ATTUNITY

AvePoint®
Unleashing the Power of SharePoint™

CloudProTEAM

CodeFluent
Entities

AIT®

time cockpit

INFRAGISTICS™
DESIGN / DEVELOP / EXPERIENCE

ppedv Training
★★★★★
einfach ausgezeichnet!

SQL
ServerCentral.com

Professional
Developer
Training

QUEST
SOFTWARE®
Simplicity At Work™

SQLSENTRY®

SYNTERGY

telerik
deliver more than expected

Tzunami
Migration Masters

IT&
Dev CONNECTIONS
powered by Microsoft®

Turbo-Boost for Databases – Sharding with LINQ, OData and SQL Azure

Rainer Stropek
software architects gmbh
rainer@software-architects.at

Abstract

Mit ODATA hat Microsoft ein Datenaustauschformat vorgestellt, das sich immer mehr zum Quasistandard vorarbeitet. ODATA = SOA ohne dem Overhead von SOAP. Es stehen mittlerweile Implementierungen auf verschiedenen Plattformen zur Verfügung. In dieser Session zeigt Rainer Stropek die Entwicklung individueller ODATA Provider, über die man eigene Datenstrukturen im ODATA Format zugänglich machen kann.

With ODATA Microsoft offers a data access format that has becomes an industriy standard more and more. ODATA = SOA without the overhead of SOAP. Today Microsoft and other vendors offer implementations of ODATA on various platforms. In this session Rainer Stropek demonstrates how to implement a custom ODATA provider that is tailored to specific needs.

Introduction

- software architects gmbh
- Rainer Stropek

  Developer, Speaker, Trainer

  MVP for Windows Azure

  rainer@timecockpit.com

  twitter FOLLOW ME @rstropek

http://www.timecockpit.com

http://www.software-architects.com

AIT®

time cockpit

CloudProTEAM

Introduction

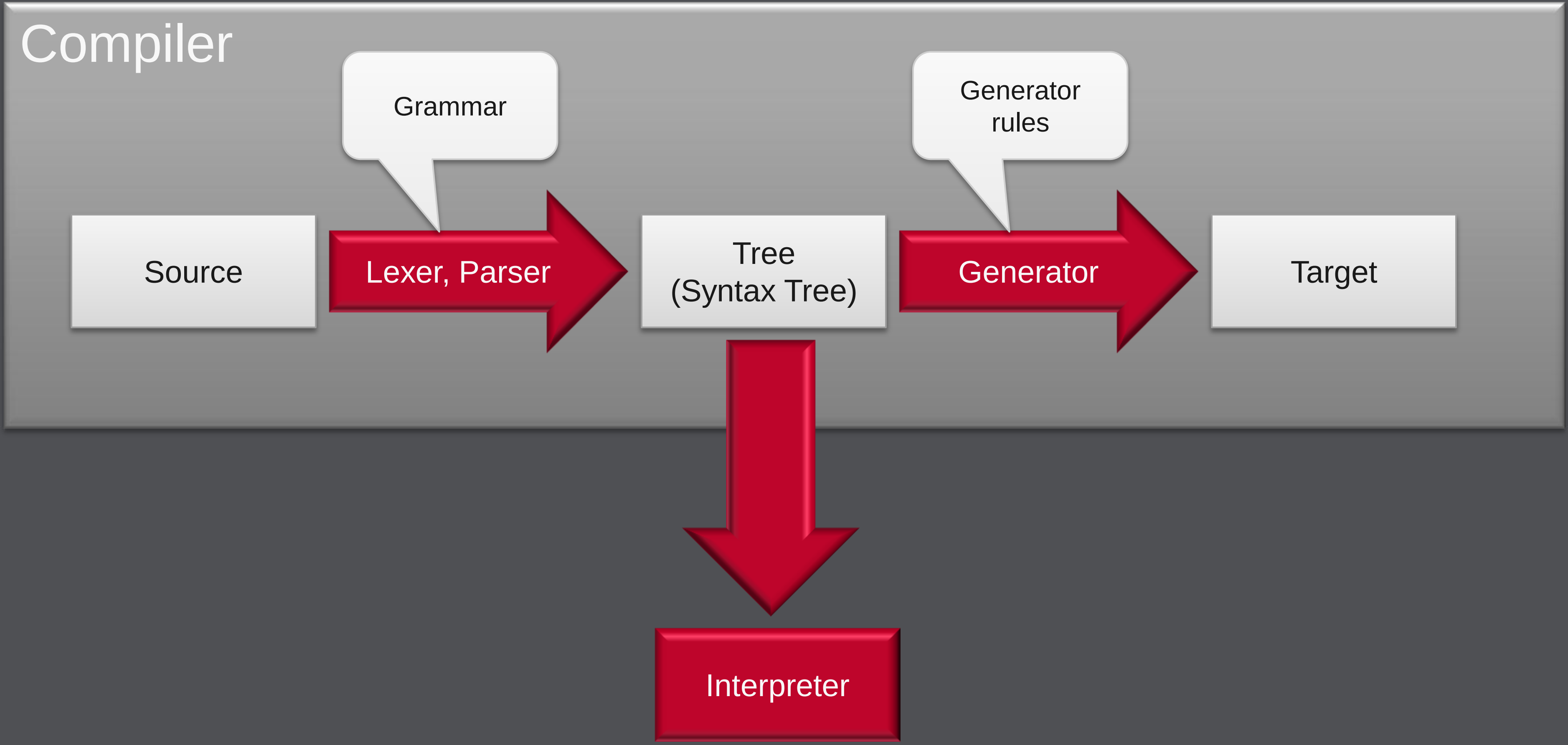
- You all know what **OData** is?
  - Short demo to introduce OData?

