Version 2, 1 oct 2024 08:33

Server Processes or Threads?

research from the parmington foundation

Gus Bjorklund,

head groundskeeper

the parmington foundation

Waltham, MA USA
29 sep 2024
02 oct 2024

Abstract

In this talk we discuss the results of a series of test where we compared the performance of OpenEdge database servers with and without multithreading.

Multiple servers each serving multiple clients compared to multithreaded servers serving multiple clients.

Database Server Test Machine

Dell PowerEdge 2050 iii (circa 2009) with Centos 7.9

- 2 XEON E5450 3 GHz 4-core processors
- 64 GB ECC RAM
- PERC/6 RAID controller
- 12 TB disk storage (6 Hitachi 2 TB 7200 rpm SATA disk drives)
  - /boot (xfs), /home (xfs) and swap on 4x2 TB sata drives in RAID 0.
  - /bi (ext3) on 2 tb drive
  - /ai (ext3) on 2 tb drive

“furgal test” time: 0.42 seconds. (228 mb/sec)

4GL Client Test Machine

Lenovo ThinkCentre M82

- Intel Core i5-3470 Quad-Core CPU
- 32 GB RAM
- 1 TB PCIe M.2 SSD
- Operating system is Rocky Linux 10

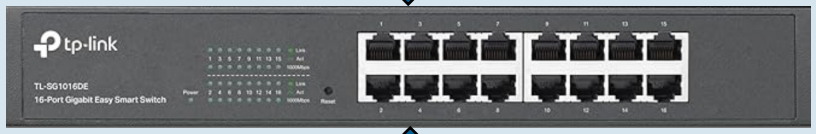
“furgal test” time: 2.0 seconds. (48 MB/sec)

Network Setup

Dell database server

cat5 ethernet cable

gigabit ethernet switch



cat5 ethernet cable

Lenovo client machine

iperf bandwidth client to server machine is 801 Mbits/second
ping time client to server machine is 0.092 millisecond

Database Parameters On Database Server (server.pf)

- -Mn 200 # maximum number of servers
- -n 500 # maximum number of users (connections)
- -H michaelson # database server host IP
- -S 5108 # broker's login port
- -minport 5100 # lowest port number to use
- -maxport 7100 # highest port number
- -prefetchDelay # enable delay of first message for query result
- -prefetchFactor 100 # fill message buffer before sending
- -prefetchNumRecs 100 # max records to add to message

- -threadedserver 0 # threaded server off (1 means on)
- -Ma 1 # maximum clients per server
- -Mi 1 # minimum clients per server
- -Mm 16384 # maximum message size

ATM Benchmark Refresher: Description

- Standard “Secret Bunker” Benchmark
  - uses a baseline config always the same since Bunker #2
- Simulates ATM withdrawal transaction
- 150 concurrent users
  - execute as many transactions as possible in given time
- Highly update intensive
  - fetch 3 rows
  - update 3 rows
  - create 1 row with 1 index entry

```
find account where account.id = theAccount exclusive-lock.
    assign account.balance = account.balance + delta.

create history1.
    assign history1.account   = theAccount
           history1.teller    = theTeller
           history1.branch    = theBranch
           history1.delta     = delta
           history1.balance   = account.balance
          .

find teller where teller.id = theTeller exclusive-lock.
    assign teller.balance = teller.balance + delta.

find branch where branch.id = theBranch exclusive-lock.
    assign branch.balance = branch.balance + delta.
```

ATM Benchmark Refresher: the standard baseline config for database operation

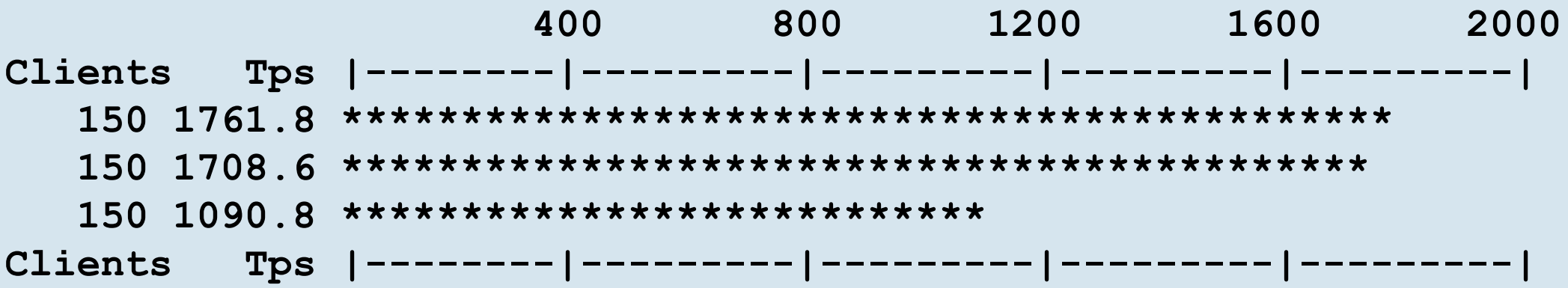
```
-maxAreas 90      # maximum storage areas
-n 250            # maximum number of connections
-L 10240          # lock table entries
-B 64000          # main buffer pool number of buffers
-spin 5000        # spinlock retries
-nap 10           # initial nap time
-napmax 250       # maximum nap time
-bibufs 32        # before image log buffers

1 BIW             # before image writer
1 APW             # asynchronous page writer
```

Some OpenEdge 12.8 default values

```
-threadedserver 1    # threaded server on
-Mn 5                # maximum number of servers
-Ma 125              # maximum clients per server
-Mi 1                # minimum clients per server
-Mm 8192             # maximum message size
-prefetchNumRecs 100 # max records to send in 1 message
```

Mm should be 16384
prefetchNumRecs should be at least 1000
We need Mn 200 so we can do -Ma 1 -Mi 1

Baseline Test: 150 Self-serving clients (no network clients)



Client-server over network, 12.8 default network settings (except Mn 200)

Threaded server enabled, -Mm 8192 -Mi 1, -Ma 125

```
                      200        400        600        800       1000
Clients   Tps  |---------|----------|----------|----------|----------|
   150  530.5  ****************************
   150  632.9  ************************************
   150  830.9  ****************************************************
Clients   Tps  |---------|----------|----------|----------|----------|
```

