copenhagen goto; conference

NewSQL Database for New Real-time Applications

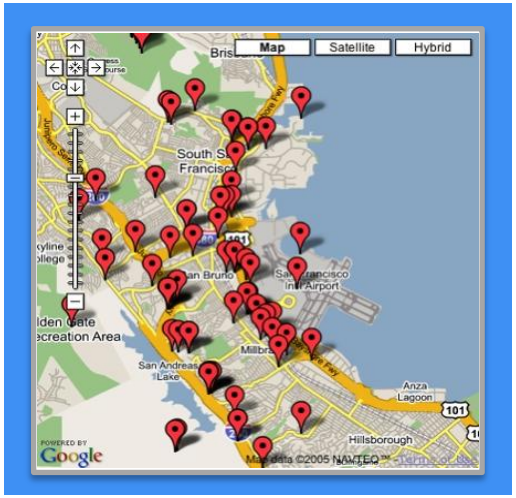
PhD Peter Idestam-Almquist
Starcounter AB

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

STARCOUNTER

New real time applications



- Millions of simultaneous online users.
- High degree of concurrency.
- Interactive applications (95% reads; 5% writes).

ACID TRANSACTIONS/SECOND

Starcounter database

- We claim you can run the database of a large webshop like amazon.com on a single off-the-shelf server using Starcounter.

Old products:
slow, complex, expensive.

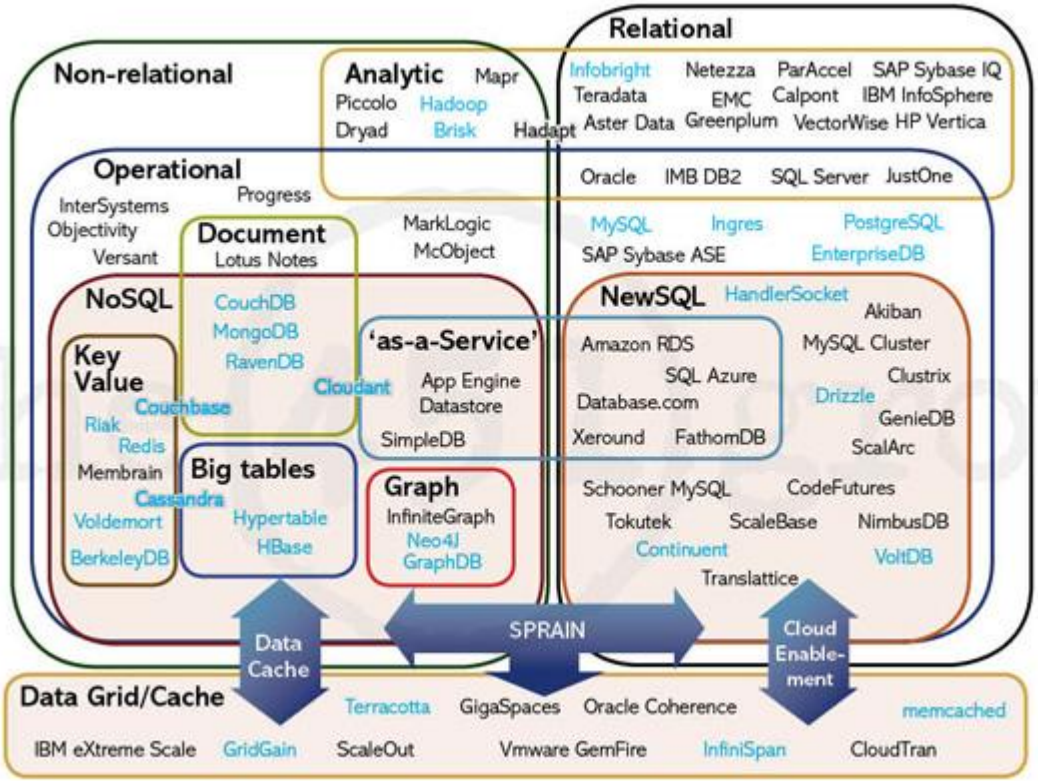
The new generation:
easy, fast, game changing.

