In-Memory Computing SUMMIT NORTH AMERICA 2019

Automating Query Caching with Data Grids

Roland Lee
VP of Product

HEIMDALL DATA

Agenda

- Intro to Database Proxy concept
- Query caching
- Other use cases
- Demo

Executive Summary

- Database Proxies:
  - Improves SQL read/write performance and reliability
  - Deployment requires no application changes

Database Proxy Vendors

Feature	HEIMDALL DATA	ScaleArc	MariaDB	ProxySQL
Automated Failover	✓	✓	✓	✓
Read/Write split	✓	✓	✓	✓
Database Vendor Neutral	✓			
Automated Cache invalidation	✓			
Reduces network latency	✓			

IMDG vs. Database Proxies

Amazon ElastiCache

GridGain

Pivotal GemFire®

redis

ORACLE®

hazelcast

HEIMDALL DATA

MariaDB®

ScaleArc

ProxySQL

- Best scale & performance
- Greenfield applications
- Requires code changes

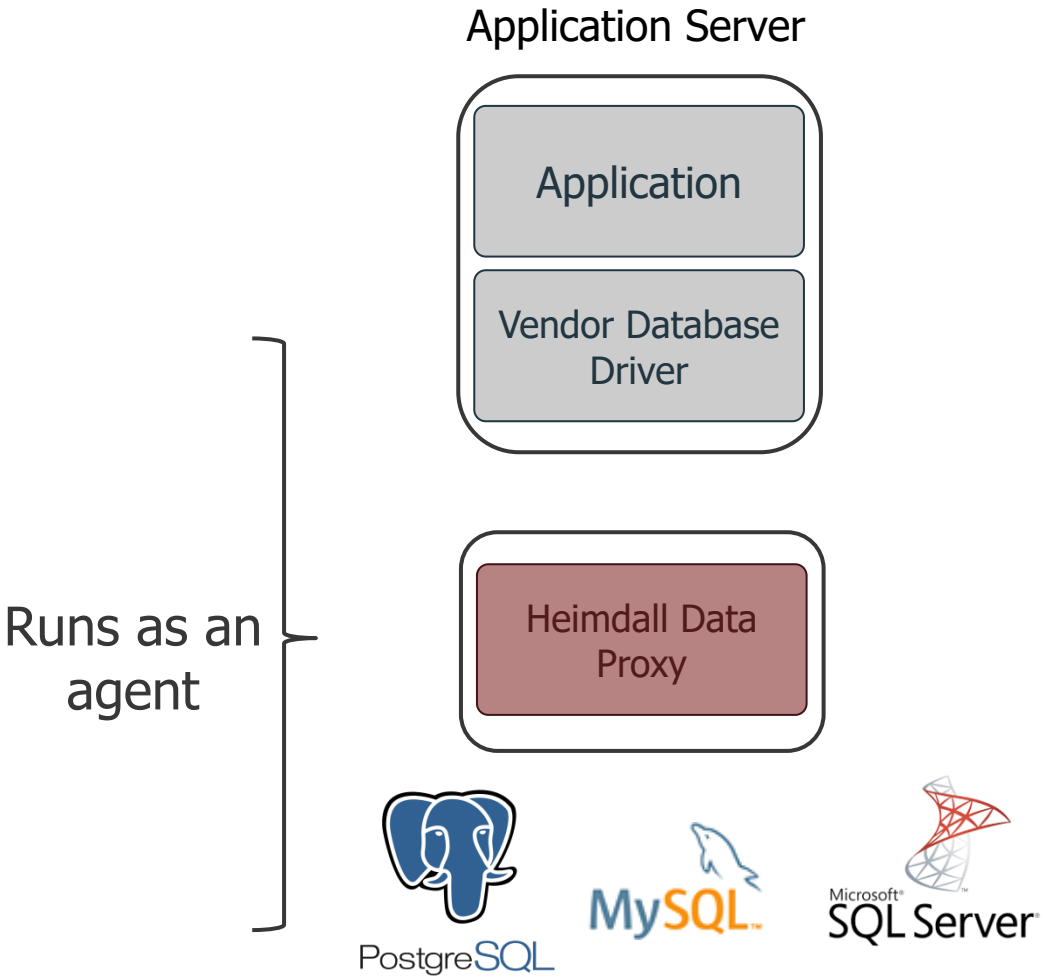
- May be “good enough”
- Existing applications, small dev
- **No code changes**