- You are all familiar with LINQ & **`IQueryable`**?
  - Short demo to introduce `IQueryable`?

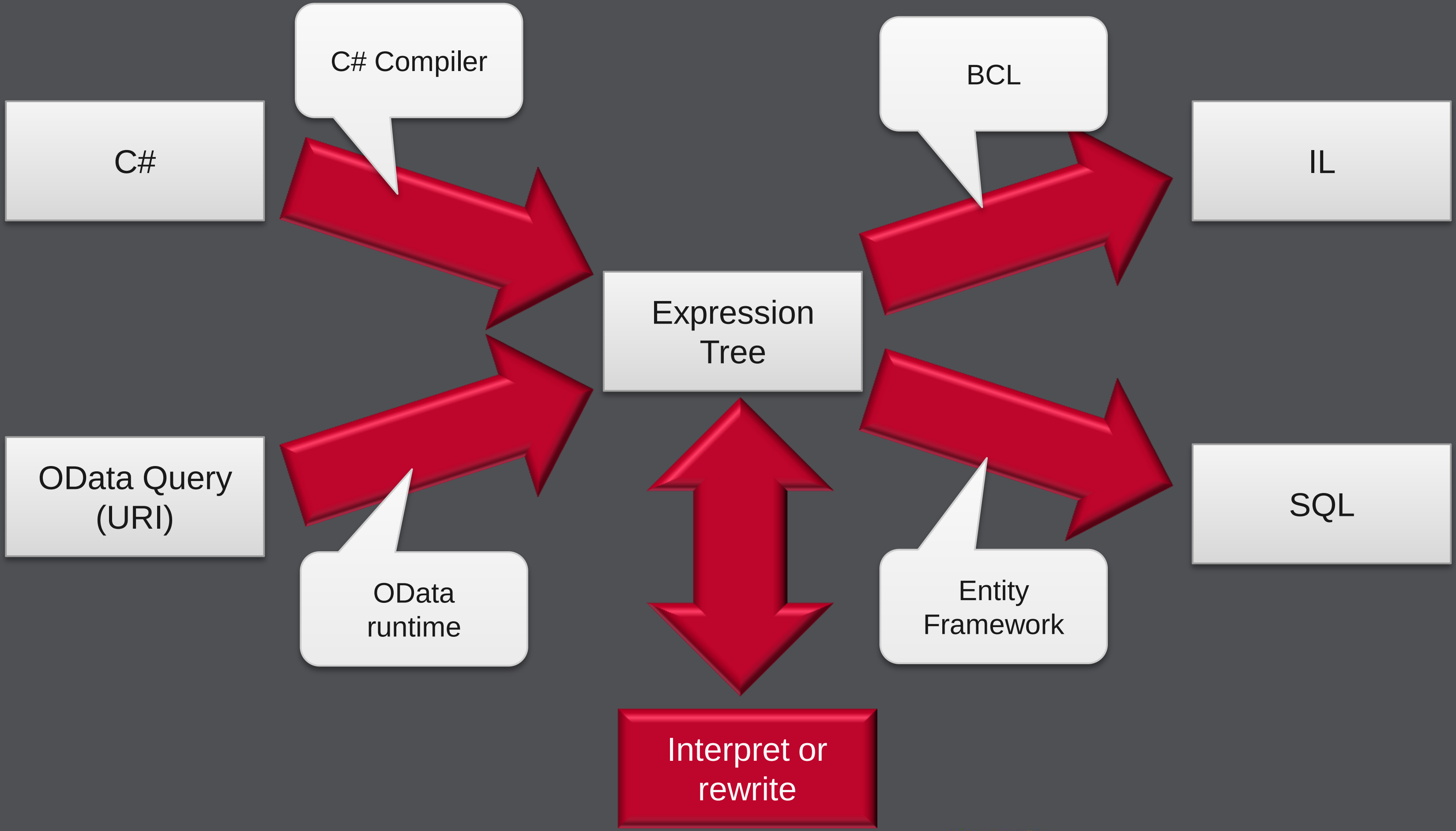
What is the basic idea of OData and IQueryable regarding database queries?