STARCOUNTER

Outline

- ✓ Positioning
- ✓ Consistency
- ✓ Performance
- ✓ Code examples

Database landscape



Matthew Aslett, The 451 Group

STARCOUNTER

NoSQL and NewSQL

NoSQL:

- New breed of non-relational database products;
- Rejections of fixed table schema and join operations;
- Designed to meet scalability requirements of distributed architectures;
- And/or schema-less data management requirements.

NewSQL:

- New breed of relational database products;
- Retain SQL and ACID;
- Designed to meet scalability requirements of distributed architectures;
- Or improve performance so horizontal scalability is no longer a necessity.

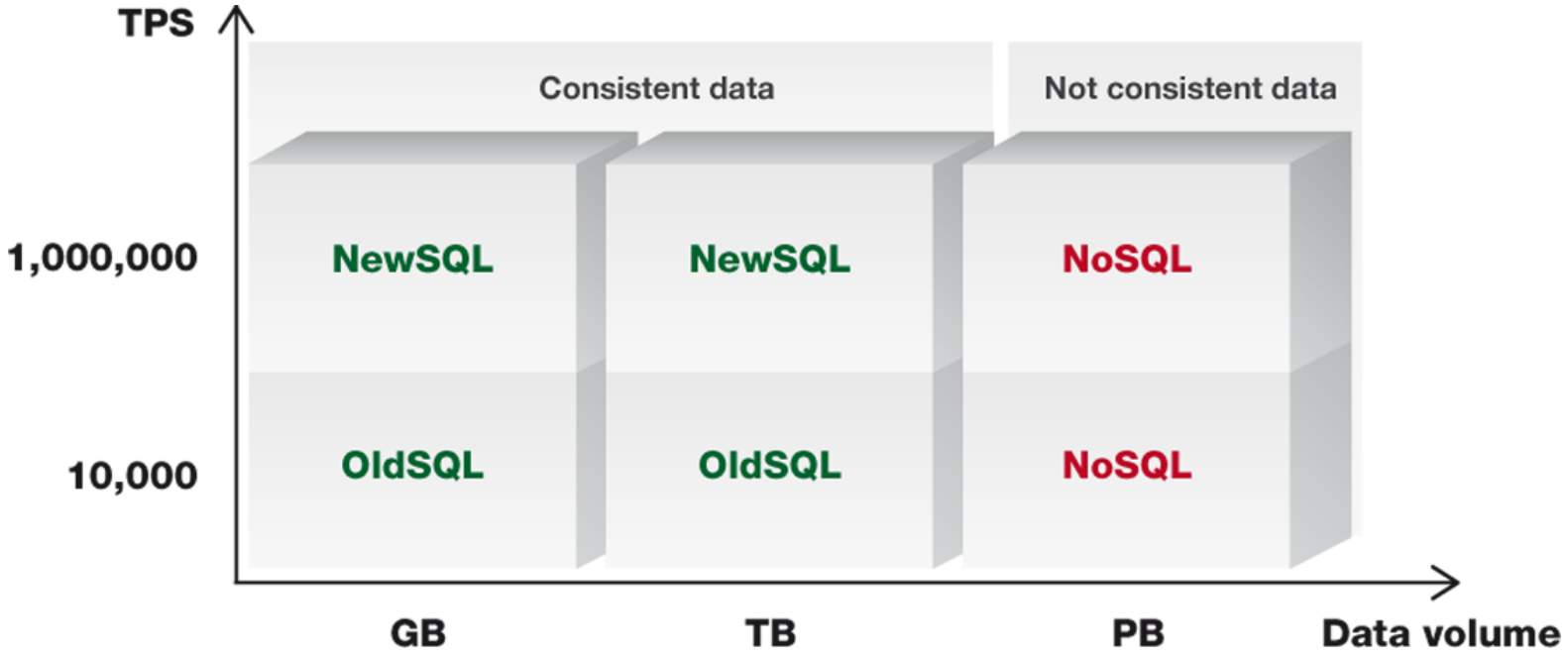
Matthew Aslett, The 451 Group

Data management challenge

- You have:
  - big data volumes,
  - many simultaneous online users (updating data).
- You want:
  - high performance (throughput and latency),
  - consistent data.

