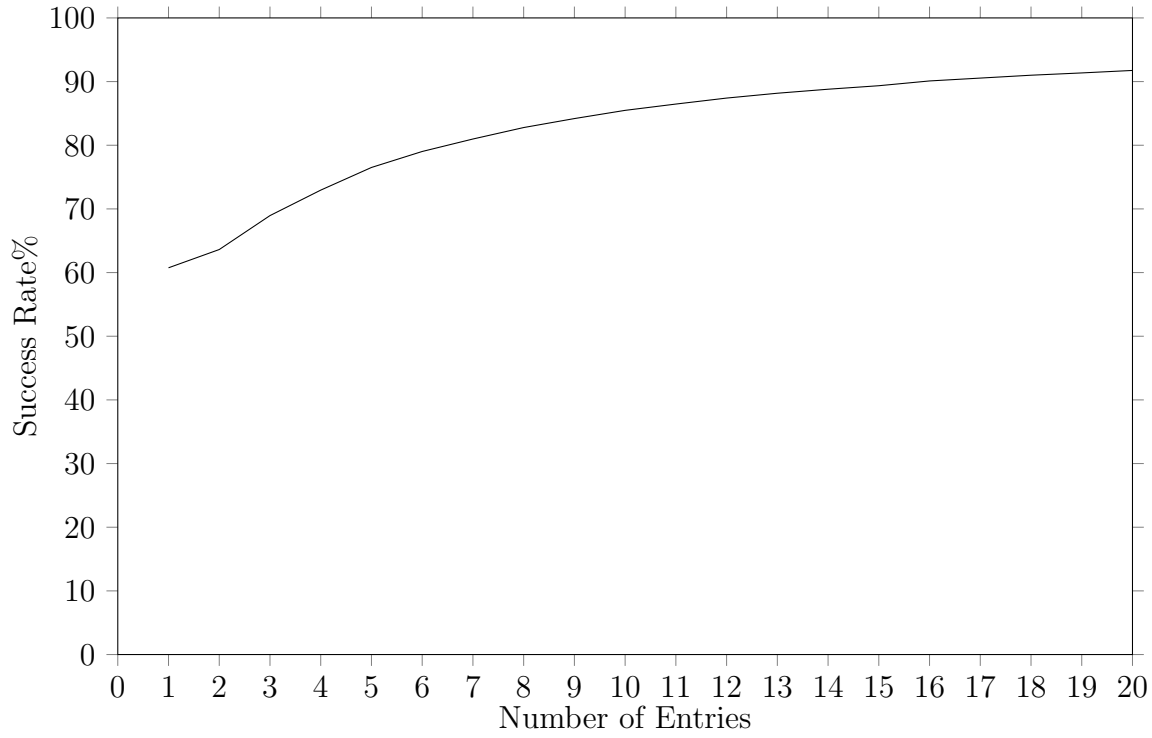

Figure 4: Impact of number of entries on success rate.

1.1.3 Number of Read Operations Per Transaction.

Impact of changing the number of read operations per transaction while keeping the number of clients, number of entries, write operations per transaction and duration constant.

The table 3 consists of varying number of read operations per transaction and the corresponding success rate while keeping the other parameters constant. The figure 5 depicts a plot of the entire output and figure 6 depicts a part of the output. We can see that as the number of read operations per transaction increases the success rate decreases. From the table we can see that for 1 read per transaction, the success rate is around 84.12% whereas, for 20 read operations per transaction the success rate is 38.50%. We can also see that the success rate is quite similar for different clients in the figure 6.

- Clients: 4, Entries: 9, Write Operations Per Transaction: 1, Duration: 3s.

Reads	**Success(%)**
1	84.12
2	74.95
3	68.63
4	63.60
5	59.83
6	56.36
7	53.45
8	51.78
9	49.49

Reads	Success(%)
10	47.80
11	46.75
12	44.95
13	43.86
14	42.55
15	42.32
16	40.77
17	39.78
18	39.30
19	38.95
20	38.50

Table 3: Observations recorded after varying the number of read operations per transaction.

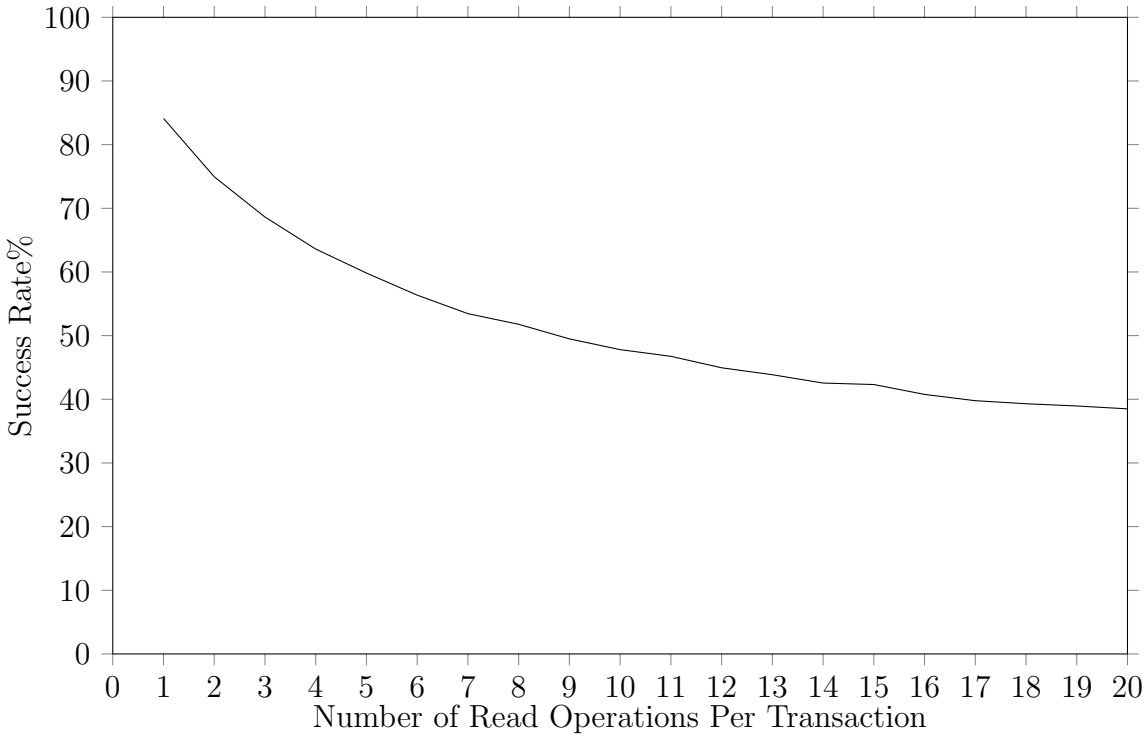

Figure 5: Impact of the number of read operations on success rate.

```
50> opty:start(4,9,1,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:100912, OK:84893, -> 84.12577295068971 %
4: Transactions TOTAL:100953, OK:84870, -> 84.06882410626727 %
2: Transactions TOTAL:100791, OK:84775, -> 84.1096923336409 %
3: Transactions TOTAL:100821, OK:84832, -> 84.14120074190893 %
Stopped
ok
51> opty:start(4,9,2,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 2 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:76044, OK:56984, -> 74.93556362106149 %
3: Transactions TOTAL:75768, OK:56903, -> 75.10162601626017 %
4: Transactions TOTAL:76115, OK:57034, -> 74.93135387242988 %
2: Transactions TOTAL:75864, OK:56651, -> 74.67441737846673 %
Stopped
ok
52> opty:start(4,9,3,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 3 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:61531, OK:42214, -> 68.6060684858039 %
4: Transactions TOTAL:61586, OK:42307, -> 68.69580748871496 %
3: Transactions TOTAL:61430, OK:42154, -> 68.62119485593358 %
1: Transactions TOTAL:61392, OK:42119, -> 68.60665884805837 %
Stopped
ok
53> opty:start(4,9,4,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 4 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:51533, OK:32777, -> 63.60390429433567 %
1: Transactions TOTAL:51559, OK:32890, -> 63.79099672220175 %
3: Transactions TOTAL:51614, OK:32897, -> 63.73658309760918 %
4: Transactions TOTAL:51606, OK:33046, -> 64.03518970662326 %
Stopped
ok
54> opty:start(4,9,5,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 5 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
4: Transactions TOTAL:44609, OK:26600, -> 59.62922280257347 %
1: Transactions TOTAL:44609, OK:26883, -> 60.263623932390324 %
3: Transactions TOTAL:44559, OK:26664, -> 59.83976301083956 %
2: Transactions TOTAL:44652, OK:26730, -> 59.86294006987369 %
Stopped
ok
```

Figure 6: Output when verying the number of read operations per transaction.

1.1.4 Number of Write Operations Per Transaction.

Impact of changing the number of write operations per transaction while keeping the number of clients, number of entries, number of read operations per transaction and duration constant.

The table 4 consists of varying number of write operations per transaction and the corresponding success rate while keeping the other parameters constant. The figure 7 depicts a plot of the entire output and figure 8 depicts a part of the output. We can see that as the number of write operations per transaction increases, the success rate decreases. From the table we can see that for 1 write operation per transaction, the success rate is around 84.09% whereas, for 20 write operations per transaction, the success rate is around 37.80%. We can also see that the success rate is quite similar for different clients from the figure 7.

- Clients: 4, Entries: 9, Read Operations Per Transaction: 1, Duration: 3s.

Writes	**Success(%)**
1	84.09
2	74.65
3	68.10
4	62.95
5	59.01
6	55.63
7	52.90
8	50.42
9	48.42
10	46.90
11	45.10
12	43.78
13	42.52
14	41.59
15	40.38
16	40.12
17	39.55
18	38.81
19	38.27
20	37.80

Table 4: Observations recorded after varying the number of write operations per transaction.

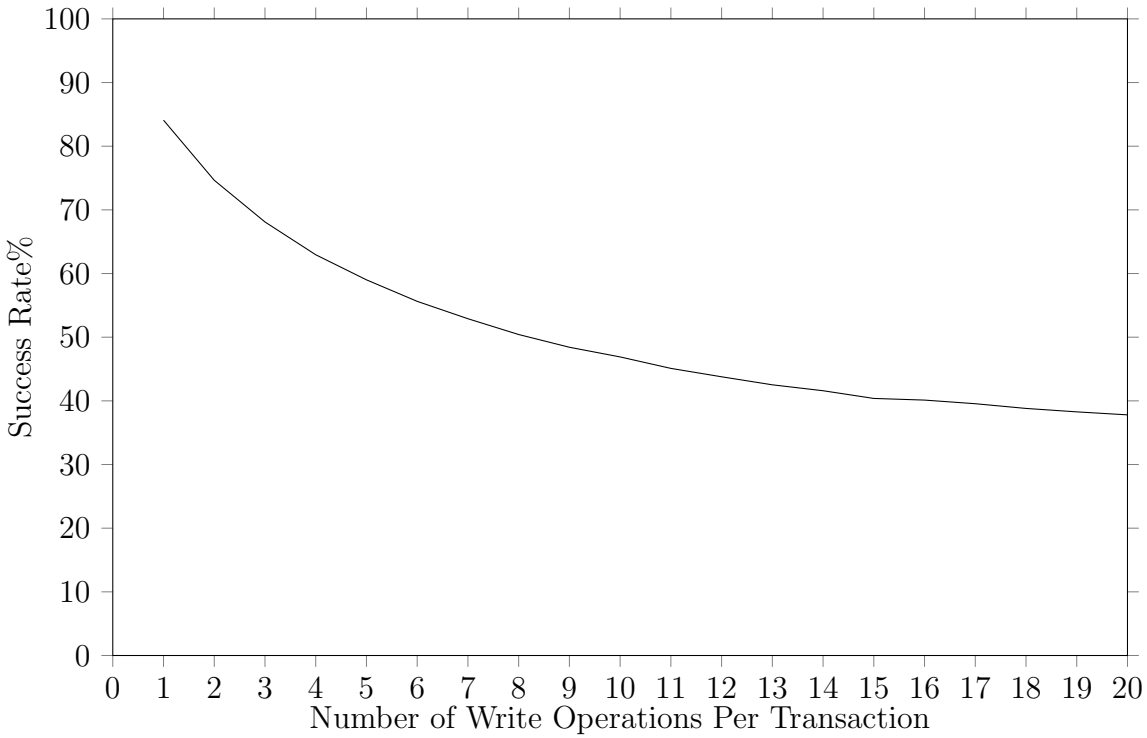

Figure 7: Impact of number of write operations per transaction on success rate.

```
72> opty:start(4,9,1,1,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:103213, OK:86796, -> 84.09405791906059 %
3: Transactions TOTAL:103386, OK:86853, -> 84.00847310080668 %
4: Transactions TOTAL:103194, OK:86788, -> 84.10178886369363 %
2: Transactions TOTAL:103261, OK:86846, -> 84.10338850098294 %
Stopped
ok
73> opty:start(4,9,1,2,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
4: Transactions TOTAL:99642, OK:74413, -> 74.68035567331044 %
1: Transactions TOTAL:99590, OK:74577, -> 74.88402450045186 %
3: Transactions TOTAL:99558, OK:74303, -> 74.63287731774443 %
2: Transactions TOTAL:99568, OK:74184, -> 74.50586533826129 %
Stopped
ok
74> opty:start(4,9,1,3,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 3 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:94219, OK:64245, -> 68.18688374956218 %
4: Transactions TOTAL:94435, OK:64224, -> 68.0086832212633 %
1: Transactions TOTAL:94213, OK:64165, -> 68.10631229235881 %
3: Transactions TOTAL:94413, OK:64579, -> 68.40053806149577 %
Stopped
ok
75> opty:start(4,9,1,4,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 4 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:90674, OK:56997, -> 62.85925403092397 %
3: Transactions TOTAL:90717, OK:57355, -> 63.22409250746828 %
4: Transactions TOTAL:90973, OK:57164, -> 62.83622613302848 %
2: Transactions TOTAL:90854, OK:57160, -> 62.91412596033196 %
Stopped
ok
76> opty:start(4,9,1,5,3).
Starting: 4 CLIENTS, 9 ENTRIES, 1 RDxTR, 5 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:87368, OK:51566, -> 59.02160974269756 %
3: Transactions TOTAL:87359, OK:51450, -> 58.89490493251983 %
4: Transactions TOTAL:87607, OK:51791, -> 59.11742212380289 %
2: Transactions TOTAL:87549, OK:51606, -> 58.94527635952438 %
Stopped
ok
```

Figure 8: Output when varying the number of write operations per transaction.