ARCHITECTURE

Architecture of C#, OData and IQueryable



Architecture of C#, OData and IQueryable



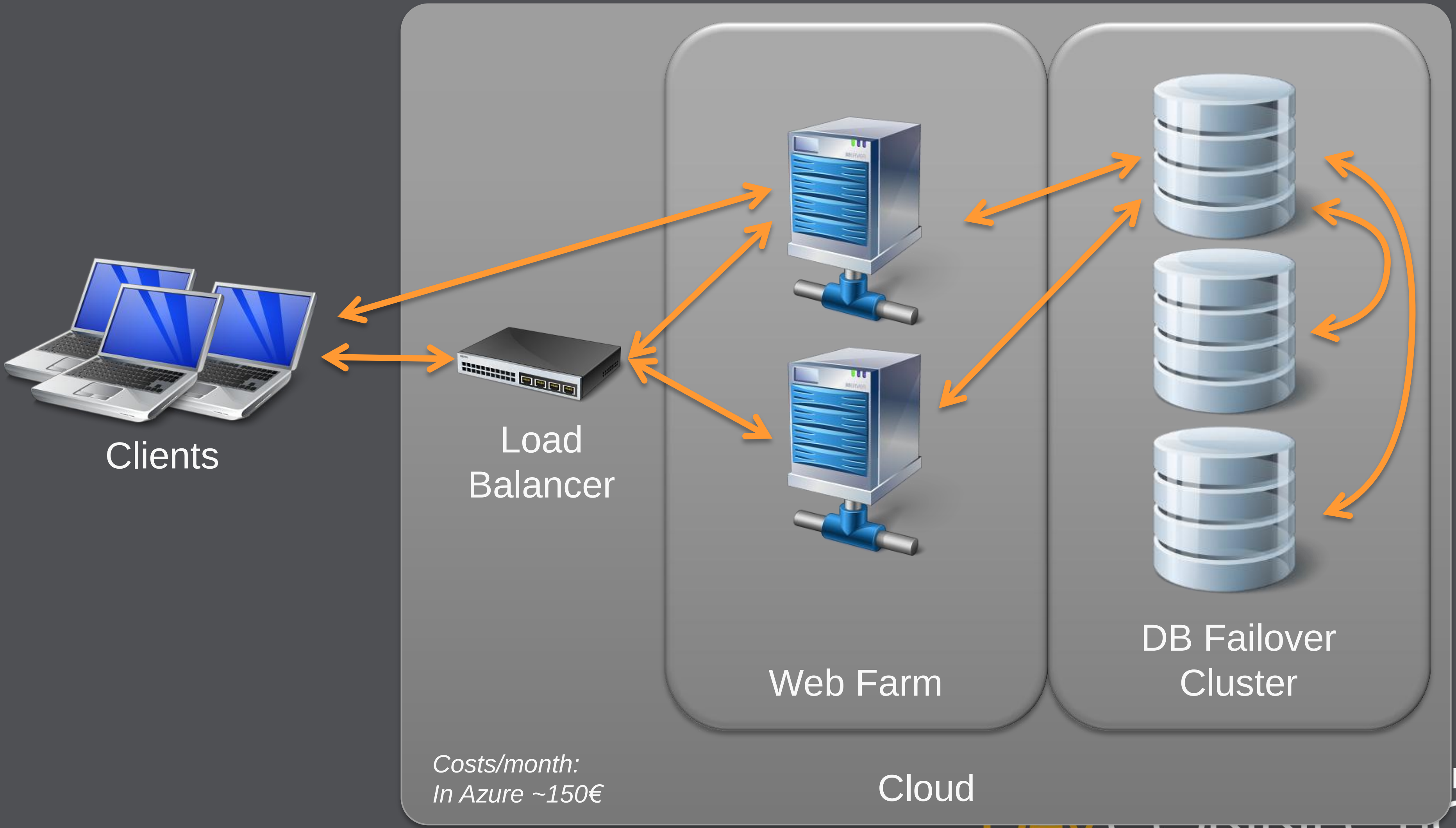
Why could it be necessary to build a custom OData and IQueryable provider?

DEMO SCENARIO

Imagine...

- Based on a real story...
- Current Situation
  - You have a real estate search engine
  - You have lots of data (>=5GB; >=10 Mio. rows)
  - You have lots of users (some hundred thousands queries per day)
  - Users can filter and sort by any column
  - Availability is crucial
- Goal
  - Offer a real estate search SDK for internal (LINQ, EF) and external (OData) use

