Performance tuning inefficient legacy applications can be a tremendous drain on development resources in an organization. Handling availability of the database layer of these applications can be an even larger challenge. ScaleArc can help improve both challenges by improving the availability of the database layer and accelerating common repetitive database queries from the application.
Executive Summary

The past five years or so has seen a shift in people’s expectations for application performance. As they’ve seen their personal apps deliver continuous uptime and high performance, they’ve grown more frustrated with the slower performers in business. Despite these business apps being critical to corporate performance, users have “just dealt with it” for years whenever they’re slow and unreliable. One challenge, of course, has been that these enterprise apps are much more complex than the personal apps, making it harder to update them. Plus, making the necessary changes to “update” the application might be out of the hands of an organization when they don’t control the source code, or are costly and time prohibitive to update when they do.

A compelling alternative has emerged. Rather than recoding apps for improved uptime and performance in a database environment, consider the benefits of database load balancing software. A pioneer in this technology, ScaleArc has set out to make business application platforms more highly available and perform better by managing the traffic into the database servers. The ScaleArc software acts as an intermediary between the application and database layers. Sitting in between, it buffers database outages from impacting application availability and transparently accelerates the application’s performance, all without code or usability changes.

This whitepaper explains the benefits of the ScaleArc software and shows benchmarking results of one feature set – app-transparent caching. In our tests, the ScaleArc software enabled a 5x improvement in the operations processed per minute and a drop in CPU consumption from a peak of 100% to a peak of 60%.

Our testing demonstrates that the ScaleArc software provides an easy way to improve application uptime and performance with no changes to the application code.

The Lingering Impact of Availability and Performance Problems

Availability and performance of customer-facing business-critical applications should be at the forefront of any IT organization. The end users attempting to access an application do not care about where in the technology stack a performance problem or an outage started. All they know - and care about - is that the application they are trying to access - to shop, buy music, book tickets, post a social media comment, or communicate with you - is slow or unavailable. This situation, no matter how brief, impacts them and you.

The memory of these application performance and availability problems lingers, creating a negative perception of your business. Even momentary interruptions in service can be enough to drive away customers and expose an area of weakness for your competitors to exploit.

Availability and performance within the organization are usually handled on a micro instead of macro level - that is, each application stack has different teams of people managing it, and the availability and performance discussions are usually limited to the technology area managed by an individual team. Each application stack team develops robust availability solutions for its piece of the puzzle, but the teams rarely communicate with each other, much less design their pieces with the other parts of the technology stack in mind. The availability solution of one layer might not just “plug-in” to the availability architecture of the other layers. Service interruptions from one layer can have unexpected cascading effects further up the stack. Each team works to performance tune their layer, but a positive change in one layer can negatively impact others, which might have adverse effects on the entire system.
Layers and Their Dependencies

Let’s look at the layers of this infrastructure and application stack, starting with on-premises server deployments.

If your servers are currently within your own datacenter, your collective IT teams together will manage all of these layers. All of the data and applications need to be stored somewhere, so enterprise storage is used as a central storage medium underneath all layers of the technology stack. Physical servers (ideally configured with a virtualization solution) connect to the storage using networking and/or storage interconnect fabric. Operating systems are installed either on the physical servers or on virtual machines deployed on the virtual server hosts. The applications are then deployed and configured inside the operating system. The end users connect to the application by any number of options, and when the stack is healthy, the end user experience should be uninterrupted. Life is good.

If your virtual machines are hosted in the cloud, but your IT organization still manages the applications inside the virtual machines, the infrastructure stack responsibilities shift. Everything underneath the operating system becomes a black box, and your options to manage performance and architect availability are handled through the management interface on the cloud portal, and not hands-on by your IT teams.

When the applications and databases themselves are hosted in the cloud as a dedicated service, the infrastructure becomes even more abstracted away from visibility by the administrators.

The cloud-based databases and applications are still subject to the same availability rules of the platform underneath, but your options are restricted to what is offered through the cloud platform provider.

