



Making PBXT Faster

PERCONA PERFORMANCE CONFERENCE 2009

Vladimir Kolesnikov

PrimeBase Technologies GmbH

www.primebase.org





Contents

- **Introduction**
- **Performance Improvements**
- **Architecture Overview**
- **Performance Tune-up**



About PrimeBase XT

- **A pluggable storage engine for MySQL 5.1+**
- **Transactional, ACID compliant (v1.0+)**
- **Open source (GPL), community project**
- **Designed and built specifically for MySQL**
- **Developed by PrimeBase Technologies:**
<http://www.primebase.org>
- **Hosted by Sourceforge.net:**
<http://sourceforge.net/projects/pbxt>



Design Principles

- **MVCC, all versions are stored on disk**
- **Writes sequentially/write once (log-based)**
- **Never updates in place**
- **No undo, non-committed data is garbage (collected by background threads)**
- **File-per-table, no table spaces**



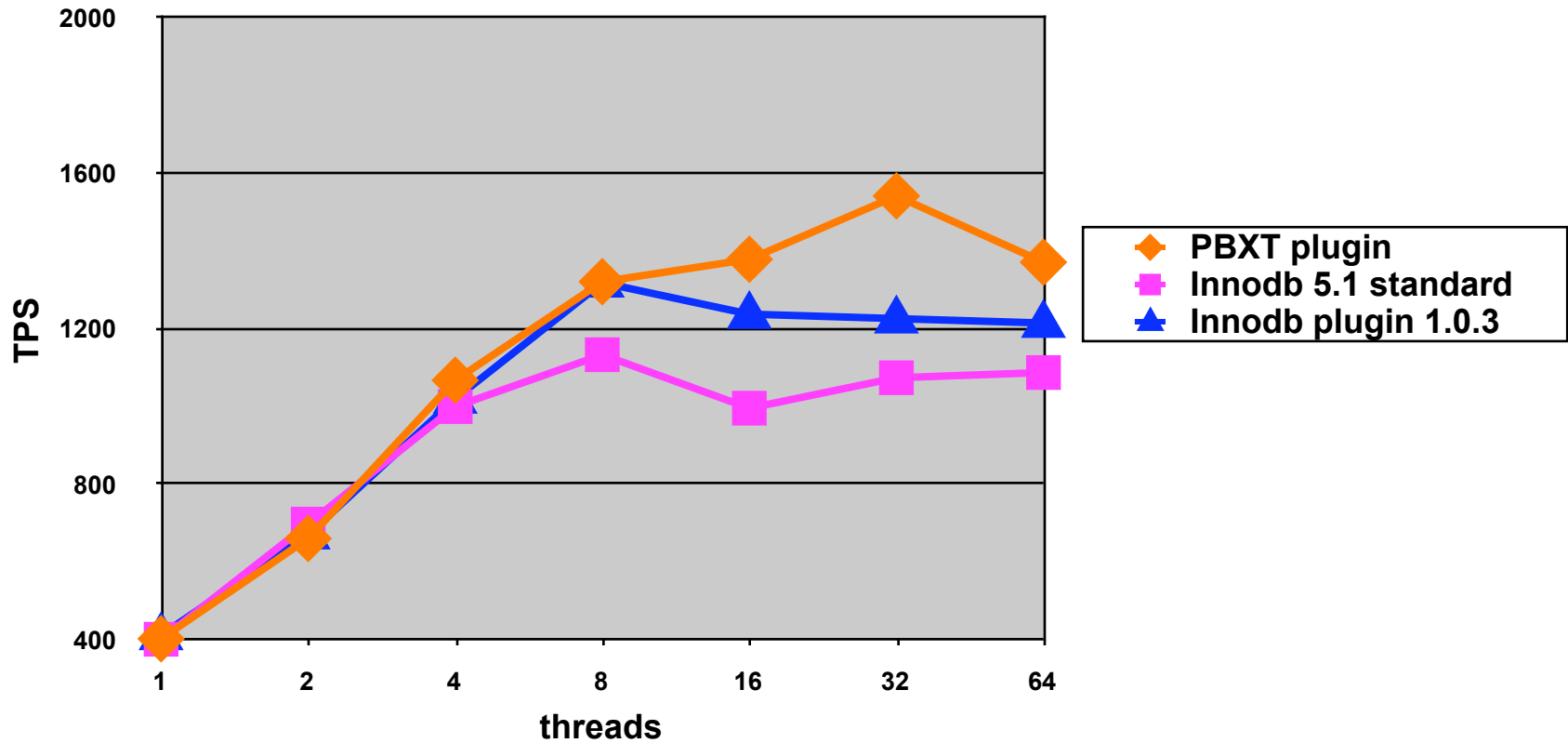
PBXT Performance

Improvements since v.1.0.07 RC

- **Reduced sleep times, added more spinning**
- **Minimized the number of memcpy()'s**
- **Implemented parallel index updates**
- **Added index scan cache page handles**
- **Improved selectivity estimation**



