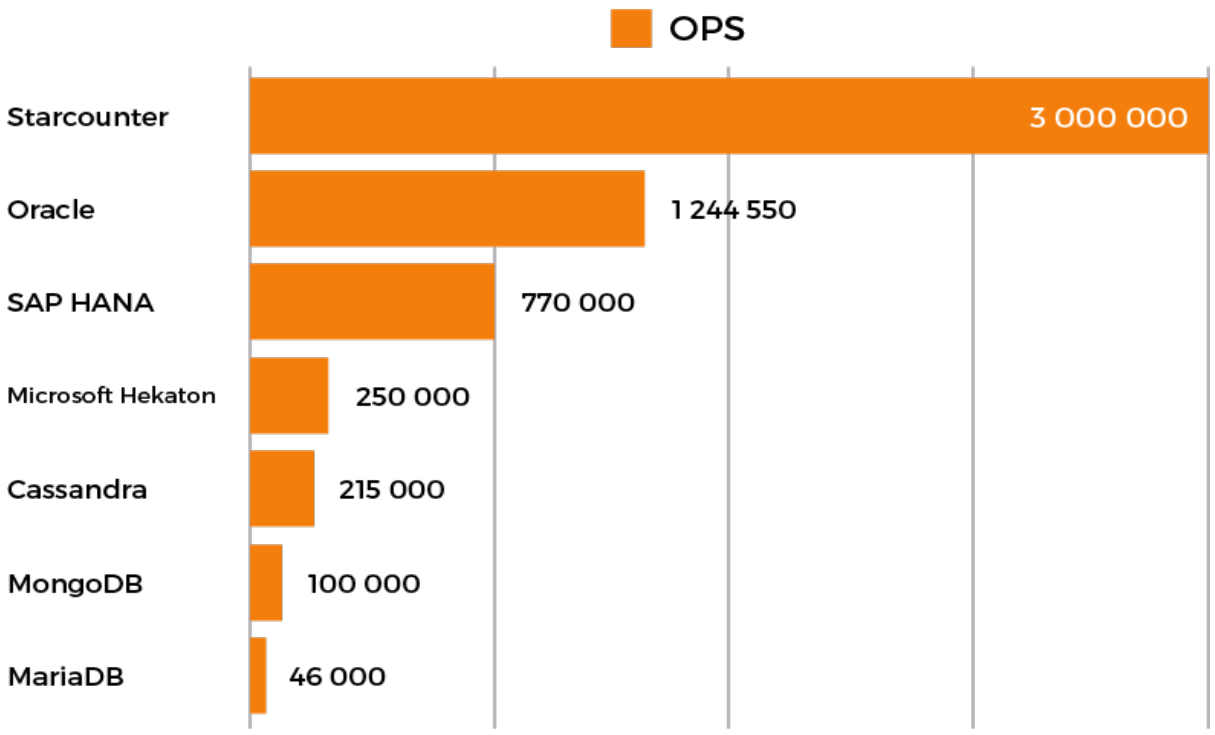
STARCOUNTER PERFORMANCE

Starcounter in-memory application platform ships with high performance ACID database capable of millions read/write transactions per second on a single server. Running a single machine instead of server fleet for equal performance reduces costs for hardware, licenses and maintenance, and good for ecology.

Considering patterns and workloads met in business applications, Starcounter is ready out of the box for serving hundreds thousands of simultaneous users, complex data dependencies and event chains.

In addition, Starcounter provides efficient networking layer, including the built-in Web server, which allows leveraging hardware resources and transfer channels with a very good efficiency. This power brings new opportunities for building modern Web applications with rich GUI and high responsiveness using Starcounter native .NET API and Starcounter client-side library for data binding, hereby substantially reducing lines of code and system complexity.



Benchmark:

We evaluate Starcounter's performance with a simple retail demo. The demo runs a flow of operations typical for business apps: a mix of analytic reads and highly concurrent writes; each operation entails an ACID database transaction involving multiple data records. In the demo, we arbitrary bind 1.5 million customers to 0.5 million accounts and generate a workload of 5% transfers and 95% reads.

Transfer operation moves some money from a source account to a destination account given. If the source account doesn't have enough money, then transfer is cancelled with corresponding result returned.

Read operation returns a total sum on all accounts of a given customer. For each operation, account numbers and sums are generated randomly.

Result:
Starcounter performs at 3 million full ACID operations per second.

Setup:
In our retail demo, we implement a full-stack scenario using Starcounter. It means that the entire client-server-database-server-client trip is presented: real clients approach server app over network by calls to the app's API, while app itself operates the database by invoking ACID transactions and SQL queries. We connect client machines to a server with a Gigabit Ethernet LAN. The client machines continuously invoke transfer and read operations by REST API calls to the server app. Server machine is Xeon E5-2680v2 (10 cores) with 128 Gb RAM and an SSD.

Comparison to other databases:
Among performance results officially provided by major database vendors and enterprise benchmark performers, we have chosen those which showed the best numbers (operations per second) while invoking maximum core database features. In any ambiguous case, we adjusted the numbers towards better values. For every benchmark selected, we estimated number of CPU cores involved, and put the final results in a diagram above. This form, in result, allows comparing total cost of ownership with performance offered.

Methodology
We used three rules for choosing benchmarks comparable to our retail demo:
ACID benchmarks are preferred to non-ACID benchmarks;
Benchmarks using SQL interface are preferred to benchmarks testing pure storage engine;
Benchmarks with transactions involving different records (complex dependencies by data, hard to scale) are preferred to benchmarks with transactions involving only one record (no dependencies by data, easy to scale).

Benchmark sources: Oracle NoSQL | SAP HANA | Microsoft Hekaton | Cassandra | MariaDB / FusionIO