Regardless of your system architecture, any of these layers should be designed for maximum resiliency and performance. The availability and performance solutions of each layer should be managed carefully as part of a larger business continuity design. Each layer usually (or with enough work put into it) possesses solid options for availability. For example, enterprise storage is hardened against failures with multiple controllers and backplanes, and with redundantly configured
disks that can buffer the SAN from component failures to prevent an outage. Networking is designed to be redundant with multiple switches and redundant communication paths to insulate against port or switch failure. Web applications can be configured with multiple web server nodes for a scale-out architecture that can buffer against individual node failures. Database systems have numerous features for availability as well.

Layers and Their Performance Characteristics

The higher you go in this application stack, the more complex the availability solution becomes, and the more dependent it is on the layers underneath. Performance is even more complex, because far more variables exist within system performance. For example, take a look at this common web application architecture.

This model works and scales well until you reach the database tier. Most of the enterprise database platforms in the world have only one active database server that can actually change table data. Secondary copies of the data can be configured for read-only access, but rarely is the usage of these secondary copies of data seamless by the application, since the app would need recoding to direct reads to the secondaries. Plus, any interruption in service at the data layer can knock out an entire application stack and contribute to an application outage.

Take Microsoft’s enterprise database platform, SQL Server, which can be deployed either locally in your organization’s datacenter or in the public cloud. SQL Server contains numerous world-class availability features, such as the flagship “AlwaysOn” Availability Group feature. Failover clusters, mirroring, and data replication are also solid options for improving the database availability. Each one of these features is quite powerful when architected properly, but each also comes with its own availability signature and design requirements. Azure SQL Database, SQL Server’s sibling in the Microsoft Azure cloud, contains its own availability characteristics, depending on the platform options you select.

The applications assume that the data layers will always be there and usually make no attempt to insulate themselves from a database outage. An active database connection from an application will be dropped during a SQL Server failover, and without the proper coding, the active transaction will be lost. The end user sees and experiences application errors. Sometimes the application even needs to be manually restarted if the database connectivity is interrupted, which results in an even longer and impactful outage.

This disconnect is only magnified if the application is from a third-party vendor and your IT groups cannot modify the application code to change this basic assumption. Even applications that your company built in-house can require a major overhaul to change this basic behavior.

Expectations and Modern Times

The true challenge in architecting for uptime and performance is in managing the end-users’ expectations.

Your users have their expectations set in the consumer-grade ‘app’ world that we live in today. These ‘apps’ are always available, and because of their modern architecture, generally perform well out in the wild.
Then your users go to work and sit down at the legacy application that their organization depends upon. This application’s back-end architecture is monolithic in nature, and the legacy codebase would require a tremendous amount of re-tooling by the vendor to modernize the availability architecture and bring the performance back to within the modern end users’ expectation of how an ‘app’ should operate.

The experience of consumer-grade applications became the new enterprise application expectation, but most of the legacy applications that your business cannot function without fail even the most basic expectations of today’s business users.

So if the application vendor will not modernize an application to meet the expectations of the users, and we, the administrators of the application within an organization, are caught between the unstoppable business and the immovable third-party vendor, what do we do?

What if you could add an intermediary between the application and database to improve this basic assumption without needing to change the application?

Now you can.

ScaleArc acts as the transparent database availability solution that completes the database availability requirements that your business demands without requiring changes to the application server.

The ScaleArc database load balancing software deploys transparently between applications or web servers and your databases.

The software provides a range of capabilities as it directs traffic into the database servers, including:

- failover that’s transparent to the application - ScaleArc holds queries in its queue during database failover and sends them onto the new primary after failover is complete
- automatic read/write split - so you don’t have to recode your apps to leverage AlwaysOn scale out
- dynamic load balancing, based on server response time - ScaleArc delivers geo-aware load balancing that goes far beyond the “round robin” load balancing in SQL Server 2016
- connection pooling and multiplexing - to increase server efficiency
- query response caching - with no application changes, to speed application throughput
• real-time and historical analytics - to simplify troubleshooting and highlight good candidates for caching

These benchmark tests focused on SQL Server - the ScaleArc software also supports MySQL and Oracle databases.

These benchmark tests also focused on application performance improvements via caching. The ScaleArc also provides app-transparent failover. By queueing transactions during failover, the ScaleArc software shields users from seeing application errors, enabling continuous availability despite database downtime, whether planned or unplanned.

