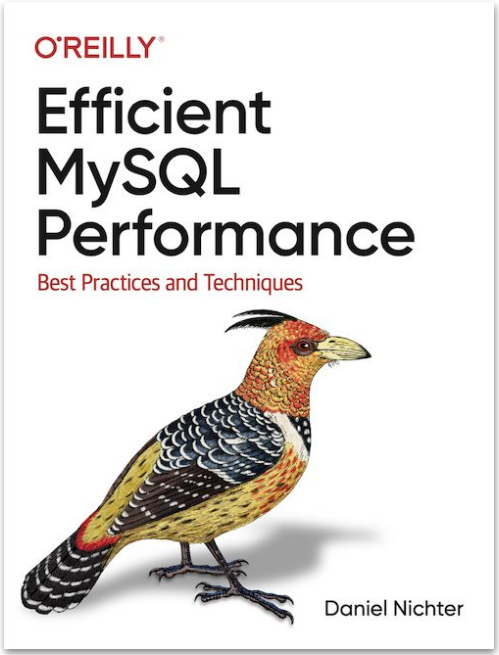
MySQL Performance for Developers

Daniel Nichter @ Percona Live 2023

About Me

- 19 years with MySQL
- hackmysql.com
- Percona
- Block (Square, Cash App, et al.)
- *Efficient MySQL Performance* (O'Reilly 2021)



This is not a sales pitch.

You can learn from MySQL manual, blog posts, and other books.

Intended Audience

- ✗ DBAs
- ✓ Engineers *using* MySQL
  - New to MySQL
  - Not new but want to “level up”
  - Leads/mentors for others using MySQL

Focus

A *path* for learning

how to understand and achieve

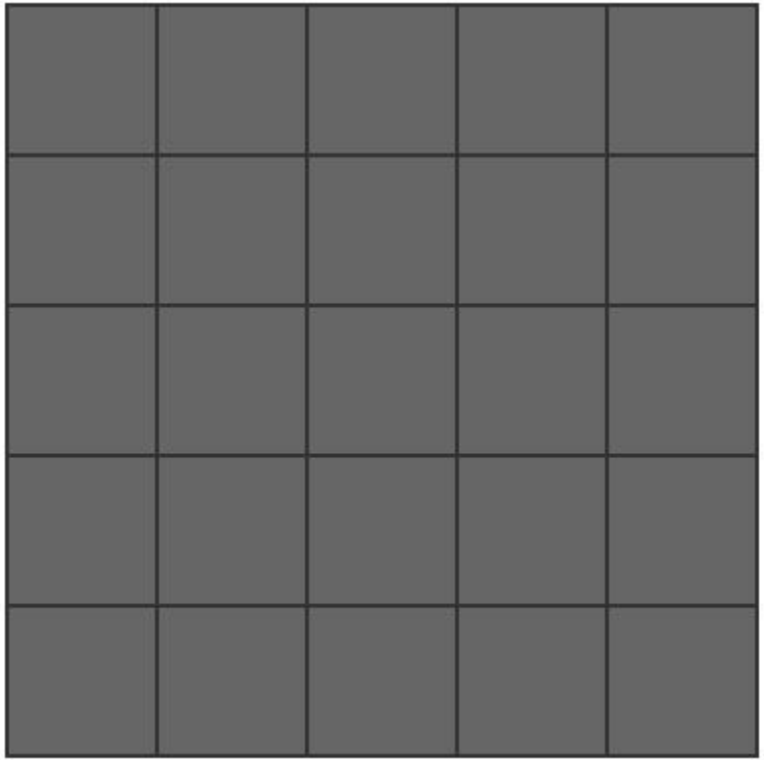
better MySQL performance.