1.1.5 Number of Read and Write Operations Per Transaction.

Impact of changing the number of read and write operations per transaction while keeping the number of clients, number of entries and duration constant.

The table 5 consists of varying the ratio of number of read and write operations per transaction for a fixed number of operations and the corresponding success rate while keeping the other parameters constant. The figure 9 depicts a part of the output which includes when read and write operations are zero respectively. Figure 10 depicts a plot of the entire output. We can see that for bigger differences between read and write operation, the success rate is larger than that for smaller differences between them. For larger number of reads than writes, the success rate is higher than when it is vice versa. We can also see that the success rate is quite similar for different clients from the figure 9.

- Clients: 5, Entries: 9, Duration: 3s.

```
108> opty:start(5,9,0,7,3).
Starting: 5 CLIENTS, 9 ENTRIES, 0 RDxTR, 7 WRxTR, DURATION 3 s
Stopping...
5: Transactions TOTAL:104035, OK:104035, -> 100.0 %
3: Transactions TOTAL:103973, OK:103973, -> 100.0 %
1: Transactions TOTAL:104065, OK:104065, -> 100.0 %
2: Transactions TOTAL:104262, OK:104262, -> 100.0 %
4: Transactions TOTAL:104192, OK:104192, -> 100.0 %
Stopped
ok
109> opty:start(5,9,7,0,3).
Starting: 5 CLIENTS, 9 ENTRIES, 7 RDxTR, 0 WRxTR, DURATION 3 s
Stopping...
5: Transactions TOTAL:27346, OK:27346, -> 100.0 %
3: Transactions TOTAL:27339, OK:27339, -> 100.0 %
4: Transactions TOTAL:27347, OK:27347, -> 100.0 %
1: Transactions TOTAL:27391, OK:27391, -> 100.0 %
2: Transactions TOTAL:27262, OK:27262, -> 100.0 %
Stopped
ok
110> opty:start(5,9,1,6,3).
Starting: 5 CLIENTS, 9 ENTRIES, 1 RDxTR, 6 WRxTR, DURATION 3 s
Stopping...
5: Transactions TOTAL:72141, OK:35875, -> 49.72900292482777 %
1: Transactions TOTAL:72088, OK:35897, -> 49.79608256575297 %
3: Transactions TOTAL:72138, OK:35977, -> 49.87246666112174 %
2: Transactions TOTAL:72020, OK:35470, -> 49.25020827547903 %
4: Transactions TOTAL:72042, OK:36028, -> 50.009716554232256 %
Stopped
ok
111> opty:start(5,9,6,1,3).
Starting: 5 CLIENTS, 9 ENTRIES, 6 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
5: Transactions TOTAL:31826, OK:16285, -> 51.16885565261107 %
4: Transactions TOTAL:31833, OK:16113, -> 50.617283950617285 %
2: Transactions TOTAL:31768, OK:16101, -> 50.68307731050113 %
1: Transactions TOTAL:31754, OK:16057, -> 50.566857718712605 %
3: Transactions TOTAL:31777, OK:16205, -> 50.996003398684586 %
Stopped
ok
```

Figure 9: Output when varying the number of read and write operations per transaction while keeping total number of operations constant.

Sl. No.	Reads Operations	Write Operations	Success(%)
1	0	7	100.00
2	1	6	49.72
3	2	5	42.26
4	3	4	40.03
5	4	3	40.36
6	5	2	42.42
7	6	1	50.65
8	7	0	100.00

Table 5: Observations recorded after changing the number of read and write operations keeping number of operations constant.

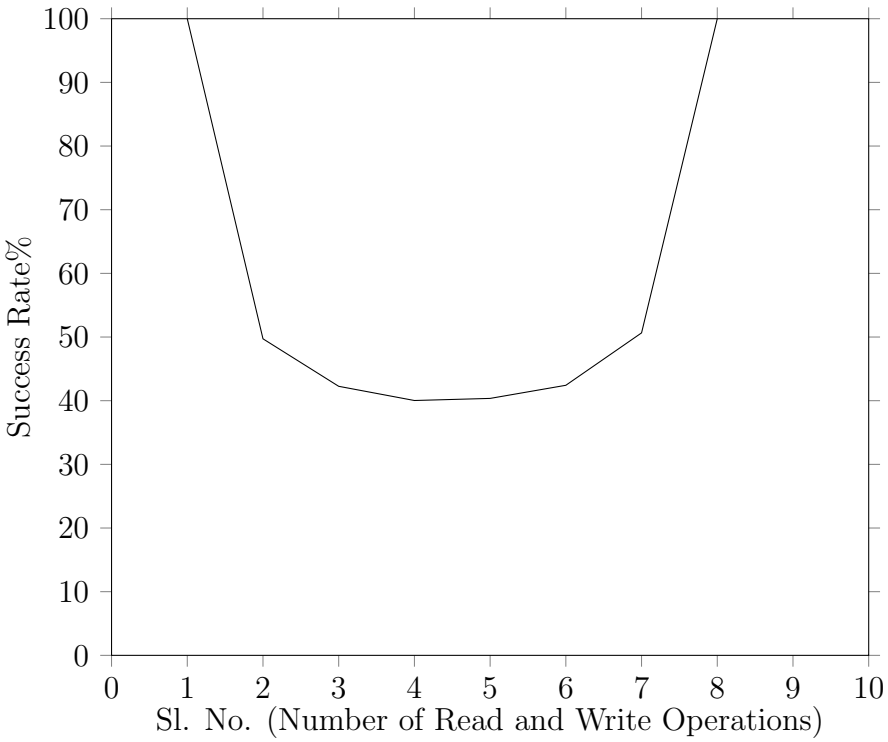

Figure 10: Impact of number of ratio of read and write operations per transaction on success rate.

1.1.6 Percentage of Accessed Entries.

Impact of different percentage of accessed entries with respect to the total number of entries.

Table 6 and 7 contains the observations recorded when varying the percentage of accessed entries with respect to the total number of entries. The figure 11 and 13 depicts a plot of the same for the respective experiments. Figure 12 and 14 depicts only a part of the outputs. We can see that as the percentage of accessed entries increases from 10% to 100%, the success rate decreases.

Below are two experiments:

- Experiment 1: Clients: 5, Entries: 10, Read Operations Per Transaction: 2, Write Operations Per Transaction: 2, Duration: 3s.

Accessed Entries(%)	Success(%)
10	87.92
20	75.24
30	69.14
40	66.46
50	63.91
60	62.60
70	61.53
80	60.75
90	60.29
100	59.68

Table 6: Observations recorded after changing the percentage of accessed subsets of entries.

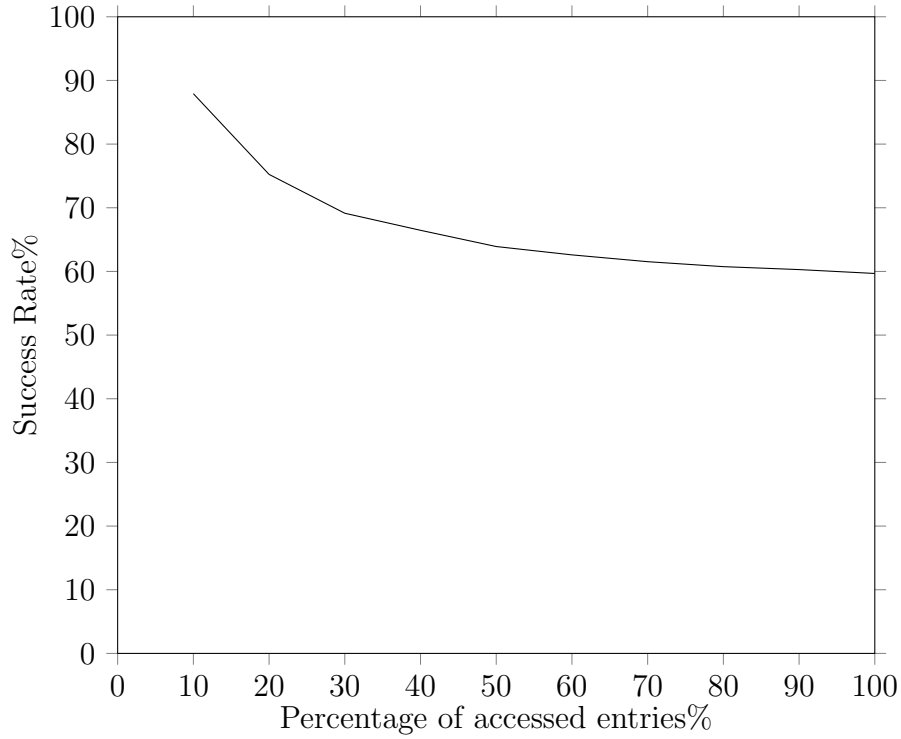

Figure 11: Impact of different percentage of accessed entries with respect to the total number of entries on the success rate.

```
53> c(client).
{ok,client}
54> opty:start(5,10,2,2,3).
Starting: 5 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:58673, OK:51651, -> 88.03197382100797 %
2: Transactions TOTAL:58546, OK:51539, -> 88.031633245653 %
4: Transactions TOTAL:58453, OK:51350, -> 87.84835679947993 %
1: Transactions TOTAL:58671, OK:51505, -> 87.78612943362138 %
5: Transactions TOTAL:58629, OK:51560, -> 87.94282692865306 %
Stopped
ok
55> c(client).
{ok,client}
56> opty:start(5,10,2,2,3).
Starting: 5 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:49768, OK:37437, -> 75.22303488185179 %
4: Transactions TOTAL:49621, OK:37378, -> 75.32697849700732 %
3: Transactions TOTAL:49642, OK:37291, -> 75.11985818460175 %
1: Transactions TOTAL:49859, OK:37569, -> 75.35048837722377 %
5: Transactions TOTAL:49800, OK:37445, -> 75.19076305220884 %
Stopped
ok
57> c(client).
{ok,client}
58> opty:start(5,10,2,2,3).
Starting: 5 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:47924, OK:33067, -> 68.9988314831817 %
5: Transactions TOTAL:48109, OK:33256, -> 69.12635889334636 %
4: Transactions TOTAL:48106, OK:33319, -> 69.26163056583378 %
3: Transactions TOTAL:47943, OK:33132, -> 69.10706463925912 %
1: Transactions TOTAL:47931, OK:33166, -> 69.19530157935365 %
Stopped
ok
59> c(client).
{ok,client}
60> opty:start(5,10,2,2,3).
Starting: 5 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:46792, OK:30763, -> 65.74414429817062 %
4: Transactions TOTAL:46980, OK:30955, -> 65.88974031502767 %
3: Transactions TOTAL:46843, OK:30708, -> 65.55515231731529 %
5: Transactions TOTAL:46926, OK:30981, -> 66.02096918552614 %
1: Transactions TOTAL:46899, OK:30786, -> 65.64319068636858 %
Stopped
ok
```

Figure 12: Output when percentage of accessed entries with respect to the total number of entries is varied.

- Experiment 2: Clients: 7, Entries: 10, Read Operations Per Transaction: 2, Write Operations Per Transaction: 2, Duration: 3s.