Sysbench r/w OLPT, 10m Rows, SSD





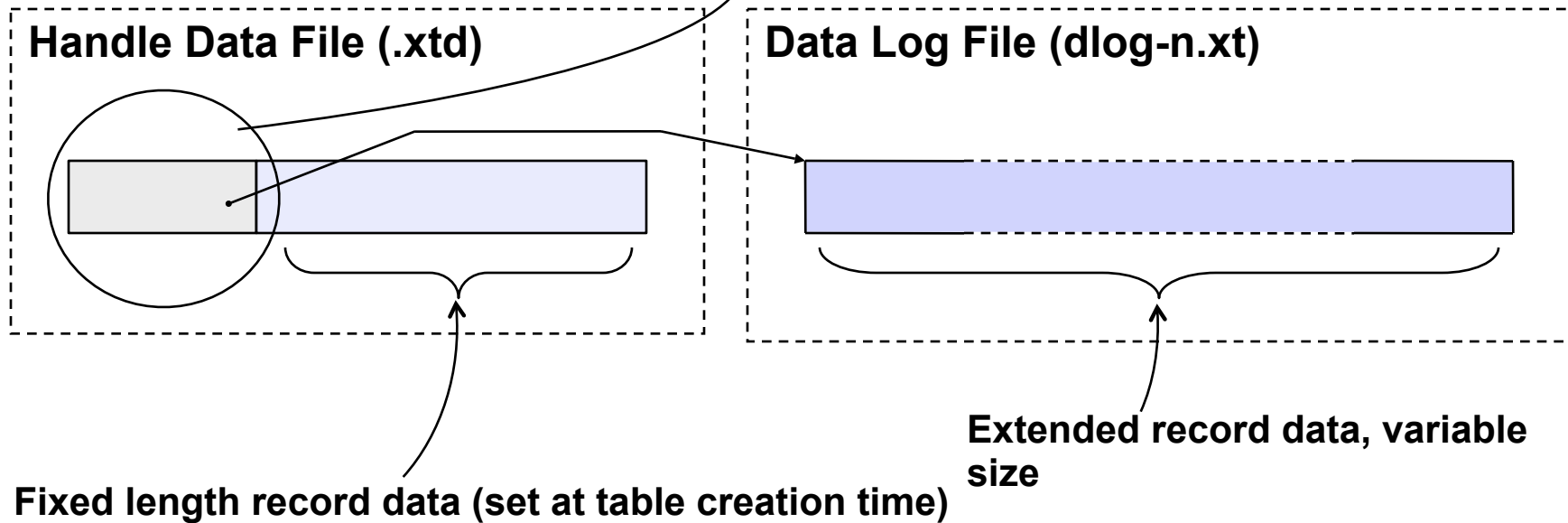
PBXT Architecture Overview



Record Structure

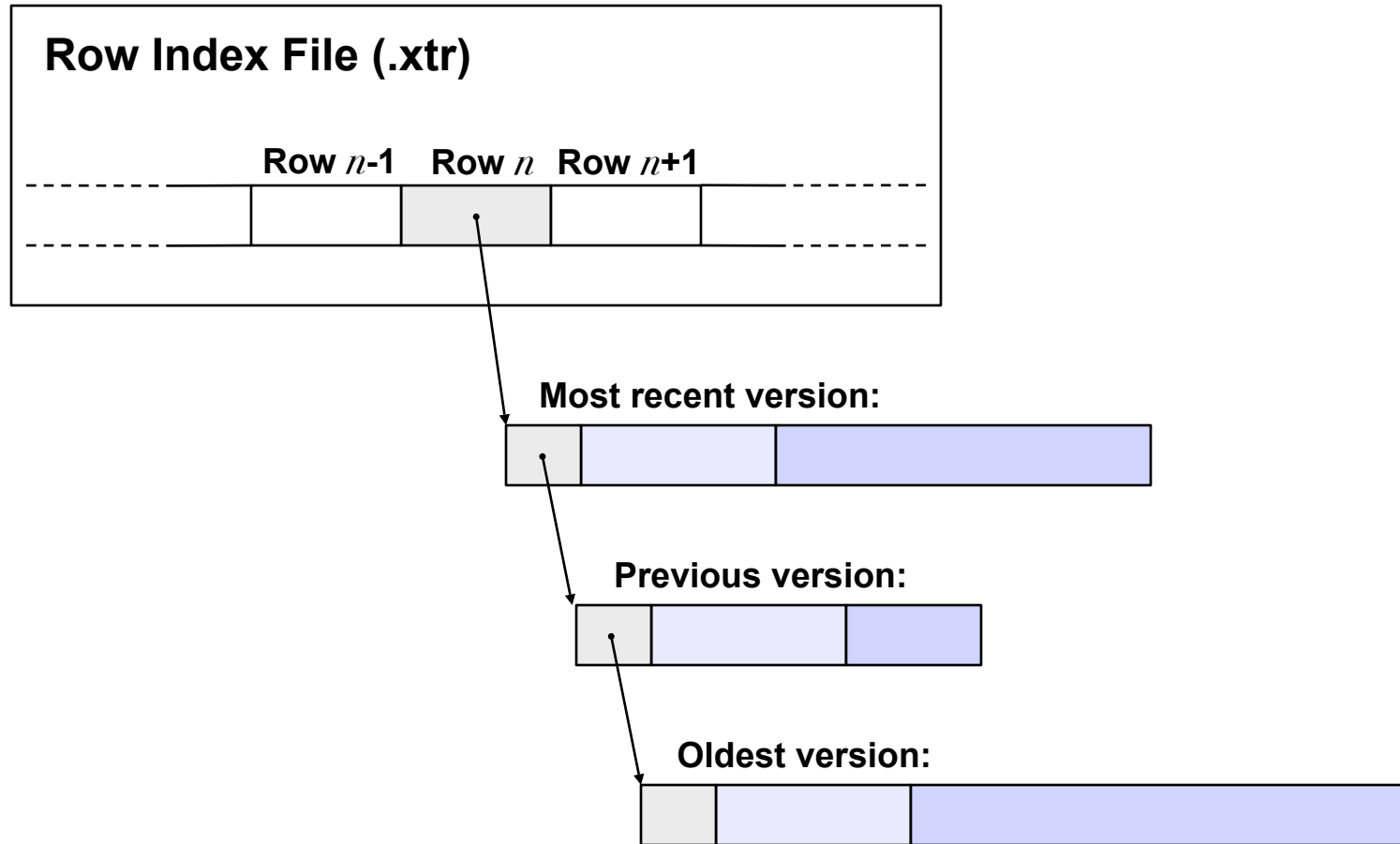
Control Data (14 - 26 bytes):

Status	Prev. version	Xaction ID	Row ID	Ext. Data Ref
---------------	----------------------	-------------------	---------------	----------------------