Value

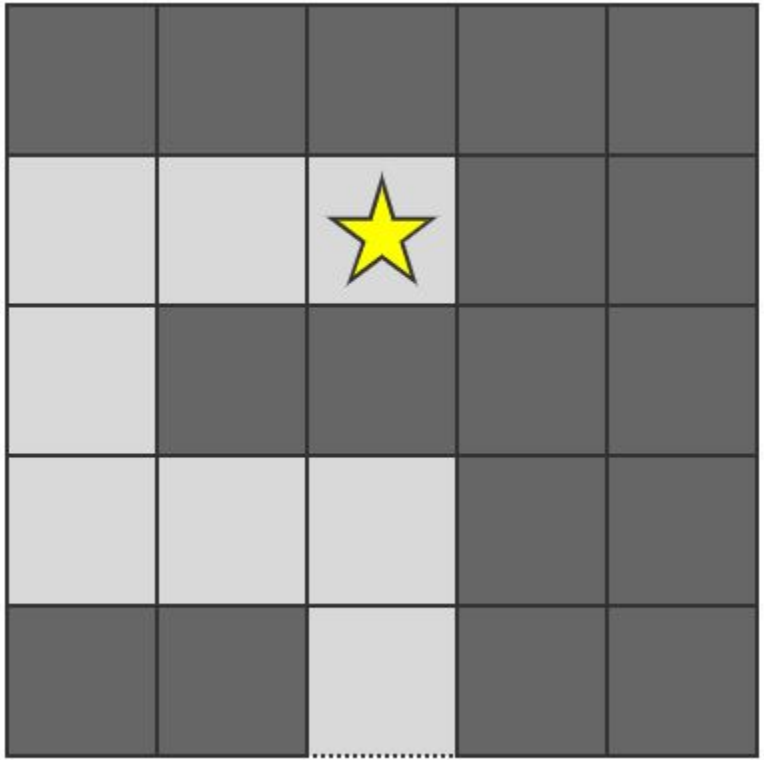
- How to start; where to go…
- Save time (efficiency)...
- Success with MySQL…
- Success at work…
- Success in life (happiness?)

hackmysql.com/path

An Unknown Path



An Efficient Path



An Efficient Path

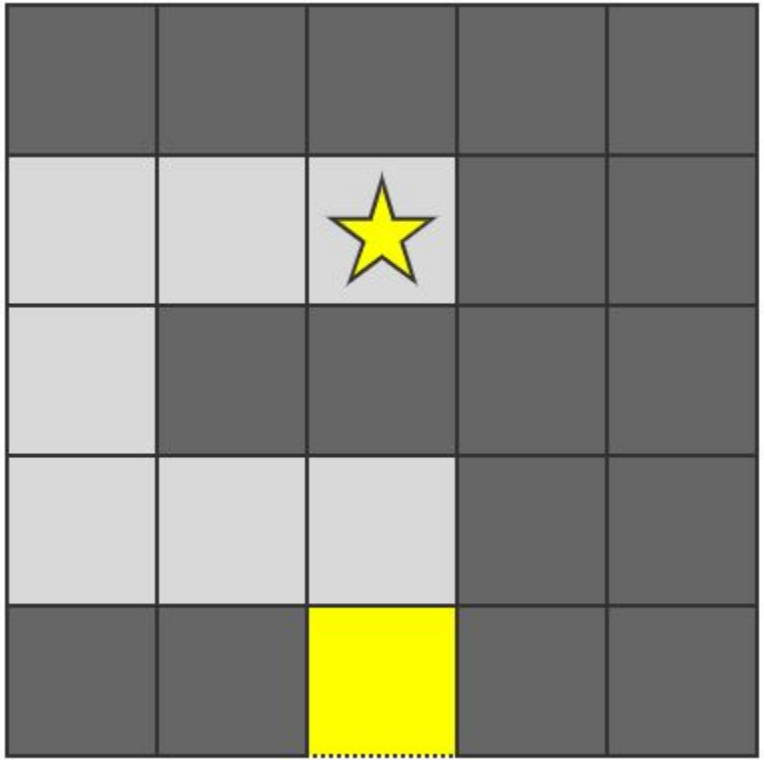
1. Query Response Time
2. Indexes and Indexing
3. Data Storage and Access
4. Sharding
5. Server Metrics
6. Replication Lag
7. Transactions
8. Cloud