Typical Architecture



Clients

Load Balancer

Web Farm

DB Failover Cluster

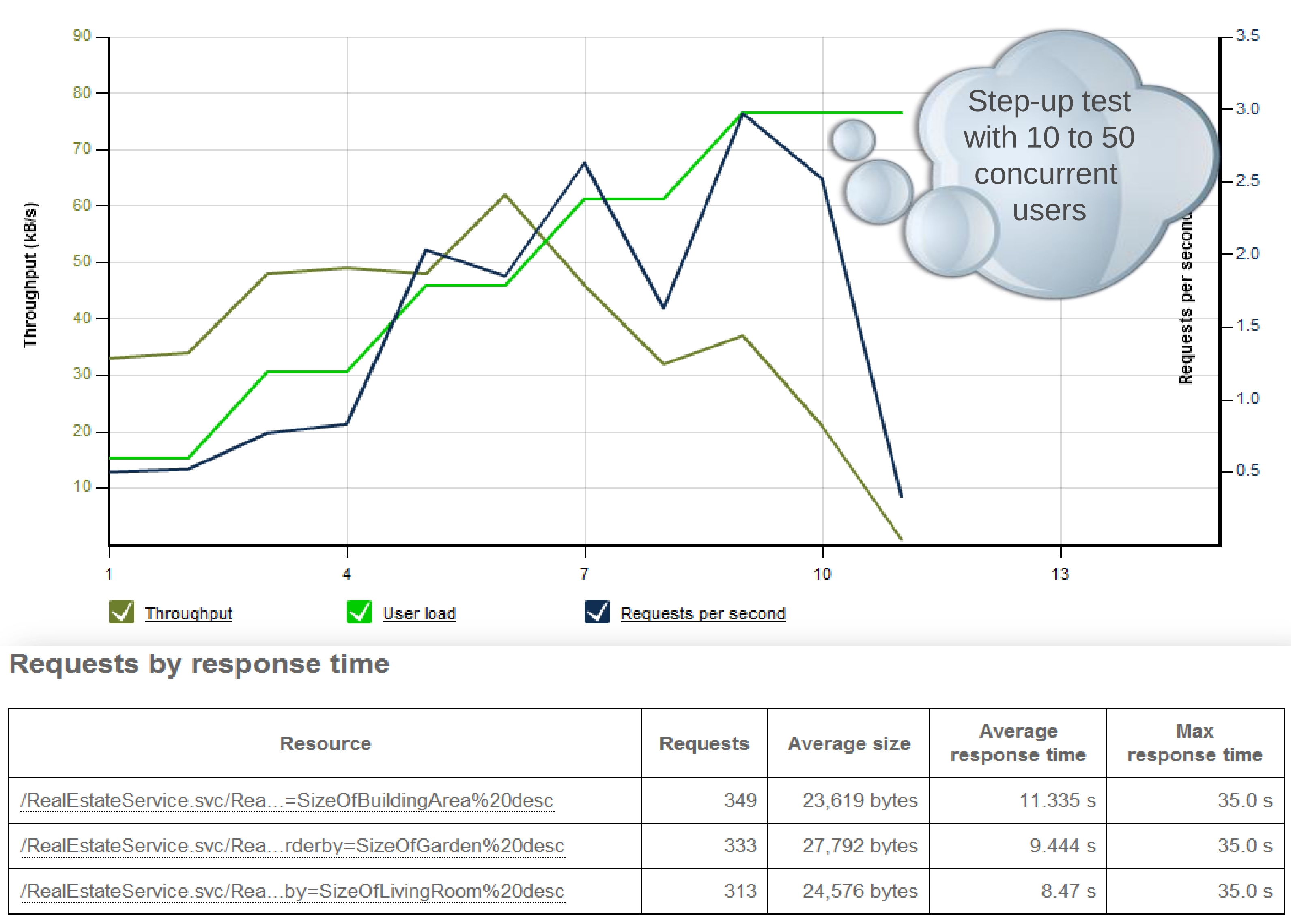
Costs/month:
In Azure ~150€

Cloud

Problems

- Web servers – no bottleneck
  - Scale up & out possible
  - Limited need for CPU
- Database – bottleneck and limited scalability
  - No load balancing cluster
  - Only scale up
  - Limited possiblities for indexing

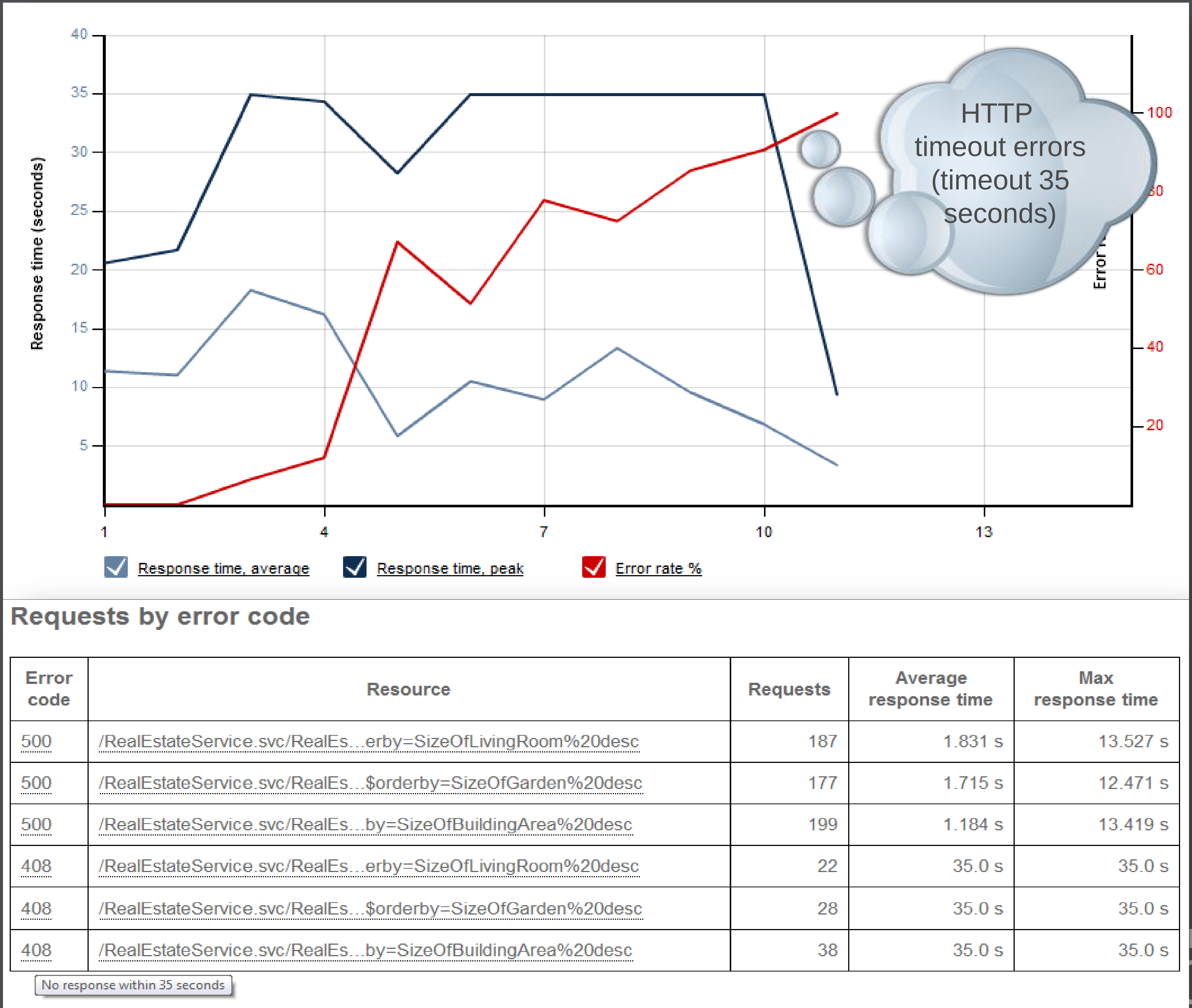
Load Test Results
(Created with LoadStorm.com)