Accessed Entries(%)	**Success(%)**
10	83.20
20	67.15
30	60.13
40	56.75
50	54.16
60	52.98
70	51.85
80	51.11
90	50.67

Accessed Entries(%)	**Success(%)**
100	50.23

Table 7: Observations recorded after changing the percentage of accessed subsets of entries.

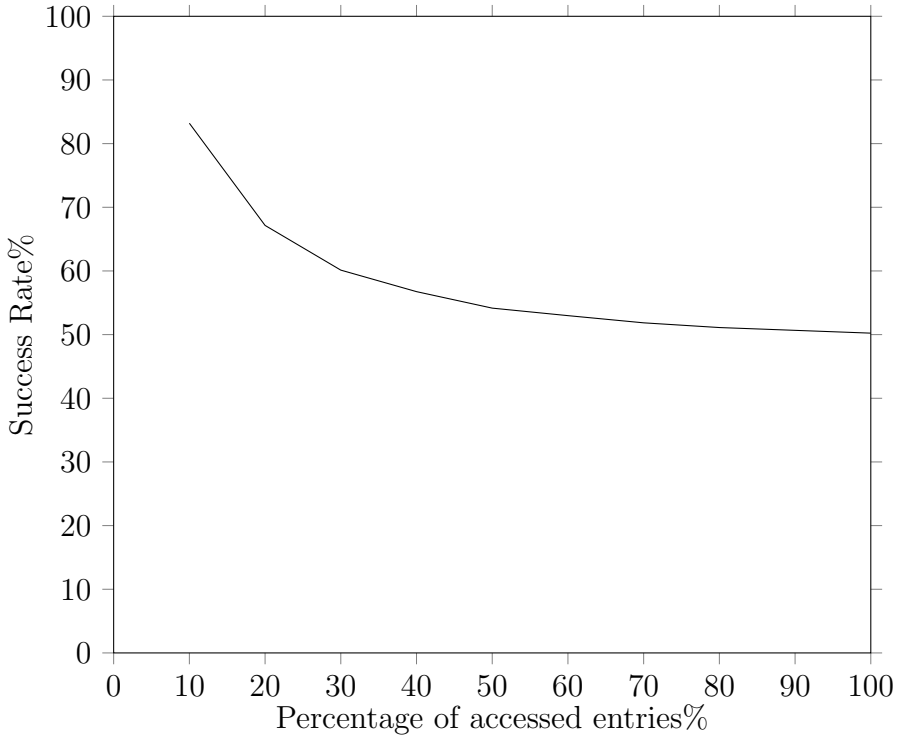

Figure 13: Impact of different percentage of accessed entries with respect to the total number of entries on the success rate.

```
4> c(client).
{ok,client}
5> opty:start(7,10,2,2,3).
Starting: 7 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:46680, OK:38718, -> 82.94344473007712 %
3: Transactions TOTAL:46944, OK:38881, -> 82.82421608725289 %
6: Transactions TOTAL:46935, OK:39154, -> 83.42175348886758 %
2: Transactions TOTAL:46732, OK:38836, -> 83.10365488316357 %
7: Transactions TOTAL:47189, OK:39373, -> 83.43681790247727 %
5: Transactions TOTAL:46729, OK:38947, -> 83.34652999208201 %
4: Transactions TOTAL:46904, OK:39032, -> 83.21678321678321 %
Stopped
ok
6> c(client).
{ok,client}
7> opty:start(7,10,2,2,3).
Starting: 7 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
5: Transactions TOTAL:40439, OK:27201, -> 67.2642745864141 %
4: Transactions TOTAL:40408, OK:27107, -> 67.08325084141754 %
3: Transactions TOTAL:40376, OK:27289, -> 67.58718050326927 %
2: Transactions TOTAL:40410, OK:27091, -> 67.04033655035882 %
7: Transactions TOTAL:40395, OK:27225, -> 67.39695506869663 %
6: Transactions TOTAL:40406, OK:27230, -> 67.39098153739543 %
1: Transactions TOTAL:40387, OK:27022, -> 66.90766830910937 %
Stopped
ok
8> c(client).
{ok,client}
9> opty:start(7,10,2,2,3).
Starting: 7 CLIENTS, 10 ENTRIES, 2 RDxTR, 2 WRxTR, DURATION 3 s
Stopping...
2: Transactions TOTAL:38174, OK:22840, -> 59.83129878975219 %
4: Transactions TOTAL:38139, OK:23179, -> 60.7750596502268 %
1: Transactions TOTAL:38132, OK:23040, -> 60.421693066191125 %
5: Transactions TOTAL:38191, OK:23060, -> 60.380717970202404 %
3: Transactions TOTAL:38100, OK:22794, -> 59.826771653543304 %
7: Transactions TOTAL:38183, OK:23022, -> 60.29384804756043 %
6: Transactions TOTAL:38180, OK:22952, -> 60.11524358302776 %
Stopped
ok
```

Figure 14: Output when percentage of accessed entries with respect to the total number of entries is varied.

1.2 Distributed Execution.

The code has been updated as shown below in figure 15. The server node is opty-srv@DESKTOP-SIIVR1N where opty-srv is the locally registered name of the server and DESKTOP-SIIVR1N is the name of my PC which varies for different PCs.

```
start(Clients, Entries, Reads, Writes, Time) ->
    spawn('opty-srv@DESKTOP-SIIVR1N', fun() -> register(s, server:start(Entries))end), %%Spawn opty-srv@DESKYOP_SIIVR1N server
    L = startClients(Clients, [], Entries, Reads, Writes),
    io:format("Starting: ~w CLIENTS, ~w ENTRIES, ~w RDxTR, ~w WRxTR, DURATION ~w s~n",
        [Clients, Entries, Reads, Writes, Time]),
    timer:sleep(Time*1000),
    stop(L).

stop(L) ->
    io:format("Stopping...~n"),
    stopClients(L),
    waitClients(L),
    {s, 'opty-srv@DESKTOP-SIIVR1N'} ! stop,  %%Stop
    io:format("Stopped~n").
```

Figure 15: opty.erl code changes.

To create two nodes, we run the following commands:

- On one terminal(server):

  erl -sname opty-srv

  Then we compile all the files using:

  cover:compile.directory().

  The above two commands are depicted in figure 16.

```
C:\Users\PEARL\Downloads\DistributedSystems\Opty\Opty-source-files\Q2\Base\opty-source-files>erl -sname opty-srv
Eshell V13.1.2  (abort with ^G)
(opty-srv@DESKTOP-SIIVR1N)1> cover:compile_directory().
[{ok,client},
 {ok,validator},
 {ok,handler},
 {ok,entry},
 {ok,opty},
 {ok,server},
 {ok,store}]
(opty-srv@DESKTOP-SIIVR1N)2>
```

Figure 16: Server.

- On the other terminal(client):

  erl -sname bar

  Compile all files again just to be sure:

  cover:compile.directory().

  The above commands are depicted in figure 17.

```
C:\Users\PEARL\Downloads\DistributedSystems\Opty\Opty-source-files\Q2\Base\opty-source-files>erl -sname bar
Eshell V13.1.2  (abort with ^G)
(bar@DESKTOP-SIIVR1N)1> cover:compile_directory().
[{ok,client},
 {ok,handler},
 {ok,validator},
 {ok,opty},
 {ok,entry},
 {ok,server},
 {ok,store}]
(bar@DESKTOP-SIIVR1N)2>
```

Figure 17: Client.

Below are two cases run on the client:

- opty:start(5,10,1,1,3).

  Clients: 5, Entries: 10, Read operations per transaction: 1, Write operations per transaction: 1, Duration: 3s

- opty:start(6,10,1,1,3).

  Clients: 6, Entries: 10, Read operations per transaction: 1, Write operations per transaction: 1, Duration: 3s

The output is depicted in the figure 18.

```
(bar@DESKTOP-SIIVR1N)2> net_adm:ping('opty-srv@DESKTOP-SIIVR1N').
pong
(bar@DESKTOP-SIIVR1N)3> opty:start(5,10,1,1,3).
Starting: 5 CLIENTS, 10 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
3: Transactions TOTAL:5781, OK:4949, -> 85.60802629302889 %
1: Transactions TOTAL:5756, OK:4918, -> 85.4412786657401 %
2: Transactions TOTAL:5767, OK:4942, -> 85.69446852783076 %
5: Transactions TOTAL:5759, OK:4984, -> 86.54280256989061 %
4: Transactions TOTAL:5776, OK:4978, -> 86.1842105263158 %
Stopped
ok
(bar@DESKTOP-SIIVR1N)4> opty:start(6,10,1,1,3).
Starting: 6 CLIENTS, 10 ENTRIES, 1 RDxTR, 1 WRxTR, DURATION 3 s
Stopping...
1: Transactions TOTAL:5068, OK:4228, -> 83.42541436464089 %
2: Transactions TOTAL:5081, OK:4292, -> 84.47156071639441 %
5: Transactions TOTAL:5081, OK:4171, -> 82.09013973627239 %
6: Transactions TOTAL:5067, OK:4220, -> 83.28399447404776 %
3: Transactions TOTAL:5069, OK:4204, -> 82.93549023476031 %
4: Transactions TOTAL:5076, OK:4189, -> 82.52561071710008 %
Stopped
ok
```

Figure 18: Output after performing distributed execution.

1.3 Timey.

1.3.1 Number of Clients.

Comparison of impact of number of clients on success rate between optimistic concurrency control and timestamp ordering concurrency control.

The table 8 contains varying number of clients and the corresponding success rate while keeping the other parameters constant for optimistic concurrency control(Opty) and timestamp ordering concurrency control(Timey). The figure 19 depicts a plot of the same. The red curve is for Opty and the blue curve for Timey. We can see that for Opty and Timey, the success rate decreases as the number of clients increases as there is more probability of experiencing transaction conflicts when many clients try to access the entries. But we can also see that the success rate for Timey is more than that for Opty with the same parameters. For example, using 5 clients we see that the success rate for Opty is 79.75% whereas, when using Timey it is 94.75%. For a single client it remains 100% as there is no chance of transaction conflicts taking place. This can be seen in figure 20, which depicts a part of the otput. We also notice the difference in the total number of transactions. The total number of transactions in the case of Timey is more compared to Opty and also the total number of successful transactions compared to the total is also more and hence the higher success rate.

- Entries: 9, Read Opertaions Per Transaction: 1, Write Operations Per Transaction: 1, Duration: 3s.

Client(s)	Success% (Opty)	Success% (Timey)
1	100.00	100.00
2	94.15	98.90
3	88.60	97.33
4	84.28	96.35
5	79.75	94.75
6	75.55	93.64
7	71.48	92.29
8	68.03	90.88
9	64.57	89.65
10	61.46	88.26
11	59.26	87.24
12	57.23	86.10
13	54.89	85.14
14	53.47	83.98
15	51.76	83.24
16	49.99	82.51
17	48.78	81.44
18	47.30	80.51
19	46.27	79.99
20	44.56	79.25

Table 8: Observations recorded after changing only the number of clients.

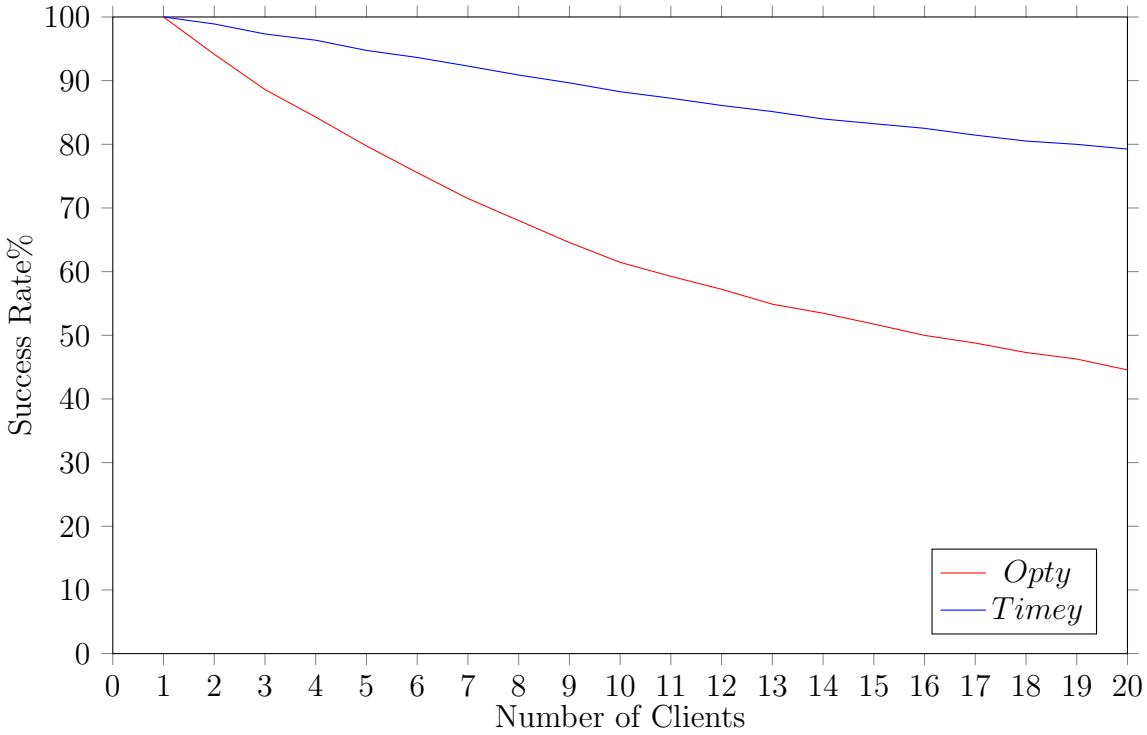

Figure 19: Impact of number of clients on success rate.