Row Structure



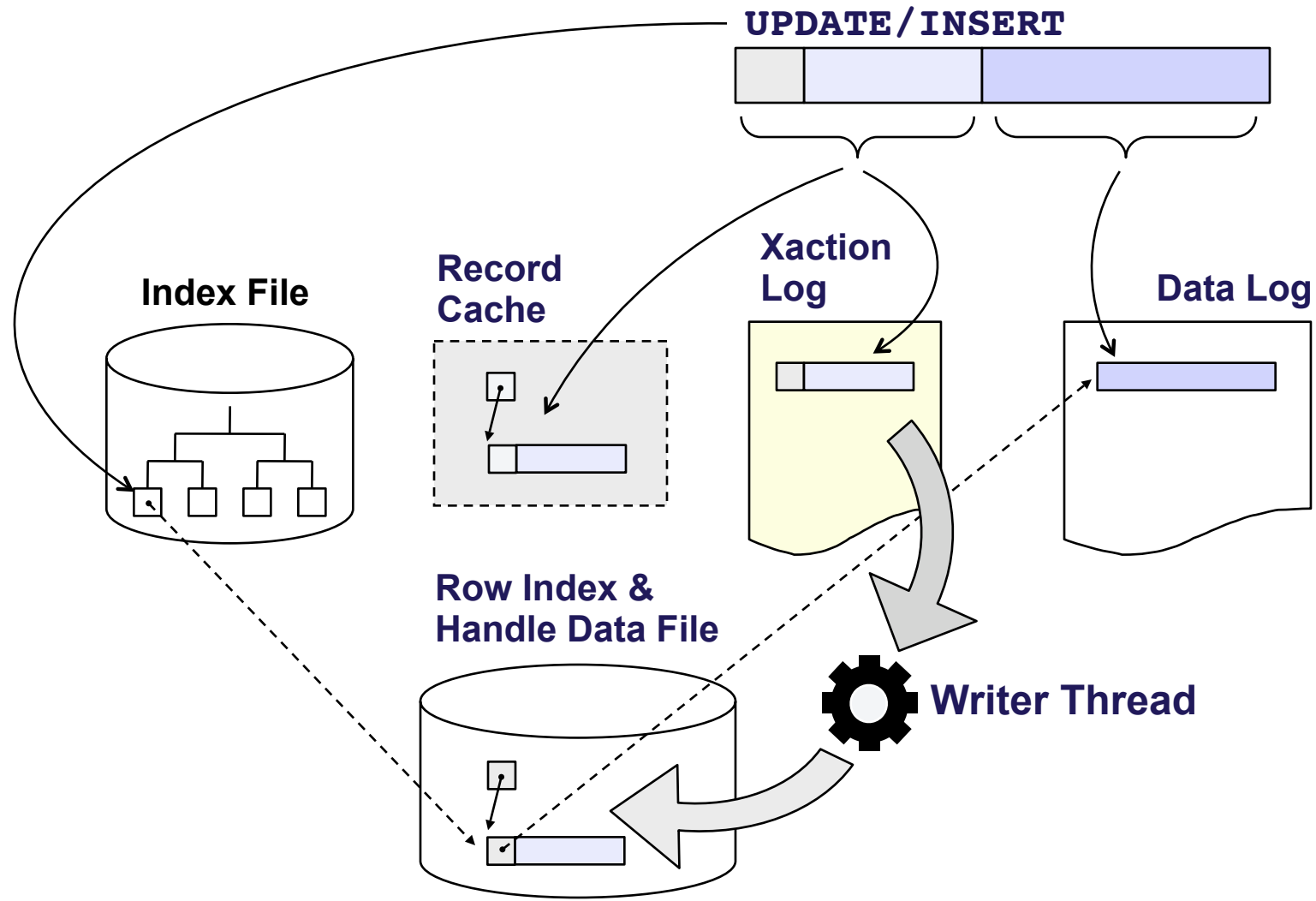


Background Threads

-  **Writer**
-  **Sweeper**
-  **Compactor**
-  **Checkpointner**

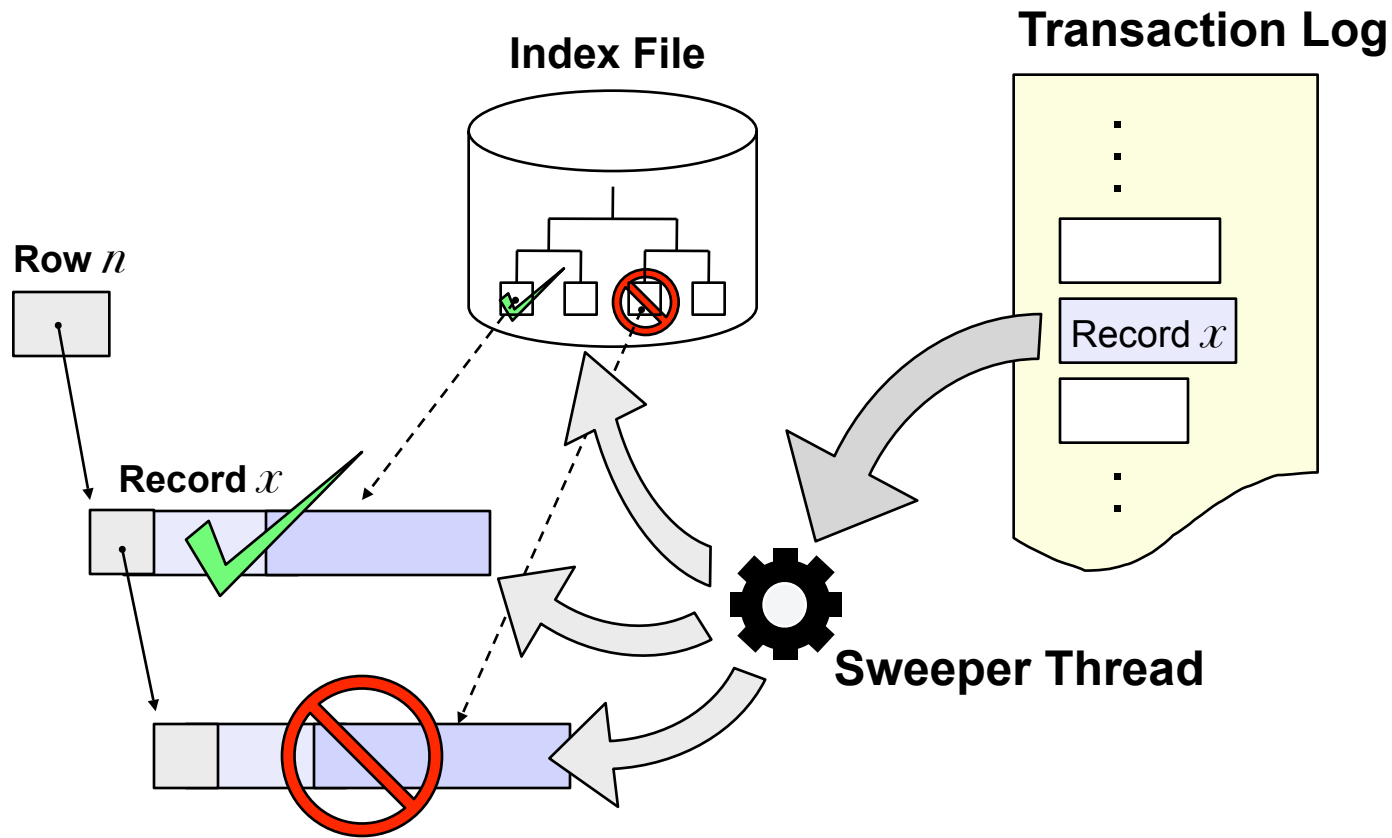


Writer Thread



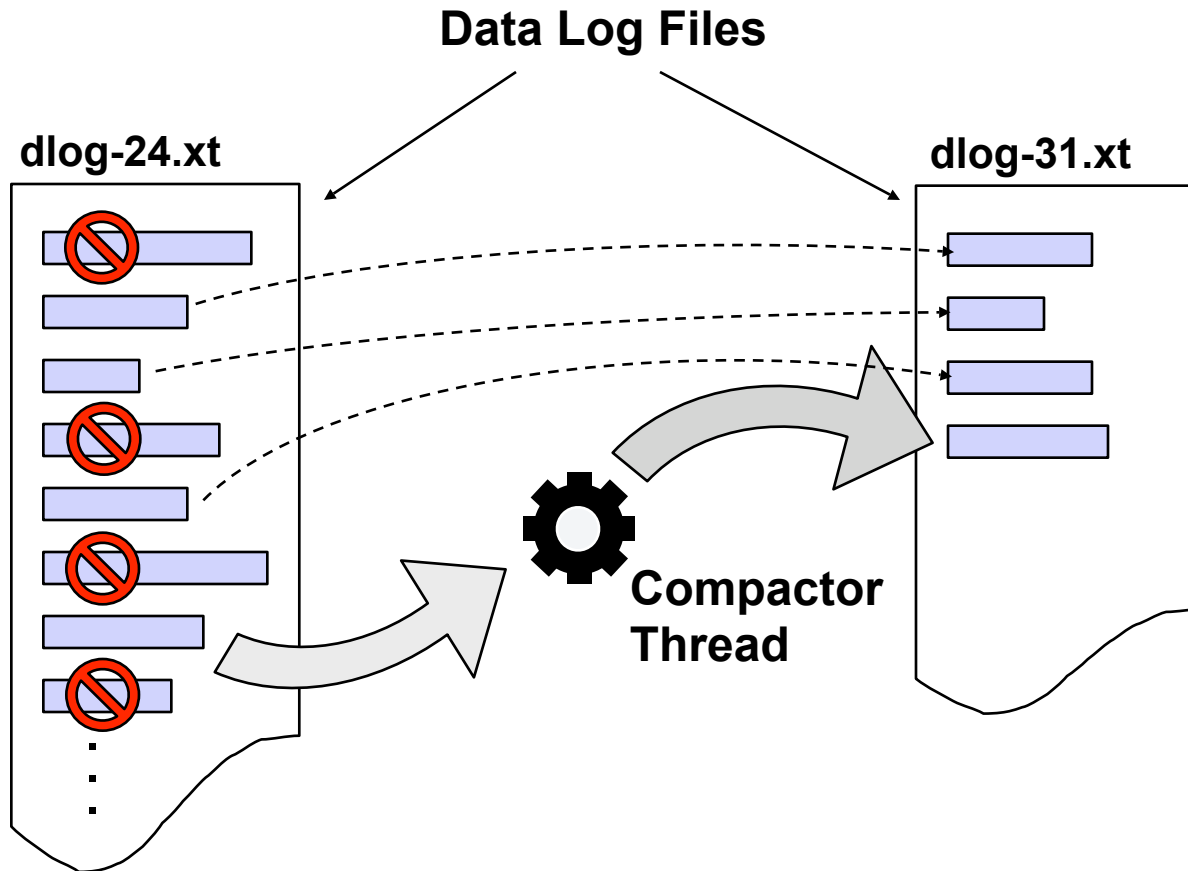


Sweeper Thread



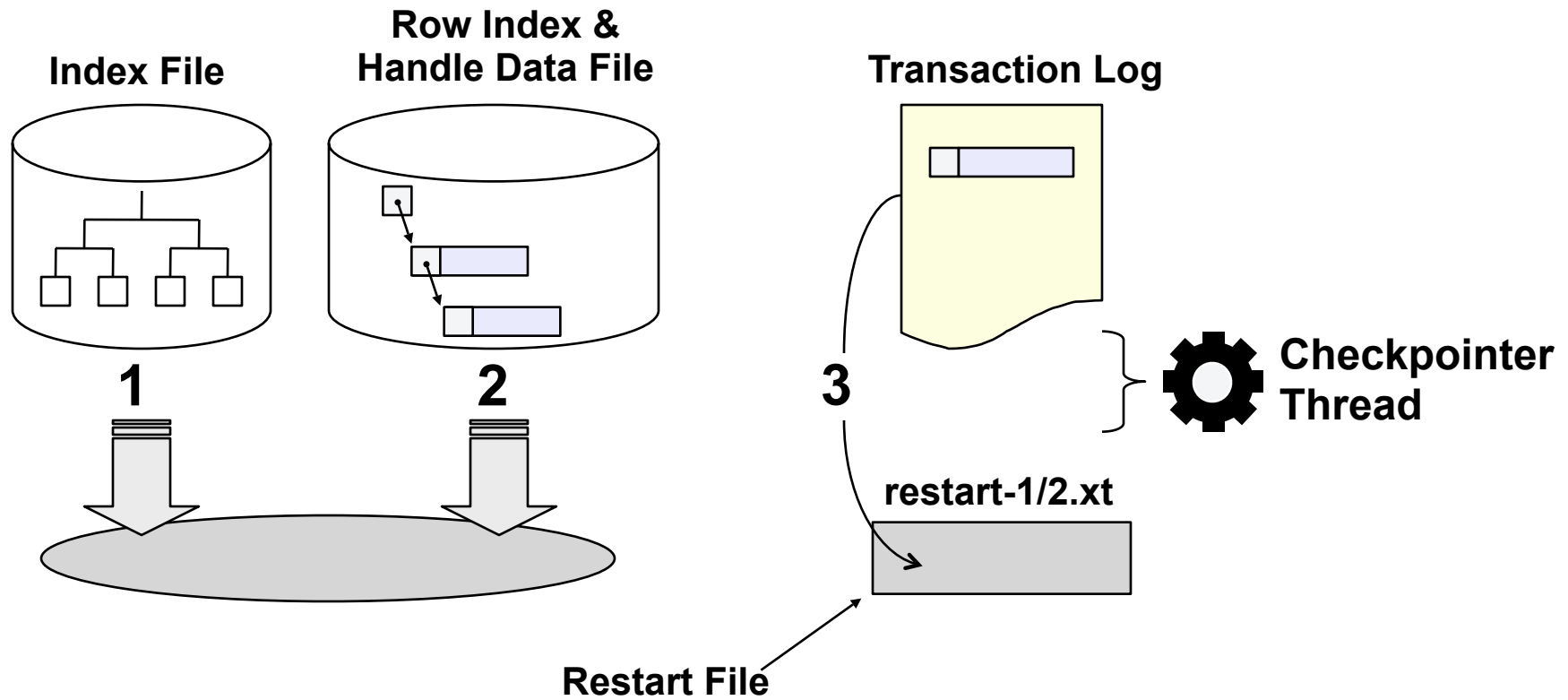


Compactor Thread





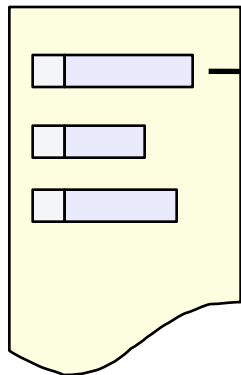
Checkpoint Thread



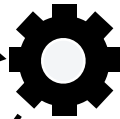


Recovery

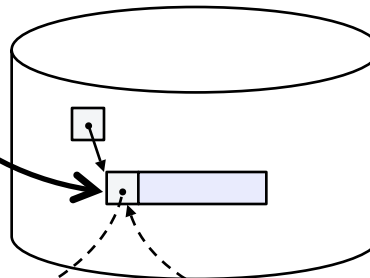
Transaction Log



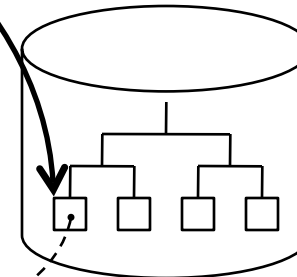
Recovery Process



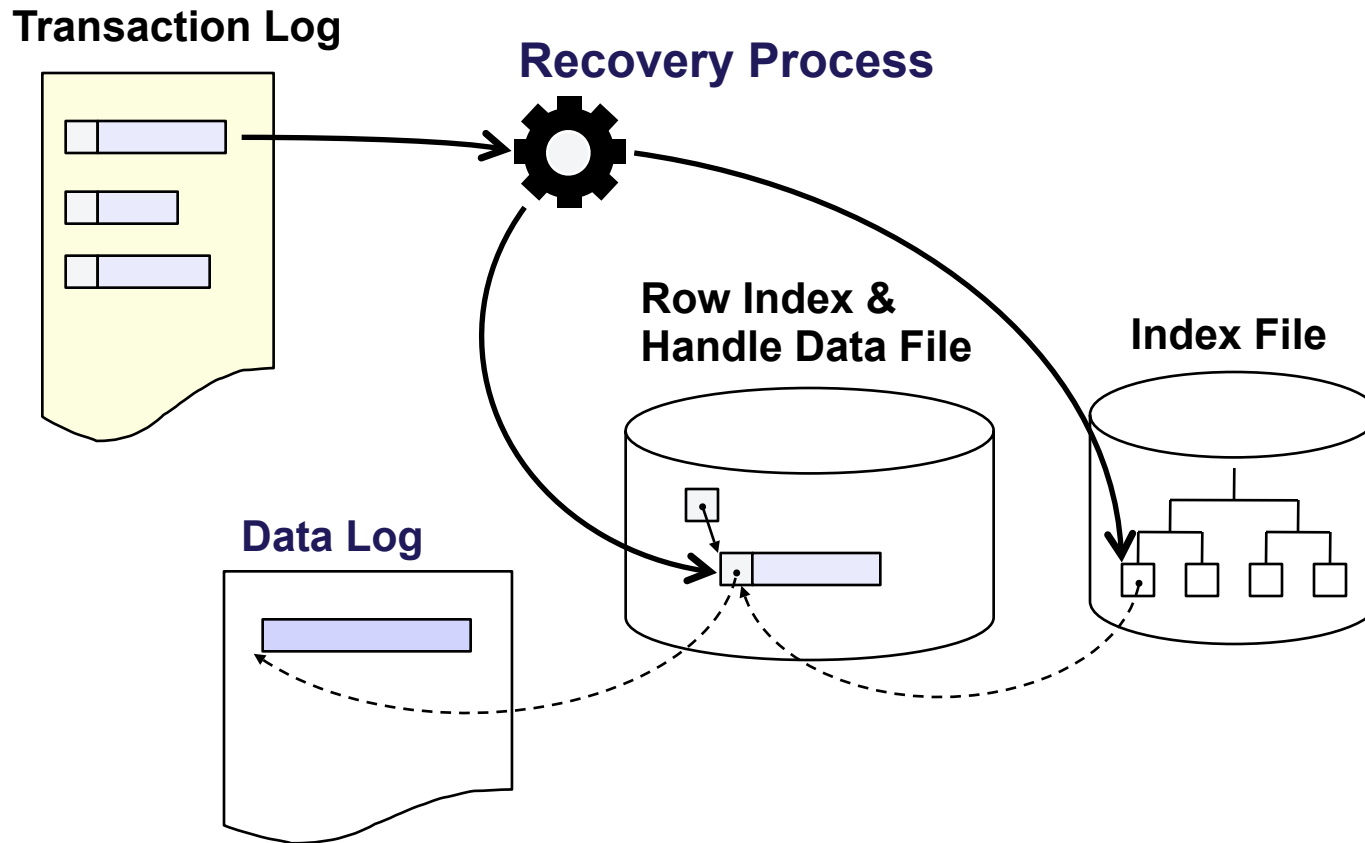
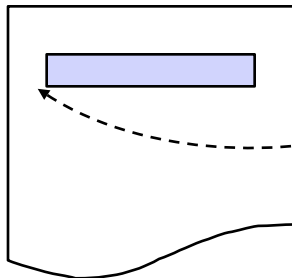
Row Index &
Handle Data File



Index File



Data Log





The PBXT_STATISTICS Table

6 groups (46 counters total)

- Transactions
- Record Files & Cache
- Index Files & Cache
- Transaction Log
- Data Log
- Background Threads



PBXT Counters - Transactions

- **Commit Count**
- **Rollback Count**
can be high if there are many deadlocks
- **Read Statements**
- **Write Statements**



PBXT Counters - Transactions

- **Select Row Count**
- **Insert Row Count**
- **Update Row Count**
- **Delete Row Count**
- **Wait for Xact Count**
incremented every time a transaction or a background thread has to wait for another one; high count means high number of concurrent row updates