Client-server over network, -Ma 1

Threaded server enabled, -Mm 16384, -Mi 1, -Ma 1

```
                      200        400        600        800       1000
Clients   Tps |--------|---------|---------|---------|---------|
   150  518.8 **************************
   150  628.9 ********************************
   150  848.0 ******************************************
Clients   Tps |--------|---------|---------|---------|---------|
```

Client-server over network, -Ma 1

Threaded server OFF, -Mm 16384, -Mi 1, -Ma 1

```
                      300        600        900       1200       1500
Clients   Tps |---------|----------|----------|----------|----------|
   150 1203.9 **************************************
   150 1171.3 ************************************
   150 1197.8 **************************************
Clients   Tps |---------|----------|----------|----------|----------|
```

Client-server over network, -Ma 5

Threaded server enabled, -Mm 16384, -Mi 5, -Ma 5

```
                      300        600        900       1200       1500
Clients   Tps |--------|---------|---------|---------|---------|
   150 1159.8 **********************************
   150 1060.9 ********************************
   150  800.7 ************************
Clients   Tps |--------|---------|---------|---------|---------|
```

Client-server over network, -Ma 5

Threaded server OFF, -Mm 16384, -Mi 5, -Ma 5

```
                      300        600        900       1200       1500
Clients   Tps |--------|---------|---------|---------|---------|
   150 1170.0 ************************************
   150 1251.5 ****************************************
   150 1204.0 ************************************
Clients   Tps |--------|---------|---------|---------|---------|
```

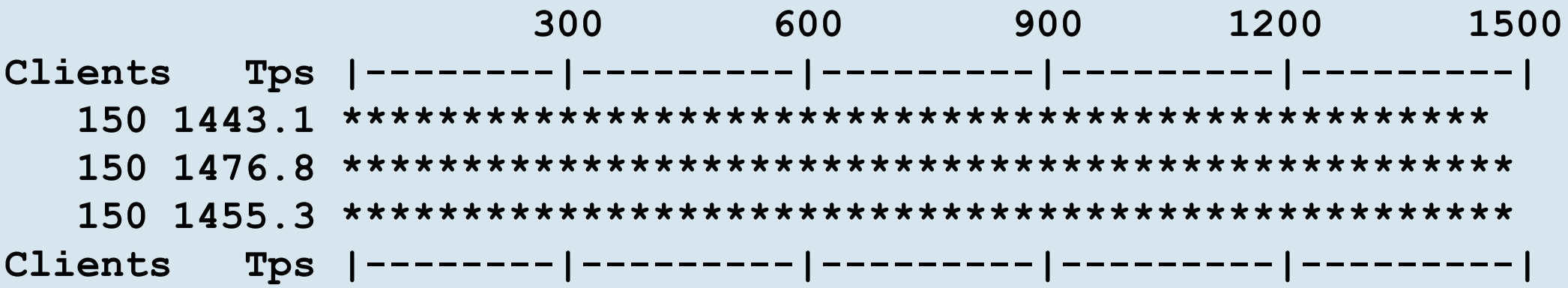
Client-server over network, -Ma 10

Threaded server enabled, -Mm 16384, -Mi 10, -Ma 10

```
                        300         600         900        1200        1500
Clients    Tps  |---------|----------|----------|----------|----------|
   150 1162.7  **************************************
   150 1165.5  **************************************
   150 1165.9  **************************************
Clients    Tps  |---------|----------|----------|----------|----------|
```

Client-server over network, -Ma 10

Threaded server OFF, -Mm 16384, -Mi 10, -Ma 10



Client-server over network, -Ma 20

Threaded server OFF, -Mm 16384, -Mi 20, -Ma 20

```
                      400        800       1200       1600       2000
Clients   Tps |---------|----------|----------|----------|----------|
   150 1551.9 ************************************
   150 1564.9 ************************************
   150 1553.1 ************************************
Clients   Tps |---------|----------|----------|----------|----------|
```

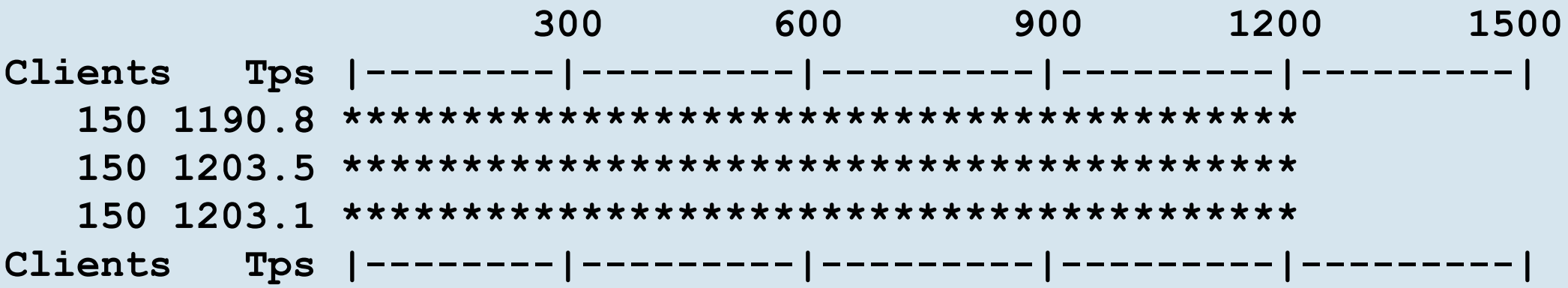
Client-server over network, -Ma 20

Threaded server enabled, -Mm 16384, -Mi 20, -Ma 20

```
                        300         600         900        1200        1500
Clients    Tps |---------|----------|----------|----------|----------|
   150 1205.1 **************************************
   150 1203.5 **************************************
   150 1199.2 **************************************
Clients    Tps |---------|----------|----------|----------|----------|
```