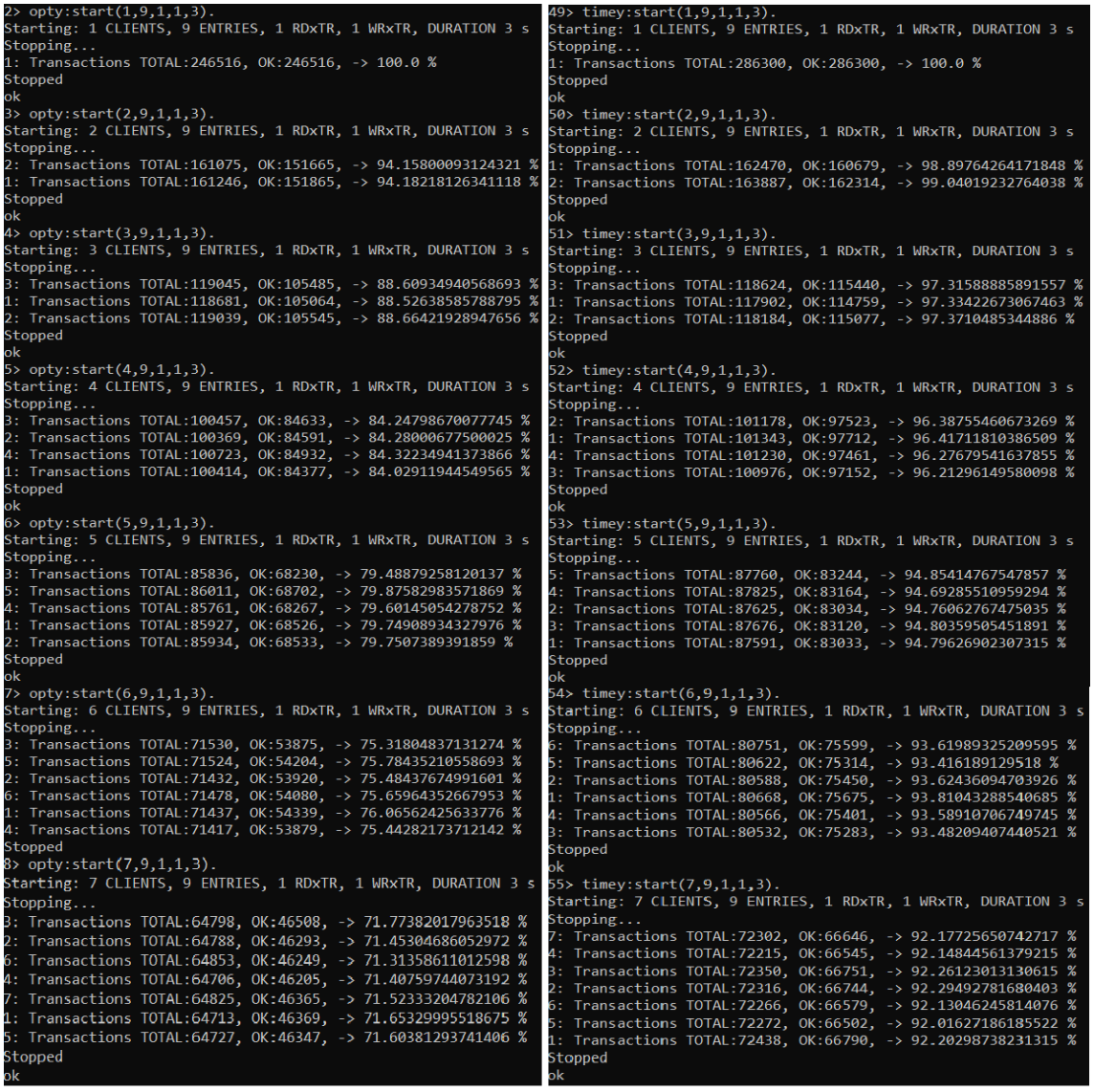

Figure 20: Output when changing the number of clients for Opty vs Timey.

1.3.2 Number of Entries.

Comparison of impact of number of entries on success rate between optimistic concurrency control and timestamp ordering concurrency control.

The table 9 consists of the varying number of entries and their corresponding success rate while keeping the other parameters constant for optimistic concurrency control(Opty) and timestamp ordering concurrency control(Timey). The figure 21 depicts a part of the output and the figure 22 depicts a plot of the values of the entire output. The red curve is for Opty and the blue curve for Timey. We can see that for Opty and Timey, the success rate increases as the number of entries increases as there is less probability of experiencing transaction conflicts with more entries. More entries means more places for clients to write data in the stores which reduces conflicts. But we can also see that the success rate for Timey is more than that for Opty with the same parameters. For example, using 6 entries, we see that the success rate using Opty is 79.02% where as when using Timey is 94.53%. The total number of transactions in the case of Timey when compared to Opty is lesser for few clients but gradually starts to increase for larger number of entries but there more number of successful transactions in the case of Timey and hence better success rate. The difference in total number of transactions is not very large. The margin of increase in the success rate for very large number of entries is not much as can be noticed from the table 9.

- Clients: 4, Read Opertaions Per Transaction: 1, Write Operations Per Transaction: 1, Duration: 3s.

Entry	Success% (Opty)	Success% (Timey)
1	60.75	72.10
2	63.62	83.88
3	68.95	89.06
4	72.95	91.85
5	76.51	93.36
6	79.02	94.53
7	80.98	95.29
8	82.78	95.85
9	84.19	96.20
10	85.48	96.62
11	86.47	96.97
12	87.41	97.19
13	88.17	97.40
14	88.80	97.55
15	89.35	97.63
16	90.10	97.92
17	90.55	98.01
18	91.00	98.15
19	91.36	98.21
20	91.75	98.25

Table 9: Observations recorded after changing only the number of entries.

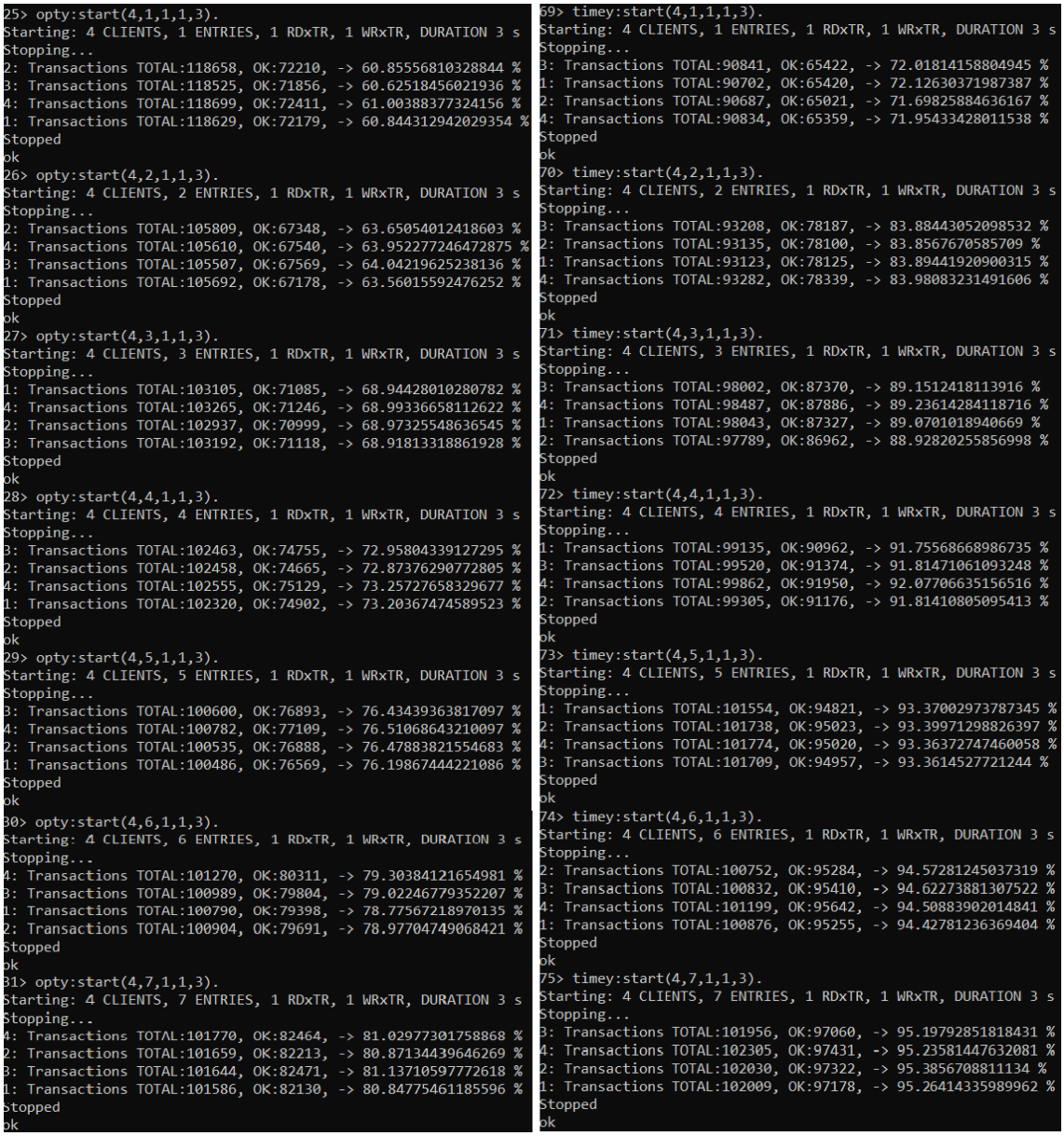

Figure 21: Output when changing the number of entries for Opty vs Timey.

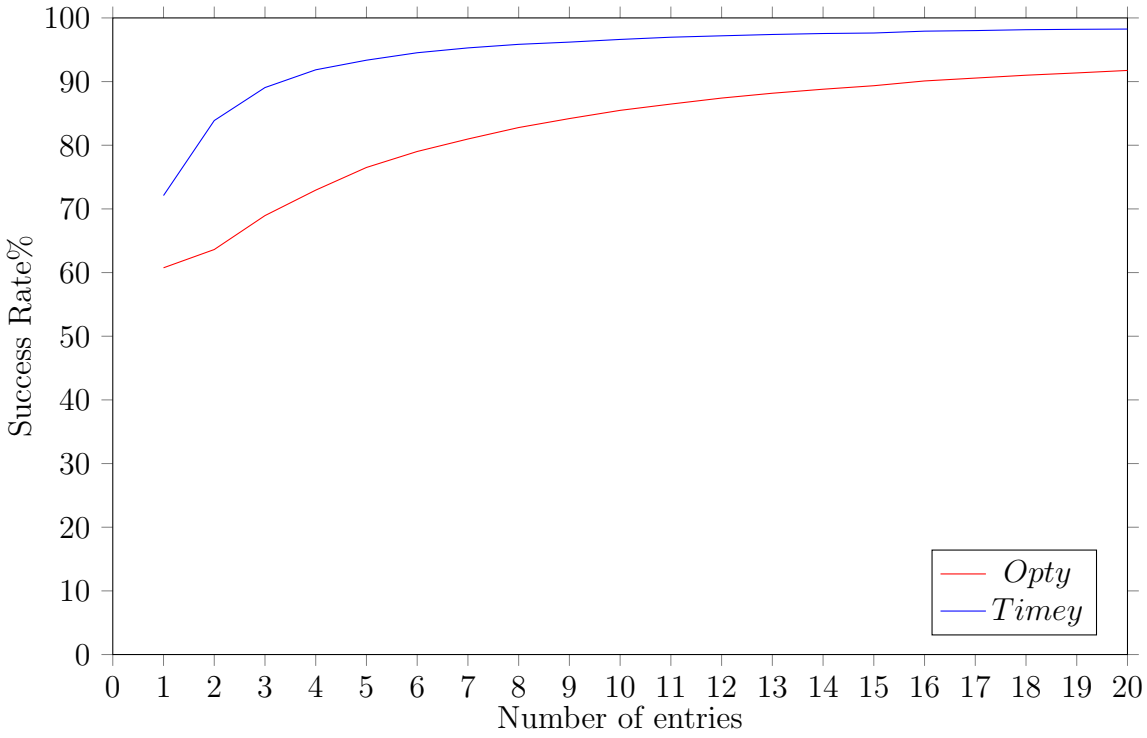

Figure 22: Impact of number of entries on success rate.