PBXT Counters – Records/Indexes

- **Record/Index Bytes Read**
bytes read from files (filesystem cache, OS page cache, disk)
- **Record/Index Bytes Written**
bytes written to files (filesystem cache, OS page cache)
- **Record/Index File Flushes**
number of fsync/msync calls; can be affected by pbxt_data_file_grow_size
- **Record/Index Flush Time**
fsync/msync duration in ms



PBXT Counters – Records/Indexes

- **Record/Index Cache Hits**
- **Record/Index Cache Misses**
- **Record/Index Cache Frees**
16K cache pages; the freer removes LRU pages when cache use is > 95%
- **Record/Index Cache Usage**
- **Table/Index Scan Count**
number of times table scan was initiated (SELECT, UPDATE, DELETE, ALTER TABLE)



PBXT Perf. Counters – Indexes

- **Index Log Bytes In**
- **Index Log Bytes Out**
- **Index Log File Syncs**
- **Index Log Sync Time**



PBXT Counters – Transaction Log

- **Xact Log Bytes In**
x-log is read by writer, sweeper, recovery
- **Xact Log Bytes Out**
this is done in foreground!
- **Xact Log File Syncs**
- **Xact Log Sync Time**
- **Xact Log Cache Hits**



PBXT Counters – Transaction Log

- **Xact Log Cache Misses**
only the sweeper can cause x-log cache miss, which means it's far behind