@Point_along_the_path

- @Interesting_point_1
- @Interesting_point_N

1. @Action_item_1
2. @Action_item_N

Query Response Time

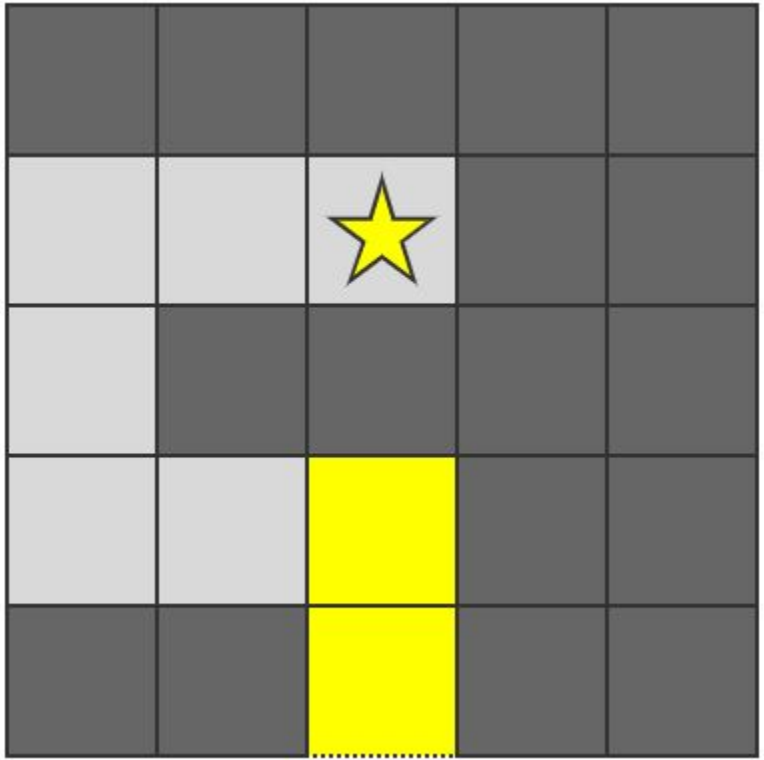


Query Response Time

- MySQL does nothing
- Performance is query response time

1. Choose a tool
2. Enable/configure MySQL *query* metrics
3. Analyze slow queries
4. Teach other engineers

Indexes and Indexing

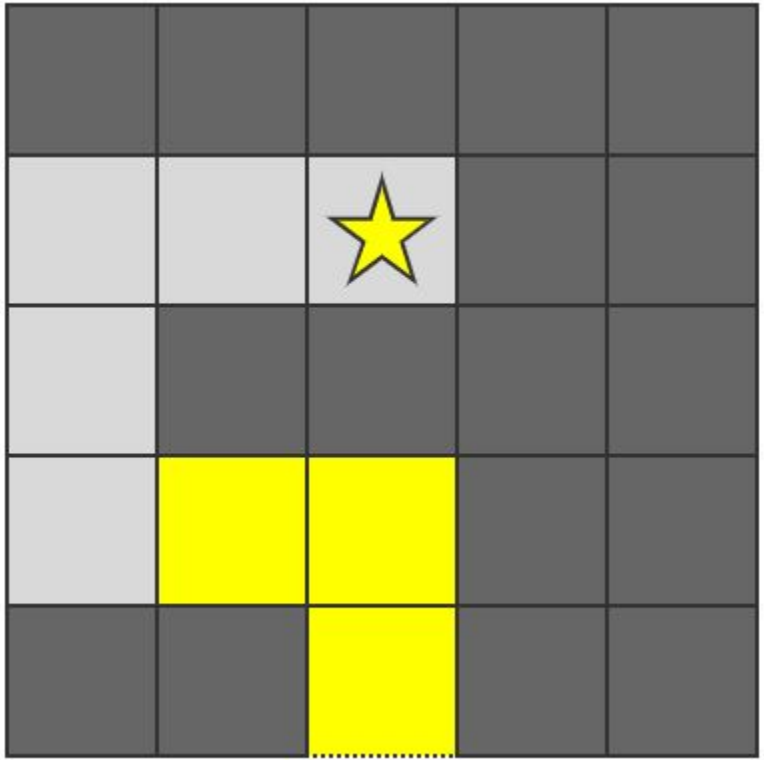


Indexes and Indexing

- *Leftmost Prefix Requirement*
- Learn the 5 before you dive:
  - WHERE
  - GROUP BY
  - ORDER BY
  - Covering Index
  - Join Tables
- Indexes provide leverage against data