1.3.3 Number of Read Operations Per Transaction.

Comparison of impact of number of read operations per transaction on success rate between optimistic concurrency control and timestamp ordering concurrency control.

The table 10 consists of the varying number of read operations per transaction and the corresponding success rate while keeping the other parameters constant for optimistic concurrency control(Opty) and timestamp ordering concurrency control(Timey). The figure 23 depicts a plot of the same. The red curve is for Opty and the blue curve for Timey. We can see that for Opty and Timey, the success rate decreases as the number of read operations per transaction increases. But we can also see that the success rate for Timey is more than that for Opty with the same parameters. For example, using 7 read operations per transaction, we see that the success rate using Opty is 53.45% whereas, when using Timey is 89.84% . There are less number of total transactions in the case of Timey and more number of successful transactions with respect to the total, as can be seen in figure 24 which depicts only a part of the output.

- Clients: 4, Entries: 9, Write Operations Per Transaction: 1, Duration: 3s.

Reads	Success% (Opty)	Success% (Timey)
1	84.12	96.35
2	74.95	93.92
3	68.63	92.58
4	63.60	91.86
5	59.83	90.97
6	56.36	90.32
7	53.45	89.84

Reads	Success% (Opty)	Success% (Timey)
8	51.78	89.52
9	49.49	89.04
10	47.80	88.80
11	46.75	88.61
12	44.95	88.62
13	43.86	88.84
14	42.55	88.56
15	42.32	88.74
16	40.77	88.76
17	39.78	88.76
18	39.30	89.05
19	38.95	89.12
20	38.50	89.24

Table 10: Observations recorded after changing only the number of reads operations.

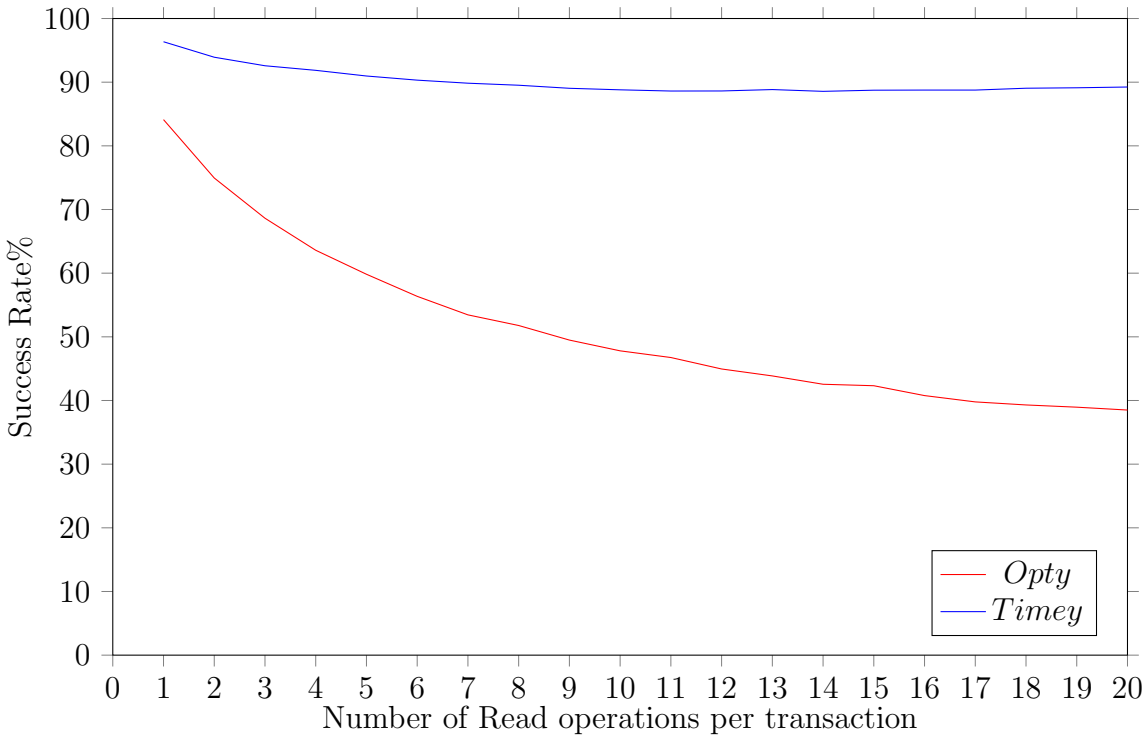

Figure 23: Impact of number of read operations per transaction on success rate.

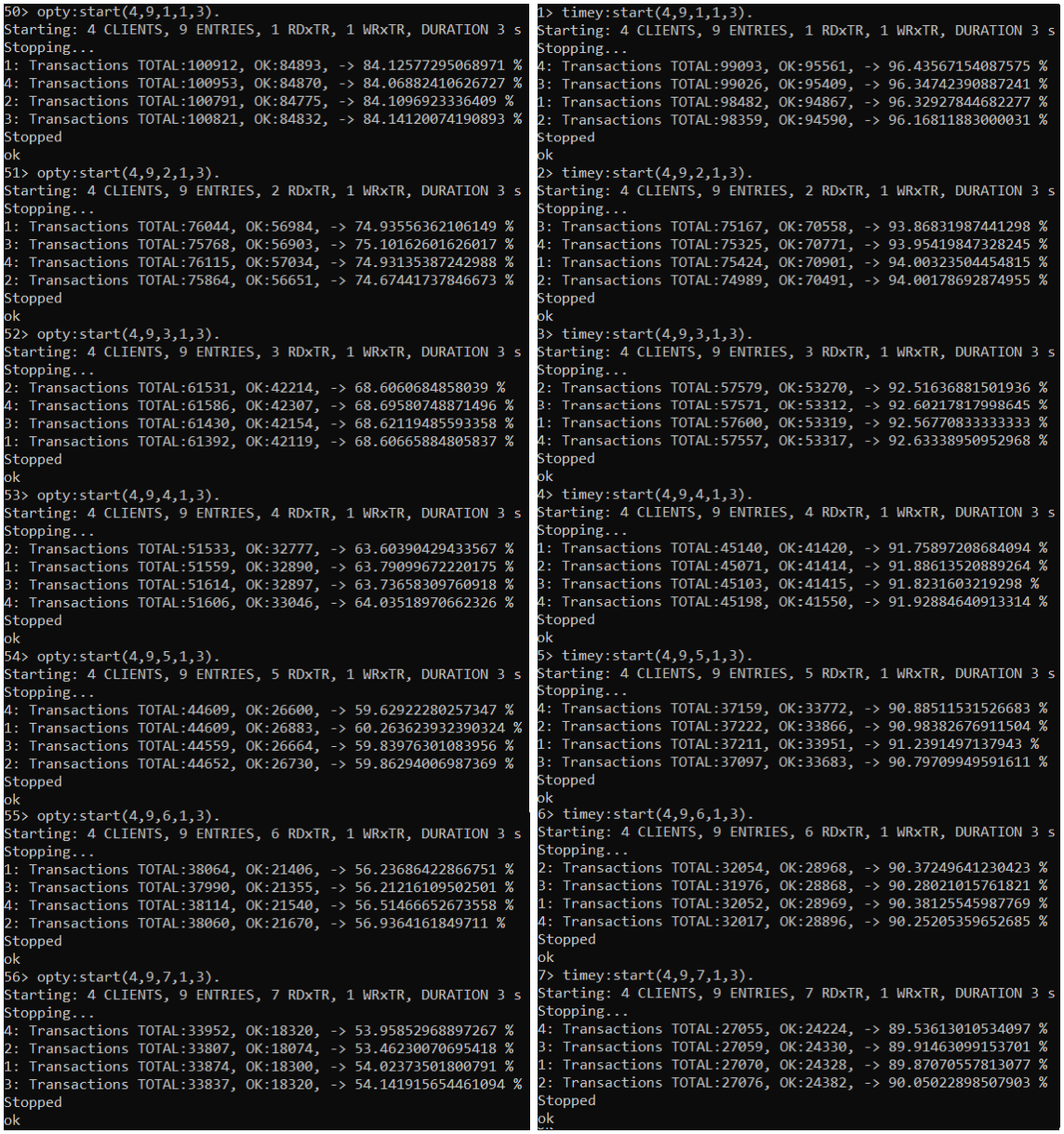

Figure 24: Output when changing the number of read operations per transaction for Opty vs Timey.

1.3.4 Number of Write Operations Per Transaction.

Comparison of impact of number of write operations per transaction on success rate between optimistic concurrency control and timestamp ordering concurrency control.

The table 11 consists of the varying number of write operations per transaction and the corresponding success rate while keeping the other parameters constant for optimistic concurrency control(Opty) and timestamp ordering concurrency control(Timey). The figure 25 depicts only a part of the output and figure 26 depicts a plot of the entire output. The red curve is for Opty and the blue curve for Timey. We can see that for Opty and Timey, the success rate decreases as the number of write operations per transaction increases. But we can also see that the success rate for Timey is more than that for Opty with the same parameters upto a certain number of write opeartions, after which it appears to be lesser than for Opty. For example, using 7 write operations per transaction, we see that the success rate using Opty is 52.90% where as when using Timey is 56.78% and when using 10 write operations per transaction, we see that the success rate is 46.90% for Opty whereas, 45.71% for Timey. There are lesser number of total transactions in the case of Timey and but more number of successful transactions with respect to the total upto a certain number of write operations per transaction and hence, higher success rate.

- Clients: 4, Entries: 9, Read Operations Per Transaction: 1, Duration: 3s.

Writes	**Success% (Opty)**	**Success% (Timey)**
1	84.09	96.25
2	74.65	88.85
3	68.10	80.89
4	62.95	74.15
5	59.01	67.14
6	55.63	61.59
7	52.90	56.78
8	50.42	53.32
9	48.42	49.75
10	46.90	45.71
11	45.10	44.36
12	43.78	42.00
13	42.52	40.50
14	41.59	38.79
15	40.38	37.30
16	40.12	35.95
17	39.55	35.10
18	38.81	34.12
19	38.27	33.22
20	37.80	32.59

Table 11: Observations recorded after changing only the number of write operations.

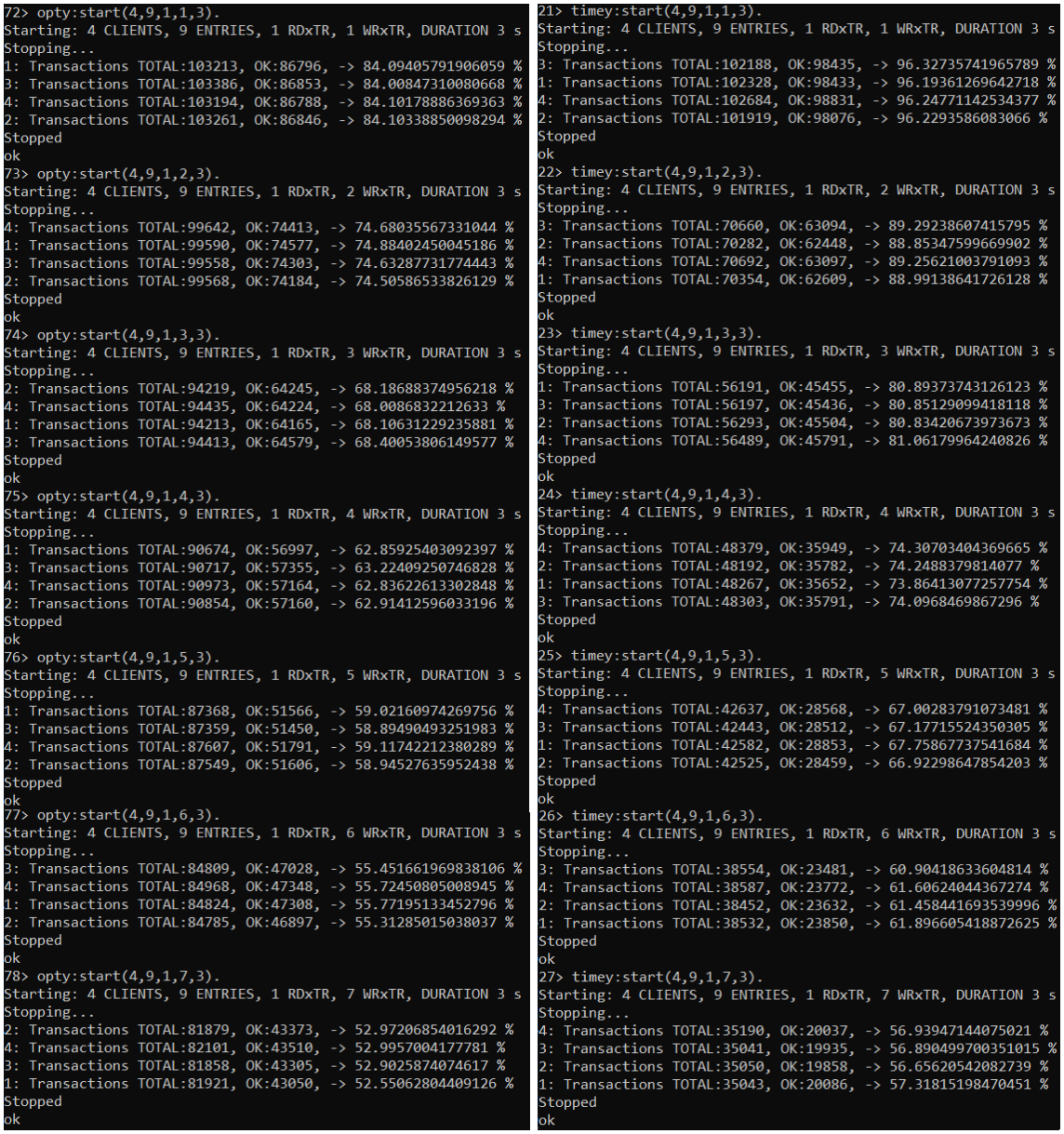

Figure 25: Output when changing the number of write operations per transaction for Opty vs Timey.