Step-up test with 10 to 50 concurrent users

Requests by response time

Resource	Requests	Average size	Average response time	Max response time
/RealEstateService.svc/Rea…=SizeOfBuildingArea%20desc	349	23,619 bytes	11.335 s	35.0 s
/RealEstateService.svc/Rea…rderby=SizeOfGarden%20desc	333	27,792 bytes	9.444 s	35.0 s
/RealEstateService.svc/Rea…by=SizeOfLivingRoom%20desc	313	24,576 bytes	8.47 s	35.0 s



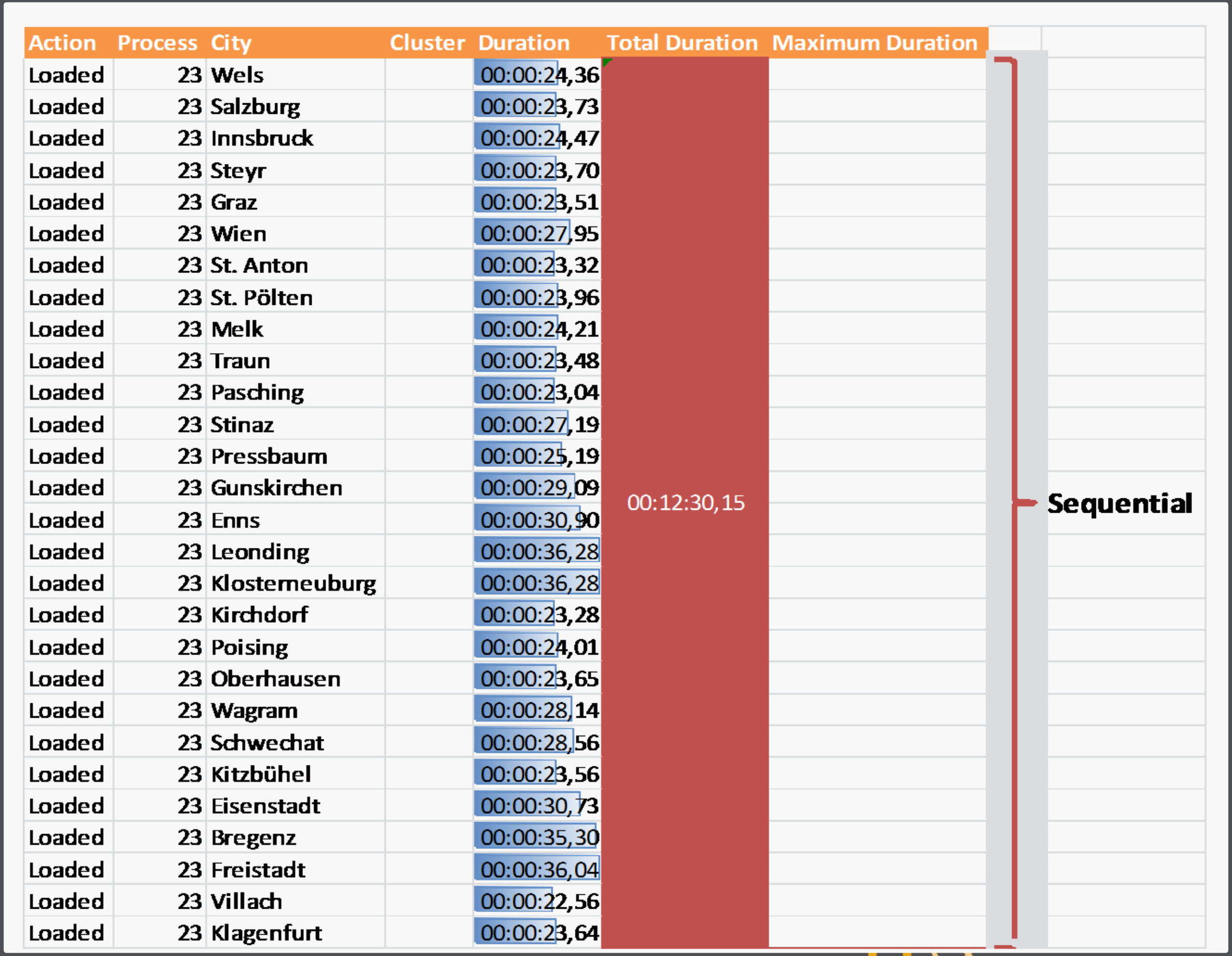
HTTP timeout errors (timeout 35 seconds)