1. EXPLAIN slowest queries
2. Consider leftmost prefixes and the 5 on those queries
3. Run `pt-duplicate-key-checker`
4. Browse MySQL manual section 8.2.1

Data Storage and Access

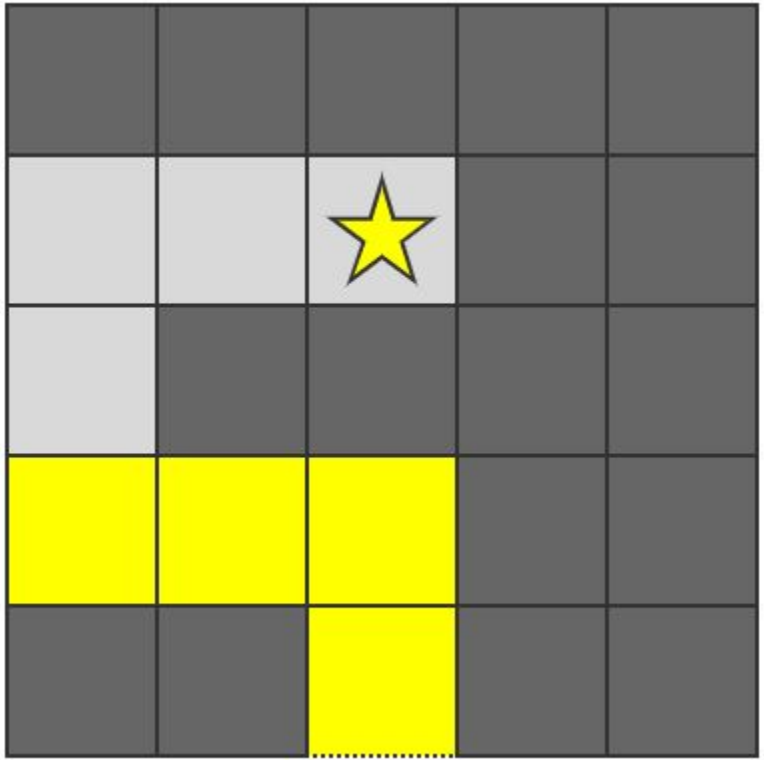


Data Storage and Access

- Data is dead weight—rocks
- Engineers celebrate “less”; they cope with “more”
- *Access patterns* and *working set size* frame performance