Client-server over network, -Ma 50

Threaded server enabled, -Mm 16384, -Mi 50, -Ma 50



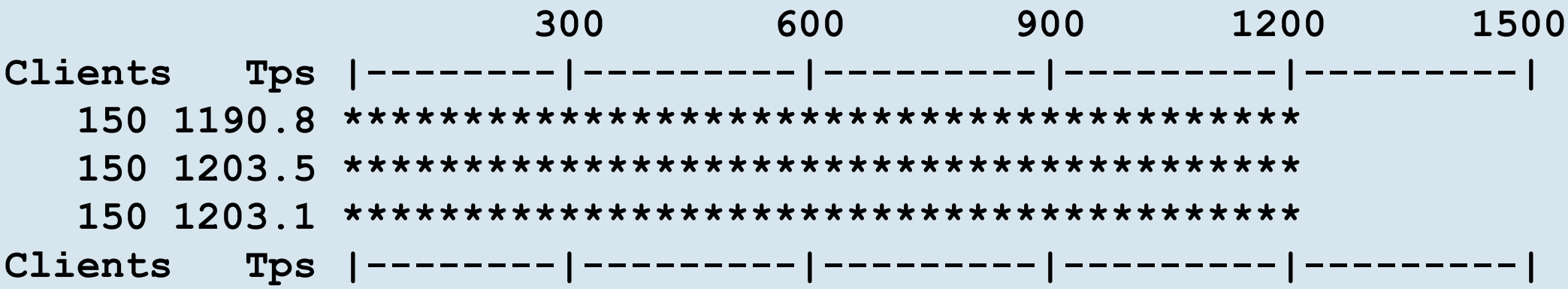
Client-server over network, -Ma 50

Threaded server OFF, -Mm 16384, -Mi 50, -Ma 50

```
                      400        800       1200       1600       2000
Clients   Tps |---------|----------|----------|----------|----------|
   150 1623.5 ************************************
   150 1599.6 ************************************
   150 1630.5 ************************************
Clients   Tps |---------|----------|----------|----------|----------|
```