STARCOUNTER

Your alternatives



STARCOUNTER

Outline

- ✓ Positioning
- ✓ Consistency
- ✓ Performance
- ✓ Code examples

ACID transactions

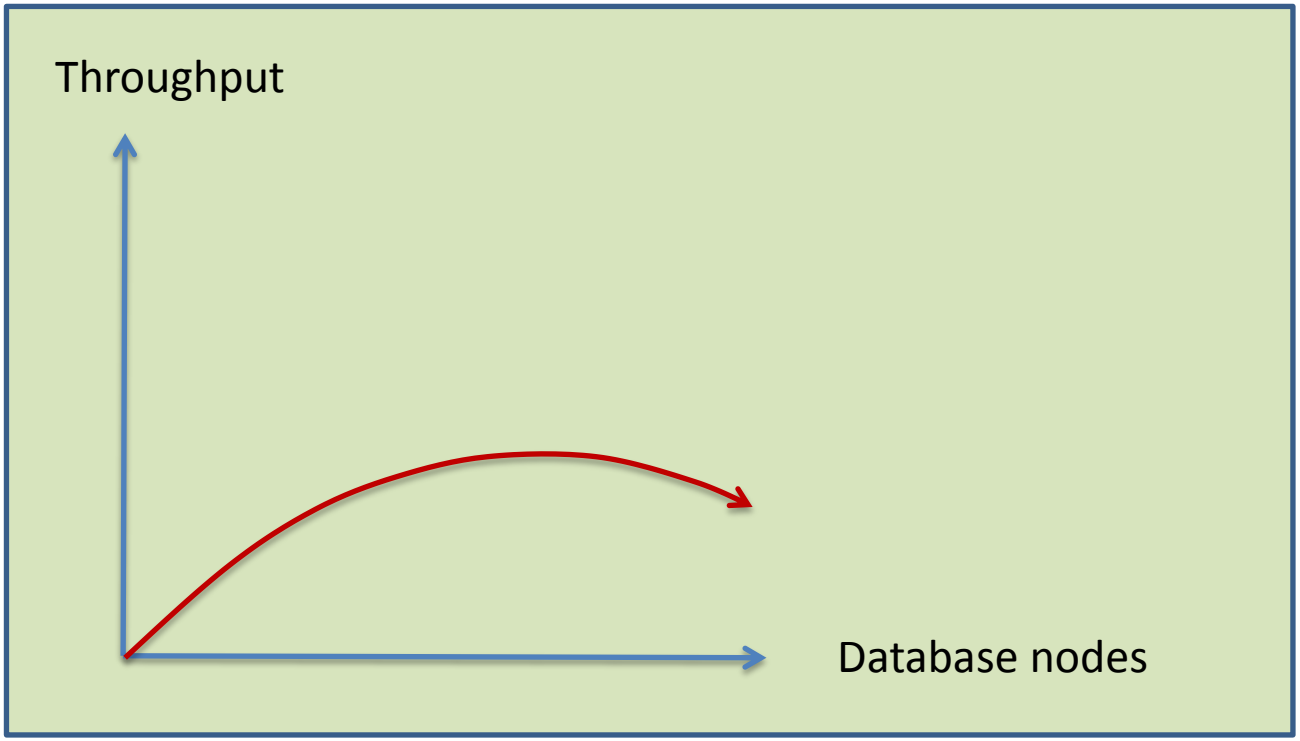
- ACID transactions guarantee consistent data:
- Atomicity - either entire transaction or nothing;
- Consistency - valid state before and after transaction;
- Isolation - no transaction interfers with another transaction;
- Durability - committed transactions will remain after crash or power loss.