1. Review and archive/delete data—*carefully*
2. Review access patterns of slowest queries
3. Consider other data stores

Sharding

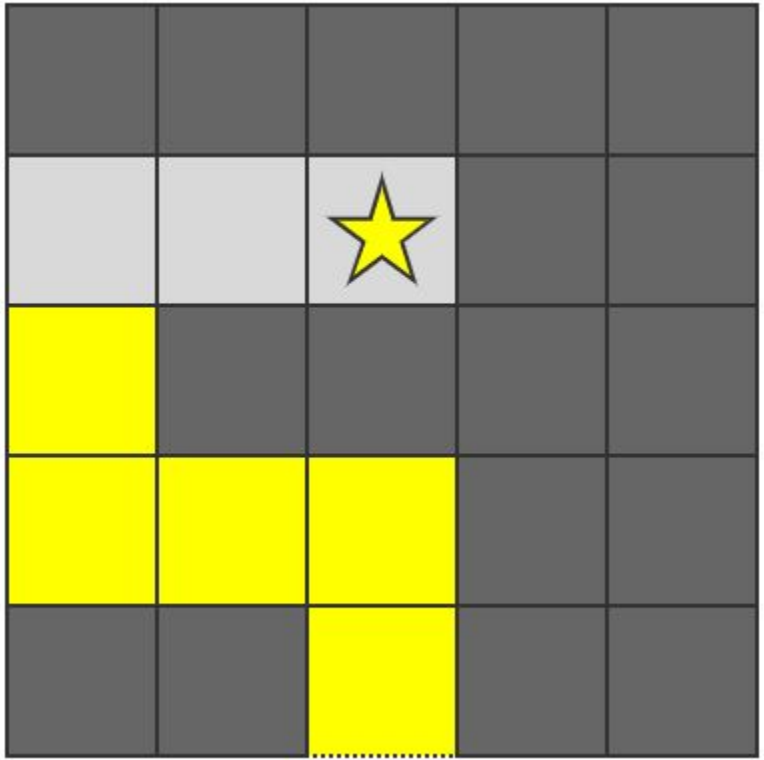


Sharding

- For writes and ops, not (necessarily) data size
- Vitess and Planet Scale
- NewSQL: TiDB, CockroachDB, Neon

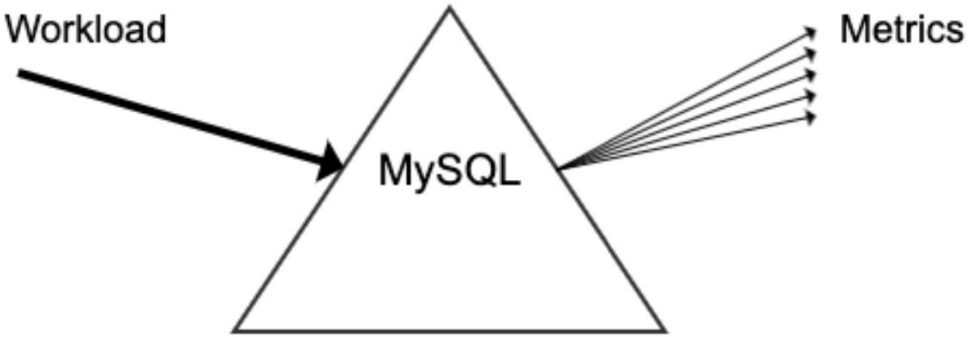
1. Measure and monitoring data size—plan ahead
2. Read about Vitess, Planet Scale, TiDB, and CockroachDB—plan *way* ahead

Server Metrics



Server Metrics

- Raw elements; chemistry required
- Reflect the workload (queries, data, & access patterns)
- *Insight* is proportional to metric *resolution*