**Performance Through Efficiency**

Performance problems underneath business-critical applications are one of the most common challenges in enterprise IT organizations today. To demonstrate the efficacy of ScaleArc in transparently improving the performance of these legacy applications, a demonstration of the potential improvements in performance was assembled to illustrate the power of ScaleArc. ScaleArc is introduced into a synthetic database load test environment to present the impact of ScaleArc.

**Test Environment and Methodology**

The testing environment consisted of a trio of SQL Server 2016 virtual machines configured for a commonly found highly available architecture, leveraging a three-node SQL Server Availability Group configuration, with two synchronous (assumed to be locally deployed) and one asynchronous (assumed to be at a disaster recovery facility) secondary server.

**DVDStore version 3, freely available at http://github.com/dvdstore/ds3, was used to generate and simulate a highly concurrent and active workload against the SQL Server database. It is an open-source cross-database OLTP workload simulator and can generate load tests against SQL Server, Oracle, MySQL, MariaDB, and**
PostgreSQL database platforms. The included scripts are used to create sample data files and then load them into the database. The included executable, which is database platform specific, is then used to generate a synthetic workload, similar to concurrent activity from an online store that sells DVDs.

DVDStore was configured for a highly active load test with a batch file used to repeat the process. The only change between test cycles was to the destination platform - directly to the database server or via ScaleArc.

```bash
\sqlserverds3\ds3sqlserverdriver.exe --target=(destination IP address) --n_threads=80
--run_time=15 --ramp_rate=10 --db_size=10GB --warmup_time=0 --think_time=0.085
--n_searches=3 --search_batch_size=5 --n_line_items=5
--windows_perf_host=(servername)
--detailed_view=N --out_filename=loadtestresults.txt
```

The output from the test cycle is two metrics - orders placed total and orders per minute generated. The final orders per minute averages are more representative of a real-world number, as the results show how the database engine ramps up the query result cache and the engine reaches steady state.

DVDStore is an ideal candidate for such a scalability test with ScaleArc. Being that it is designed to run across a number of database platforms, the database design is very generic and ANSI standard, with very little functionality that is SQL Server-specific. This architecture is similar to a large number of third-party legacy applications on the market today, and the impact of the ScaleArc appliance in these tests is similar to expected results on these systems in the field.

After the Availability Group test cycles, ScaleArc was introduced into the architecture in between the DVDStore workload driver and the SQL Server Availability Group listener.
ScaleArc was then configured for partial query caching, a key feature of ScaleArc. The engine is capable of query introspection and caching only those queries that are deemed appropriate for caching. The DVDStore queries that are appropriate for caching including product reviews and inventory lists. Tasks such as new user registration and orders being placed were deemed not appropriate for caching in this test and were omitted from the query cache rules. ScaleArc does support automatic cache invalidation, which would enable eCommerce entities to cache even data not typically suitable for caching, such as shopping cart data, since the software would flush the cache whenever the cart content changed.

The testing environment included the following infrastructure configuration:

<table>
<thead>
<tr>
<th>Component</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtualization Platform</td>
<td>VMware vSphere 6.0, ESXi 6.0 Update 2 Patch 3 (4192238)</td>
</tr>
<tr>
<td>Virtual Machines</td>
<td>1x8 vCPUs, 64GB vRAM, Windows Server 2012R2 Datacenter 64-bit</td>
</tr>
<tr>
<td>Compute Hardware</td>
<td>HP DL580 G7, 4x10 Intel Xeon E7-4870 CPUs @ 2.40GHz, 512GB RAM, 2x10GbE networking interconnects</td>
</tr>
<tr>
<td>Storage</td>
<td>Pure Storage //m20 All-flash SAN</td>
</tr>
<tr>
<td>SQL Server</td>
<td>SQL Server 2016 RTM CU01 64-bit, Cost Threshold of Parallelism = 25, MaxDOP = 0</td>
</tr>
<tr>
<td>Testing Software</td>
<td>DVDStore3, build as of 2016.09.01, with 100GB specified database workload</td>
</tr>
<tr>
<td>Database</td>
<td>Specified at 100GB within DVDStore, but database indexes and statistics made final DB size 250GB. SQL Server 2016 compatibility. Default DVDStore configuration.</td>
</tr>
<tr>
<td>ScaleArc</td>
<td>ScaleArc VMware-specific virtual appliance, 4 vCPUs, 64GB RAM</td>
</tr>
</tbody>
</table>