Isolation levels

- Different isolation levels to trade off between performance and consistency:
- Read uncommitted - dirty reads;
- Read committed - non-repeatable reads;
- Repeatable reads - phantom reads;
- Serializable (required for ACID)
  - as executing transactions sequentially;
  - often relaxed to snapshot isolation.

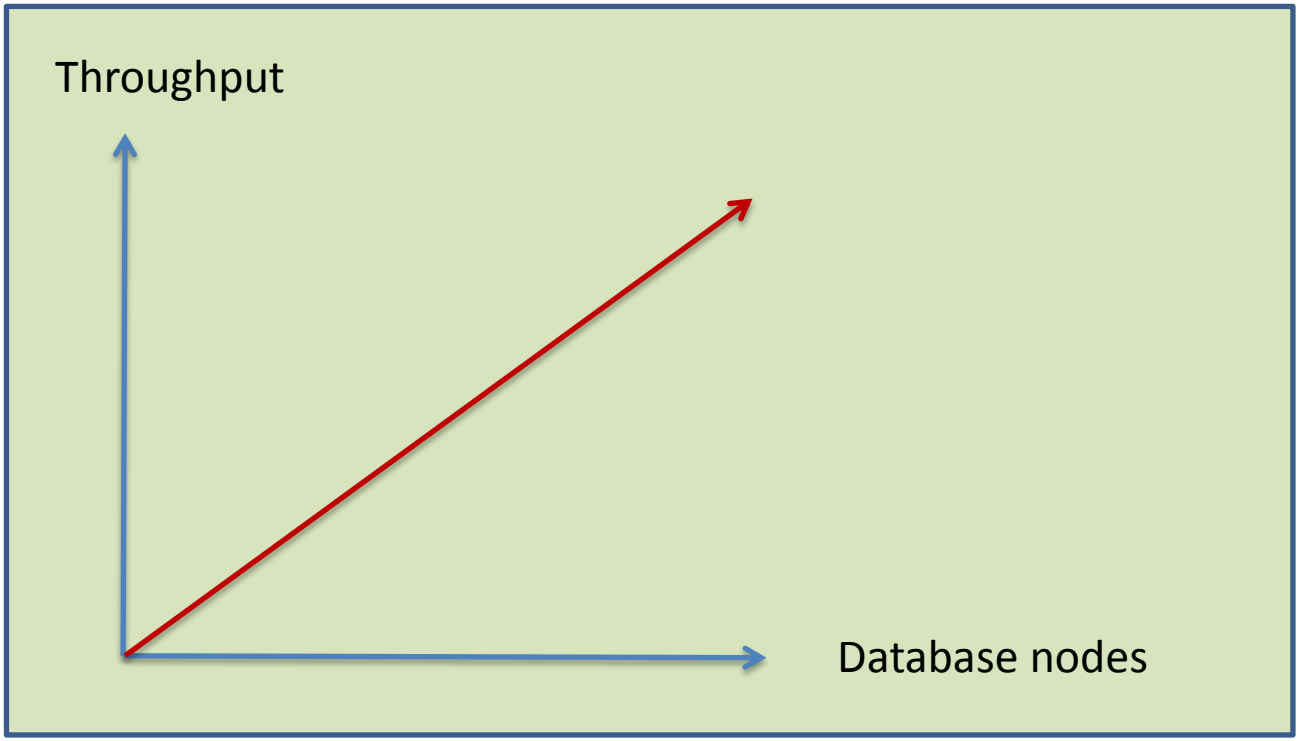
Scale out and global consistency

Distributed transactions (two-phase commits) gives (high degree of) global consistency, but do not scale.



Scale out and local consistency

Horizontal scaling (shared-nothing) scales linearly, but gives no global consistency (only local consistency).



CAP theorem (Brewer)

- A distributed system can satisfy two but not three out of:
- Consistency - all nodes see the same data at the same time;
- Availability - every request recieves a response whether it succeeded or failed;
- Partition tolerance - operates despite of message loss or failure of part of the system.