- **Xact Log Cache Usage**
sooner or later this becomes full, it's ok



PBXT Counters – Data Log

- **Data Log Bytes In**
- **Data Log Bytes Out**
- **Data Log File Syncs**
performed on commit, in foreground
- **Data Log Sync Time**



PBXT Counters – Background Threads

- **Bytes to Checkpoint**

usually a value in range [0; pbxt_checkpoint_frequency]

- **Log Bytes to Write**

amount of data to be processed by the writer



PBXT Counters – Sweeper

- **Log Bytes to Sweep**

bytes to read from x-log in order to clean transactions

- **Sweeper Wait on Xact**

incremented every 1/10 of sec when we cannot sweep a transaction because we wait for another transaction to finish

- **Dirty Xact Count**

number of transactions to sweep



XTSTAT

- a part of PBXT source tree
- uses mysql client library

```
vkolesnikov@ubuntu8:~/projects/pbxt/bin$ ./xtstat --display=rec,ind,xlog,sweep
```

```
-- PBXT System Variables --
```

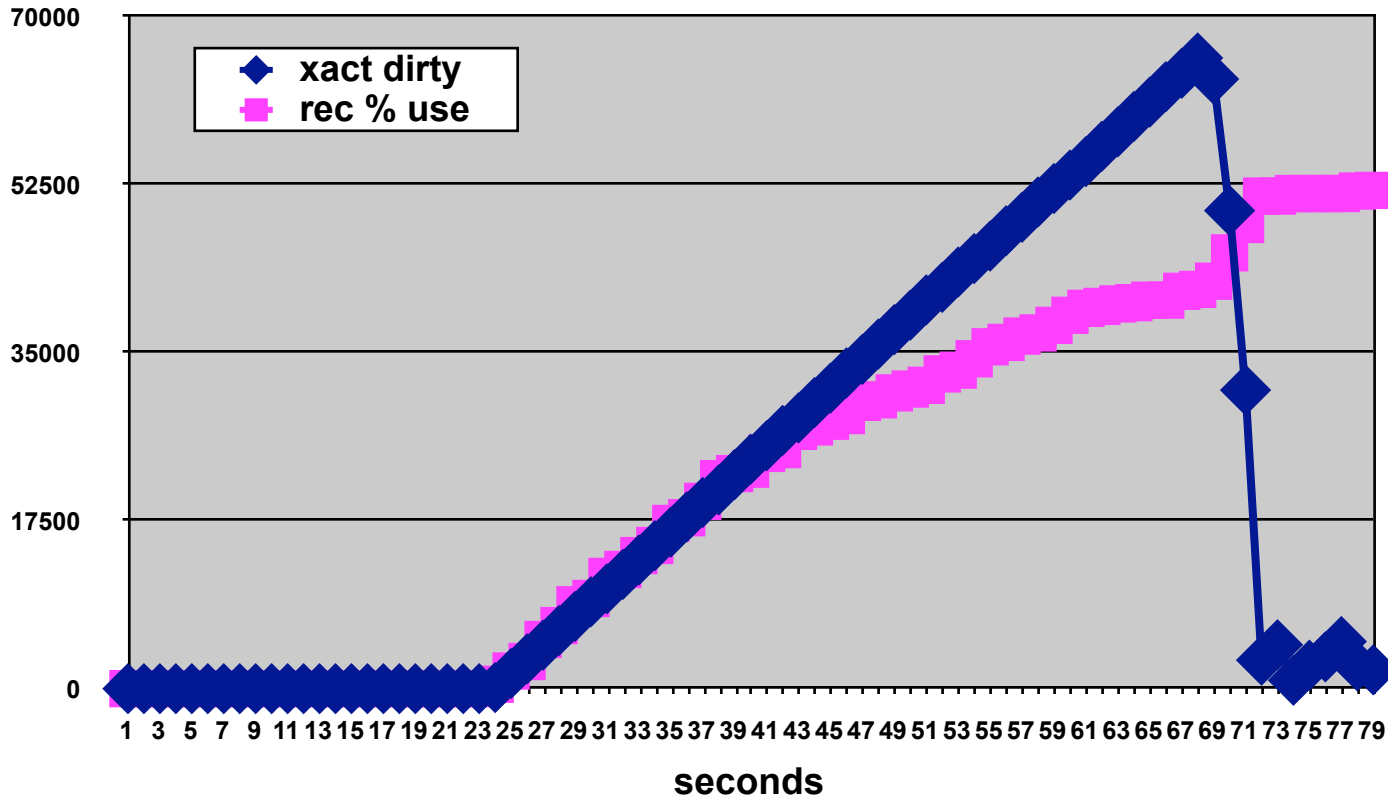
```
pbxt_auto_increment_mode = 0
pbxt_checkpoint_frequency = 12MB
pbxt_data_file_grow_size = 1MB
pbxt_record_cache_size = 100MB
pbxt_row_file_grow_size = 1MB
pbxt_sweeper_priority = 0
pbxt_transaction_buffer_size = 1MB
```

```
Display options: rec,ind,xlog,sweep
```