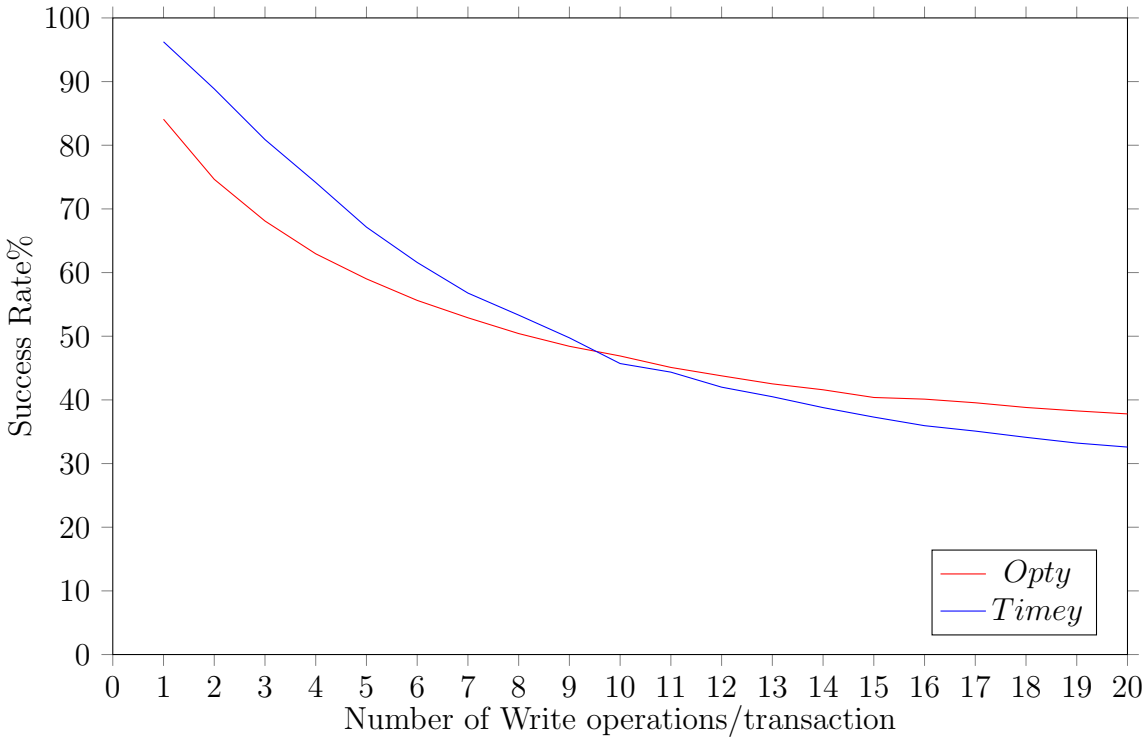

Figure 26: Impact of number of write operations per transaction on success rate.

1.3.5 Number of Read and Write Operations Per Transaction.

Comparison of impact of ratio of number of read and write operations per transaction when keeping total number of operations constant between optimistic concurrency control and timestamp ordering concurrency control.

The table 12 consists of the varying number of read and write operations per transaction keeping the total number of operations constant and the corresponding success rate while keeping the other parameters constant for optimistic concurrency control(Opty) and timestamp ordering concurrency control(Timey). The figure 27 depicts a plot of the same. The red curve is for Opty and the blue curve for Timey. We can see that for Opty and Timey, the success rate decreases as the number of read operations per transaction increases. But we can also see that the success rate for Timey is more than that for Opty with the same parameters. For example, using 2 read and 5 write operations per transaction, we see that the success rate using Opty is 42.26% whereas, when using Timey is 46.81%, and when using 5 read and 2 write operations, the success rate using Opty is 42.42% whereas, when using Timey is 72.67%. There are lesser number of total transactions in the case of Timey but more number of successful transactions with respect to the total as can be seen in figure 28, which depicts the data from the transactions. Hence, higher success rate.

- Clients: 5, Entries: 9, Duration: 3s.

Sl. No.	Reads	Writes	Success% (Opty)	Success% (Timey)
1	0	7	100	97.55
2	1	6	49.72	54.16
3	2	5	42.26	46.81
4	3	4	40.03	50.52

Sl. No.	Reads	Writes	Success% (Opty)	Success% (Timey)
5	4	3	40.36	60.00
6	5	2	42.42	72.67
7	6	1	50.65	86.68
8	7	0	100	100.00

Table 12: Observations recorded after changing the number of read and write operations keeping number of operations constant.

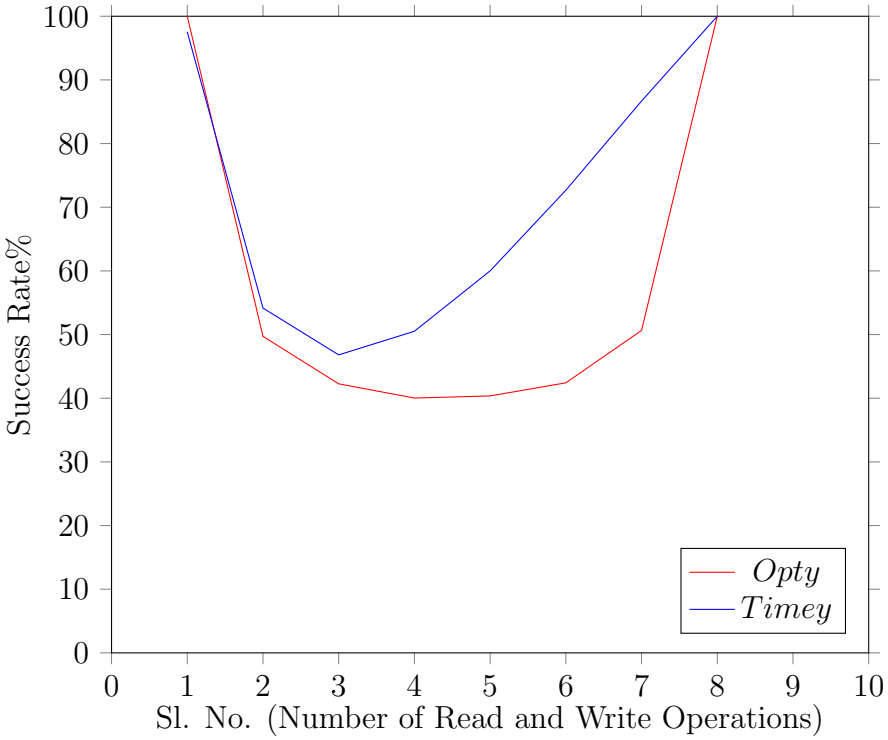

Figure 27: Impact of ratio of number of read and write operations per transaction keeping total number of transactions constant on success rate.

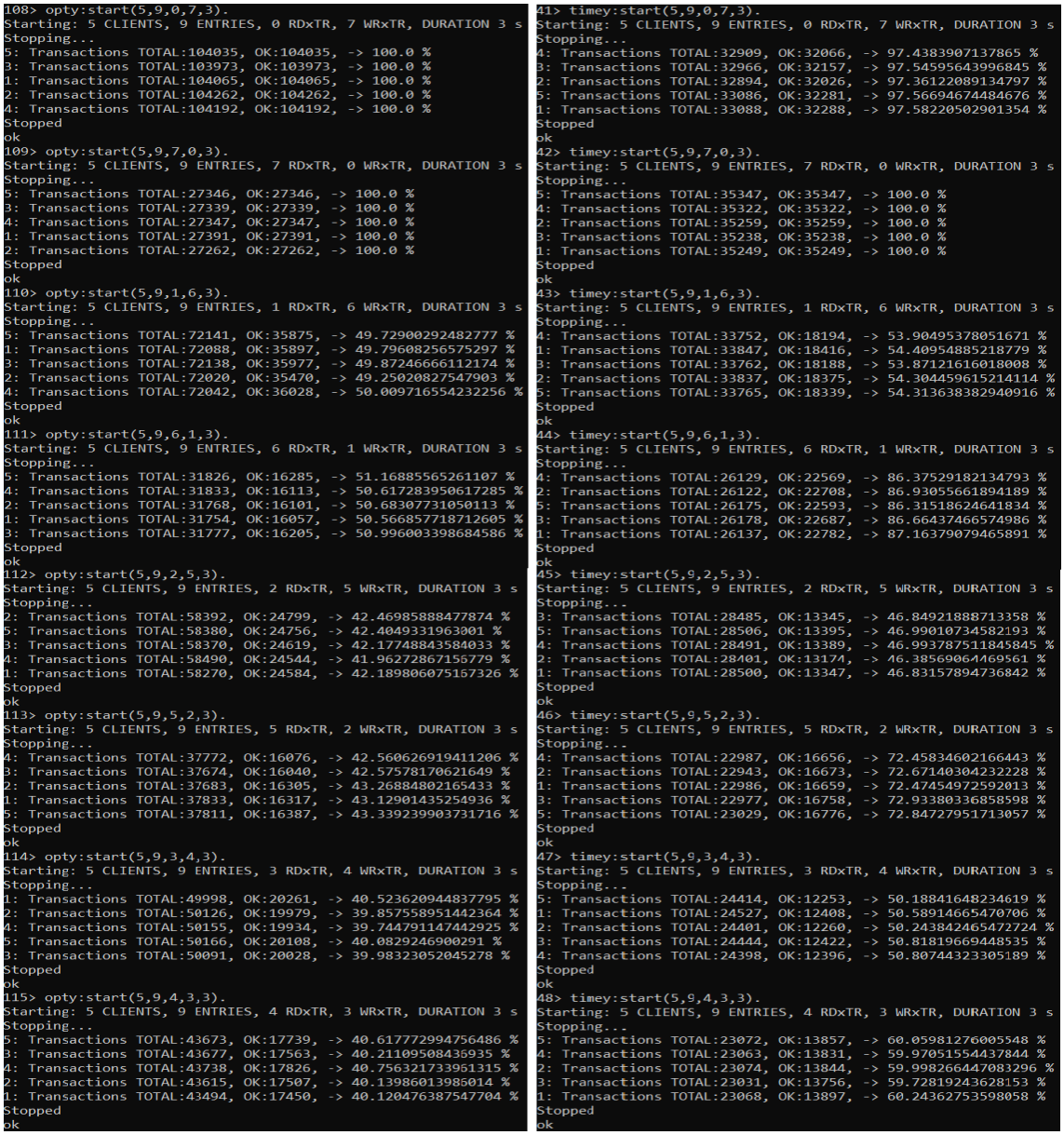

Figure 28: Output when changing the ratio of read and write operations per transaction for Opty vs Timey.

1.3.6 Percentage of Accessed Entries.

Impact of different percentage of accessed entries with respect to the total number of entries using Optimistic Concurrency Control and Timestamp Ordering Concurrency Control.

Table 13 and 14 contains the observations recorded when varying the percentage of accessed entries with respect to the total number of entries. The figure 29 and 31 depicts a plot of the same. Figure 30 and 32 depicts only a part of the output for each of the two experiments. We can see that as the percentage of accessed entries increases from 10% to 100%, the success rate decreases. We can see that the success rate is higher for Timey than Opty and also that the total number of transactions for Timey are lower than that for Opty.