STARCOUNTER

Our conclusion

- You cannot achieve both high performance and consistency by scaling-out.
- To achieve both high performance and consistency you should:
  - **Scale-in** - execute all transactions in RAM (performance) on the same computer (consistency);
  - **Scale-up** - get a powerful multi-core server with a lot of RAM (performance).

Do I need ACID?

- When dealing with business critical data like stock quantities or money.
- For multi-user applications transactional conflicts will occur and need to be managed by:
  - database (DBMS),
  - application (hard for developers),
  - end user: "Sorry we have just sold you a product we already have sold to someone else".

STARCOUNTER

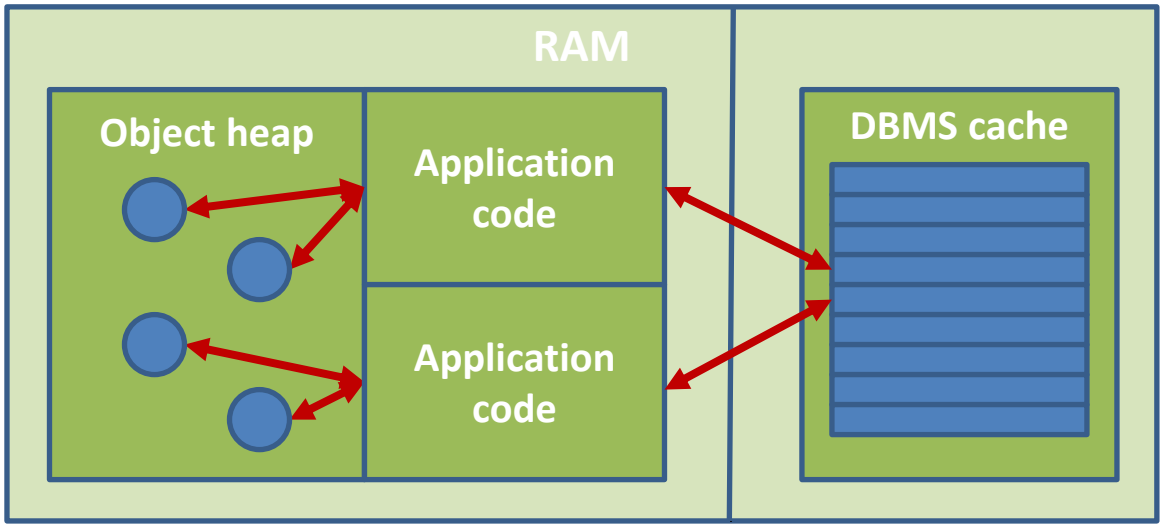
Outline

- ✓ Positioning
- ✓ Consistency
- ✓ Performance
- ✓ Code examples

Application performance

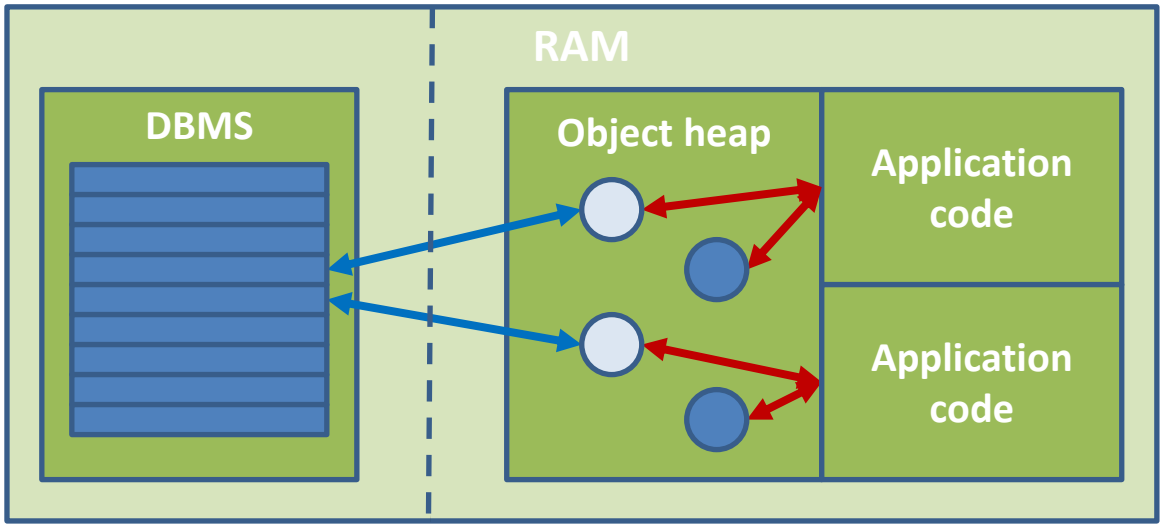
- For the performance of an application the interaction between the application and the database is crucial.
- Our invention: VMDBMS,
  which integrates the application runtime (virtual machine - VM) and the database management system (DBMS).