rec	rec	rec	rec	rec	rec	rec	ind	ind	ind	ind	xlog	xlog	xlog	xlog	xlog	xlog	sweep
in	out	syncs/ms	hits	miss	frees	%use	out	hits	miss	%use	in	out	syncs	msec	hits	miss	waits
69.3M	27.2M	<t/77.0t	541t	2013	0	63.3	51.1M	6309t	936	18.4	288K	156M	9051	427t	15.6t	9	13.4t
50658	0	0/0	1906	1	0	63.4	0	13.7t	1	18.5	0	146K	33	640	63	0	46
111	0	0/0	212	0	0	63.4	0	9330	0	18.5	0	32256	2	985	9	0	10
16915	0	0/0	1708	0	0	63.4	0	13.7t	0	18.5	0	174K	49	295	53	0	73
3996	0	0/1002	226	0	0	63.5	0	8713	0	18.6	0	25088	1	995	1	0	12
5946	40	2/1000	478	0	0	63.5	0	9623	0	18.6	0	65024	2	998	2	0	16
0	0	0/1001	56	0	0	63.5	0	9210	0	18.6	0	0	0	1001	0	0	10
36111	0	1/330	842	1	0	63.5	0	10.8t	0	18.6	0	51200	2	998	2	0	11
1554	0	0/0	1383	0	0	63.5	769K	12.6t	0	18.6	0	59392	2	996	25	0	14
49349	0	0/0	1475	2	0	63.6	886K	12.8t	2	18.6	0	152K	26	663	42	0	6