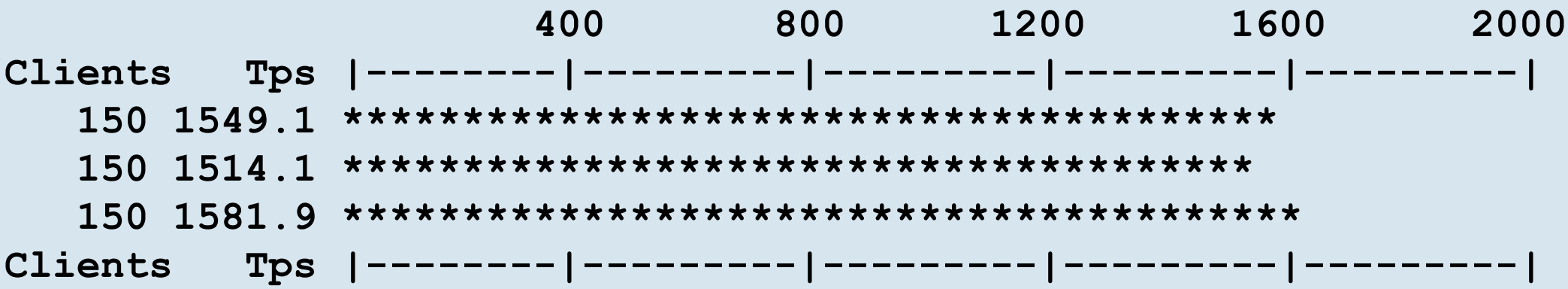
Client-server over network, -Ma 75

Threaded server enabled, -Mm 16384, -Mi 75, -Ma 75



Client-server over network, -Ma 75

Threaded server OFF, -Mm 16384, -Mi 75, -Ma 75



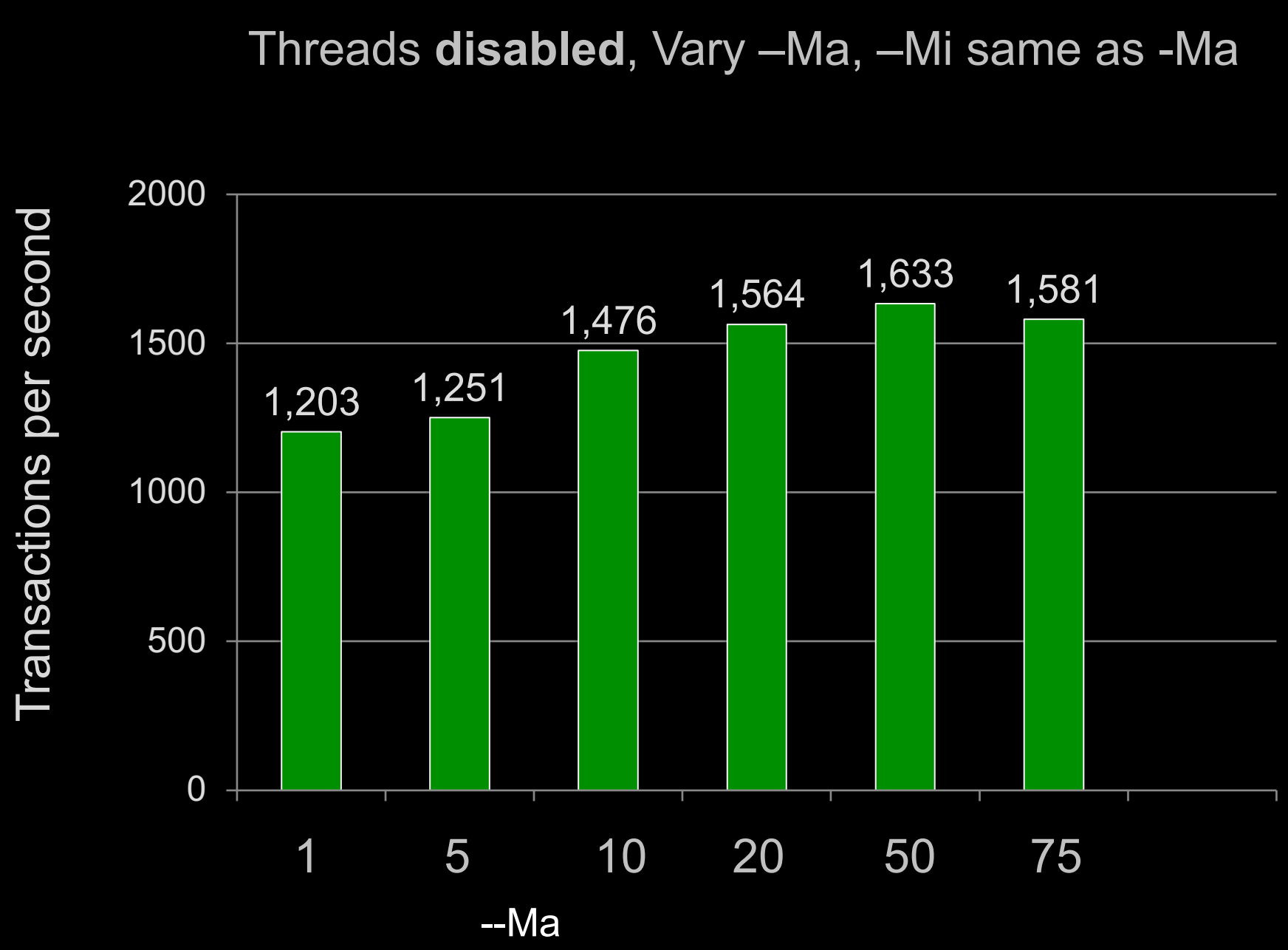
Results

# clients	threads	-Mi	-Ma	tps
150	N/A	N/A	N/A	1761
150	ON	1	125	830
150	ON	1	1	848
150	OFF	1	1	1203
150	ON	1	5	1159
150	OFF	1	5	1251
150	ON	1	10	1165
150	OFF	1	10	1476
150	ON	1	20	1205
150	OFF	1	20	1564
150	ON	1	50	1203
150	OFF	1	50	1633
150	ON	1	75	1207
150	OFF	1	75	1581

Self-serving

Default network


Threads **enabled**, Vary –Ma, –Mi same as -Ma
Transactions per second
2000
1500
1000
500
0
848
1,159
1,165
1,205
1,203
1,205
1
5
10
20
50
75
--Ma


Threads **disabled**, Vary –Ma, –Mi same as -Ma
Transactions per second
2000
1500
1000
500
0
1,203
1,251
1,476
1,564
1,633
1,581
1
5
10
20
50
75
--Ma

What have we learned ?

- For small update transactions, multithreaded server is not as fast as unthreaded servers

- Threaded throughput flattens at –Ma 20 at about 1200 tps
- Multi server throughput flattens at –Ma 20 at about 1600 tps
- Self-serving clients are faster at 1761 tps

- Testing with other types of transactions is needed.

But:

But: There's More !!

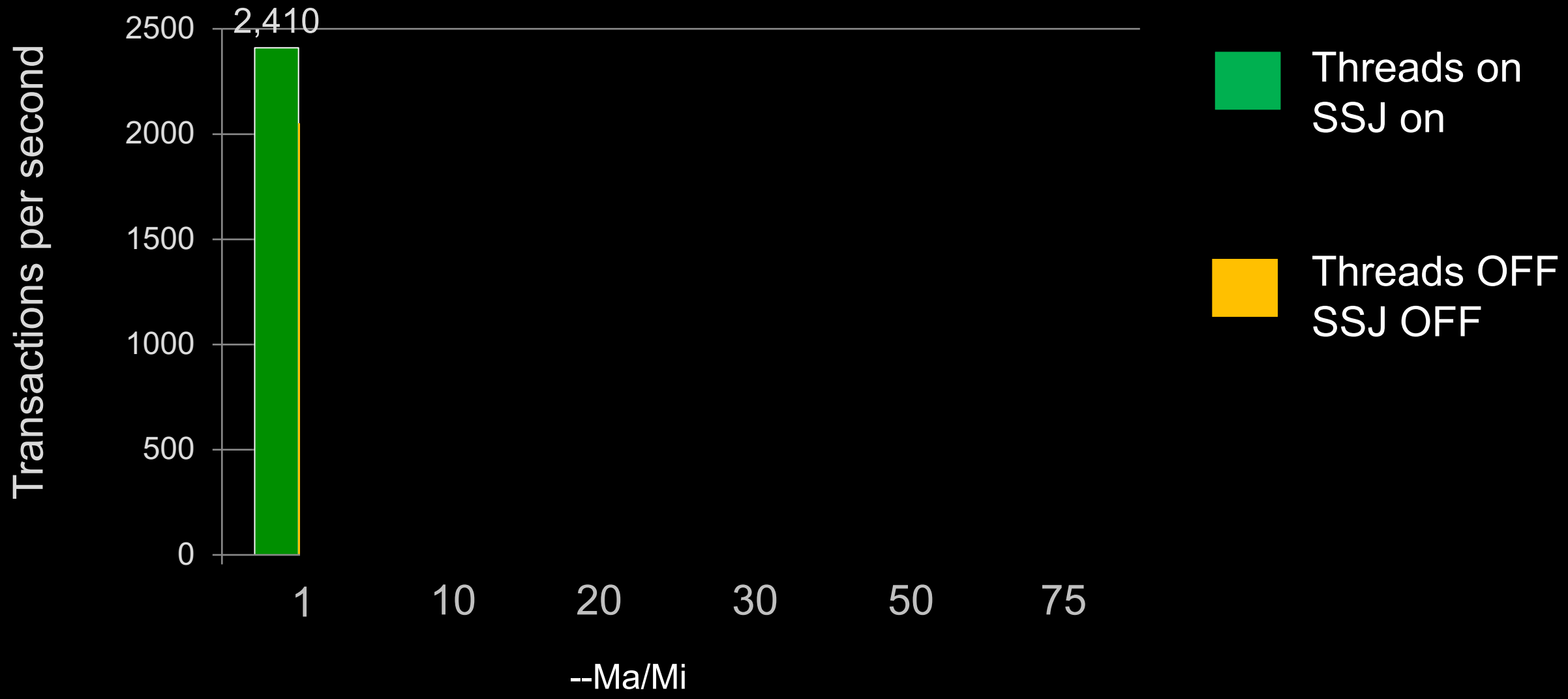
Let’s look at server-side joins

150 clients, as before
113 execute standard atm transaction
(3 updates, 1 create)
37 execute a join of branch and tellers
(reads only)