Traditional DBMS



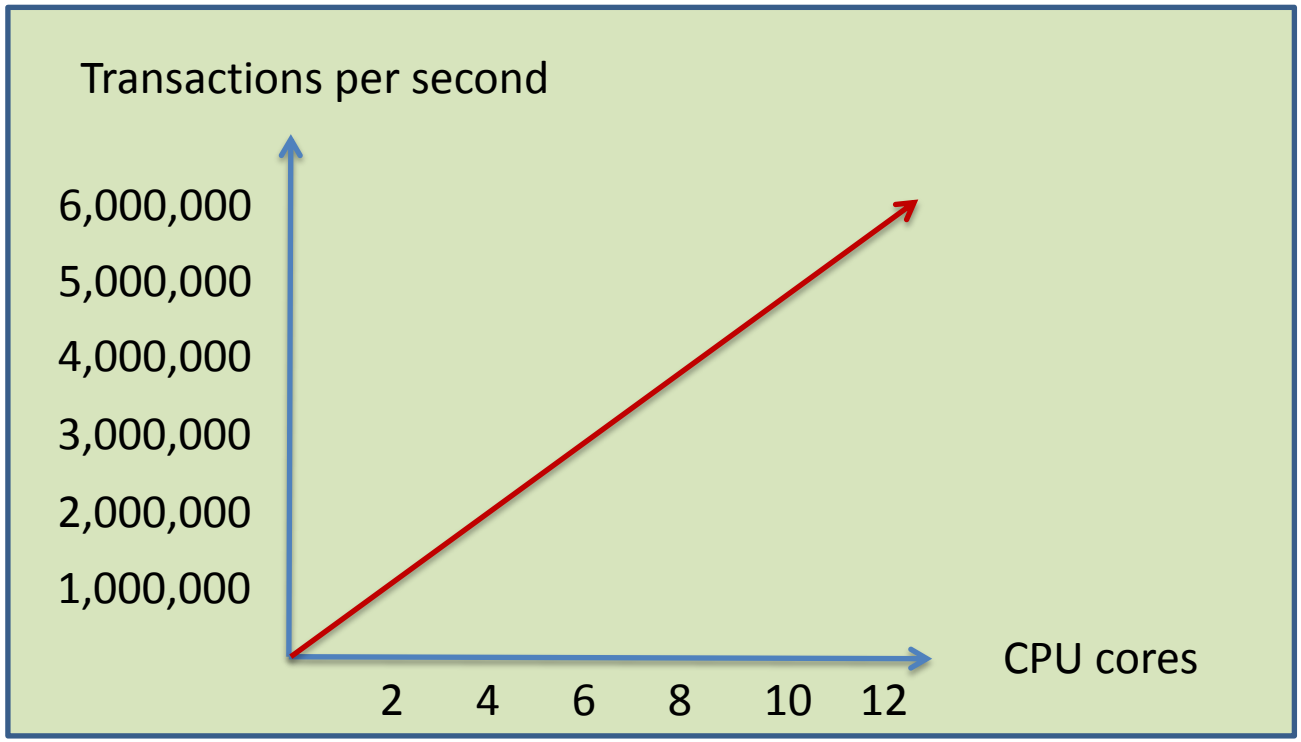
- Data is copied back and forth between the application (heap) and the database.
- Business objects store temporary local copies of the data.

