PERCONA

MySQL 5.5
Guide to InnoDB Status

Bill Karwin, Percona Inc.

Me

- Software developer
- C, Java, Perl, PHP, Ruby
- SQL maven
- MySQL Consultant at Percona
- Author of *SQL Antipatterns: Avoiding the Pitfalls of Database Programming*

Bill Karwin

MyISAM or InnoDB?

- MyISAM can store more compactly
- MyISAM supports FULLTEXT indexes
- MyISAM uses primary keys and secondary keys in the same way

BUT...

- MyISAM does not support transactions
- MyISAM allows updates to write non-atomically
- MyISAM is susceptible to corruption on crashes
- MyISAM relies on filesystem cache

InnoDB Plugin 1.0

- Enable in /etc/my.cnf:

```
[mysqld]
ignore-builtin-innodb
plugin-load=innodb=ha_innodb_plugin.so
```

- Verify InnoDB plugin is enabled:

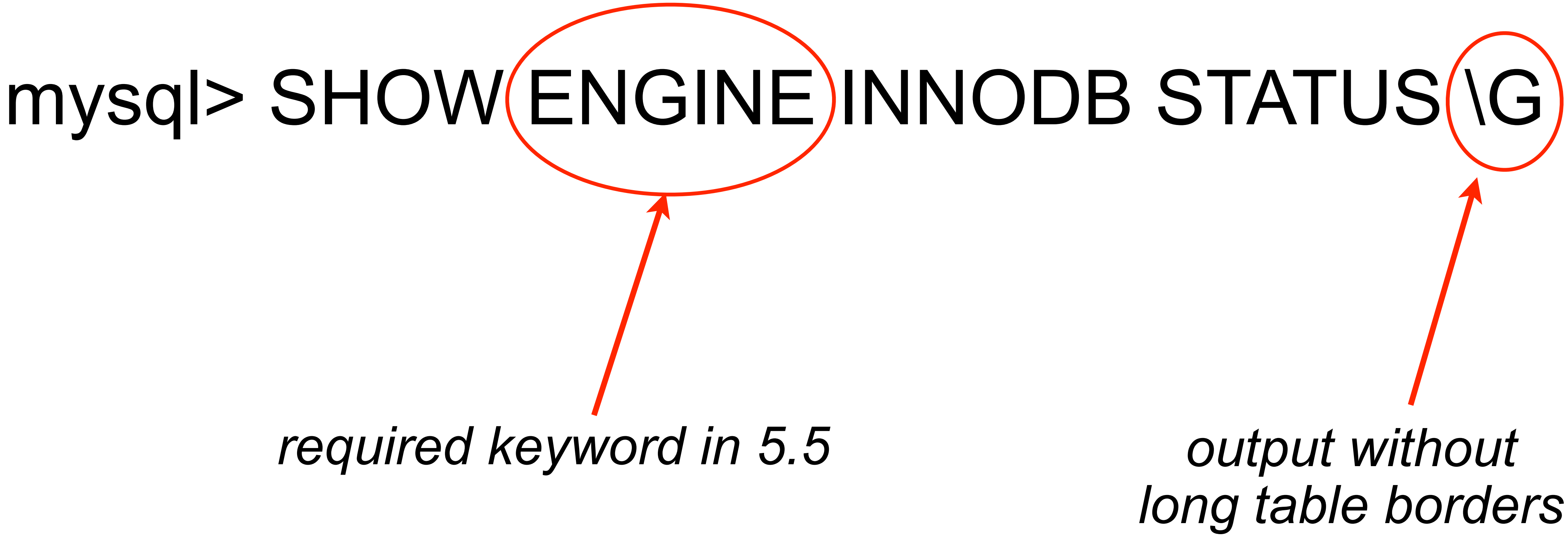
```
mysql> SHOW PLUGINS;

+------------+----------+----------------+---------------------+---------+
| Name       | Status   | Type           | Library             | License |
+------------+----------+----------------+---------------------+---------+
| InnoDB     | ACTIVE   | STORAGE ENGINE | ha_innodb_plugin.so | GPL     |
+------------+----------+----------------+---------------------+---------+
```

InnoDB Plugin 1.1

- Not necessary to enable InnoDB Plugin 1.1
- Default storage engine in MySQL 5.5

How to Show InnoDB Status



```
mysql> SHOW ENGINE INNODB STATUS \G
```

required keyword in 5.5

output without long table borders

=====================================

INNODB MONITOR OUTPUT

=====================================

```
=====================================
111001 19:29:44 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 23 seconds
```

if this is less than 20-30 seconds,
statistics may be inaccurate.
run this command again.

InnoDB Monitor Sections

- Background Thread
- Semaphores
- Latest Foreign Key Error
- Latest Detect Deadlock
- File I/O
- Insert Buffer and Adaptive Hash Index
- Log
- Buffer Pool and Memory
- Row Operations
- Transactions

BACKGROUND THREAD

```
srv_master_thread loops: 11938931 1_second, 11935492 sleeps,
  1193884 10_second, 365 background, 365 flush
srv_master_thread log flush and writes: 12087852
```

statistics about InnoDB main thread

SEMAPHORES

```
OS WAIT ARRAY INFO: reservation count 43659756, signal count 371748016


--Thread 1276582208 has waited at log/log0log.c line 1393 for 0.0000 seconds
   the semaphore:
Mutex at 0x2ab19a048de8 '&log_sys->mutex', lock var 1
waiters flag 1
```

contention on InnoDB log file.
problems in I/O?

```
Mutex spin waits 919370636, rounds 1883832361, OS waits 21271702
RW-shared spins 104920102, OS waits 11414698; RW-excl spins 139844907, OS
   waits 9956101
Spin rounds per wait: 2.05 mutex, 4.42 RW-shared, 6.16 RW-excl
```

statistics about mutexes.
high OS waits indicates lots of contention

LATEST FOREIGN KEY ERROR

```
030709 13:00:59 Transaction:
TRANSACTION 0 290328284, ACTIVE 0 sec, process no 3195, OS thread id 34831
inserting
15 lock struct(s), heap size 2496, undo log entries 9
MySQL thread id 25, query id 4668733 localhost heikki update
insert into ibtest11a (D, B, C) values (5, 'khDk' ,'khDk')
Foreign key constraint fails for table test/ibtest11a:
  CONSTRAINT `0_219242` FOREIGN KEY (`A`, `D`) REFERENCES `ibtest11b` (`A`,`D`) ON
   DELETE CASCADE ON UPDATE CASCADE
Trying to add in child table, in index PRIMARY tuple:
 0: len 4; hex 80000101; asc ....;; 1: len 4; hex 80000005; asc ....;; 2:
 len 4; hex 6b68446b; asc khDk;; 3: len 6; hex 0000114e0edc; asc ...N..;; 4:
 len 7; hex 00000000c3e0a7; asc .......;; 5: len 4; hex 6b68446b; asc khDk;;
But in parent table test/ibtest11b, in index PRIMARY,
the closest match we can find is record:
RECORD: info bits 0 0: len 4; hex 8000015b; asc ...[;; 1: len 4; hex
80000005; asc ....;; 2: len 3; hex 6b6864; asc khd;; 3: len 6; hex
0000111ef3eb; asc ......;; 4: len 7; hex 800001001e0084; asc .......;; 5:
len 3; hex 6b6864; asc khd;;
```