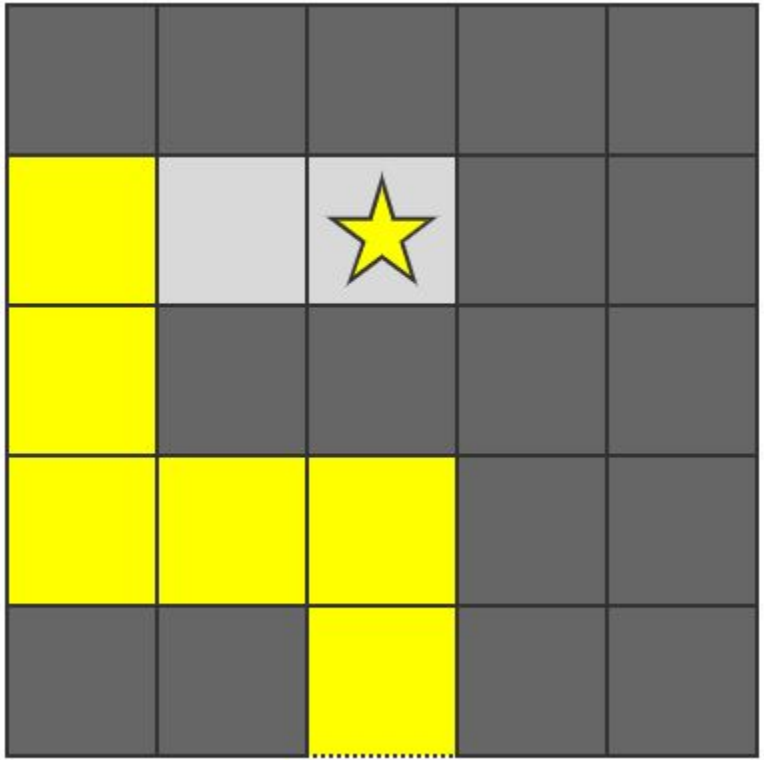


Server Metrics

- Raw elements; chemistry required
- Reflect the workload (queries, data, & access patterns)
- *Insight* is proportional to metric *resolution*

1. Learn about the common, useful metrics
2. Increase resolution
3. Clean up dashboards, charts
4. Try PMM

Replication Lag

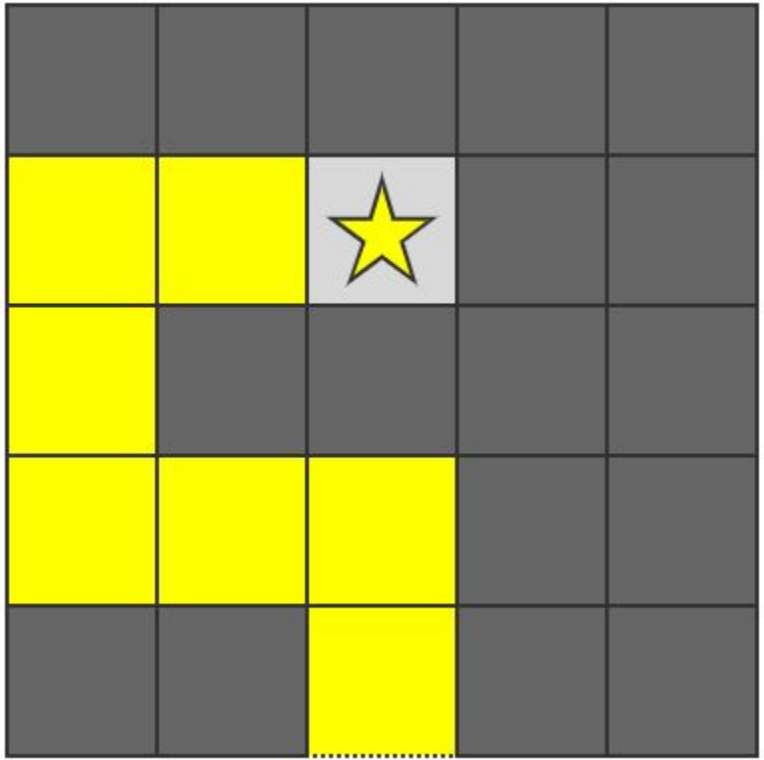


Replication Lag

- Lag is data loss
- Caused by application (other rare causes notwithstanding)
- Monitor and page 24x7x365