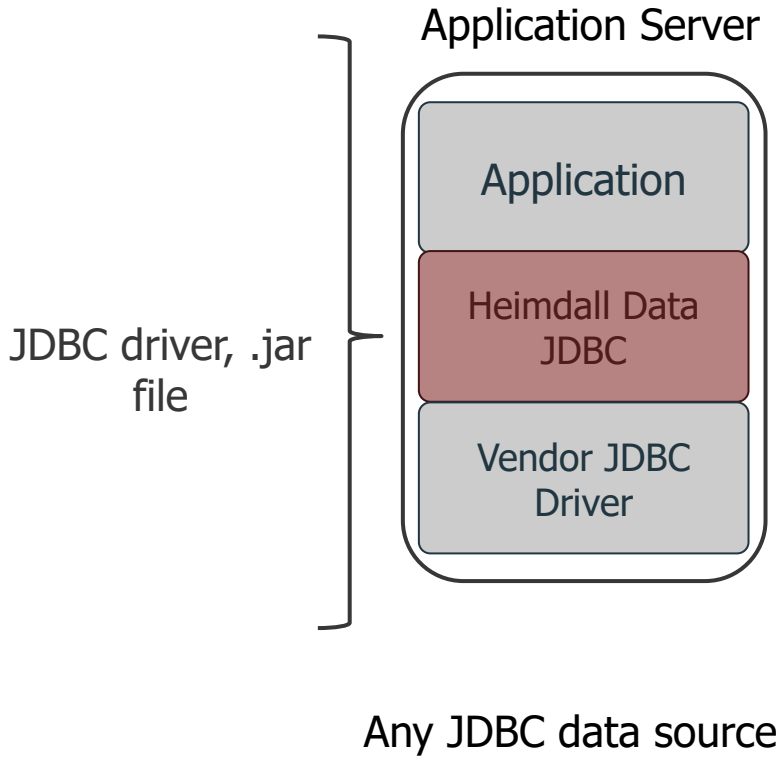
Transparent Database Proxy

Click to add text

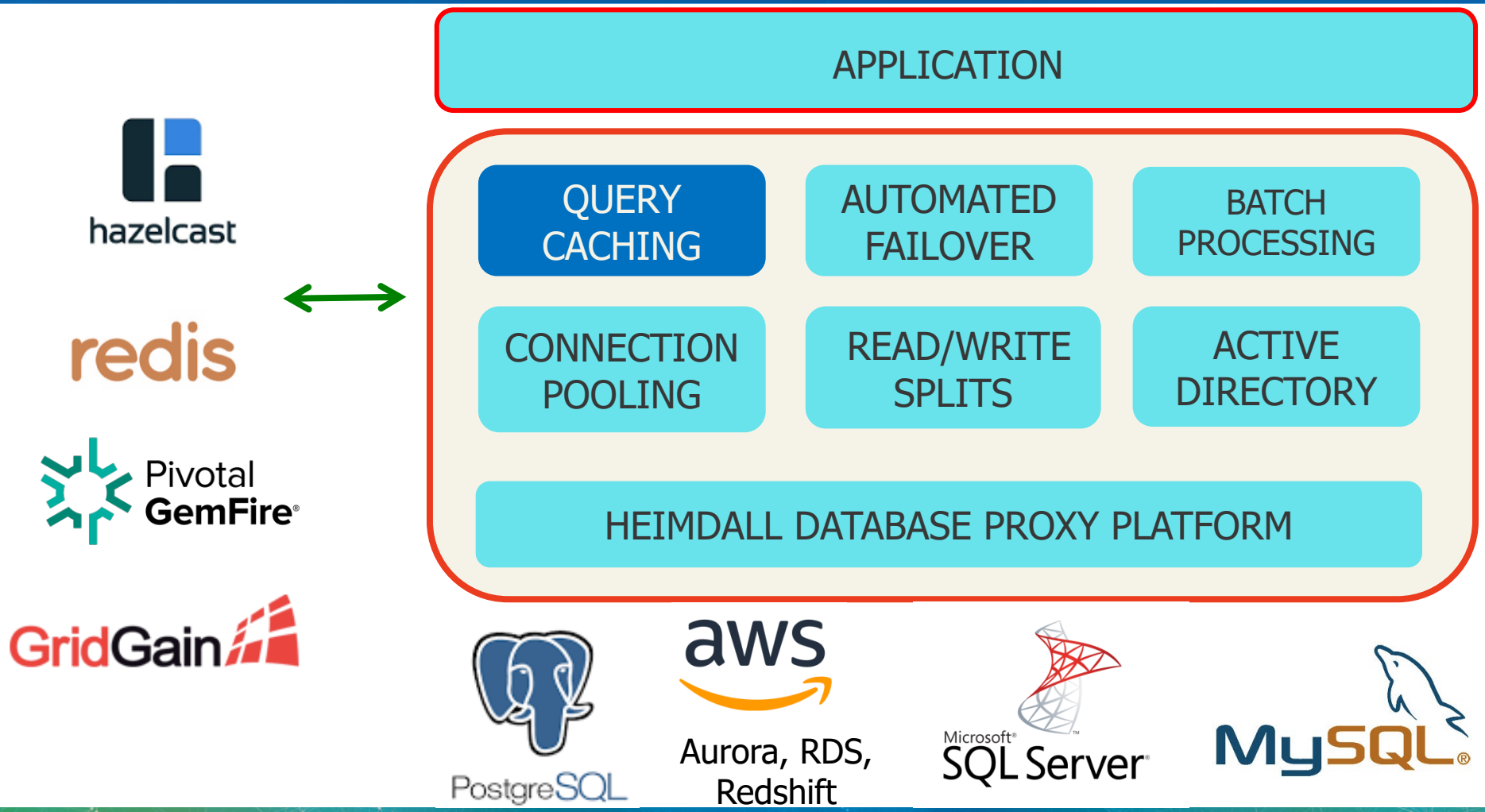
In-Memory Computing SUMMIT NORTH AMERICA 2019