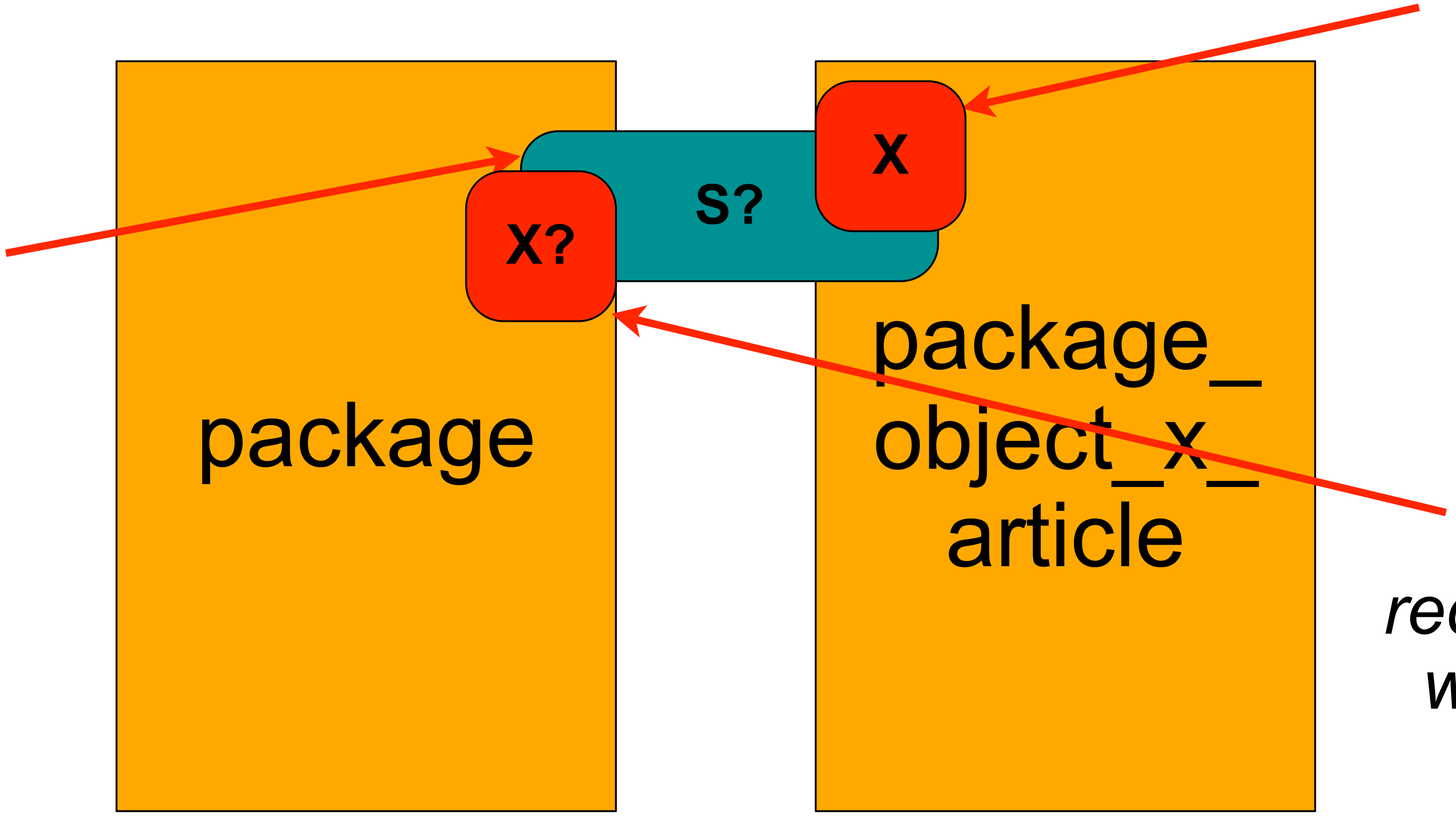
these values failed

because no match

LATEST DETECTED DEADLOCK

Transaction 1

Transaction 2



UPDATE gets X lock

SELECT JOIN requests S lock, waits for T2...

UPDATE requests X lock, waits for T1...

**** DEADLOCK! ****

LATEST DETECTED DEADLOCK (1 of 5)

```
110919 8:08:12
*** (1) TRANSACTION:
TRANSACTION A4BA03E, ACTIVE 10 sec, process no 17229, OS thread id
   1357232448 starting index read
mysql tables in use 3, locked 3
LOCK WAIT 288 lock struct(s), heap size 47544, 4025 row lock(s), undo log
   entries 45046
MySQL thread id 959109, query id 171532998 192.168.30.61 ads
   Copying to tmp table
INSERT INTO ad_article_conflicts (article_id, object_id, object_type,
   created_dts, proc_id, conflict_count, country)
SELECT distinct article_id, package.orig_package_id,
   'package', now(), 1316437206, 1,'US'
   FROM package_object_x_article, package
   WHERE package_object_x_article.object_id = package.package_id
   AND package_object_x_article.object_type= 'package'
   AND package.package_id IN (...)
```

waiting

needs S locks

LATEST DETECTED DEADLOCK (2 of 5)

```
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 102 page no 1784 n bits 1192 index
   `object_id_object_type` of table `ads`.`package_object_x_article` trx id
   A4BA03E lock mode S waiting
Record lock, heap no 13 PHYSICAL RECORD: n_fields 3; compact format; info
   bits 32
0: len 4; hex 0000060f; asc ;;
1: len 1; hex 01; asc ;;
2: len 4; hex 00000001; asc ;;
```

waiting for S lock on this table

already got the S lock on the other table `package`

LATEST DETECTED DEADLOCK (3 of 5)

```
*** (2) TRANSACTION:
TRANSACTION A4B9F50, ACTIVE 11 sec, process no 17229, OS thread id
   1274140992 starting index read
mysql tables in use 1, locked 1
984 lock struct(s), heap size 145848, 1115 row lock(s), undo log entries
   2221
MySQL thread id 959226, query id 171538155 192.168.30.150 ads Updating
UPDATE package SET live_dts = '2011-09-16 19:00:00', section_type =
   'exampleradio', display_type = 'old_style', expiration_dts = '2011-09-20
   00:00:00', title = 'Example Radio', package_type = 'with_landing',
   content_source = NULL WHERE package . . .
```

waiting

needs X lock on `package`

LATEST DETECTED DEADLOCK (4 of 5)

```
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 102 page no 1784 n bits 1192 index
   `object_id_object_type` of table `ads`.`package_object_x_article` trx id
   A4B9F50 lock_mode X locks rec but not gap
Record lock, heap no 13 PHYSICAL RECORD: n_fields 3; compact format; info
   bits 32
0: len 4; hex 0000060f; asc ;;
1: len 1; hex 01; asc ;;
2: len 4; hex 00000001; asc ;;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 305 page no 74 n bits 104 index `PRIMARY` of table
   `ads`.`package` trx id A4B9F50 lock_mode X locks rec but not gap waiting
Record lock, heap no 7 PHYSICAL RECORD: n_fields 50; compact format; info
   bits 0
```

holds X lock on one table

waiting for X lock
on other table `package`

LATEST DETECTED DEADLOCK (5 of 5)

```
*** WE ROLL BACK TRANSACTION (2)
```

why this transaction?

because it judges transaction 2 has modified fewer rows

FILE I/O (1 of 2)

```
I/O thread 0 state: waiting for i/o request (insert buffer thread)
I/O thread 1 state: waiting for i/o request (log thread)
I/O thread 2 state: waiting for i/o request (read thread)
I/O thread 3 state: waiting for i/o request (read thread)
I/O thread 4 state: waiting for i/o request (read thread)
I/O thread 5 state: waiting for i/o request (read thread)
I/O thread 6 state: waiting for i/o request (write thread)
I/O thread 7 state: waiting for i/o request (write thread)
I/O thread 8 state: waiting for i/o request (write thread)
I/O thread 9 state: waiting for i/o request (write thread)
```