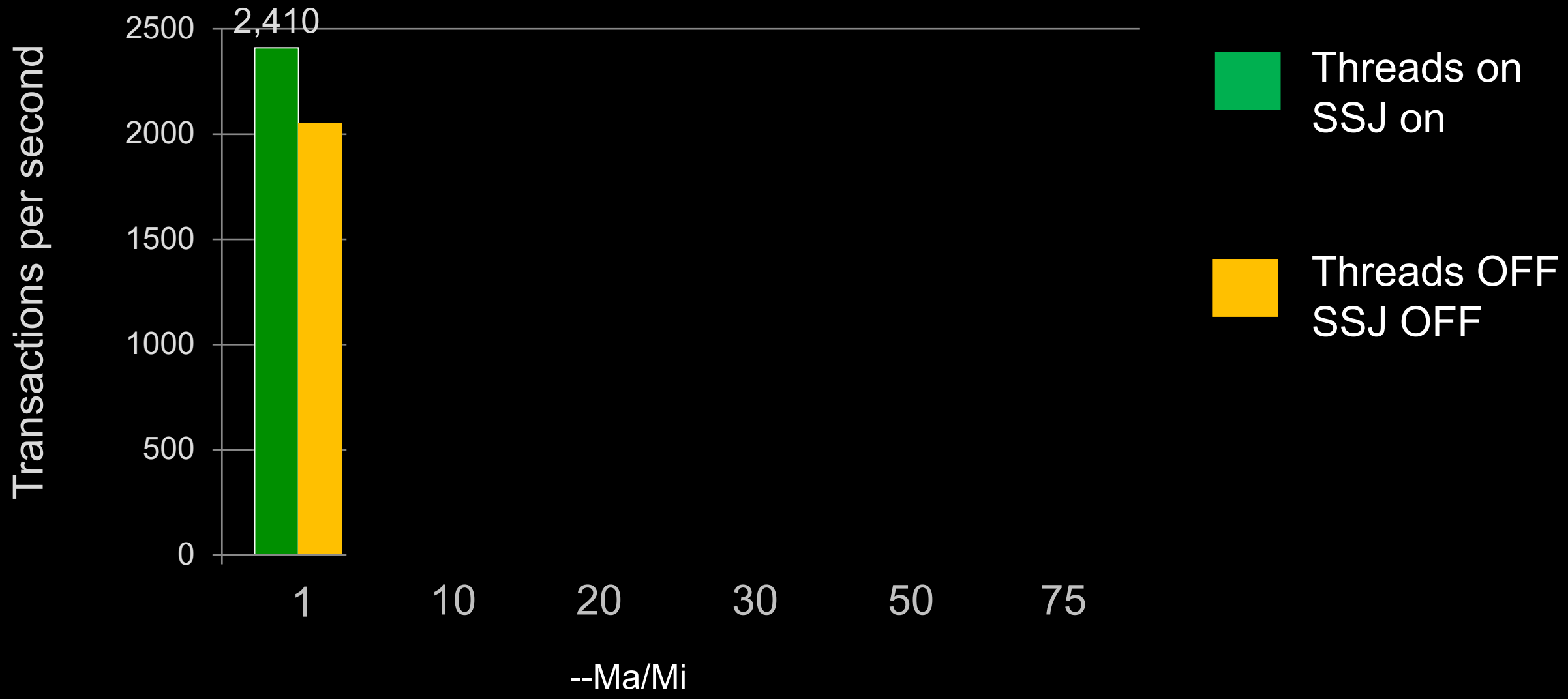
Join code

```
Assign theBranch  = random (1, lastBranch)
       .

totalBalance = 0.
numTellers = 0.
for each branch no-lock
       where (theBranch >= branch.id) and
             (branch.id <= (theBranch + 10)),
    each teller no-lock
       where (branch.id = teller.branchid):

    totalBalance = totalBalance + teller.balance.
    numTellers = numTellers + 1.
end.
```

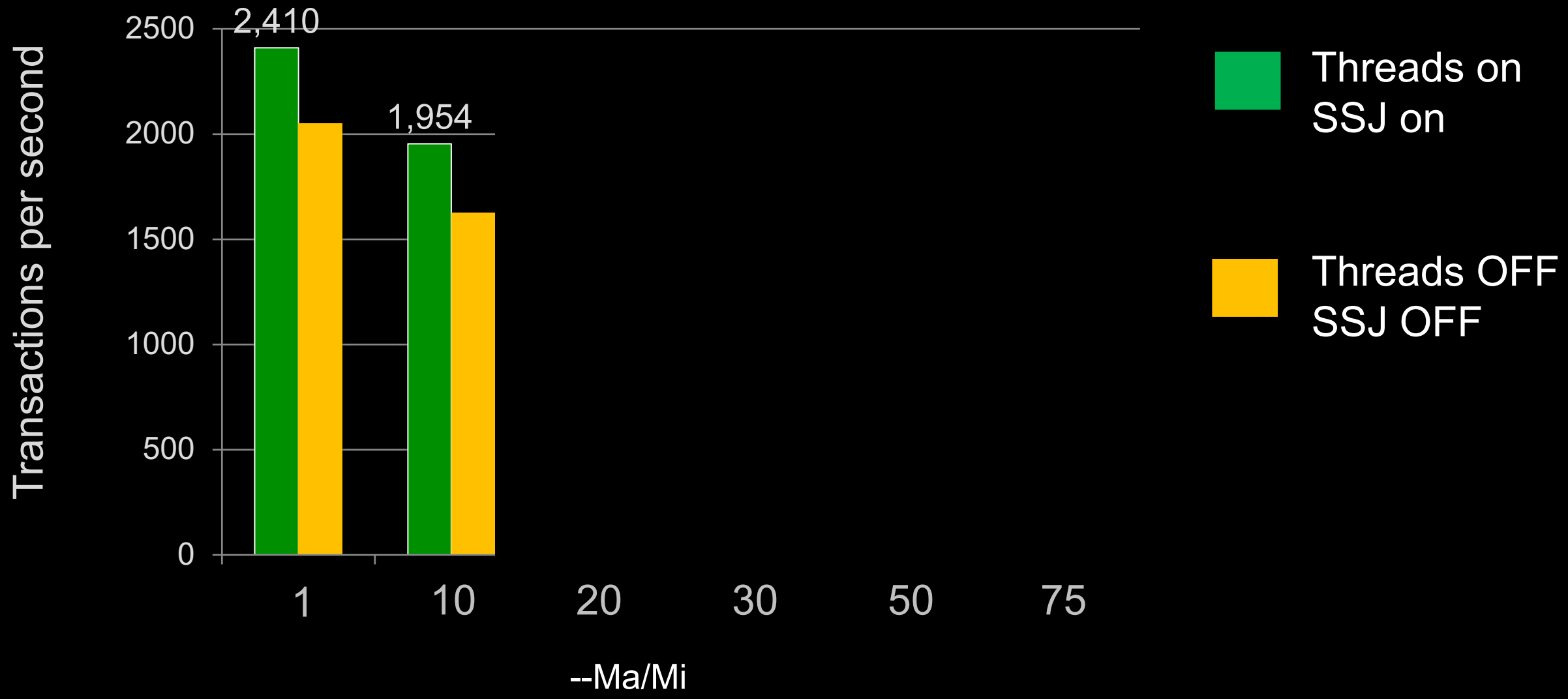
Join branch and teller
Vary –Ma, –Mi same as -Ma