Requests by error code

Error code	Resource	Requests	Average response time	Max response time
500	/RealEstateService.svc/RealEs…erby=SizeOfLivingRoom%20desc	187	1.831 s	13.527 s
500	/RealEstateService.svc/RealEs…$orderby=SizeOfGarden%20desc	177	1.715 s	12.471 s
500	/RealEstateService.svc/RealEs…by=SizeOfBuildingArea%20desc	199	1.184 s	13.419 s
408	/RealEstateService.svc/RealEs…erby=SizeOfLivingRoom%20desc	22	35.0 s	35.0 s
408	/RealEstateService.svc/RealEs…$orderby=SizeOfGarden%20desc	28	35.0 s	35.0 s
408	/RealEstateService.svc/RealEs…by=SizeOfBuildingArea%20desc	38	35.0 s	35.0 s

No response within 35 seconds

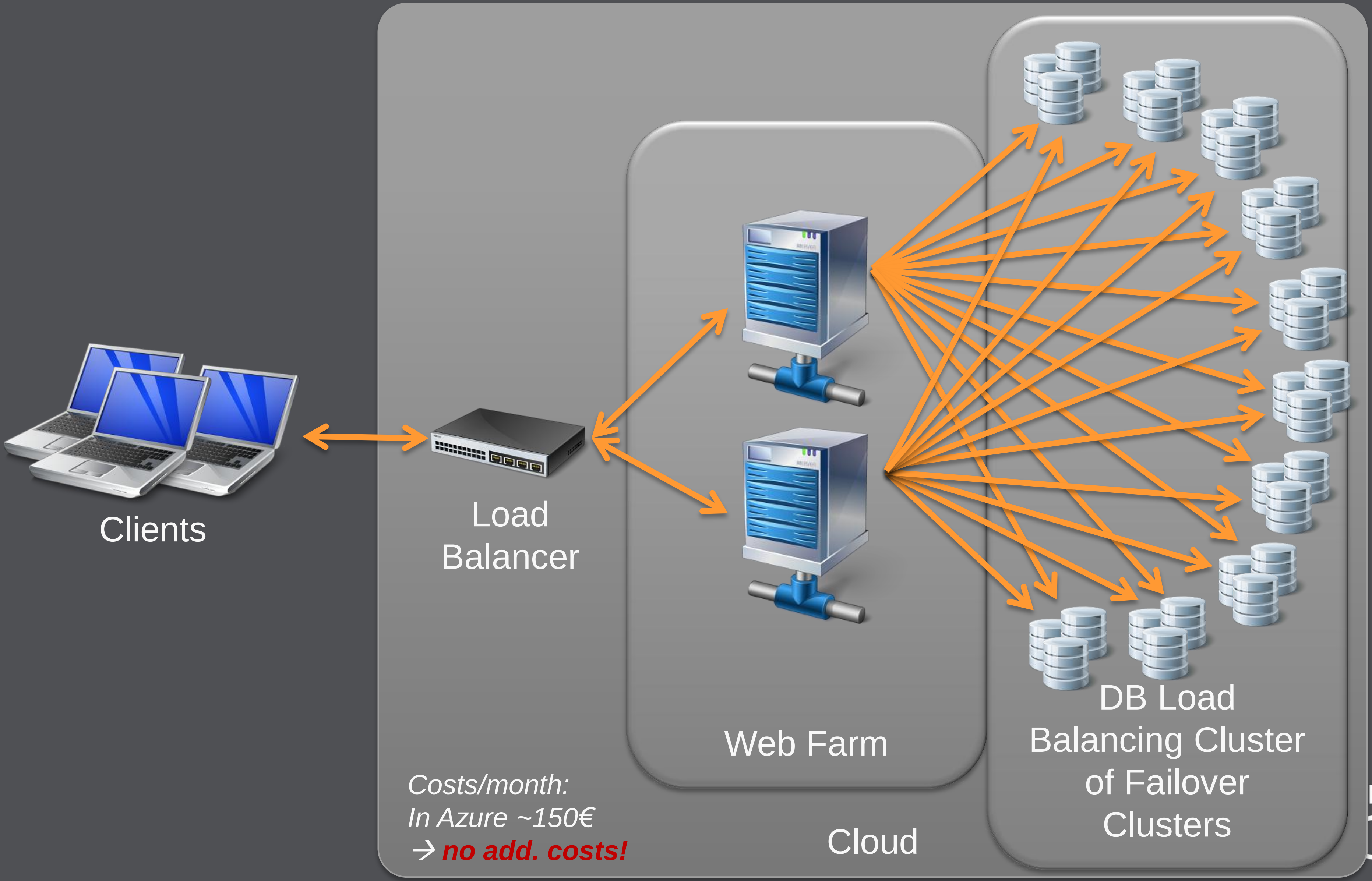
Bad Loading Performance
(Demo Data Generator)