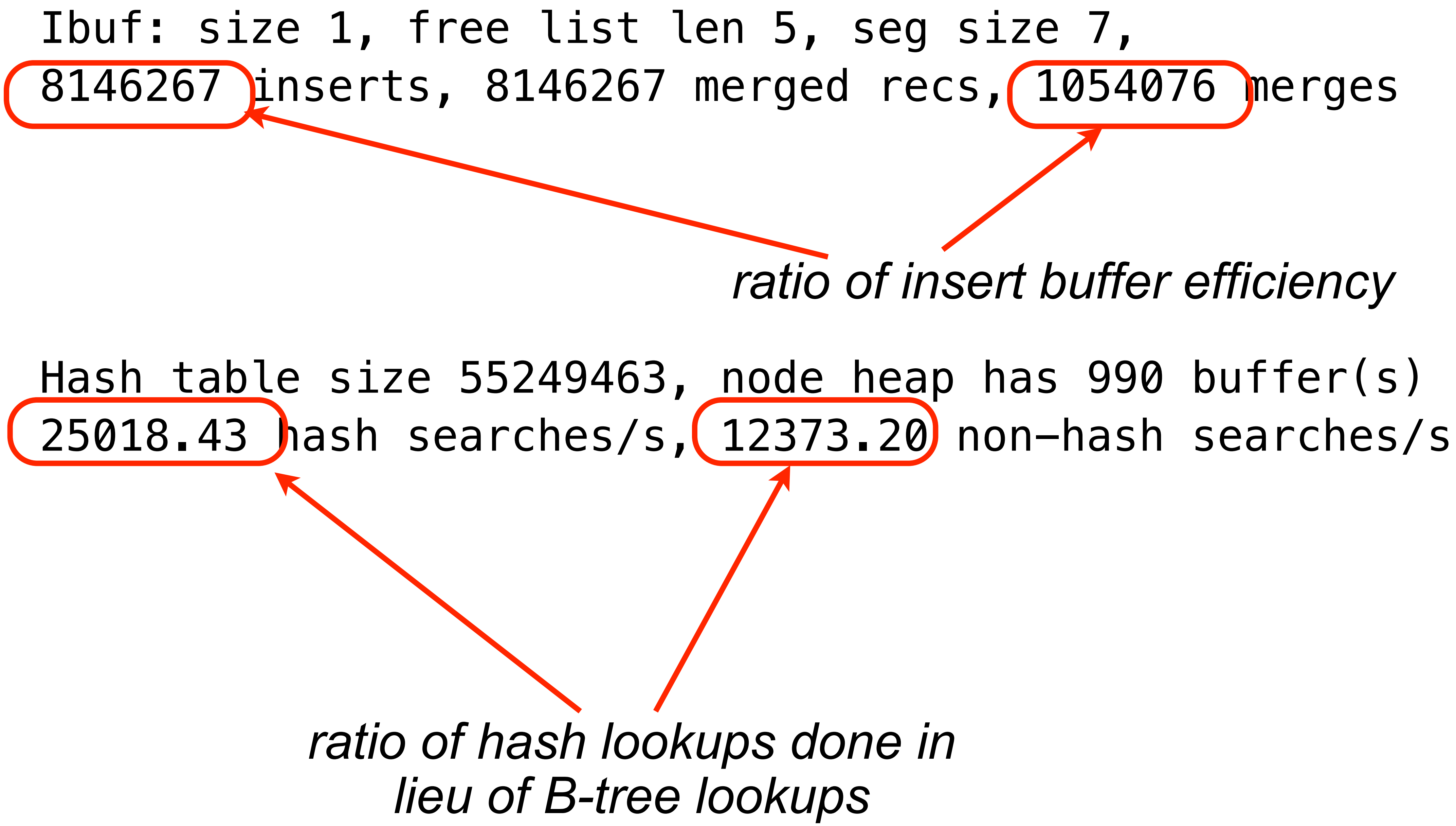
if these are busy, you can increase innodb_read_io_threads, innodb_write_io_threads

FILE I/O (2 of 2)

```
Pending normal aio reads: 0, aio writes: 0,
 ibuf aio reads: 0, log i/o's: 0, sync i/o's: 0
Pending flushes (fsync) log: 0; buffer pool: 0
907300503 OS file reads, 570173314 OS file writes, 440124224 OS fsyncs
2 pending preads, 0 pending pwrites
1182.86 reads/s, 16384 avg bytes/read, 37.52 writes/s, 31.87 fsyncs/s
```

these are all zero,
but high numbers indicate
you are I/O bound

INSERT BUFFER AND ADAPTIVE HASH INDEX



```
Ibuf: size 1, free list len 5, seg size 7,
8146267 inserts, 8146267 merged recs, 1054076 merges
```

ratio of insert buffer efficiency

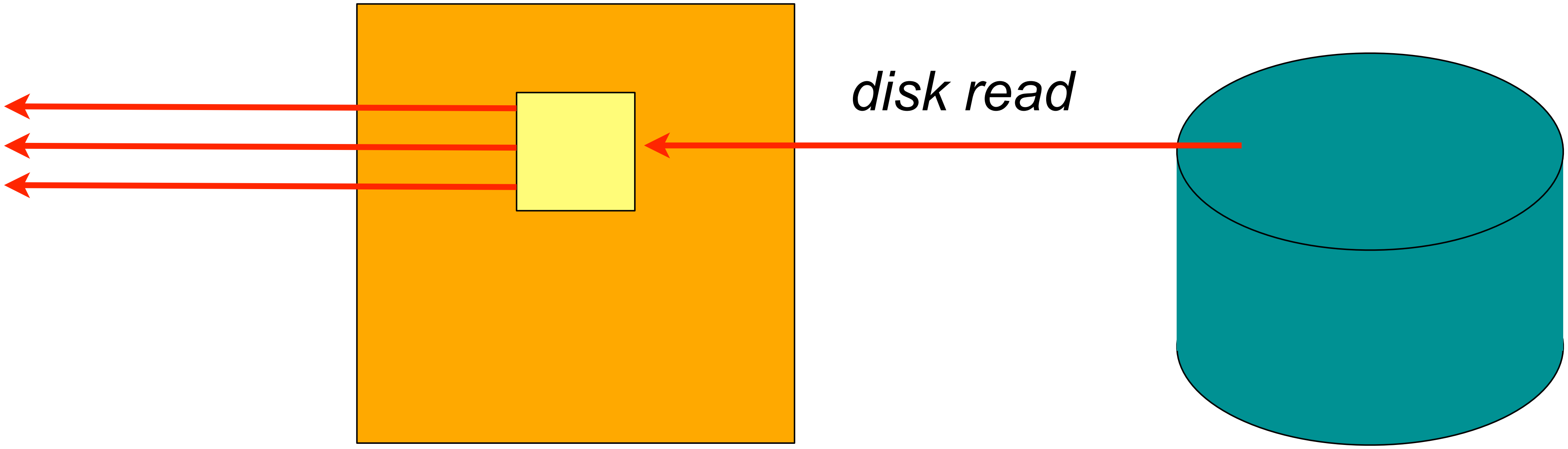
```
Hash table size 55249463, node heap has 990 buffer(s)
25018.43 hash searches/s, 12373.20 non-hash searches/s
```

ratio of hash lookups done in lieu of B-tree lookups

The Buffer Pool

The Buffer Pool

- All reads/writes use the buffer pool
- SQL reads from the buffer pool many times