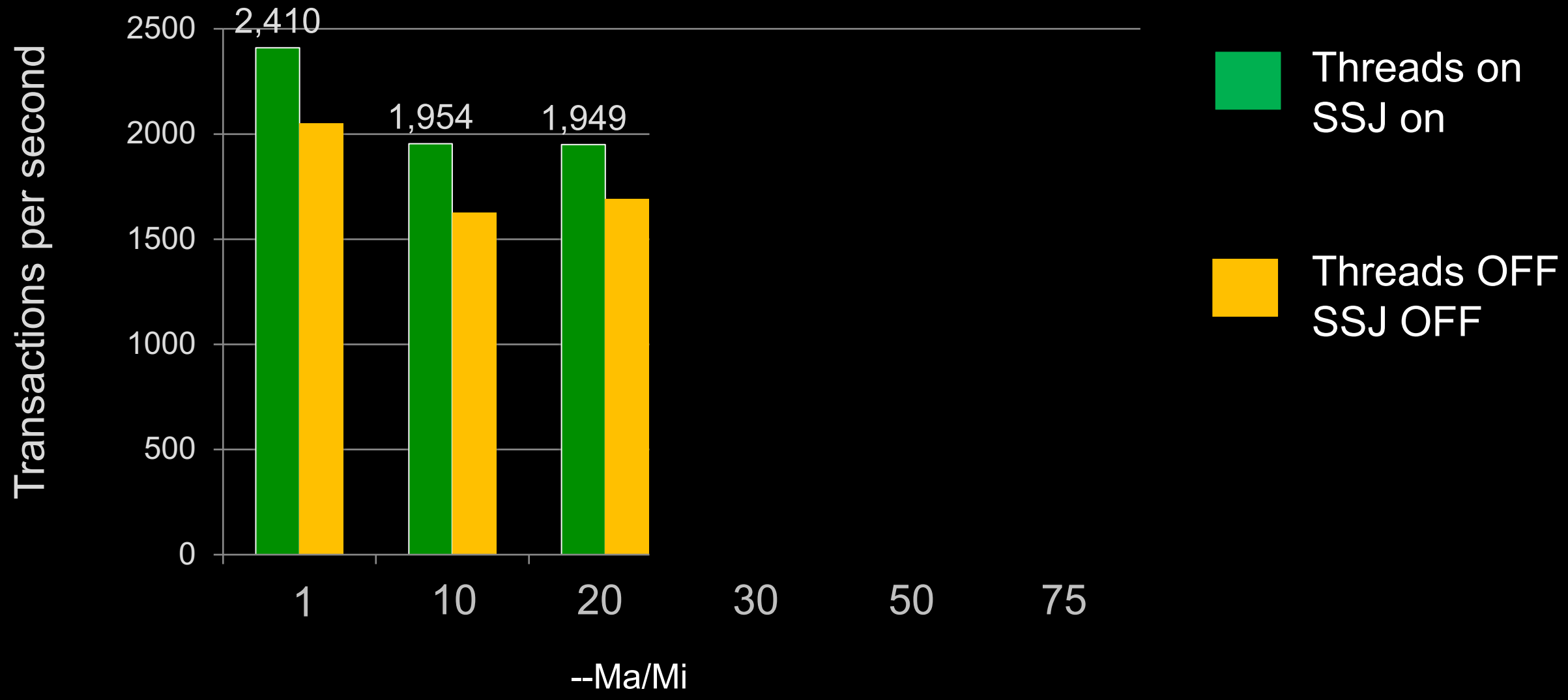
Join branch and teller
Vary –Ma, –Mi same as -Ma



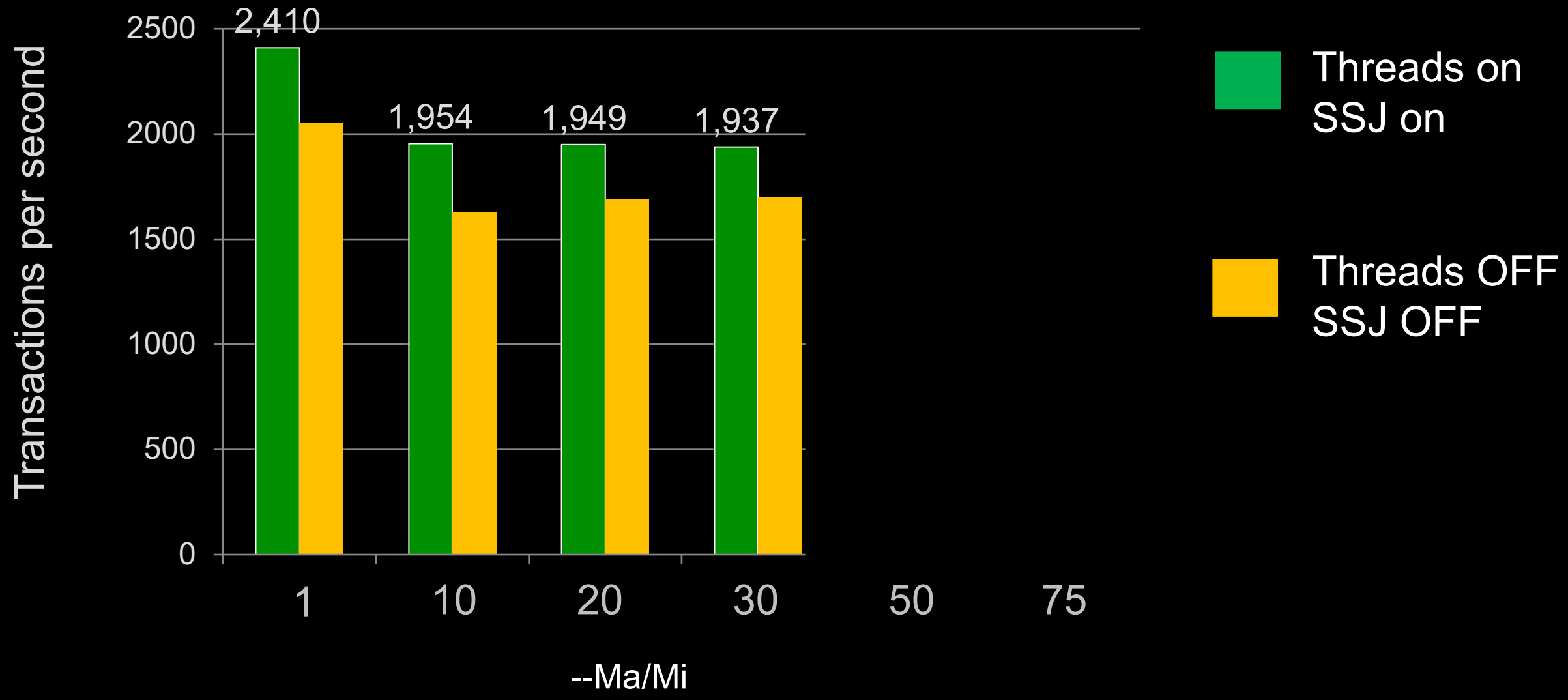
Join branch and teller
Vary –Ma, –Mi same as -Ma