Software Package Options

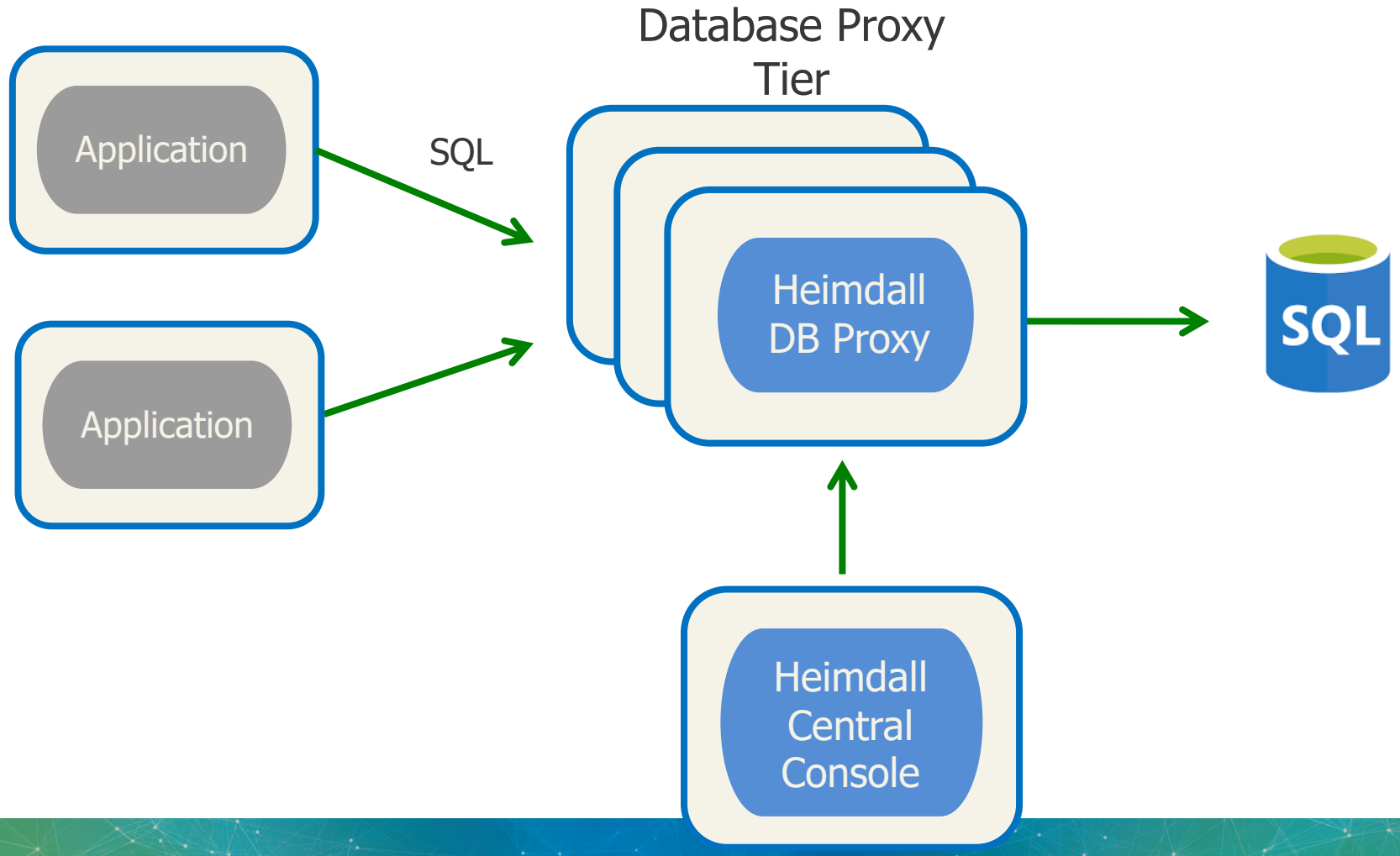




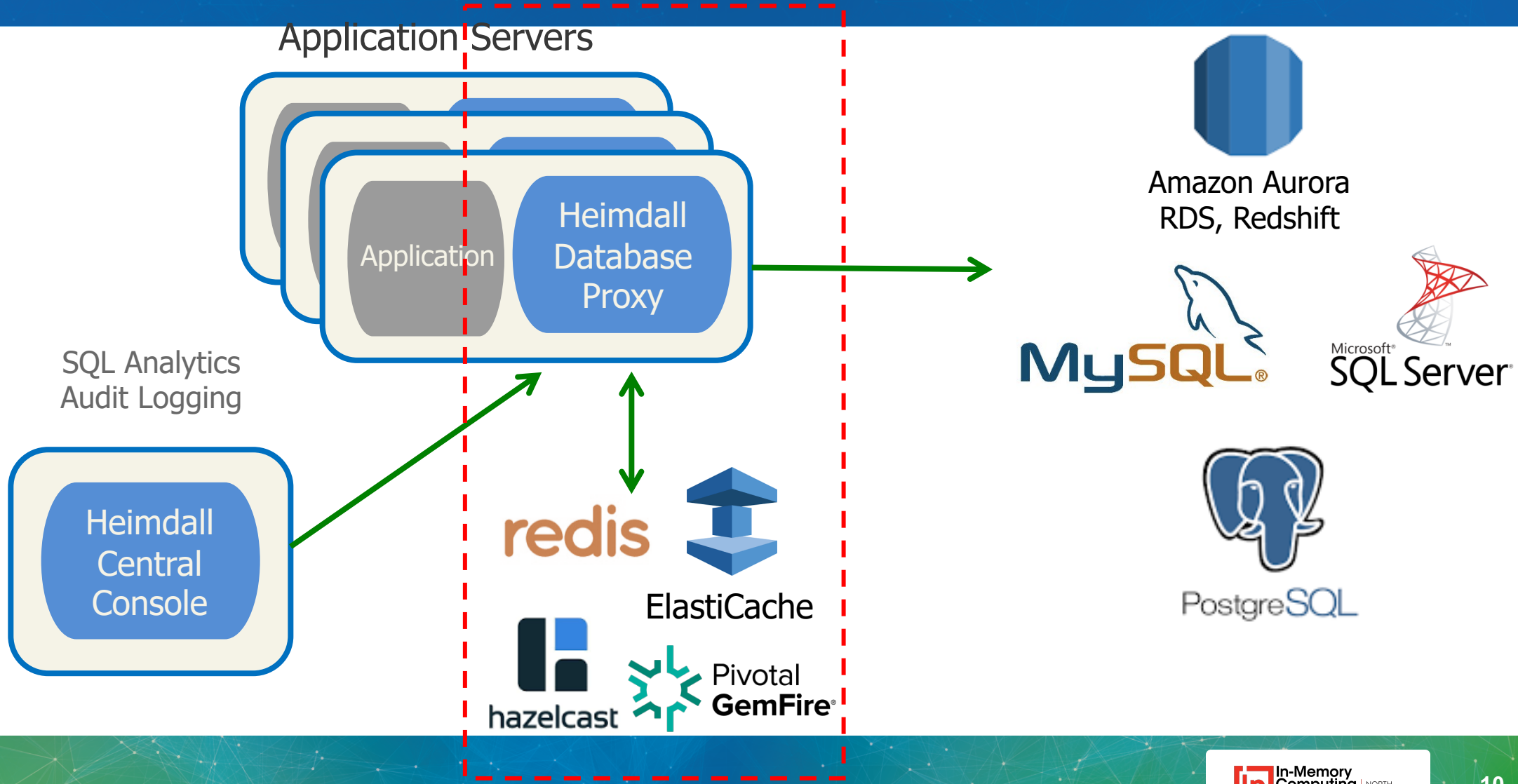