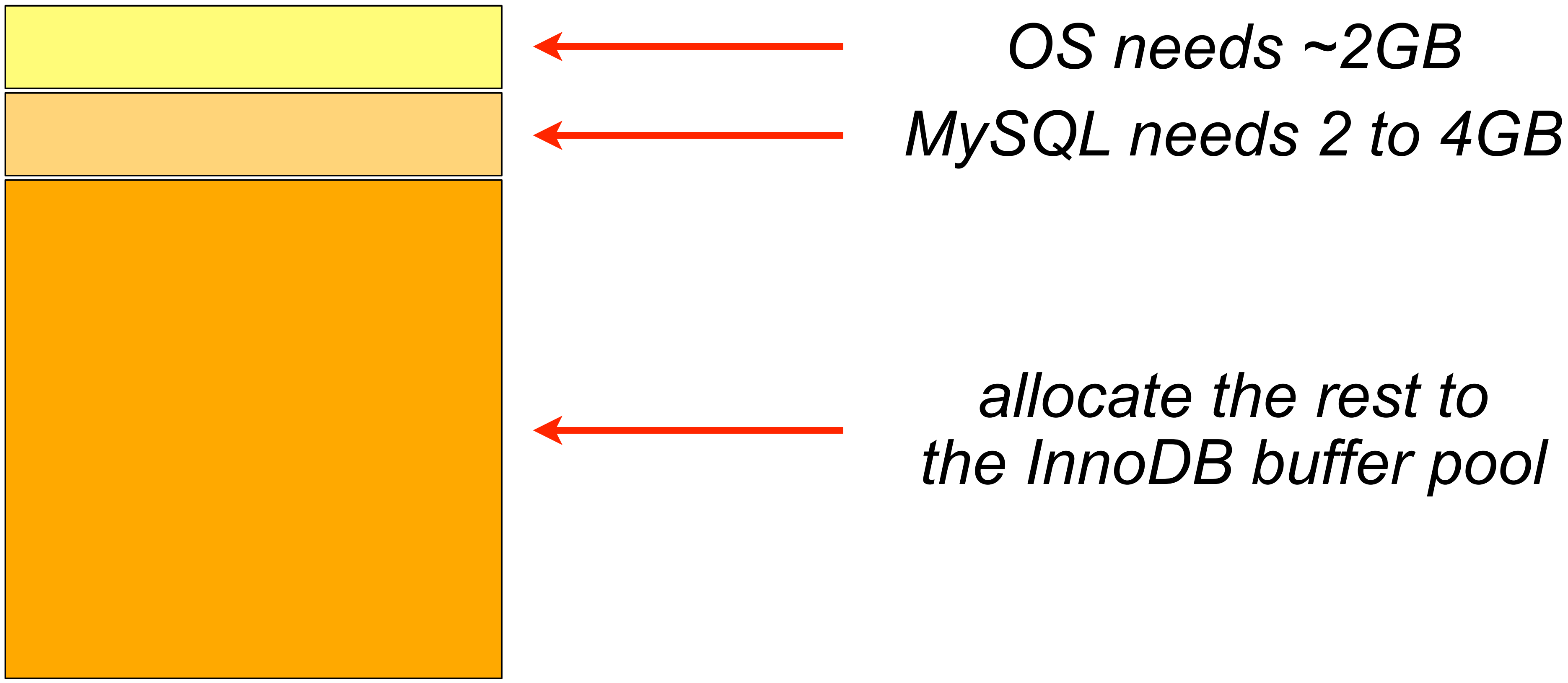


~100ns
memory access

~10,000,000ns
disk seek

The Buffer Pool

- The buffer pool is the best use of system memory.



OS needs ~2GB

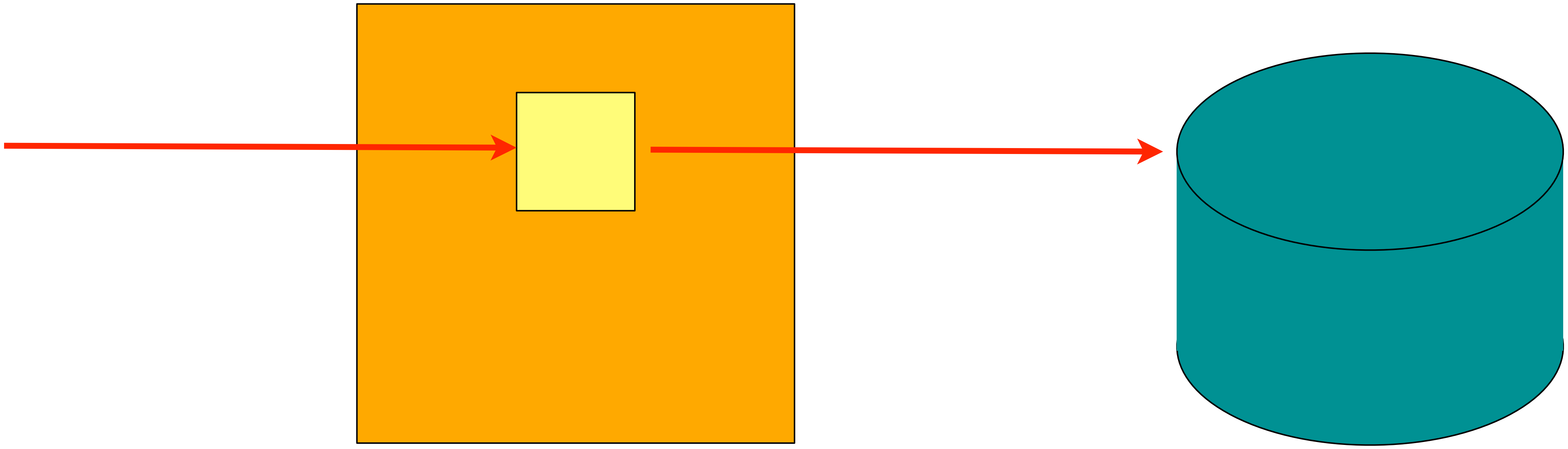
MySQL needs 2 to 4GB

allocate the rest to
the InnoDB buffer pool

http://www.mysqlperformanceblog.com/2007/11/03/choosing-innodb_buffer_pool_size/

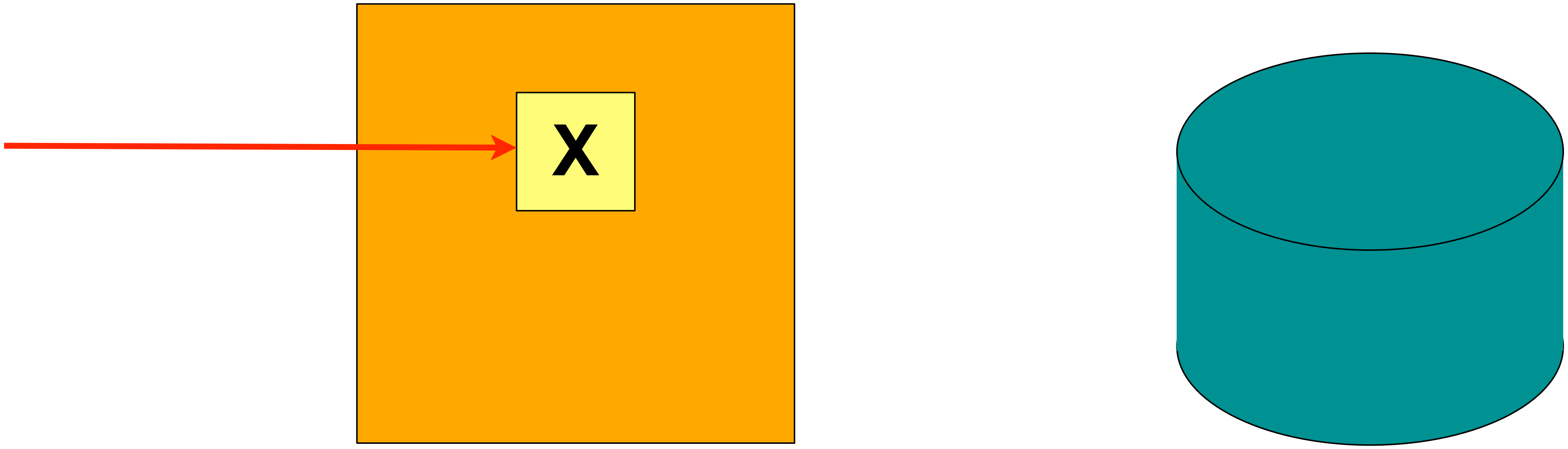
The Buffer Pool

- Random writes to disk are very slow.