Action	Process	City	Cluster	Duration	Total Duration	Maximum Duration
Loaded	23	Wels		00:00:24,36	00:12:30,15	
Loaded	23	Salzburg		00:00:23,73		
Loaded	23	Innsbruck		00:00:24,47		
Loaded	23	Steyr		00:00:23,70		
Loaded	23	Graz		00:00:23,51		
Loaded	23	Wien		00:00:27,95		
Loaded	23	St. Anton		00:00:23,32		
Loaded	23	St. Pölten		00:00:23,96		
Loaded	23	Melk		00:00:24,21		
Loaded	23	Traun		00:00:23,48		
Loaded	23	Pasching		00:00:23,04		
Loaded	23	Stinaz		00:00:27,19		
Loaded	23	Pressbaum		00:00:25,19		
Loaded	23	Gunskirchen		00:00:29,09		
Loaded	23	Enns		00:00:30,90		
Loaded	23	Leonding		00:00:36,28		
Loaded	23	Klosterneuburg		00:00:36,28		
Loaded	23	Kirchdorf		00:00:23,28		
Loaded	23	Poising		00:00:24,01		
Loaded	23	Oberhausen		00:00:23,65		
Loaded	23	Wagram		00:00:28,14		
Loaded	23	Schwechat		00:00:28,56		
Loaded	23	Kitzbühel		00:00:23,56		
Loaded	23	Eisenstadt		00:00:30,73		
Loaded	23	Bregenz		00:00:35,30		
Loaded	23	Freistadt		00:00:36,04		
Loaded	23	Villach		00:00:22,56		
Loaded	23	Klagenfurt		00:00:23,64		

Sequential

Solution = Sharding



Costs/month:
In Azure ~150€
→ ***no add. costs!***

Solution

- Split one 10GB database into ten 1GB databases („sharding“)
  - No additional costs in Azure
  - Scale out scenario for database
  - In the future: SQL Azure Federation
- Goal
  - Nearly no change for the developer
  - LINQ for internal and OData for external

How can we build a custom OData provider?

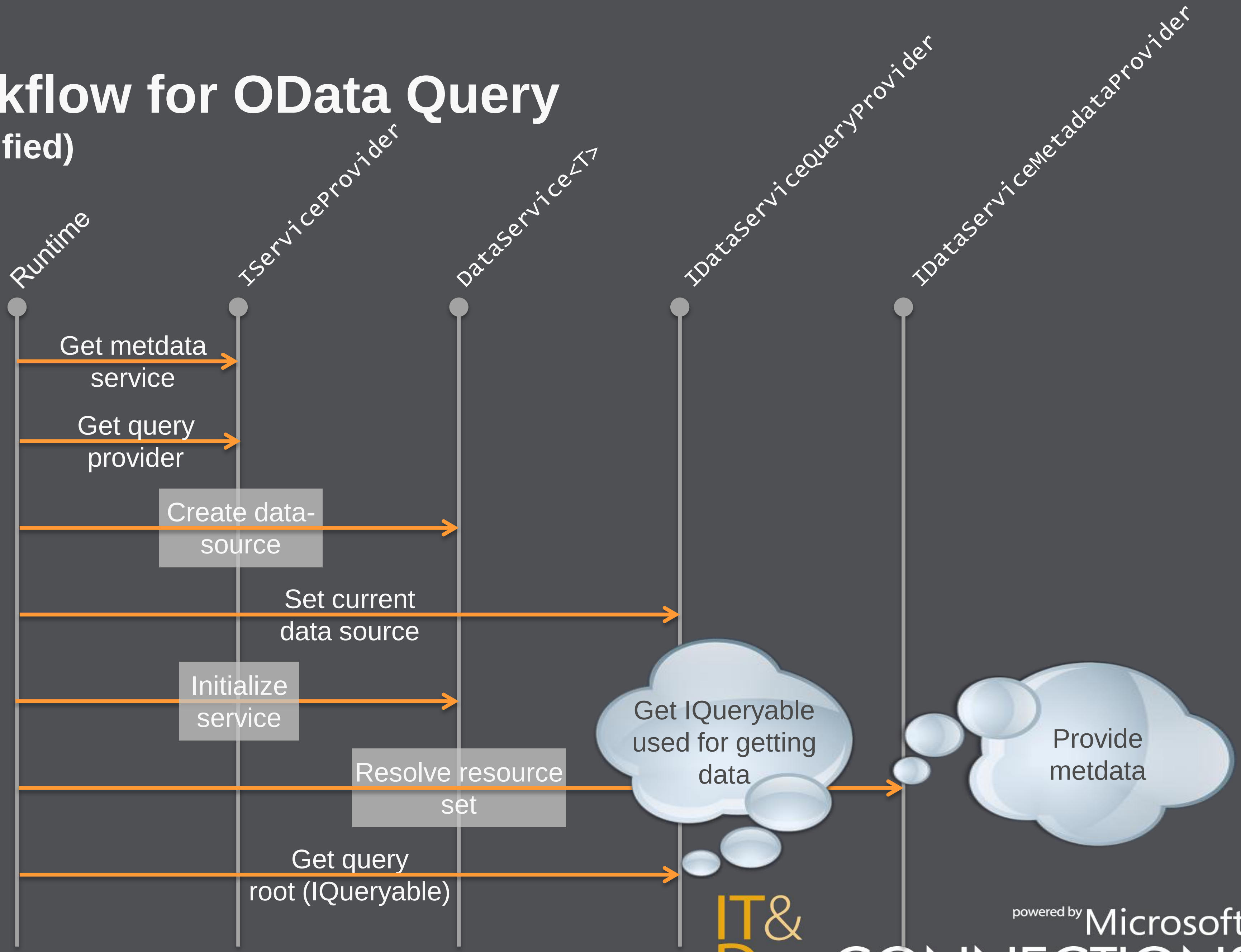
CUSTOM ODATA PROVIDER

Custom OData Building Blocks

- Derive from `DataService<T>`
  - Main entry point for developing a WCF Data Service
- Implement `IServiceProvider` to provide...
  - ...`IDataServiceMetadataProvider`
    - Build custom metadata
    - Build OData metadata with e.g. reflection
  - ... `IDataServiceQueryProvider`
    - Provide an underlying `IQueryable` for query execution