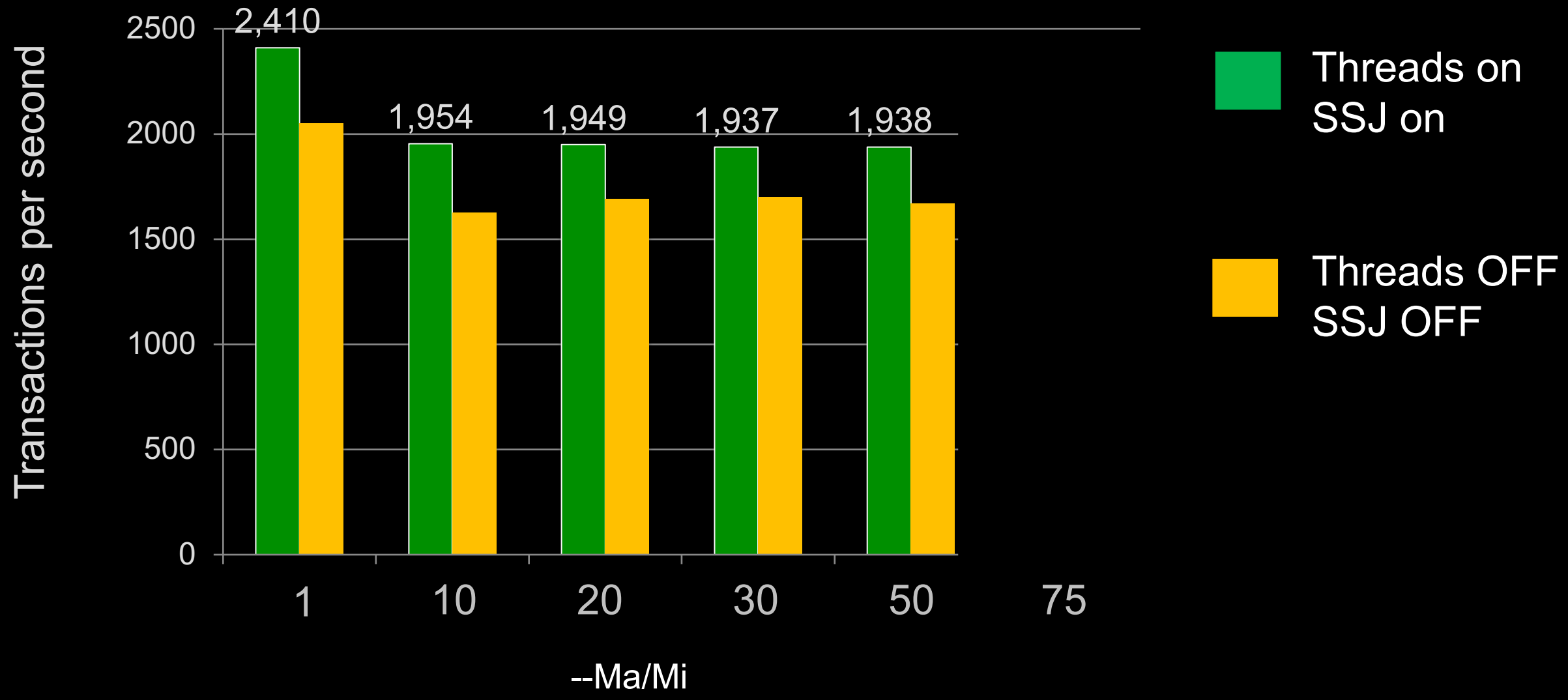
Join branch and teller
Vary –Ma, –Mi same as -Ma



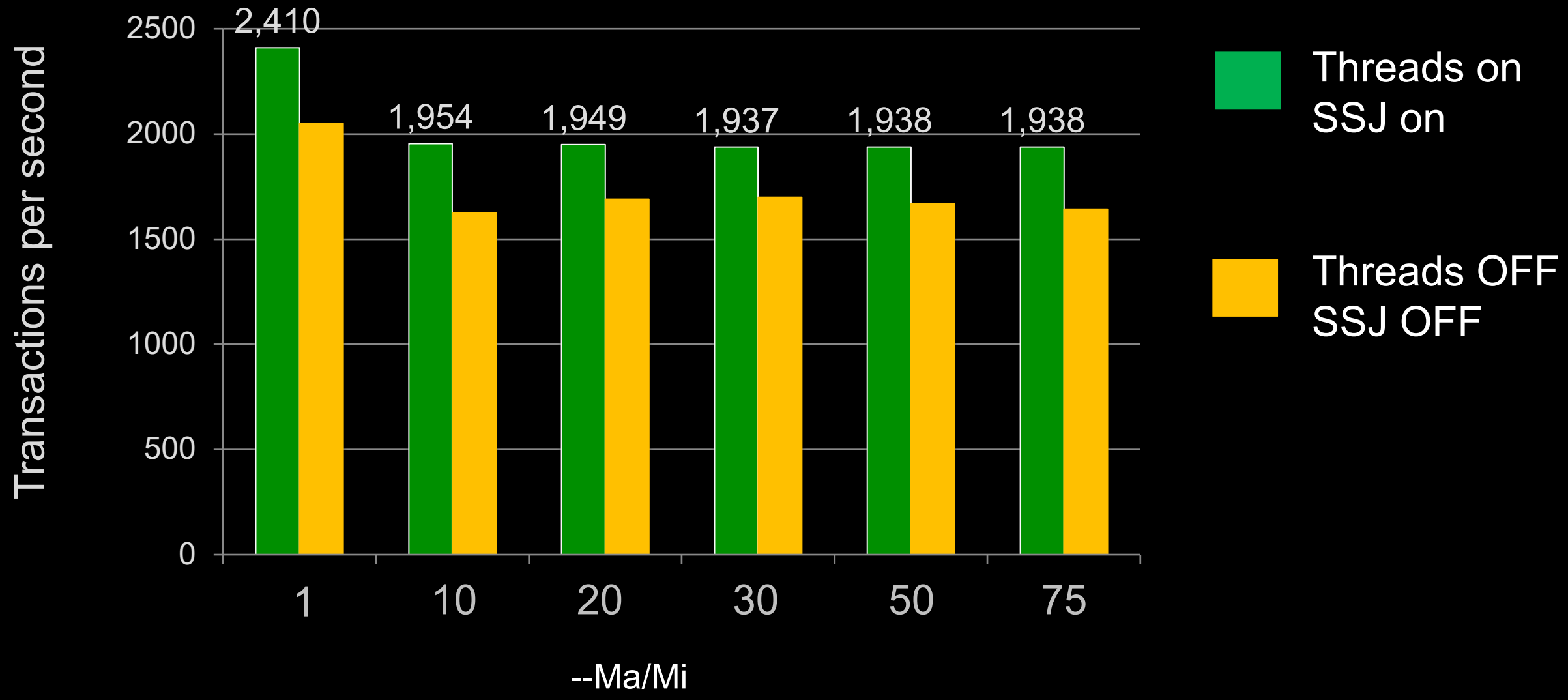
Join branch and teller
Vary –Ma, –Mi same as -Ma