**Failover Handling**

To ensure guaranteed application availability during Availability Group, failover cluster, or other brief interruptions in service, the application should have a block of code that incorporates a “retry pattern” in its database connection layer. This code allows for the application to better handle brief interruptions in database-layer service by retrying a database transaction a number of times, with a brief pause in the process for each attempt. At the time the database is brought back online, the application will retry its operation and continue on as normal. Coded correctly, the application will not reflect any connection difficulties and the end users will be insulated from the failover.


The DVDStore application does not contain application retry logic within its database connection code, and the failover handling of ScaleArc was tested with DVDStore but shown to still disconnect the benchmark test cycle when the Availability Group fails over.

If your application does not support this sort of application retry logic, or if the third-party vendor cannot or will not introduce such logic into the application, the application layer can be interrupted during these failovers. ScaleArc can help detect this interruption and help the application accommodate the disconnections more gracefully. As soon as the database is back up and in service, ScaleArc re-establishes the connection, and the application should reconnect and resume with less interruption to the end users.
First Test Cycle - Availability Groups
First, DVDStore was configured to connect to the Availability Group listener. The workload timing was configured for a 15-minute runtime, and the database buffer pool was flushed prior to each test to ensure that no data was cached prior to the start of the test cycle.

The final results of the 15-minute test cycle presented an average orders per minute (OPM) value of 5000. The ramp rate through the test cycle showed that the SQL Server buffer pool was actively caching the working set data and then reached steady-state very quickly into the test.

The CPU consumption during the test cycle showed that the SQL Server workload was very well balanced across all eight CPU cores.
The test cycled averaged 59.9% CPU consumed, with 2.99% of that CPU consumption being the Windows-level kernel time. The overhead of the replication activity with the secondary Availability Group synchronous replica caused enough overhead in CPU that the testing was unable to drive the workload to 100% CPU consumed.

The disk consumption was also profiled. During the test cycle, the initial window of the test showed SQL Server performing a large amount of retrieval operations of data from disk, but as time continued, much of the working set of data was now in the SQL Server buffer pool. The test cycle averaged 26.09MB/s read and 12.23 MB/s write.

**Second Test Cycle - Introducing ScaleArc**

The second testing cycle introduced ScaleArc into the database architecture. ScaleArc was configured to present a virtual IP address for the application to connect to, and the ScaleArc software was connected to the Availability Group listener as its database destination. The DVDStore workload timing was again configured for a 15-minute runtime, and the database buffer pool and ScaleArc appliance caches were flushed prior to each test to ensure that no data was resident in memory prior to the start of the test cycle.
Certain queries were not added to the caching rules engine, including those handling new user registration and the entire order checkout process. In a real-world environment, the business might not opt to cache these processes, whereas items such as product reviews or new inventory could be cached more readily. Many other queries, including those with product reviews and inventory browsing, were added into the query cache.

Items in this list with a cache hit percentage greater than zero were items that were added into the caching rules engine. Specific stored procedure calls, such as PURCHASE, NEW_CUSTOMER, and LOGIN were deemed cache latency sensitive by the theoretical business, and to help keep this scenario realistic, were not added into the caching scheme.

Throughout the testing, the cache utilization numbers continued to climb linearly. The connection count from the DVDStore test driver stayed constant at the pre-specified 80 concurrent connections.
ScaleArc shows the administrator the level of cache effectiveness through displaying the queries issued in a given time slice and showing the percentage of the cache hit ratio and time saved, which directly translates to performance improvements for the application.