The Buffer Pool

- Instead, keep “dirty” pages in buffer pool, to be written down later



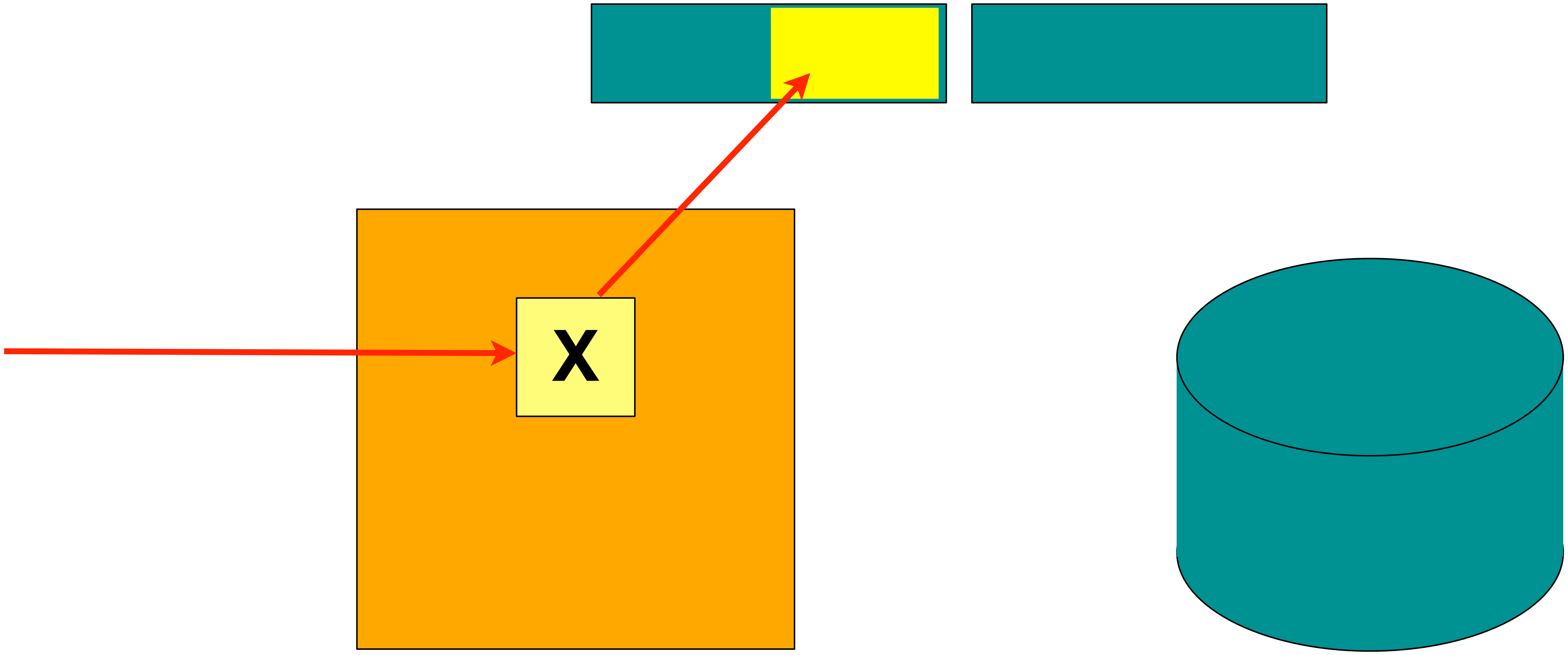
- But what happens to dirty pages in a crash?

The Log File

Simon Law

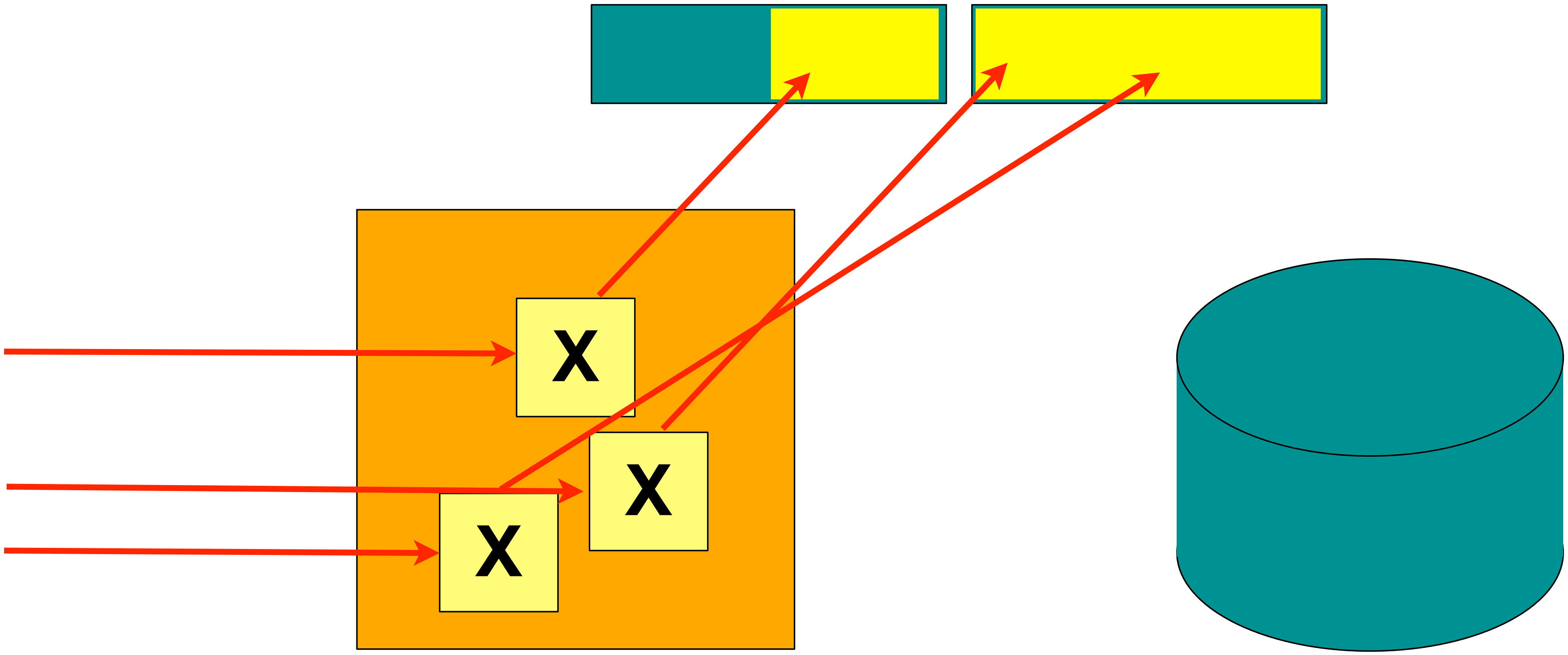
The Log File

- Sequential writes to log are much quicker.



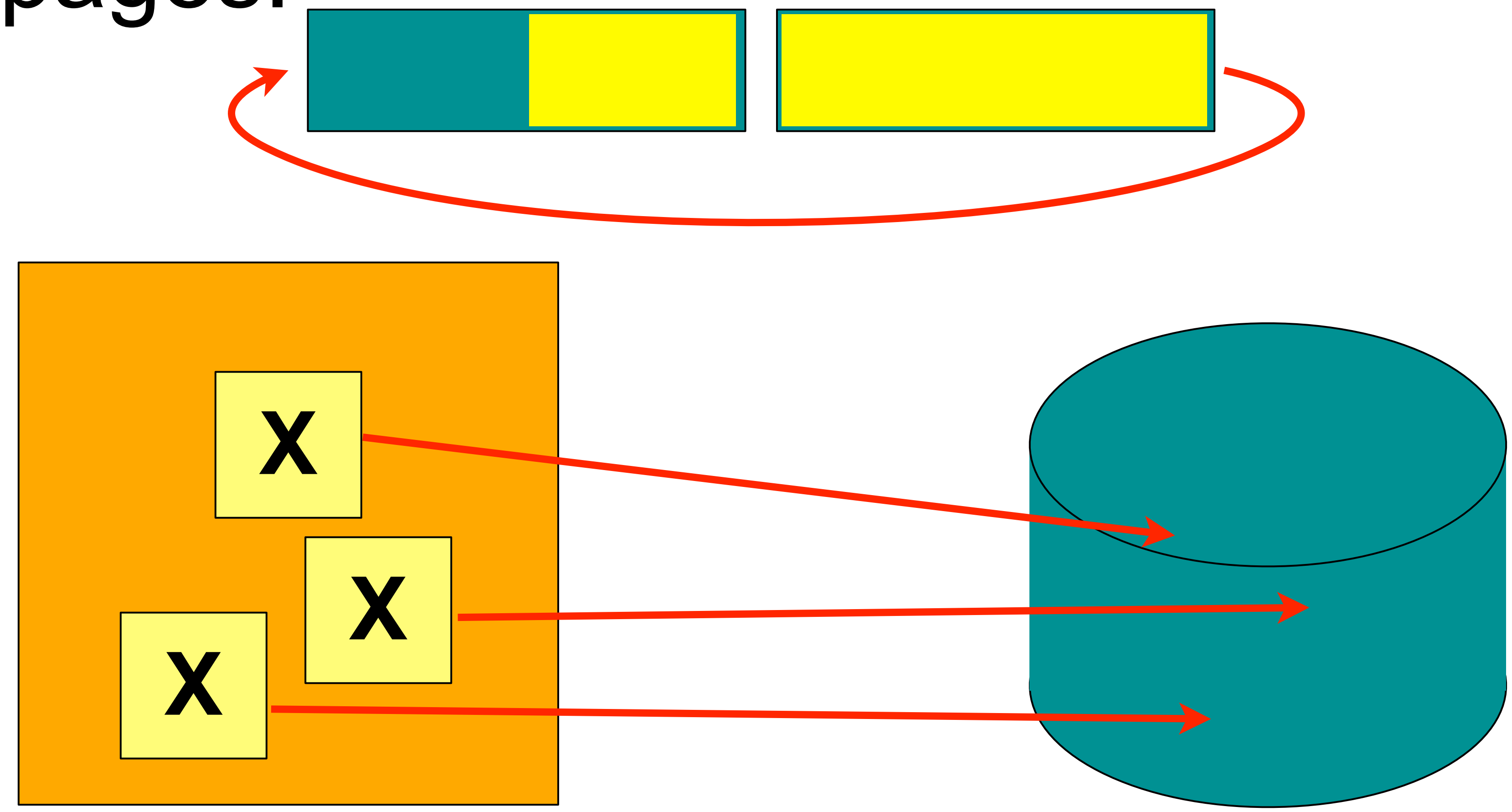
The Log File

- Many dirty pages can be recorded in the log file.