Below are two experiments done:

- Experiment 1: Clients: 5, Entries: 10, Read sets per transaction: 2, Write sets per transaction: 2, Duration: 3s.

Accessed Entries(%)	Success-Opty(%)	Success-Timey(%)
10	87.92	96.34
20	75.24	88.80
30	69.14	85.35
40	66.46	83.60
50	63.91	82.55
60	62.60	81.65
70	61.53	81.36
80	60.75	80.94
90	60.29	80.57
100	59.68	80.48

Table 13: Observations recorded after changing the percentage of accessed subsets of entries.

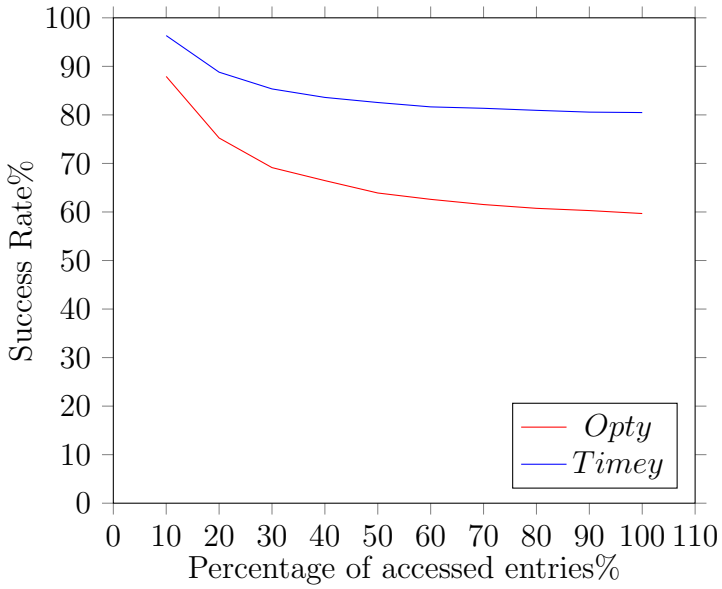

Figure 29: Impact of different percentage of accessed entries with respect to the total number of entries.

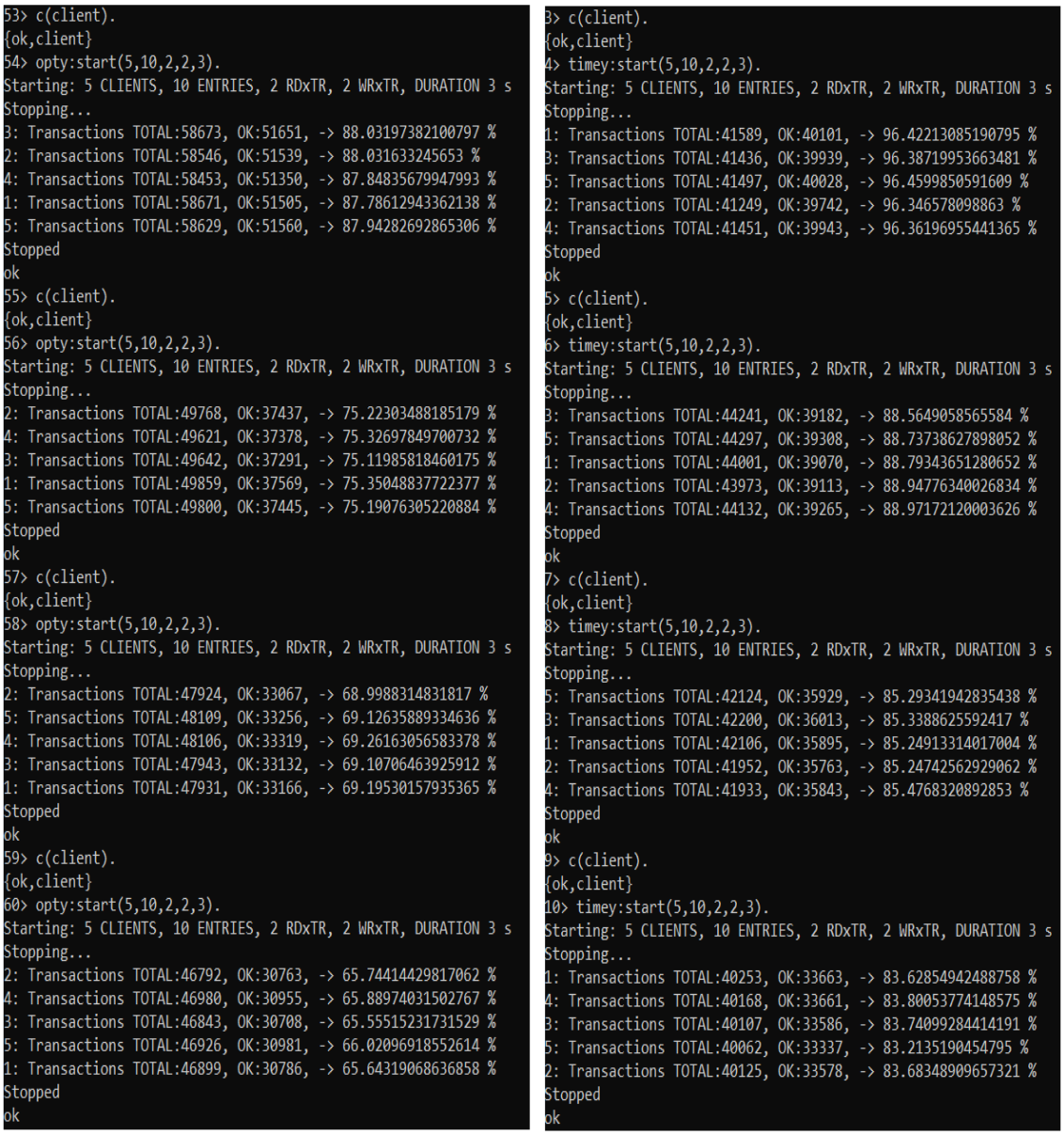

Figure 30: Output when percentage of accessed entries with respect to the total number of entries is varied for Opty vs Timey.

- Experiment 2: Clients: 7, Entries: 10, Read sets per transaction: 2, Write sets per transaction: 2, Duration: 3s.

Accessed Entries(%)	Success-Opty(%)	Success-Timey(%)
10	83.20	94.25
20	67.15	83.85
30	60.13	79.25
40	56.75	77.10
50	54.16	75.68
60	52.98	74.69
70	51.85	74.12
80	51.11	73.53
90	50.67	73.35

Accessed Entries(%)	Success-Opty(%)	Success-Timey(%)
100	50.23	72.76

Table 14: Observations recorded after changing the percentage of accessed subsets of entries.

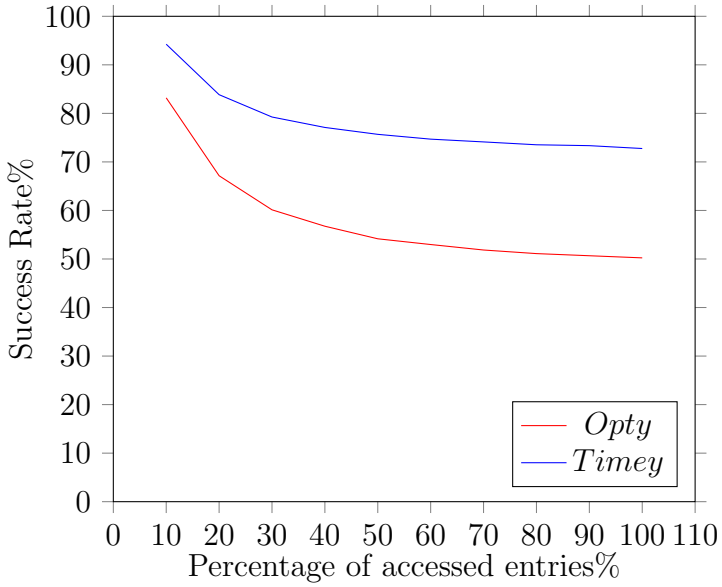

Figure 31: Impact of different percentage of accessed entries with respect to the total number of entries.

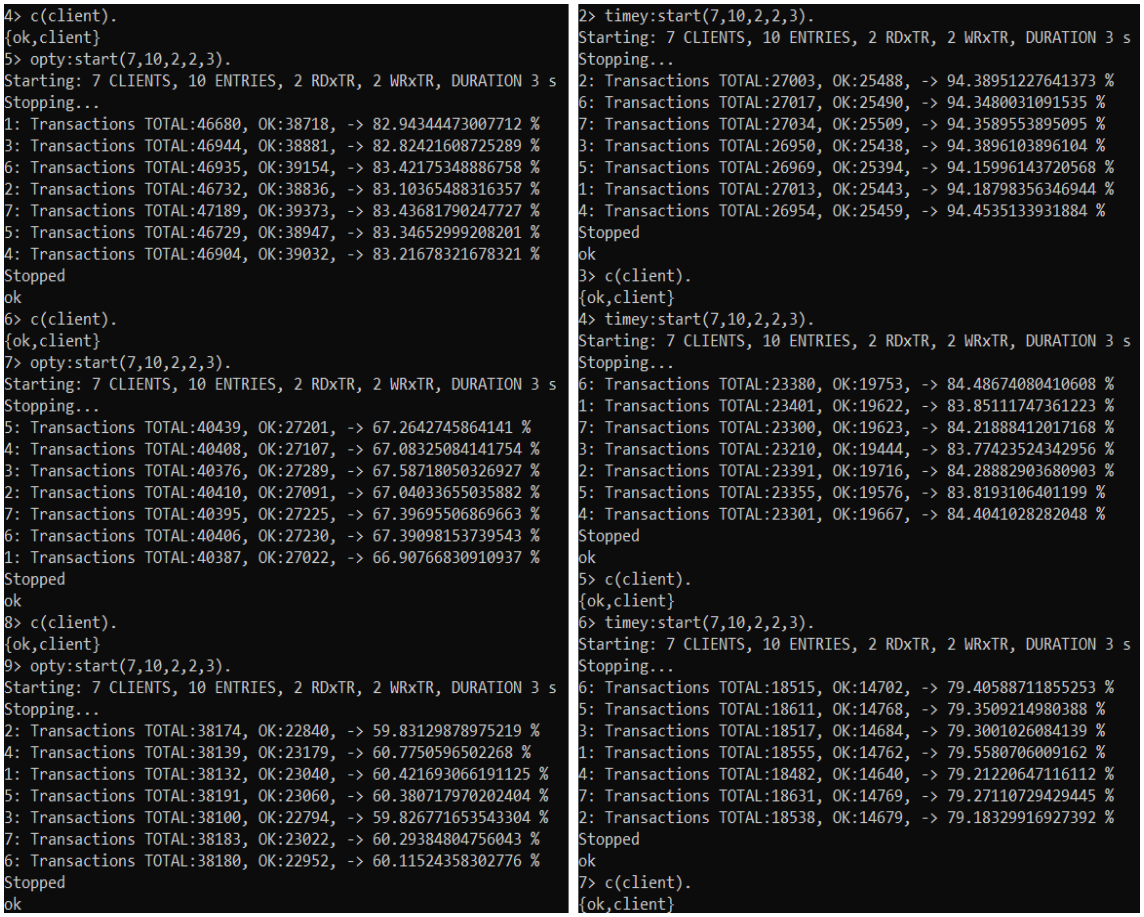

Figure 32: Output when percentage of accessed entries with respect to the total number of entries is varied for Opty vs Timey.

Open questions

2.1 Performance Parameters.

2.1.1 Number of Clients.

(Q3.1.1) What is the impact of number of clients on the success rate.

Answer: When we have a single client, there is no possibility of transaction conflicts and hence, the transaction does not fail the validation condition for backward validation. As the number of clients increases/latencies are high, the success rate decreases as more number of clients wants to read or write into the entries and hence more possibility of transactions conflicts taking place. This causes the validator to abort the conflicting transactions as they have failed the validation test. The margin of decrease in success rate decreases when larger number of clients are present, that is, there is greater margin of decrease for lesser number of clients and lesser margin of decrease for larger number of clients. Since locks are not used, the data can be changed by transactions while they are being read by other concurrent transactions as they are open to serve a large number of clients in less time. Therefore, optimistic concurrency control is more generally used in environments where there is low contention for data. Aborting a transaction leads to other transactions that read the dirty data from this transaction also to be aborted too. Any transactions that read from those transactions also need to be aborted. In other words, cascading aborts. Therefore, the success rate is not exactly the same but is similar among all clients.

2.1.2 Number of Entries.

(Q3.1.2) What is the impact of number of entries on the success rate?

Answer: As the number of entries increases, the success rate increases. This is because as the number of entries in the store increases, the order of the read and write operations in the transaction does not impact the success rate too much. More entries allows for less transaction conflicts as the processes have more entries to read or write into. Therefore, for fewer entries, the order of these operations in the transaction has a greater impact on the success rate and hence the success rate decreases as the number of entries decreases. Cascading aborts causes the success rate to differ among clients although the difference is not major.

