

# Regaining Control of the Database Stack: The New Imperative for Today's DBA

## EXECUTIVE SUMMARY

The high availability of business critical applications rests solely on the shoulders of a database administrator in most organizations. And it's no small task. Charged with protecting data, keeping applications running, managing traffic, resolving issues and conducting ongoing maintenance, DBAs are the glue holding a back-end world together. They're typically only noticed when any part of their responsibilities falls short, even minimally, but never when they're juggling everything successfully in a way that's transparent to users.

ScaleArc's database load balancing software gives DBAs better, easier control to manage and run a database stack assuring high availability. And unlike some other heroic but potentially risky methods DBAs are already using to maintain HA, with ScaleArc, there's no risk of data loss, no downtime and no application changes. These five most significant capabilities will give database control back to the DBAs:

### 1. Transactional integrity: Failovers happen

Some planned; some unplanned. Either way, DBAs must avoid data corruption that could potentially arise from transactions that are still in motion when a failover occurs. With ScaleArc, if a query dies mid-transaction, the application is alerted to retry. Incoming queries are held back while the switch to a new primary is made. Then the queries are released to be handled by the new server. All this happens automatically with no manual intervention and no business interruption.

### 2. Zero-downtime, anytime maintenance

Any database downtime triggers a deleterious domino effect. The database fails, customers get errors, customer confidence wanes, sales decrease then the bottom line suffers. But doing maintenance on a database is absolutely necessary and until now, downtime was an unfortunate side effect of maintenance. ScaleArc simplifies maintenance offering two ways to conduct routine procedures without downtime. DBAs can gradually reduce load to any given server using load balancing bias and perform diagnostics or backups. Or they can immediately reduce load in a more dramatic way by reducing the number of server connections, helpful for storage or back-end work that requires faster load reduction.

### 3. Workload flow control

DBAs need appropriate controls for database access to dictate which processes deserve the most juice and when. For example, end-of-month number crunching queries should trump any other routine processes running simultaneously. But configuring rules

to assure the pecking order stays intact is complicated and risky. ScaleArc makes it easy to apply appropriate controls using query routing that directs reporting or business intelligence workload to servers exclusively dedicated to reporting. In addition, ScaleArc syncs with Active Directory so permissions that control who should have access to which data are baked in, not bolted on.

### 4. Day one usability

Testing and deployment of database changes is tedious and time consuming. For the best return on the investment, DBAs need a faster, easier way to accomplish changes without compromising applications or data. The abstraction layer that ScaleArc inserts between the servers and the applications does just that. It's fully compatible, easy to deploy and test, and offers valuable read/write split so moving to a SQL Server version that uses AlwaysOn is far easier.

### 5. Faster troubleshooting

Which databases are consuming the most resources and why? DBAs troubleshooting performance problems ask themselves this question daily but until now haven't had visibility to easily solve problems. ScaleArc offers real-time analytics with its software allowing DBAs to spot traffic trends and proactively resolve issues. There's no more old-fashioned, labor-intensive log collection and comparison to determine whether it's the app developers or the ops teams that need to step it up.

[Read more about how to be a database hero with help from ScaleArc in the full white paper. »](#)