Database Proxy Platform



Heimdall Centralized Deployment



Heimdall Distributed Deployment



Use Cases

How Caching Works

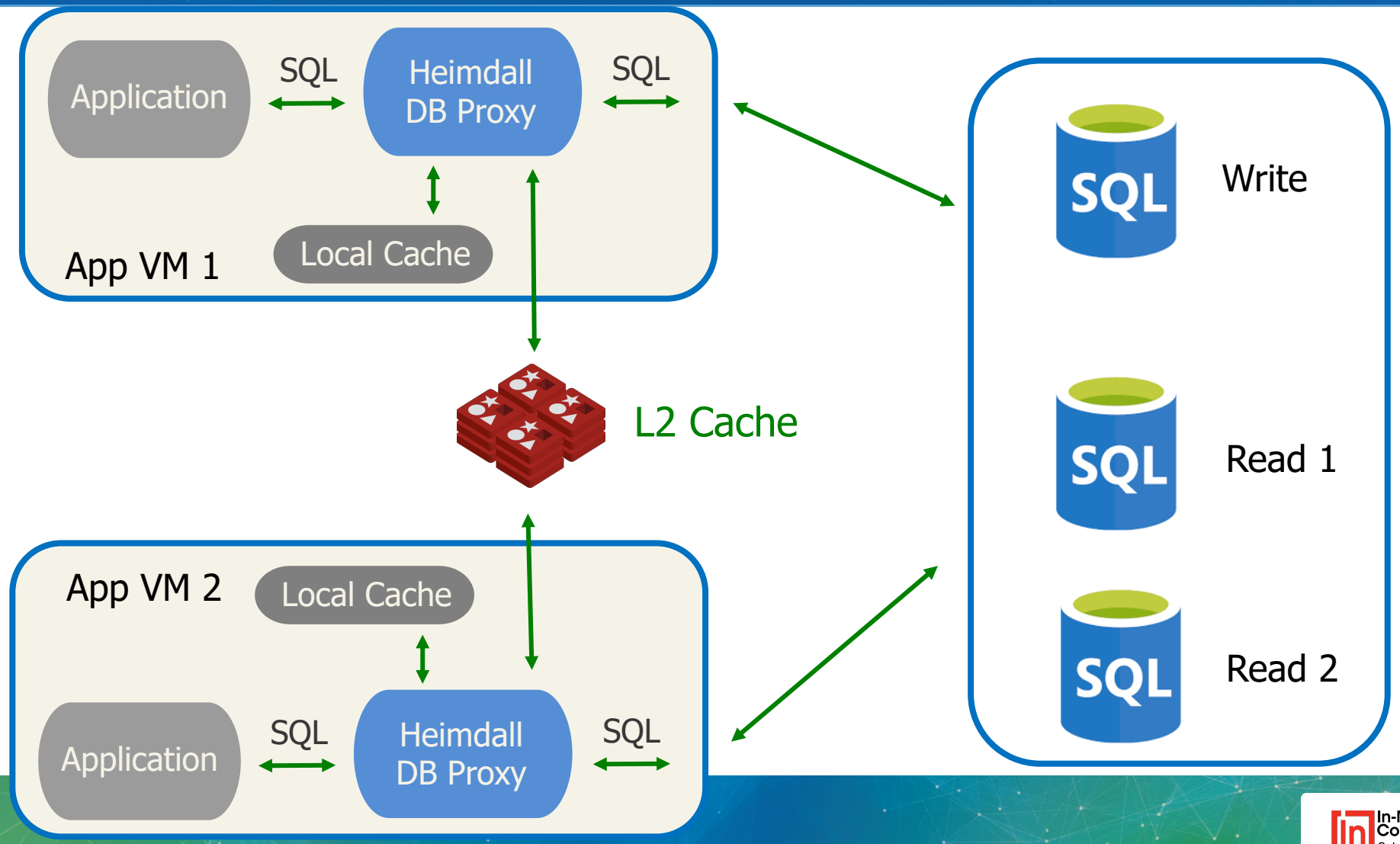
Uses real-time analysis and statistics on:

- Query frequency and variability
- Relative performance of Cache vs. Database

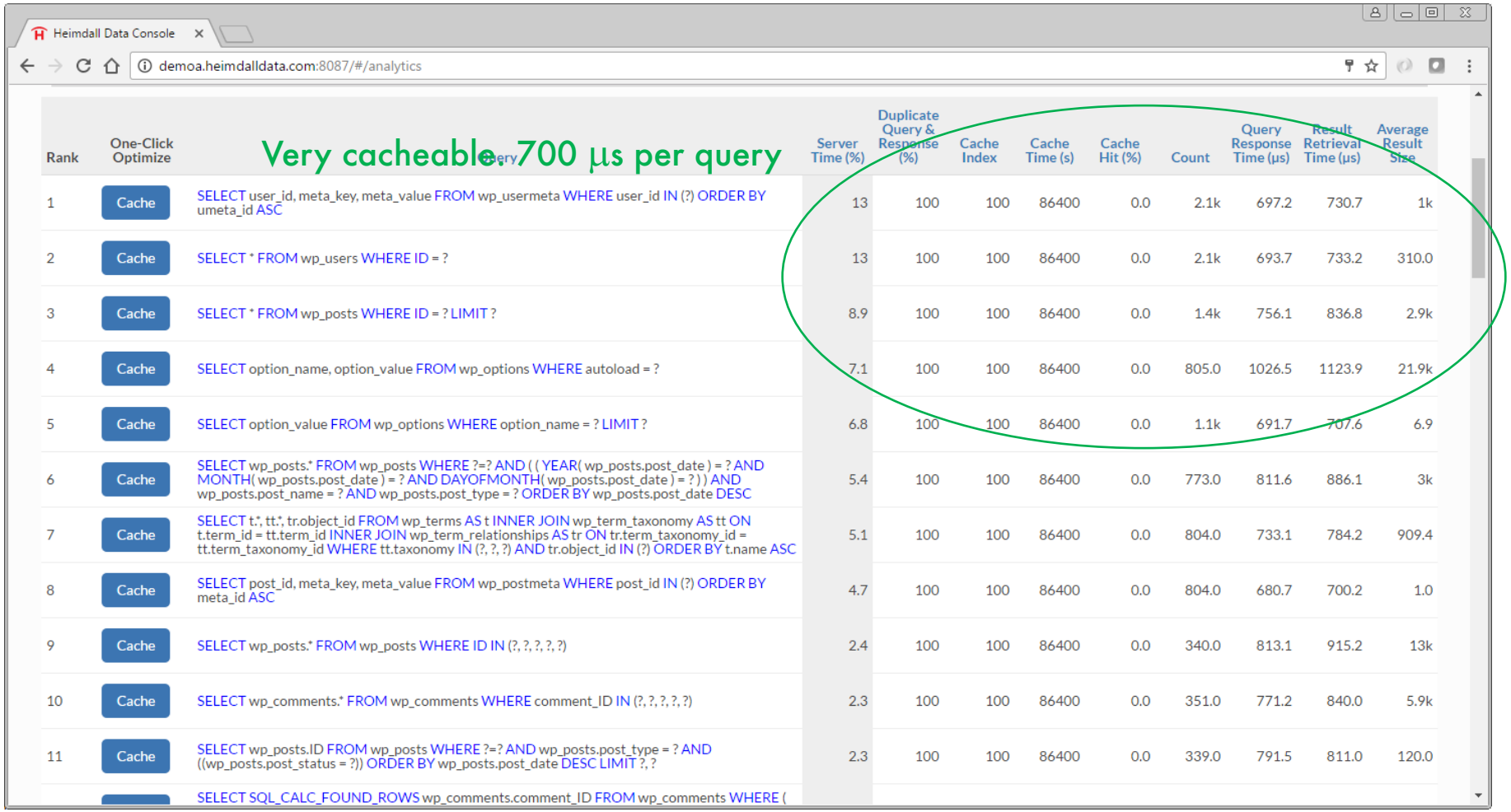
Provides:

- Auto-cache only if there is a performance benefit
- Cache recommendations and benefits

Caching and Read/Write Splits



SQL Analytics



Rank	One-Click Optimize	Query	Server Time (%)	Duplicate Query & Response (%)	Cache Index	Cache Time (s)	Cache Hit (%)	Count	Query Response Time (µs)	Result Retrieval Time (µs)	Average Result Size
1	Cache	SELECT user_id, meta_key, meta_value FROM wp_usermeta WHERE user_id IN (?) ORDER BY umeta_id ASC	13	100	100	86400	0.0	2.1k	697.2	730.7	1k
2	Cache	SELECT * FROM wp_users WHERE ID = ?	13	100	100	86400	0.0	2.1k	693.7	733.2	310.0
3	Cache	SELECT * FROM wp_posts WHERE ID = ? LIMIT ?	8.9	100	100	86400	0.0	1.4k	756.1	836.8	2.9k
4	Cache	SELECT option_name, option_value FROM wp_options WHERE autoload = ?	7.1	100	100	86400	0.0	805.0	1026.5	1123.9	21.9k
5	Cache	SELECT option_value FROM wp_options WHERE option_name = ? LIMIT ?	6.8	100	100	86400	0.0	1.1k	691.7	707.6	6.9
6	Cache	SELECT wp_posts.* FROM wp_posts WHERE ?=? AND ((YEAR(wp_posts.post_date) = ? AND MONTH(wp_posts.post_date) = ? AND DAYOFMONTH(wp_posts.post_date) = ?)) AND wp_posts.post_name = ? AND wp_posts.post_type = ? ORDER BY wp_posts.post_date DESC	5.4	100	100	86400	0.0	773.0	811.6	886.1	3k
7	Cache	SELECT t.*, tt.*, tr.object_id FROM wp_terms AS t INNER JOIN wp_term_taxonomy AS tt ON t.term_id = tt.term_id INNER JOIN wp_term_relationships AS tr ON tr.term_taxonomy_id = tt.term_taxonomy_id WHERE tt.taxonomy IN (?, ?, ?) AND tr.object_id IN (?) ORDER BY t.name ASC	5.1	100	100	86400	0.0	804.0	733.1	784.2	909.4
8	Cache	SELECT post_id, meta_key, meta_value FROM wp_postmeta WHERE post_id IN (?) ORDER BY meta_id ASC	4.7	100	100	86400	0.0	804.0	680.7	700.2	1.0
9	Cache	SELECT wp_posts.* FROM wp_posts WHERE ID IN (?, ?, ?, ?, ?)	2.4	100	100	86400	0.0	340.0	813.1	915.2	13k
10	Cache	SELECT wp_comments.* FROM wp_comments WHERE comment_ID IN (?, ?, ?, ?, ?)	2.3	100	100	86400	0.0	351.0	771.2	840.0	5.9k
11	Cache	SELECT wp_posts.ID FROM wp_posts WHERE ?=? AND wp_posts.post_type = ? AND ((wp_posts.post_status = ?)) ORDER BY wp_posts.post_date DESC LIMIT ?, ?	2.3	100	100	86400	0.0	339.0	791.5	811.0	120.0
		SELECT SQL_CALC_FOUND_ROWS wp_comments.comment_ID FROM wp_comments WHERE (									

Demo

Click to add text

In-Memory Computing SUMMIT NORTH AMERICA 2019

IMDG vs. Database Proxies

Amazon ElastiCache

GridGain

Pivotal GemFire®

redis

ORACLE®

hazelcast

HEIMDALL DATA

MariaDB®

ScaleArc

ProxySQL

- Best scale & performance
- Greenfield applications
- Requires code changes

- Good enough
- Existing applications, small dev
- **No code changes**


HEIMDALL
DATA