2.1.3 Number of Read Operations Per Transection.

(Q3.1.3) What is the impact of number of read operations per transaction on the success rate?

Answer: As the number of read operations per transaction increases, the success rate decreases. This is because there is more contention for resources from the different transaction handlers. More entries are in contention when there are increasing number of read operations per transaction. Conflicting readers also causes the transaction to be aborted as when the read transactions are trying to be resolved/have been resolved, there might be other read transactions that take place later and cause conflicts to take place. Therefore, more number of read operations per transaction causes more conflicts. Longer transactions run for a longer time than average size transactions and hence have increased risk of being subject to conflicts and being aborted and restarted. Cascading aborts causes the success rate to differ among clients. Nevertheless, they are quite similar.

2.1.4 Number of Write Operations Per Transaction.

(Q3.1.4) What is the impact of number of write operations per transaction on the success rate?

Answer: As the number of write operations per transaction increases, the success rate decreases. This is because when there are larger number of write operations per transaction, more number of entries will be accessed which increases the probability of transaction conflicts as there are usually many clients starting transactions concurrently. When the validator checks the validation conditions for the writing the data from the local copy to the store, there may be active concurrent transactions read sets that might conflict with these write sets. This causes the conflicting transaction to be aborted and hence decreases the success rate. As there are increasing write operations per transaction, it could be that while the validation of the write set is taking place, a value in the active read set by other concurrent transactions might have changed by a write operation on them. The write sets may overlap with the other concurrent write sets. Longer transactions run for a longer time than average size transactions and hence have increased risk of being subject to conflicts and being aborted and restarted. Cascading aborts causes the success rate to differ among clients although the difference is minimal.

2.1.5 Ratio of Read and Write Operations Per Transaction.

(Q3.1.5) What is the impact of ratio of read and write operation for a fixed amount of operations per transaction on the success rate?

Answer: As the difference between the number of read operations per transaction and write operations per transaction decreases, the success rate decreases. More number of read operations per transaction compared to the write operations per transaction causes more success rate than vice versa. This is so because the read operations do not modify the data in the entries whereas, the write operations do. Since it is backward validation, the data to be written depends on the previous read. Read operations use lesser number of transactions compared to write operations as the conflict validation takes place to write the values into the entries. Longer transactions run for a longer time than average size transactions and hence have increased risk of being subject to conflicts and being aborted and restarted. Cascading aborts causes the success rate to differ among clients although they are quite similar.

2.1.6 Percentage of Accessed Entries.

(Q3.1.6) What is the impact of different percentage of accessed entries with respect to the total number of entries on the success rate?

Answer: As the percentage of accessed entries increases, the success rate decreases as when the percentage of accesses entries increases, more number of subsets of entries are being accessed by the clients and therefore, more chances of conflicts taking place as they are working concurrently. Since locks are not used, the data can be modified while another read transaction is taking place and so when the read takes place, the read transaction will return different values from the ones expected. Below is the code where changes are done to incorporate the subsets of entries being generated randomly using the generateSubsetEntry(Entries) function. NumI will contain the nth index from the list of entries.

Note: The value of "s" needs to be changed and compiled for each case.

Listing 1: client.erl

```
-define(s,1).

generateSubsetEntry(Entries) ->
    lists:sublist([X ||{_,X} <- lists:sort([{rand:uniform(), E} || E <-
       lists:seq(1,Entries)])], ?s).
```

```
start(ClientID, Entries, Reads, Writes, Server) ->
    spawn(fun() -> open(ClientID, Entries, Reads, Writes, Server, 0, 0) end).

open(ClientID, Entries, Reads, Writes, Server, Total, Ok) ->
    Server ! {open, self()},
    receive
        {stop, From} ->
            io:format("~w: Transactions TOTAL:~w, OK:~w, -> ~w % ~n",
            [ClientID, Total, Ok, 100*Ok/Total]),
            From ! {done, self()},
            ok;
        {transaction, Validator, Store} ->
            Handler = handler:start(self(), Validator, Store),
            case do_transaction(ClientID, generateSubsetEntry(Entries), Reads,
                Writes, Handler) of
                ok ->
                    open(ClientID, Entries, Reads, Writes, Server, Total+1, Ok+1);
                abort ->
                    open(ClientID, Entries, Reads, Writes, Server, Total+1, Ok)
            end
    end.

    do_read(Entries, Handler) ->
    Ref = make_ref(),
    Num = rand:uniform(length(Entries)),
    NumI = lists:nth(Num,Entries),
    Handler ! {read, Ref, NumI},
    receive
        {value, Ref, Value} -> Value
    end.

do_write(Entries, Handler, Value) ->
    Num = rand:uniform(length(Entries)),
    NumI = lists:nth(Num,Entries),
    Handler ! {write, NumI, Value}.
```

2.2 Success Rate Evaluation For Individual Clients.

(Q3.2) Is the success rate the same for different clients?

Answer: No, the success rate is not exactly the same for different clients but they are very similar in all cases except when the accessed subsets are generated randomly. When the subsets are generated randomly, they are not accessed in order and therefore, they can cause the operations to overlap. This causes a difference in success rates between clients. When a transaction conflict is reported, the transaction is aborted. There might be other transactions that read this dirty data and hence need to be aborted too. Therefore, due to this cascading effect, different number of transactions are aborted for different clients. This has been demonstrated in the experiments.

2.3 Distributed Execution

(Q4) If we run this in a distributed Erlang network, where is the handler running?
Answer: The handler is running on the client side.

2.4 Other concurrency control techniques - Timey

(Q5.2) Compare all previous performance results of the transaction server when using concurrency control with backward validation with respect to timestamp ordering.

Answer: In Timey, timestamps are used to store the date and time when a value is updated in the store which are used like a reference to avoid transaction conflicts. The timestamp is assigned to a transaction when the transaction starts. The schedule in which the transactions participate is serializable. When an update/write is attempted, the timestamp of the database is compared with the original timestamp value of the modified row. If they are same, the update is performed. Else, a concurrency violation is identified. Cascading aborts are minimized which is the main reason for higher success rate as compared to Opty approach. The number of transactions are also lesser as the validation to avoid conflicts is done at an earlier stage as compared to Otpy where initially we assume that conflicts rarely happen and no locking strategy is used on the data.

2.4.1 Number of Clients.

The success rate is better when using Timey implementation than when using Opty as when more clients are present in Opty and no locking used, more entries are being accessed by the processes and hence more probability of conflicts. Whereas, in Timey, the timestamp of the object is compared with the read or write operation before it is committed which prevents conflicts. Although, the success rate does dxecrease for Timey as the number of clients increases. The number of transactions are lesser in Timey than in Opty.

2.4.2 Number of Entries.

The success rate is better when using Timey implementation than Opty. As the number of entries increases, there are more entries for transaction write operation and hence less conflicts will take place. As the number of entries increases, the success rate also increases. Then total number of transactions are lesser in Timey than in Opty.

2.4.3 Number of Read Operations Per Transaction.

The success rate is better when using Timey implementation than Opty for a smaller number of write operatios per transaction. For larger number, the sccess rate becomes lower in the case of Timey than that for Opty. The success rate decreases as the number of read operations per transaction are increased. This is because more read operations causes more entries to be accessed by the transaction process and more possibility of conflicts. The total number of transactions for Timey implementation is lesser than for Opty.In Opty since the read operation is done without any restrictions, there are more conflicts later on. In Timey a read is not done until the previous object in the entry has been written. If the timestamp of the current object is greater than the written object's timestamp, then the transaction is aborted.

2.4.4 Number of Write Operations Per Transaction.

The success rate is better when using Timey implementation than Opty. The success rate decreases as the number of write operations per transaction are increased. This is because

more write operations causes more entries to be accessed by the transaction process and more possibility of conflicts. The total number of transactions for Timey implementation is lesser than for Opty. In Opty since the write operation proceeds to validation phase without any restrictions, there are more conflicts later on as data in the entries might be getting modified in the meantime as there are no locks used. In Timey a write is not done until the previous object in the entry has been read and written y earlier transactions.

2.4.5 Ratio of Read and Write Operations Per Transaction.

Answer: As the difference between the number of read operations per transaction and write operations per transaction decreases, the success rate decreases. More number of read operations per transaction compared to the write operations per transaction causes more success rate than vice versa. This is so because the read operations do not modify the data in the entries whereas, the write operations do. The performance of Timey is better than Opty implementation as we use timestamps to prevent transaction conflicts rather than assuming that transaction conflicts will occur rarely and doing the validation just before commit phase as is done in Opty.

2.4.6 Percentage of Accessed Entries.

What is the impact of different percentage of accessed entries with respect to the total number of entries on the success rate?

Answer: As the percentage of accessed entries increases, the success rate decreases as when the percentage of accesses entries increases, more number of subsets of entries are being accessed by the clients and therefore, more chances of conflicts taking place as they are working concurrently. The Success rate is better for Timey implementation than Opty implementation. The subset of entries are generatedby the function generateSubsetEntry(Entries). NumI will contain the nth index from the list of entries.

Note: The value of "s" needs to be changed and compiled for each case.

Listing 2: client.erl

```
-define(s,1).

generateSubsetEntry(Entries) ->
    lists:sublist([X ||{_,X} <- lists:sort([{rand:uniform(), E} || E <-
        lists:seq(1,Entries)])], ?s).

start(ClientID, Entries, Reads, Writes, Server) ->
    spawn(fun() -> open(ClientID, Entries, Reads, Writes, Server, 0, 0) end).

open(ClientID, Entries, Reads, Writes, Server, Total, Ok) ->
    Server ! {open, self()},
    receive
        {stop, From} ->
            io:format("~w: Transactions TOTAL:~w, OK:~w, -> ~w % ~n",
            [ClientID, Total, Ok, 100*Ok/Total]),
            From ! {done, self()},
            ok;
        {transaction, Time, Store} ->
            Tref = make_ref(),
            Handler = handler:start(self(), Tref, Time, Store),
            case do_transaction(ClientID, generateSubsetEntry(Entries), Reads,
                Writes, Handler, Tref) of
```

```
                ok ->
                    open(ClientID, Entries, Reads, Writes, Server, Total+1, Ok+1);
                abort ->
                    open(ClientID, Entries, Reads, Writes, Server, Total+1, Ok)
            end
    end.

   do_read(Entries, Handler, Tref) ->
    Ref = make_ref(),
    Num = rand:uniform(length(Entries)),
    NumI = lists:nth(Num,Entries),
    Handler ! {read, Ref, NumI},
    receive
        {value, Ref, {ok, Value}} ->
            Value;
        {abort, Tref} ->
            abort
    end.

do_write(Entries, Handler, Value, Tref) ->
    Ref = make_ref(),
       Num = rand:uniform(length(Entries)),
    NumI = lists:nth(Num,Entries),
    Handler ! {write, Ref, NumI, Value},
    receive
        {value, Ref, ok} ->
            ok;
        {abort, Tref} ->
            abort
    end.
```

Personal opinion

Provide your personal opinion of the seminar, indicating whether it should be included in next year's course or not.

This seminar was a very interesting opportunity to implement a concrete application of the Optimistic Concurrency Control method and more generally of the material dealt with during the lectures of the Distributed Systems class. The hands-on experience helped to understand it better and in a more practical way.

The structure of the statement of this assignment was very well guided and it allowed a harmonic progression throughout all the content asked. The comparison to be done between Timestamp Ordering Concurrency Control and Optimistic Concurrency Control helped to understand both the concurrency control methods better as we had to play with the parameters and observe the performance results.

Moreover, it allowed me to learn a lot more about Erlang, a programming language that I have just learned and used in the previous seminar. I was able to successfully understand and complete the codes or portions of codes given in the statement of the assignment or attached to it.

Finally, I think that this seminar, in the way it is proposed right now, should without a doubt be included in next year's course, as it comes right after the part of the course about concurrency control, as well as the Erlang tutorial and gives us an opportunity to a hands-on experience. It provides a very good training of very recently studied material and thus contributes to a significant progress.

Conclusion

This assignment was an interesting opportunity to work on the subject of distributed systems with Erlang. It required several efforts in the subjects of algorithmic, scientific research and computer programming, that opened the gate to significant progress. I would like to respectfully thank, once again, Dr Jordi Guitart Fernandez for assigning these problems and for taking the time to read the results of my work and research presented in this report.

References

[1] [Kung81] Kung, H.T., Robinson, J.T., On Optimistic Methods for Concurrency Control, ACM Transactions on Database Systems, Vol. 6, No. 2, pp. 213-226, June 1981.

[2] [Harder84] Harder, T., Observations on Optimistic Concurrency Control Schemes, Information Systems, Vol. 9, No. 2, pp. 111- 120, November 1984.