Our invention - VMDBMS



- Data is **not moved** between the database and the application (heap).
- Data resides only in the database, and the business objects have no local copies of the data.

Starcounter read performance (SQL)



Starcounter read performance (SQL)

- 500,000 read-only ACID transactions per second and core for SQL queries.
- Scales almost linearly on the number of cores.

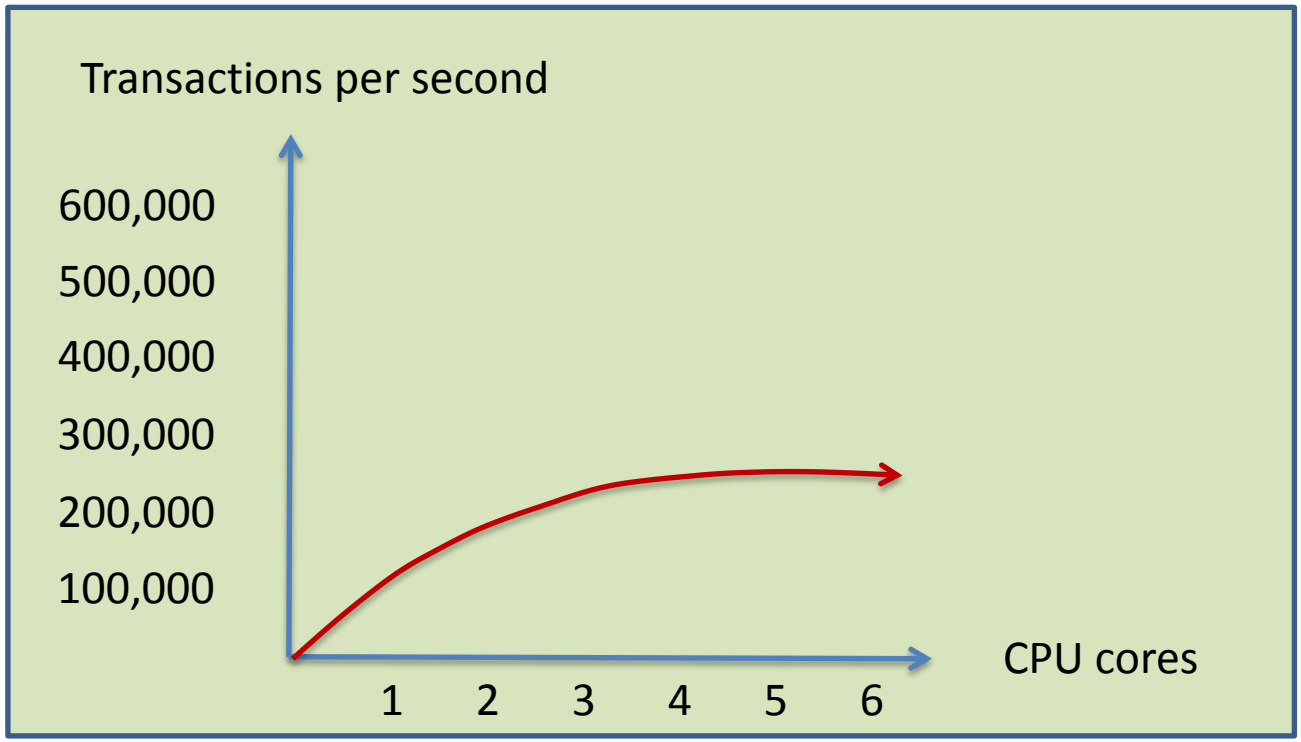
```
Int32 productId;
Int32 quantity;
...
Db.Transaction(delegate
{
  Product product =
    Db.SQL("select p from Product p where p.Id = ?",
           productId).First;
  quantity = product.Quantity;
})
```