<table>
<thead>
<tr>
<th>Row</th>
<th>Pattern</th>
<th>Total Queries</th>
<th>Cache Hit (%)</th>
<th>Server Time</th>
<th>Cache Time</th>
<th>Time Saved</th>
<th>Resp. Boost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GET_PROD_REVIEWS_BY_(<em>)(</em>)</td>
<td>40,370</td>
<td>100.00</td>
<td>28.34 ms</td>
<td>7.88 Sec</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>GET_PROD_REVIEWS_BY_(<em>)(</em>)</td>
<td>39,995</td>
<td>100.00</td>
<td>0</td>
<td>8.17 Sec</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>GET_PROD_REVIEWS_BY_...INN(*)</td>
<td>27,156</td>
<td>6.69</td>
<td>42.88 Sec</td>
<td>313.63 ms</td>
<td>2.76 Sec</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>BROWSE_BY_CATEGORY @ry_inn(*)</td>
<td>28,989</td>
<td>100.00</td>
<td>0</td>
<td>6.52 Sec</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>BROWSE_BY_TITLE @bat_(<em>)(</em>)</td>
<td>26,967</td>
<td>3.45</td>
<td>1 Min 44 Sec</td>
<td>209.6 ms</td>
<td>3.54 Sec</td>
<td>0%</td>
</tr>
<tr>
<td>6</td>
<td>GET_PROD_REVIEWS @ba_od_inn(*)</td>
<td>26,913</td>
<td>29.60</td>
<td>40.27 Sec</td>
<td>1.42 Sec</td>
<td>15.51 Sec</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>PURCHASE @customer_id...int(*)</td>
<td>26,854</td>
<td>0.00</td>
<td>21 Min 46 Sec</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>8</td>
<td>GET_PROD_REVIEWS_BY_...od_inn(*)</td>
<td>26,782</td>
<td>29.93</td>
<td>41.24 Sec</td>
<td>1.45 Sec</td>
<td>16.17 Sec</td>
<td>0%</td>
</tr>
<tr>
<td>9</td>
<td>BROWSE_BY_ACTOR @bat_(<em>)(</em>)</td>
<td>26,571</td>
<td>57.92</td>
<td>50.56 Sec</td>
<td>3.76 Sec</td>
<td>1 Min 5 Sec</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>LOGIN @username_inn_(<em>)(</em>)</td>
<td>21,602</td>
<td>0.00</td>
<td>5 Min 11 Sec</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>11</td>
<td>NEW_CUSTOMER @username_(<em>)(</em>)</td>
<td>5,332</td>
<td>0.60</td>
<td>1 Min</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

The end results show the 15-minute test cycle showed an average orders per minute (OPM) value of over 25,300, more than a 5x performance improvement than without the query caching layer in front of the Availability Group.

![ScaleArc Database Acceleration Orders Per Minute](chart.png)

The performance curve of OPM results shows the cumulative impact of the query caching at work. As the time progresses, the cache inside ScaleArc warms up and becomes more effective. Once the working set reaches steady state, the performance levels out and provides very effective performance improvements. The cache hit ratio continues to improve as the test continues over time as more of the repetitive calls are cached appropriately. ScaleArc’s dashboard shows the number of queries that are cached.
The CPU consumption within the SQL Server during each test was also profiled.

As the caching warms up, the CPU consumption drops to a steady state value of 60.2%, but the Windows kernel CPU consumption is greater than before at 4.98%. The amount of work that the SQL Server is performing shows more write activity, which is a direct result of a greater amount of write activity occurring because of more new users and more orders being placed and written into the database.

ScaleArc’s real-time analytics also show the query pattern in its caching engine and the operational statistics demonstrating the effectiveness of the query pattern.
The analytics show that for the queries being most repetitively called, the cache hit ratios improve as the cache warms up. As the cache warms up, the load for these calls is shifted from the database engine to the ScaleArc appliance, and the net improvement in performance for the application is tracked within the ScaleArc dashboard.

The disk activity was also profiled. Since ScaleArc was handling much of the repetitive data read operations, the amount of read activity is lower than on the VM without ScaleArc in front. The testing showed that the SQL Server engine had more activity related to writes, which is in line with the overall performance results. The disk activity averaged 23.86MB/s read and 42.73MB/s write. As the ScaleArc cache warmed up, the amount of read activity in the database dropped with it.