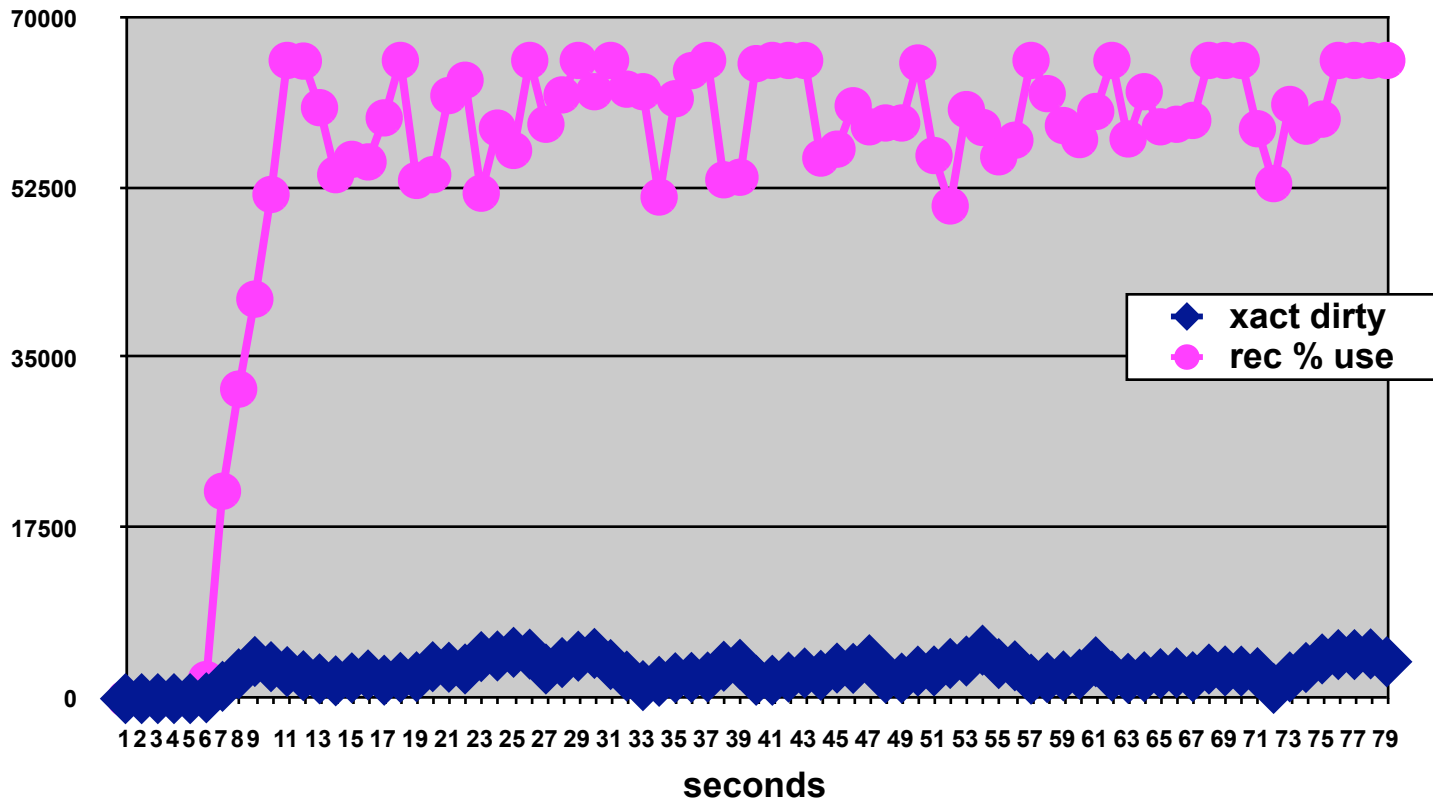


Long Transactions



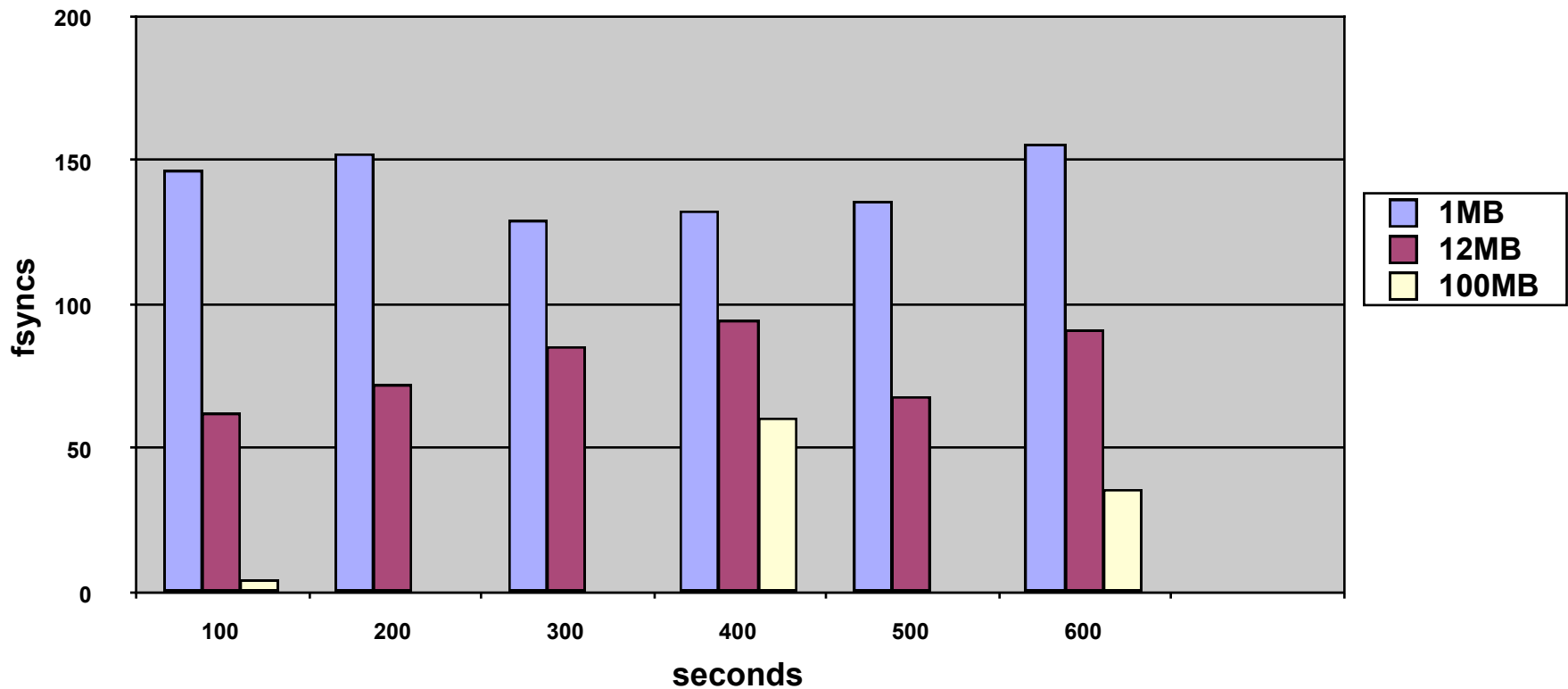


Small Record/Index Cache



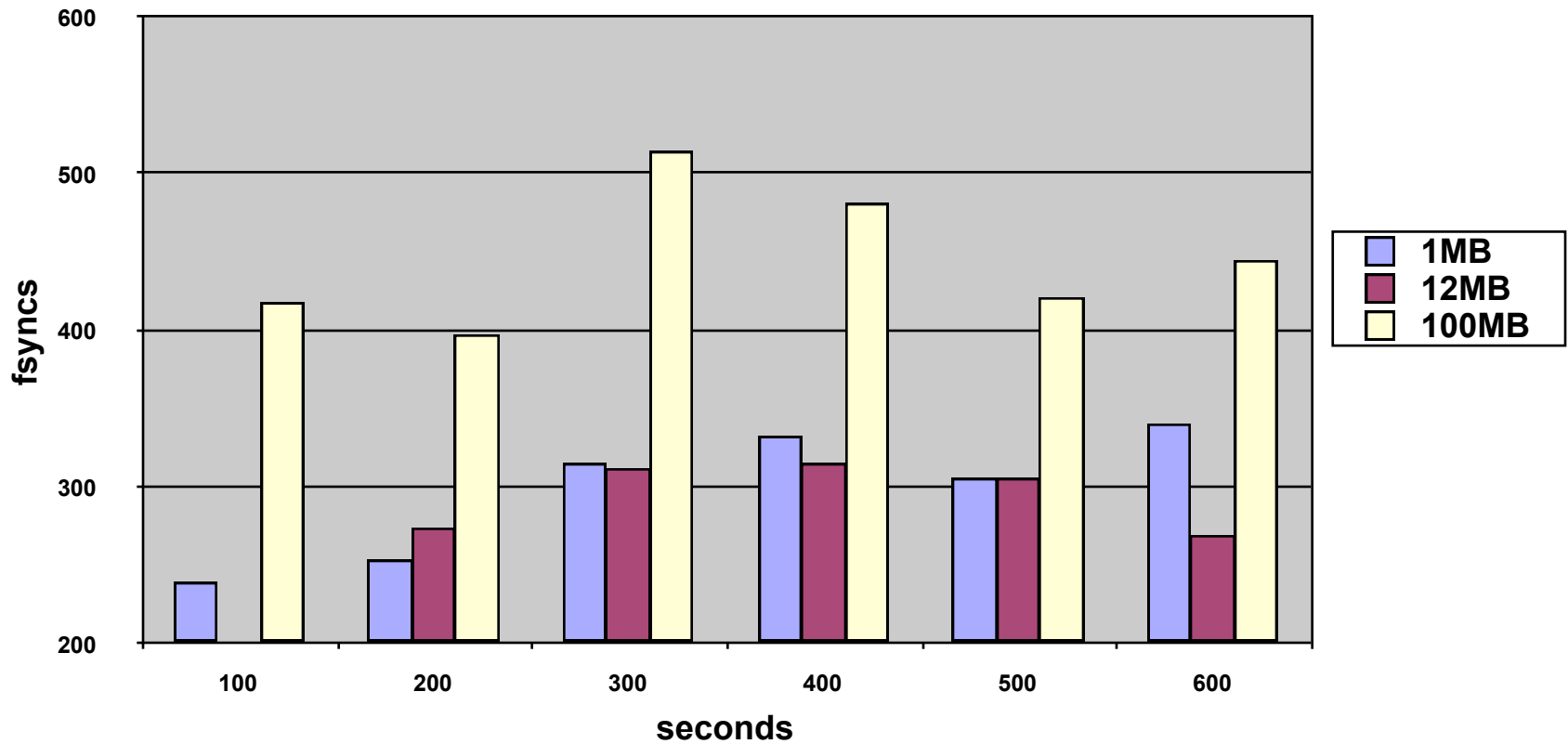


Checkpoint Intervals, Large Cache





Checkpoint Intervals, Small Cache





Other Performance Issues

- **Long-running flushes**
consider distributed data storage
- **Deadlocks**
ORDER BY a non-updated index
- **Too many sequential scans**
check your indexes, update table statistics, make sure the optimizer chooses optimal plan



Table vs Index Scan (Disk)