1. Monitor and page 24x7x365
2. Attempt sub-second measurements

Transactions

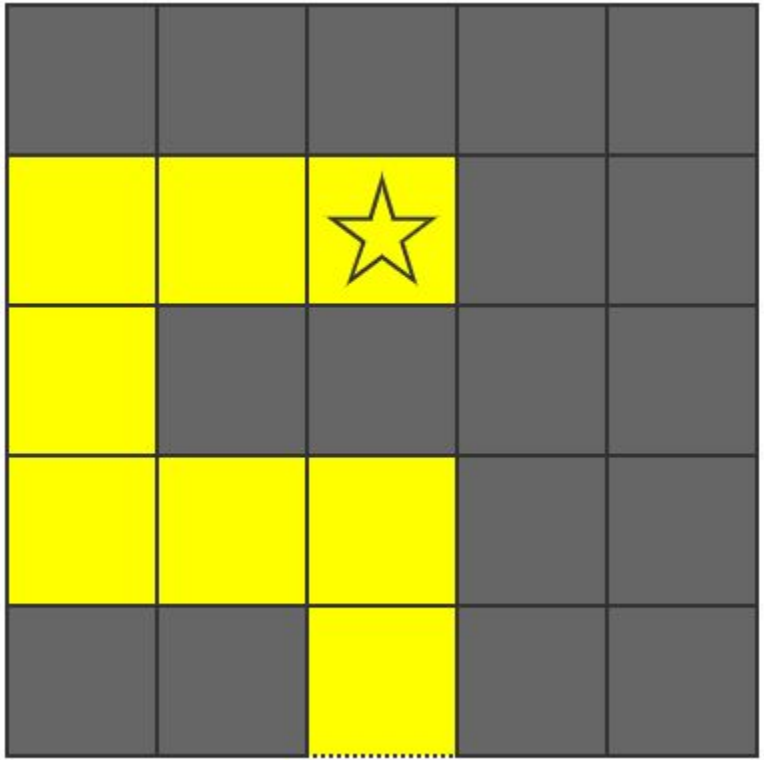


Transactions

- InnoDB locking is subtle but observable in 8.0: performance_schema.data_locks
- If a transaction isn't fleeting, it's defeating
- Transaction monitoring and reporting is nascent

1. Use 8.0 to examine locks of slow queries
2. Review code using trx
3. Try hackmysql.com/trx

Cloud



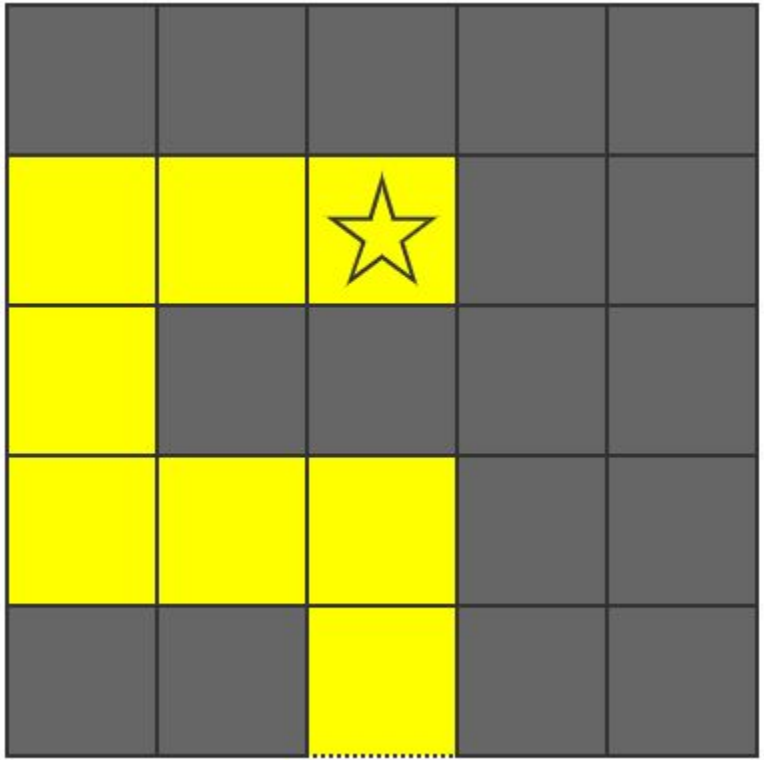
Cloud

- Network-backed storage is slow: spinning disks
- Cheap instances are too small: 1, 2, 4 vCPU
- Cloud providers are *not* DBAs
- Performance is critical for performance *and* costs

1. Review query metrics
2. Review server metrics
3. Review costs
4. Review what exists
5. Review db ops

Hack MySQL > Engineer > Db Ops

Always More to Learn…



An Efficient Path

1. Query Response Time
2. Indexes and Indexing
3. Data Storage and Access
4. Sharding
5. Server Metrics
6. Replication Lag
7. Transactions
8. Cloud

Thank You

hackmysql.com/path