The Log File

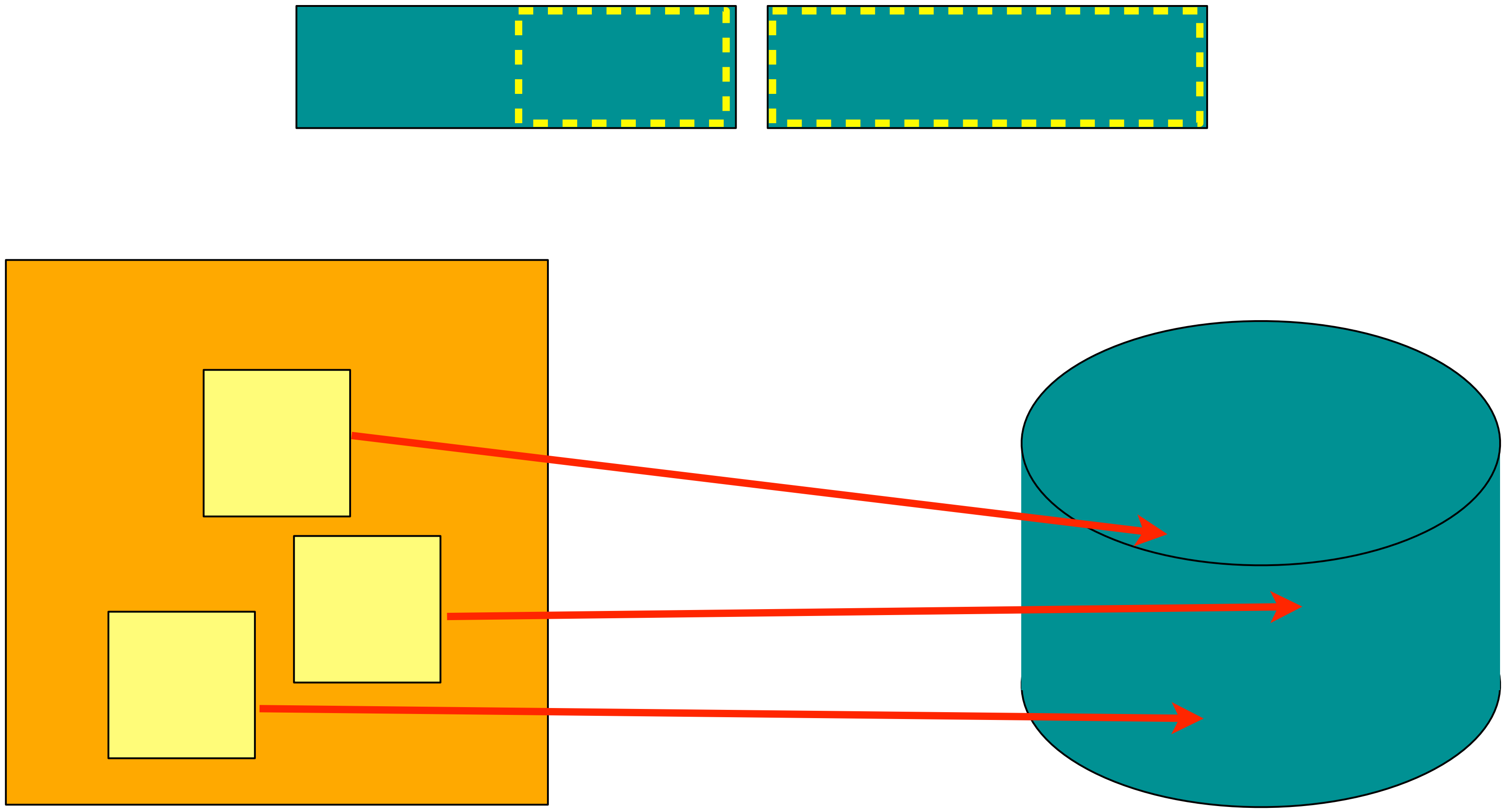
- As log file fills up, pressure rises to write down dirty pages.



- This is called a **log checkpoint.**

The Log File

- As pages are written down, they no longer dirty.



- Checkpointing frees up space in the log.

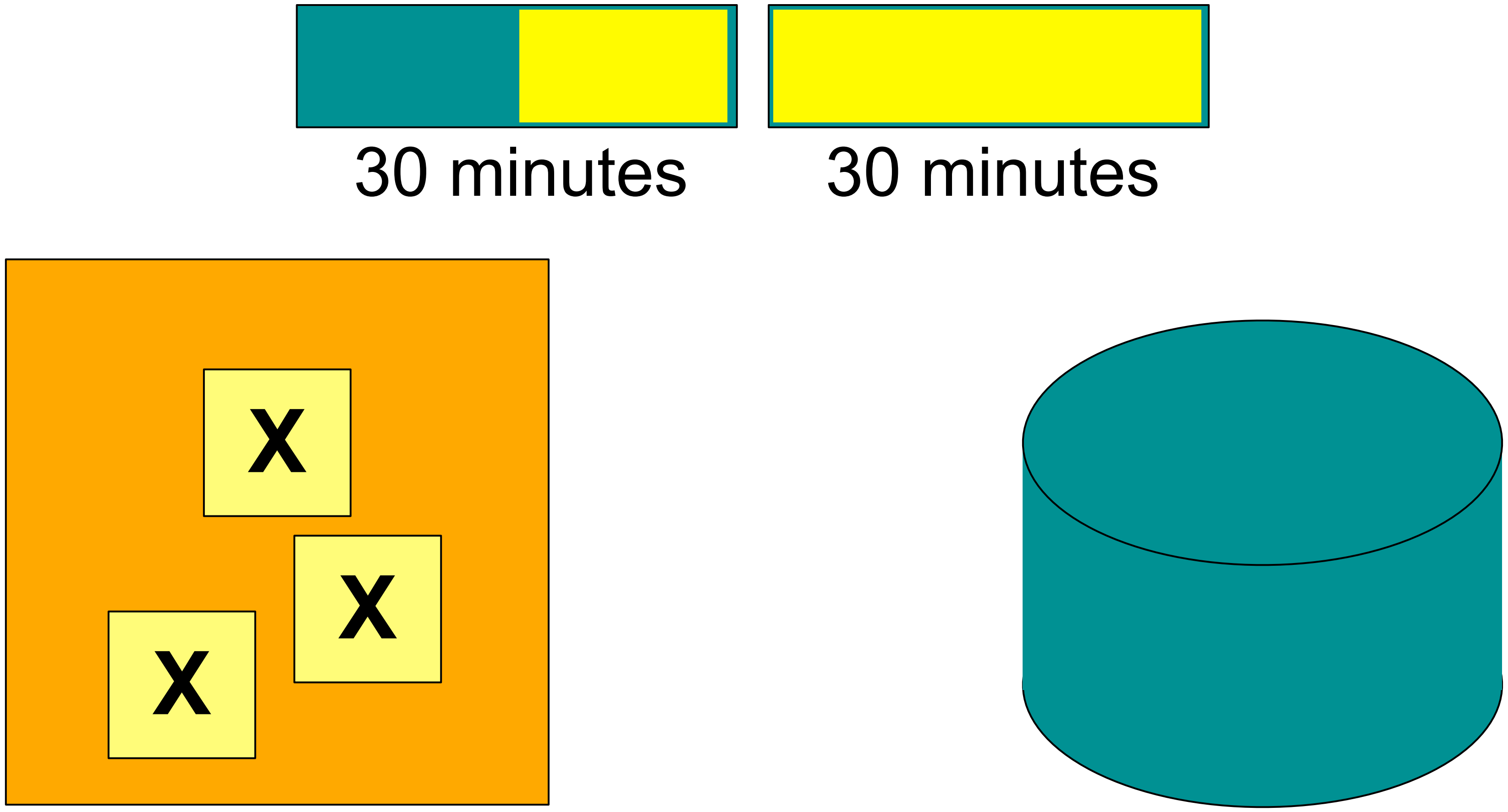
The Log File

- Pressure to checkpoint rises because:
  - The log files have a fixed size.
  - The log files contain “redo” information to replay changes for all dirty pages in case of crash.
  - If log files are getting full, InnoDB throttles incoming INSERT/UPDATE/DELETE.

http://www.mysqlperformanceblog.com/2011/04/04/innodb-flushing-theory-and-solutions/

The Log File

- 1 hour's worth of changes is a good log file size.