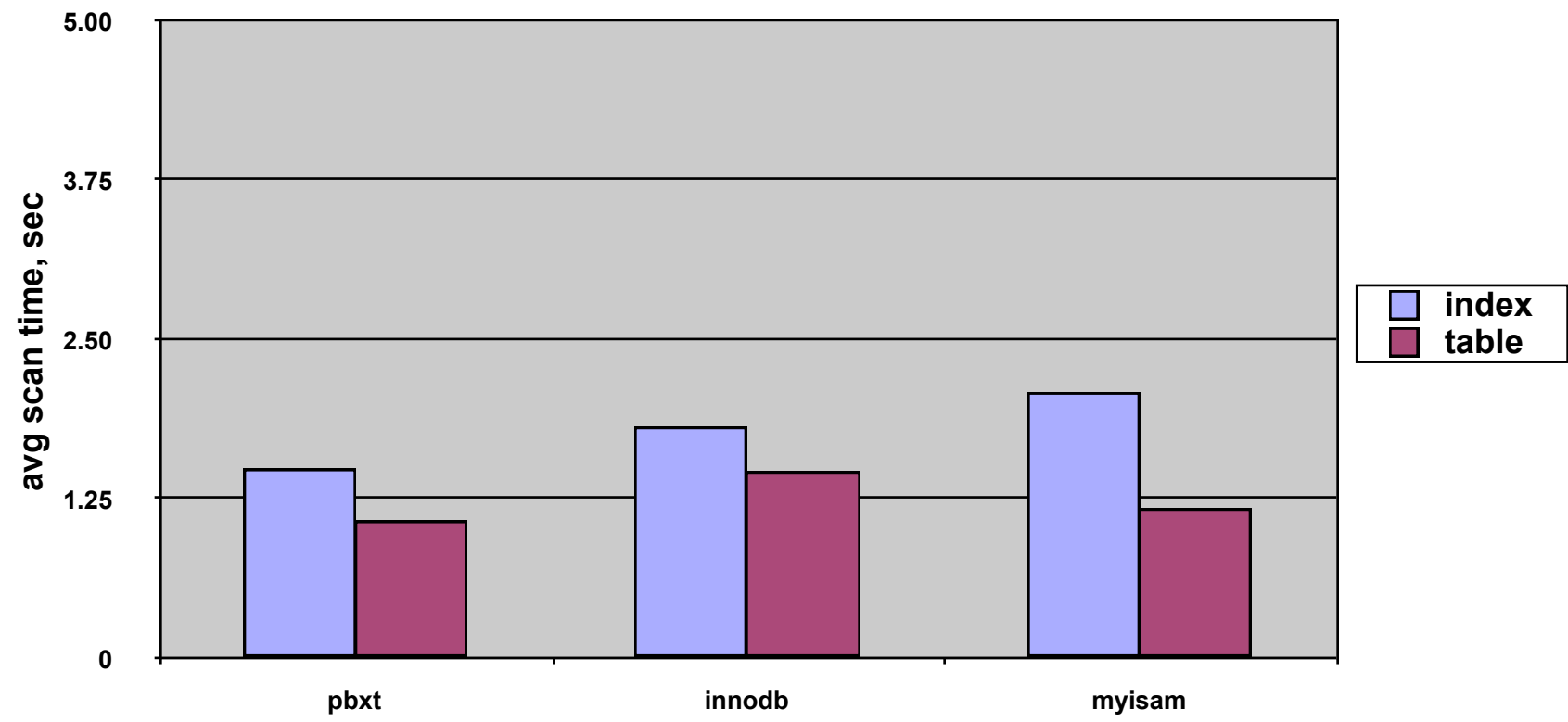
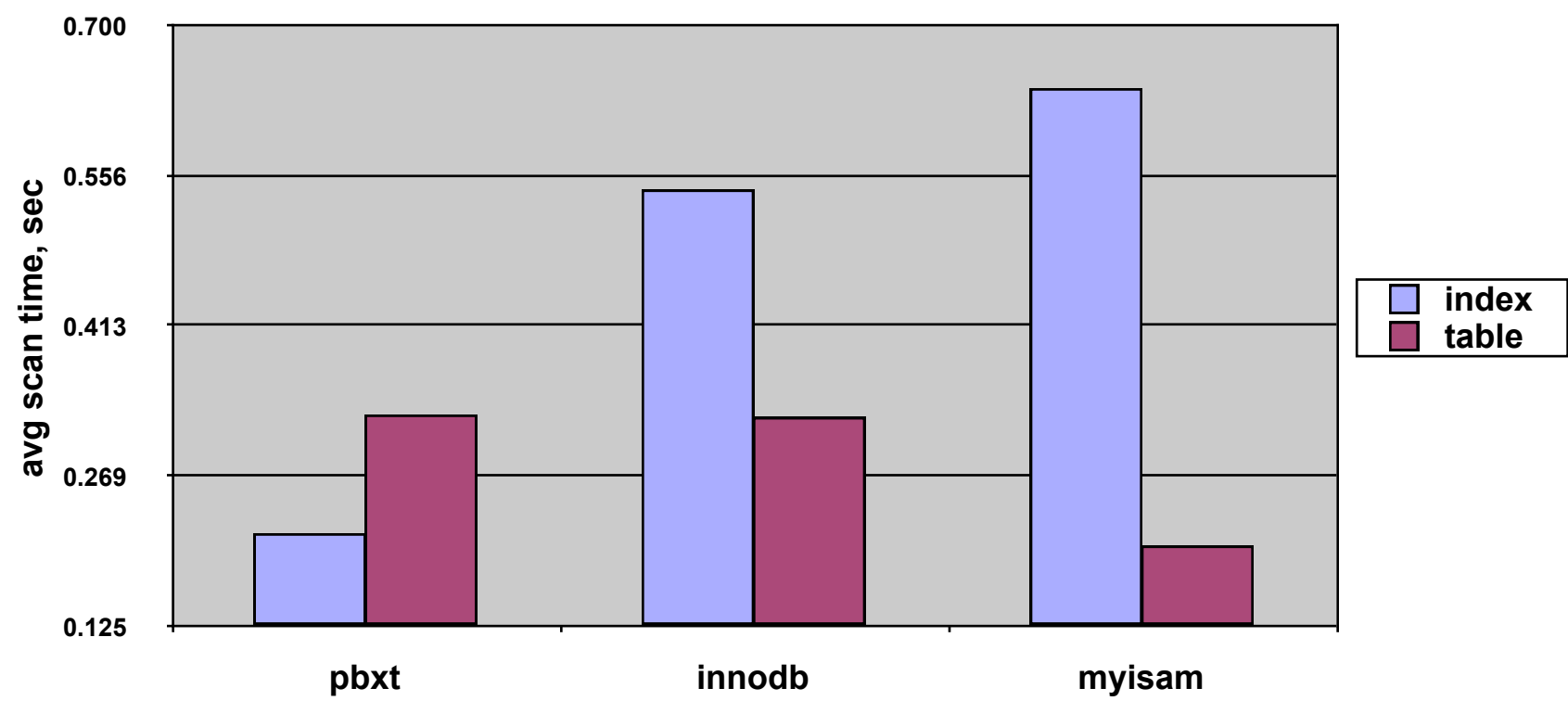




Table vs Index Scan (Cache)





Q&A

Thanks for Listening!

<http://www.primebase.org>

<http://sourceforge.net/projects/pbxt>

<http://pbxt.blogspot.com>