<table>
<thead>
<tr>
<th>Averages</th>
<th>AG Only</th>
<th>With ScaleArc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read MB/s</td>
<td>26.09</td>
<td>23.86</td>
</tr>
<tr>
<td>Write MB/s</td>
<td>12.23</td>
<td>42.73</td>
</tr>
</tbody>
</table>
The disk results show a mild reduction in disk read activity and elevated write activity. The read caching was effective in improving the efficiency of the application stack, while leaving the database engine more cycles to perform the write operations as the bottleneck was shifted away from the application itself.

**Conclusion**

The end results of comparing the outcome of the two test cycles shows an undeniable performance improvement to the DVDStore benchmark test when using ScaleArc to transparently cache database queries.

The overall SQL Server performance improvement of adding ScaleArc into the system cannot be overstated.

<table>
<thead>
<tr>
<th>Operations Per Minute</th>
<th>AG Only</th>
<th>With ScaleArc</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5,021</td>
<td>25,323</td>
<td>504%</td>
</tr>
</tbody>
</table>

Introducing ScaleArc between the application server and the database server as a caching and acceleration intermediary provided a 5x performance improvement to a legacy application without any application or database-layer changes during this test cycle with DVDStore.

In addition to the performance improvements at the database layer, the system resource consumption footprint changed as the data handling bottleneck was reduced.

<table>
<thead>
<tr>
<th>Overall CPU %</th>
<th>Kernel %</th>
<th>Read MB/s</th>
<th>Write MB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability Group Only</td>
<td>59.9%</td>
<td>2.99%</td>
<td>26.09</td>
</tr>
<tr>
<td>With ScaleArc</td>
<td>60.2%</td>
<td>4.98%</td>
<td>23.86</td>
</tr>
</tbody>
</table>

As the bottleneck inside the application for repetitive database reads was reduced through the use of ScaleArc database-layer caching, the performance consumption footprint within Windows showed greater utilization of the database engine as more data was modified. The elevated activity from data write operations is an expected and positive outcome, as more of the database-layer resources were dedicated to operations that should not be cached - data modifications. The database engine reflects the net shift in workload patterns, and the ScaleArc dashboard shows the effectiveness of the query caching patterns.

The end result showed ScaleArc delivered significant performance improvements to an application with no application or database-layer code changes or modifications to the way the application was being used. Arguably, most applications can positively benefit from these improvements in performance and availability. ScaleArc can be used to improve the performance and availability of even the most demanding business-critical application, especially those that are proven to operate with inefficient data handling.

Leverage ScaleArc to help IT deliver performance and availability levels on par with your end users’ expectations of how IT *should* perform.

**The end result showed ScaleArc delivered significant performance improvements to an application with no application or database-layer code changes or modifications to the way the application was being used. Arguably, most applications can positively benefit from these improvements in performance and availability.**
ABOUT THE AUTHOR
David Klee is a Microsoft MVP and VMware vExpert with a lifelong passion for technology. David spends his days focusing on the convergence of data and infrastructure as the Founder of Heraflux Technologies. His areas of expertise are virtualization and performance, datacenter architecture, and risk mitigation through high availability and disaster recovery. David speaks at a number of national and regional technology related events, including the PASS Summit, VMware VMworld, IT/Dev Connections, SQL Saturday events, SQL Cruise, PASS virtual chapter webinars, and many SQL Server User Groups.

ABOUT HERAFLUX TECHNOLOGIES
Heraflux Technologies is a premier IT consultancy composed of expert database, cloud, virtualization and infrastructure specialists. Heraflux provides industry-leading consulting and training on-site, remotely, or a hybrid approach to best meet your data milestones and goals. Whether you require an ongoing data adviser, a custom training class for your IT staff, a strategic plan for public cloud adoption, or a data and infrastructure health check (or anything in between), we’ve got you covered.

ABOUT SCALEARC
ScaleArc’s database load balancing software enables consumer-grade apps - that is, apps that are never down, always fast, and scale anywhere. Customers use ScaleArc to unlock the power of modern databases without writing a single line of code. ScaleArc helps organizations large and small increase revenue and save money by eliminating downtime, increasing performance, and avoiding wasteful application recoding.