30 minutes

30 minutes

http://www.mysqlperformanceblog.com/2008/11/21/how-to-calculate-a-good-innodb-log-file-size/

LOG

```
Log sequence number 16557482601459
Log flushed up to   16557473340329
Last checkpoint at  16557036774612
1 pending log writes, 0 pending chkp writes
429764749 log i/o's done, 28.82 log i/o's/second
```

bytes written to log

...15 minutes later...

```
Log sequence number 16557494601459
```

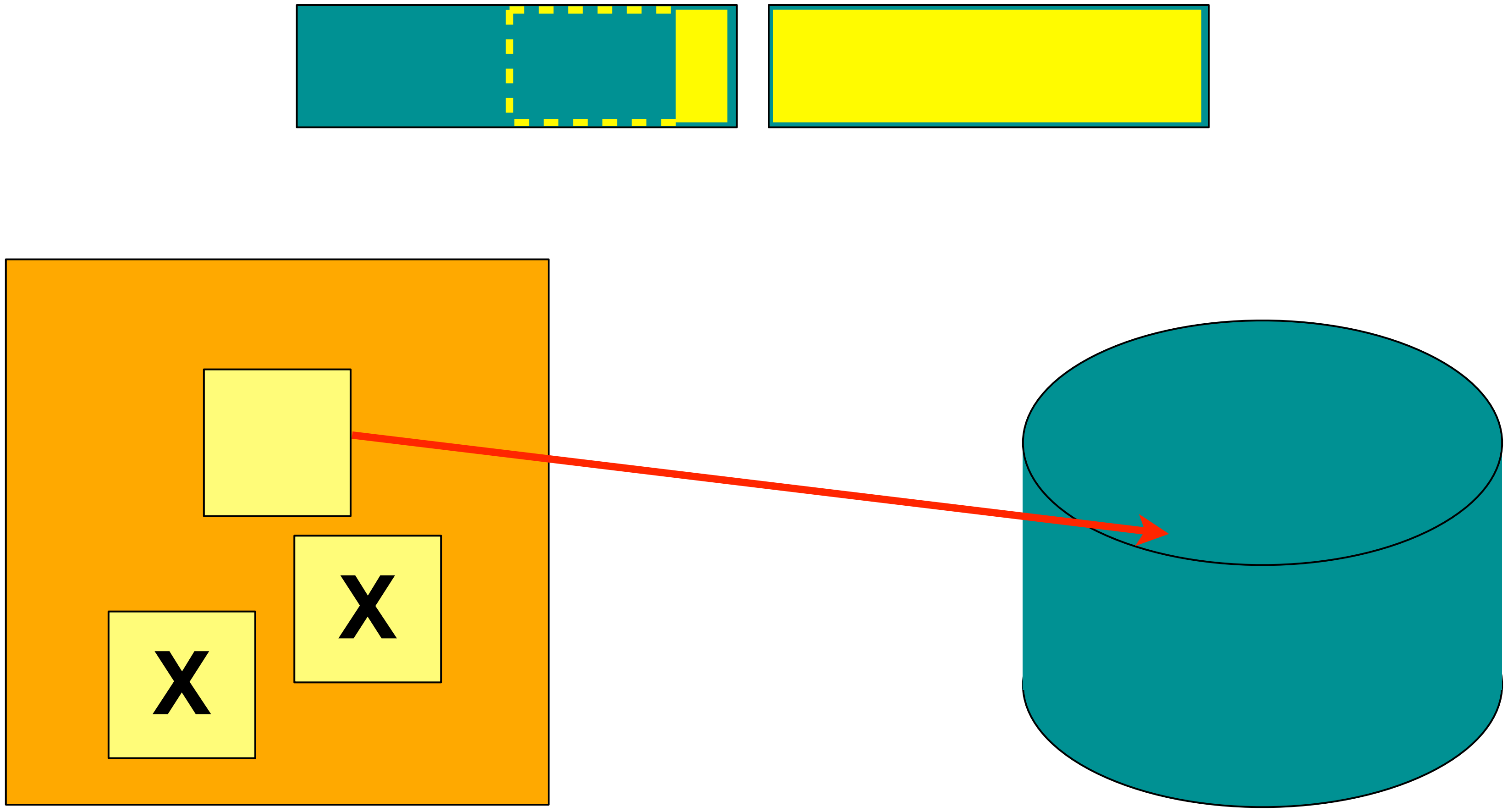
LSN is 12MB higher.
4 × 12M = 48MB/hour

- `innodb_log_file_size = 24M`

because there are two log files by default

The Log File

- Not all dirty pages are written at every checkpoint.



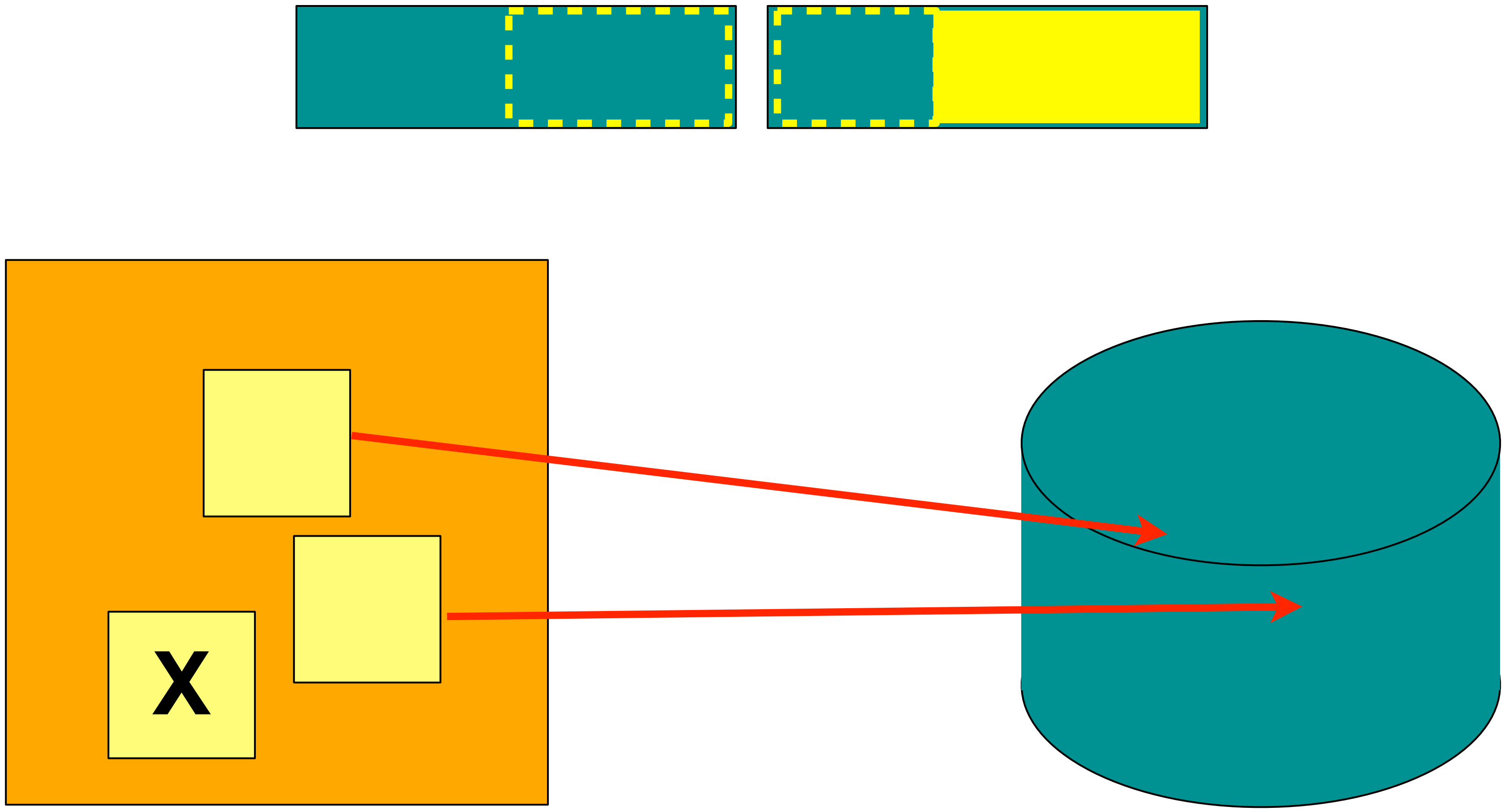
- How many pages are written? Depends...