Starcounter read performance (ref)

- You can traverse four millions of nodes in an object graph in a second (using one core).

```
Node cursor;
...
Db.Transaction(delegate
{
  while (cursor.Next != null)
  {
    cursor = cursor.Next;
  }
})
```

Starcounter write performance



STARCOUNTER

Starcounter write performance

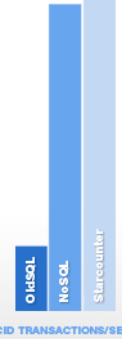
- 100,000 read-write ACID transactions per second on one core for SQL queries.
- Do not scale on the number of cores.
- Max 250,000 ACID transactions per second.

```
Int32 productId;
...
Db.Transaction(delegate
{
  Product product =
    Db.SQL("select p from Product p where p.Id = ?",
           productId).First;
  product.Quantity = product.Quantity - 1;
})
```

STARCOUNTER

A very large webshop

- 1 billions of orders per year.
- 10 billions of write transactions (400/s).
- 200 billions of read transactions (8,000/s).
- 54 GB order data per year.
- Intel Xeon, 32 cores, 1 TB RAM, 50000 USD.



STARCOUNTER

Outline

- ✓ Positioning
- ✓ Consistency
- ✓ Performance
- ✓ Code examples

Starcounter .NET object API

- Database schema (”create table”): class definitions inheriting Starcounter.Entity.
- Create object (”insert”): native ”new” operator.
- Modify object (”update”): native assignment operator (”=”).
- Delete object (”delete”): use object method Delete().
- Query objects (”select”): SQL(”select ...”).

Database schema

```
using Starcounter;

public class Employee : Entity
{
    public string Name;
    public Nullable<DateTime> HireDate;
    public decimal Salary;
    public Department Department;
    public Employee Manager;
    ...
}
```

Create object

```
public class Employee : Entity
{
    ...

    public Employee() { }
}

...

Employee e = new Employee();
```

Modify object

```
...
Department d = new Department();
...
Employee e = new Employee();

e.Name = "John";
e.HireDate = null;
e.Salary = 20000;
e.Department = d;
```

Delete object

```
...
Employee e = new Employee();

e.Name = "John";
e.HireDate = null;
e.Salary = 20000;
e.Department = d;

e.Delete();
```

Starcounter SQL

- Starcounter SQL follows SQL92 standard
- Object references:

```
SELECT e FROM Employee e
```

- Path expressions:

```
SELECT e.Name, e.Department.Name
FROM Employee e
```

- Compare to:

```
SELECT e.Name, d.Name
FROM Employee e
JOIN Department d
ON e.DepartmentId = d.Id
```

SQL in code

```
string query = "SELECT e FROM Employee e";
foreach (Employee emp in Db.SQL(query))
  emp.PrintCV();
```

```
string query = "SELECT e FROM Employee e
  WHERE e.FirstName = ?";
Employee emp =
  Db.SQL(query, "John").First;
emp.PrintCV();
```

STARCOUNTER

One-to-many relations

```
public class Employee : Entity
{
  public Employee Manager;
  public IEnumerable Staff
  {
    get {
      string query = "SELECT e FROM
        Employee e WHERE e.Manager = ?";
      return Db.SQL(query, this);
    }
  }
}
```

STARCOUNTER

Transactions

```
Db.Transaction(delegate()
{
   string query = "SELECT e FROM Employee
     e WHERE e.Name = ?";
   Employee emp =
     Db.SQL(query, "John").First;
   if (emp != null)
     emp.Name = "Bill";
});
```

Starcounter database

- Transactional database (OLTP).
- ACID compliant.
- High performance (500,000 TPS per core).
- Robust (previous versions used in production for 5 years).
- Reliable (replication and full checkpoint recovery).
- In-memory (transactions secured on disk).
- SQL query support.
- Native (.NET) object API.
- New invention: VMDBMS.

STARCOUNTER

Questions ?

More info on www.starcounter.com.