Join branch and teller
Vary –Ma, –Mi same as -Ma

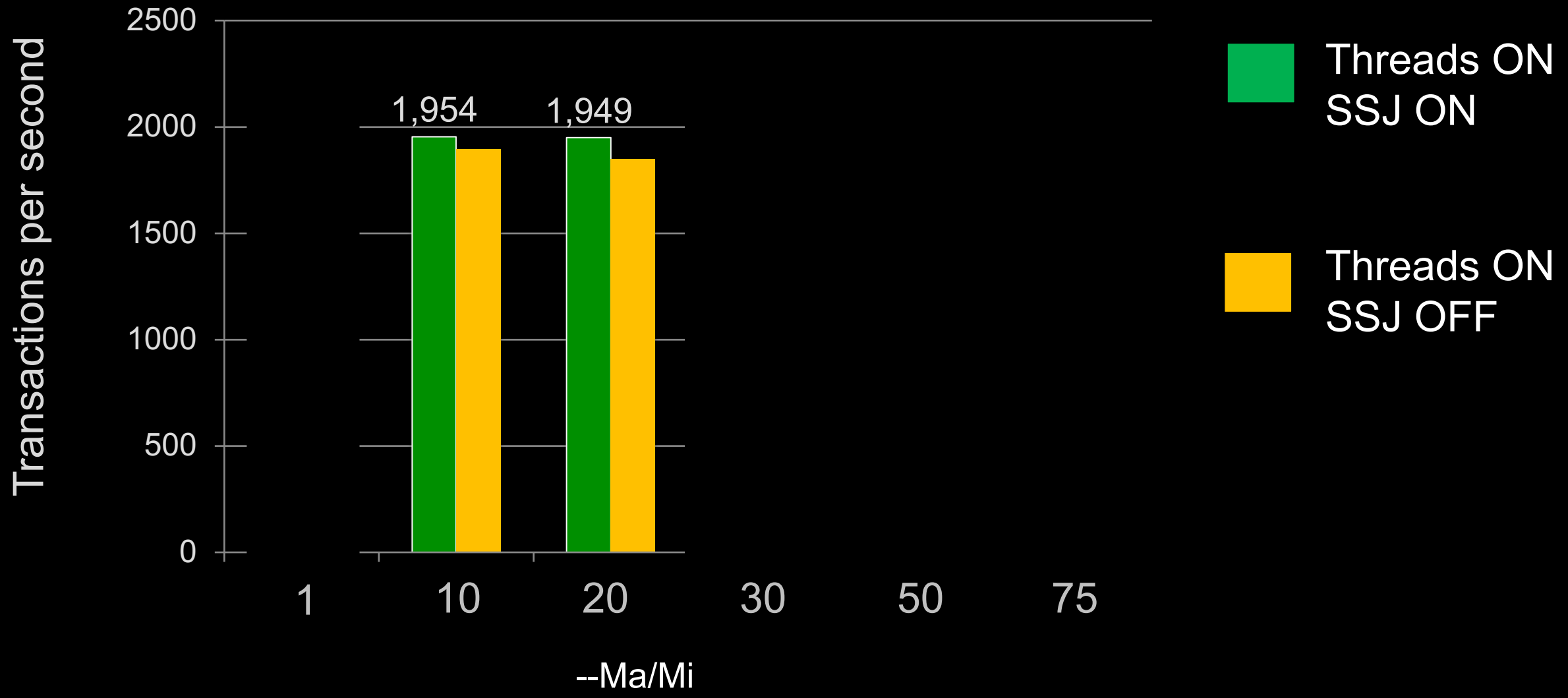


Join branch and teller
Vary –Ma, –Mi same as -Ma



Now, threads ON,
SSJ OFF

Join branch and teller
Vary –Ma, –Mi same as -Ma

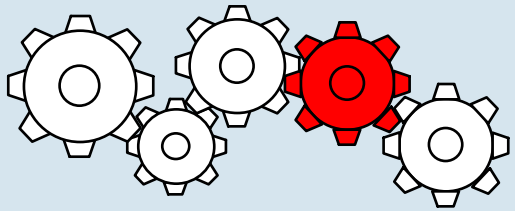


What have we learned now ?

- For small update transactions, multithreaded server is not as fast as unthreaded servers
  - Threaded throughput flattens at –Ma 20 at about 1200 tps
  - Multi server throughput flattens at –Ma 20 at about 1600 tps
  - This could be because client machine and network became overloaded
- For mix of update and readers with joins, multithreaded server with server-side joins is faster
- Slight drop with multithreaded server ON and SSJ OFF

- Testing with more types of transactions is needed.

Questions



research from the parmington foundation