IO Capacity

Shyaulis Andrjus

IO Capacity

- Faster disks can handle greater write load.



IO Capacity

- Tuning `innodb_io_capacity`
  - Default is 200, good for a single 7200rpm disk.
  - Match the IOPS your disk system can sustain.
  - Not unusual to see 800-1000 for RAID system.
  - Not unusual to see 2000-4000 for SSD system.

--- LOG (extra info from Percona Server) ---

```
Log sequence number 16557482601459
Log flushed up to   16557473340329
Last checkpoint at  16557036774612
Max checkpoint age    677822424
Checkpoint age target 656640474
Modified age          436863883
Checkpoint age        445826847
1 pending log writes, 0 pending chkp writes
429764749 log i/o's done, 28.82 log i/o's/second
```

BUFFER POOL AND MEMORY (1 of 3)

```
Total memory allocated 28626124800; in additional pool allocated 0
Internal hash tables (constant factor + variable factor)
    Adaptive hash index 458220080   (441995704 + 16224376)
    Page hash           27625592
    Dictionary cache    403718221   (110500528 + 293217693)
    File system         83536    (82672 + 864)
    Lock system         71993872    (71910152 + 83720)
    Recovery system     0 (0 + 0)
    Threads             409288   (406936 + 2352)
```

BUFFER POOL AND MEMORY (2 of 3)

```
Dictionary memory allocated 293217693
Buffer pool size        1703935
Buffer pool size, bytes 27917271040
Free buffers            0
Database pages          1702944
Old database pages      628605
Modified db pages       29305
Pending reads 1
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 1251203063, not young 0
1194.38 youngs/s, 0.00 non-youngs/s
```

```
----------------------
```

BUFFER POOL AND MEMORY (3 of 3)

```
----------------------
```

```
Pages read 1222502280, created 49476044, written 260566544
1184.30 reads/s, 61.52 creates/s, 116.82 writes/s
Buffer pool hit rate 991 / 1000, young-making rate 9 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.09/s
LRU len: 1702944, unzip_LRU len: 0
I/O sum[70075]:cur[288], unzip sum[0]:cur[0]
```

```
--------------
ROW OPERATIONS
--------------
```

```
3 queries inside InnoDB, 0 queries in queue
13 read views open inside InnoDB
Main thread process no. 7741, id 1211353408, state: sleeping
Number of rows inserted 2832396721, updated 5386467044, deleted
  427453434, read 166901591413
2008.17 inserts/s, 172.43 updates/s, 2.22 deletes/s, 4403.94
  reads/s
```

```
------------
TRANSACTIONS
------------
```

```
Trx id counter 1FE1D5398
Purge done for trx's n:o < 1FE1D4F1D undo n:o < 0
History list length 240
LIST OF TRANSACTIONS FOR EACH SESSION:

---TRANSACTION 1FE1D5334, not started, process no 7741,
  OS thread id 1268595008 flushing log
  MySQL thread id 116454235, query id 10469901493 10.80.1.110
  c_106
  COMMIT
```

keeping old row versions

trying to flush to log on COMMIT

END

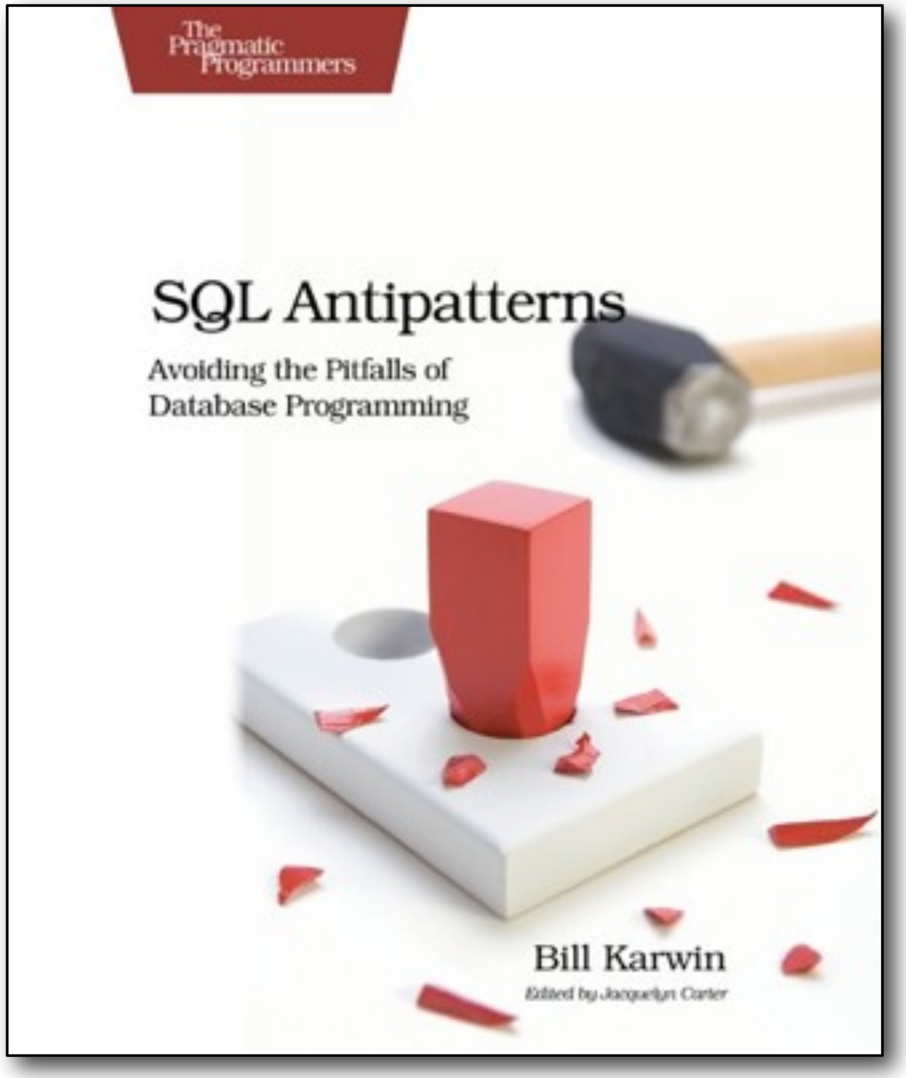
```
----------------------------
END OF INNODB MONITOR OUTPUT
============================
```

Best Tuning Parameters

- `innodb_buffer_pool_size`
  - As much as you can spare after OS and MySQL.
- `innodb_log_file_size`
  - At least enough for 60 minutes of log writes.
- `innodb_io_capacity`
  - Based on your disk IOPS.

SQL Antipatterns

20% discount code: **ZendConSQL**
(until 10/28/2011)


The Pragmatic Programmers
SQL Antipatterns
Avoiding the Pitfalls of Database Programming
Bill Karwin
Edited by Jacquelyn Carter

http://www.pragprog.com/titles/bksqla/