Workflow for OData Query

(simplified)



IDataServiceMetadataProvider

- ResourceSet consisting of ResourceType objects (=entity types)
- CustomDataServiceMetadataProvider is a simple generic implementation of IDataServiceMetadataProvider

IDataServiceMetadataProvider

```
public interface IDataServiceMetadataProvider
{
    string ContainerName { get; }
    string ContainerNamespace { get; }

    IEnumerable<ResourceSet> ResourceSets { get; }
    IEnumerable<ServiceOperation> ServiceOperations { get; }
    IEnumerable<ResourceType> Types { get; }

    IEnumerable<ResourceType> GetDerivedTypes(ResourceType resourceType);
    ResourceAssociationSet GetResourceAssociationSet(
        ResourceSet resourceSet, ResourceType resourceType,
        ResourceProperty resourceProperty);
    bool HasDerivedTypes(ResourceType resourceType);
    bool TryResolveResourceSet(string name, out ResourceSet resourceSet);
    bool TryResolveResourceType(string name, out ResourceType resourceType);
    bool TryResolveServiceOperation(string name,
        out ServiceOperation serviceOperation);
}
```

Service Ops not covered here

Generate Metadata from EF `EntityObject` type

```
var productType = new ResourceType(
    typeof(TEntity),
    ResourceTypeKind.EntityType,
    null, // BaseType
    namespaceName, // Namespace
    typeof(TEntity).Name,
    false // Abstract?
);
```

Sample contains a single type

```
// use reflection to get all properties (except entity framework specific ones)
typeof(TEntity)
    .GetProperties(BindingFlags.Public | BindingFlags.Instance)
    .Where(pi => pi.DeclaringType == typeof(TEntity))
    .Select(pi => new ResourceProperty(
        pi.Name,
        (Attribute.GetCustomAttributes(pi).OfType<EdmScalarPropertyAttribute>().Where(
              ea => ea.EntityKeyProperty).Count() == 1)
            ? ResourcePropertyKind.Primitive | ResourcePropertyKind.Key
            : ResourcePropertyKind.Primitive,
        ResourceType.GetPrimitiveResourceType(pi.PropertyType)))
    .ToList()
    .ForEach(prop => productType.AddProperty(prop));
```

Add properties (Reflection)

```
var metadata = new CustomDataServiceMetadataProvider();
metadata.AddResourceType(productType);
metadata.AddResourceSet(new ResourceSet(typeof(TEntity).Name, productType));
return metadata;
```

`IDataServiceQueryProvider`

- Provides **root `IQueryable`** for a given resource set
- `CustomDataServiceProvider` is a simple generic implementation of `IDataServiceQueryProvider`

IDataServiceQueryProvider

```
public interface IDataServiceQueryProvider
{
   object CurrentDataSource { get; set; }
   bool IsNullPropagationRequired { get; }

   object GetOpenPropertyValue(object target, string propertyName);
   IEnumerable<KeyValuePair<string, object>> GetOpenPropertyValues(
      object target);
   object GetPropertyValue(object target,
      ResourceProperty resourceProperty);
   IQueryable GetQueryRootForResourceSet(ResourceSet resourceSet);
   ResourceType GetResourceType(object target);
   object InvokeServiceOperation(ServiceOperation serviceOperation,
      object[] parameters);
}
```

How can we build a custom IQueryable?

CUSTOM IQUERYABLE

Custom IQueryable

- Use IQToolkit if possible
- Derive your implementation from `QueryProvider`
- Implement two virtual methods
  - `object Execute(Expression expression)`
  - `string GetQueryText(Expression expression)` (Optional)

Using Custom IQueryable

```
private static Query<RealEstate> CreateQueryableRoot()
{
    string shardingConnectingString = ConfigurationManager.AppSettings["ShardingDatabaseConnection"];
    int numberOfShardingDatabases = Int32.Parse(
         ConfigurationManager.AppSettings["NumberOfShardingDatabases"]);

    var connectionStrings = Enumerable.Range(1, numberOfShardingDatabases)
        .Select(i => string.Format(shardingConnectingString, i))
        .ToArray();

    var queryable = new Query<RealEstate>(
        new ShardingProvider<RealEstateEntities, RealEstate>(
            (s) => new RealEstateEntities(new EntityConnectionStringBuilder()
            {
                Metadata =
"res://*/RealEstateModel.csdl|res://*/RealEstateModel.ssdl|res://*/RealEstateModel.msl",
                Provider = "System.Data.SqlClient",
                ProviderConnectionString = s
            }.ConnectionString),
            (ctx) => ctx.RealEstate,
            connectionStrings.ToArray()));
    return queryable;
}
```

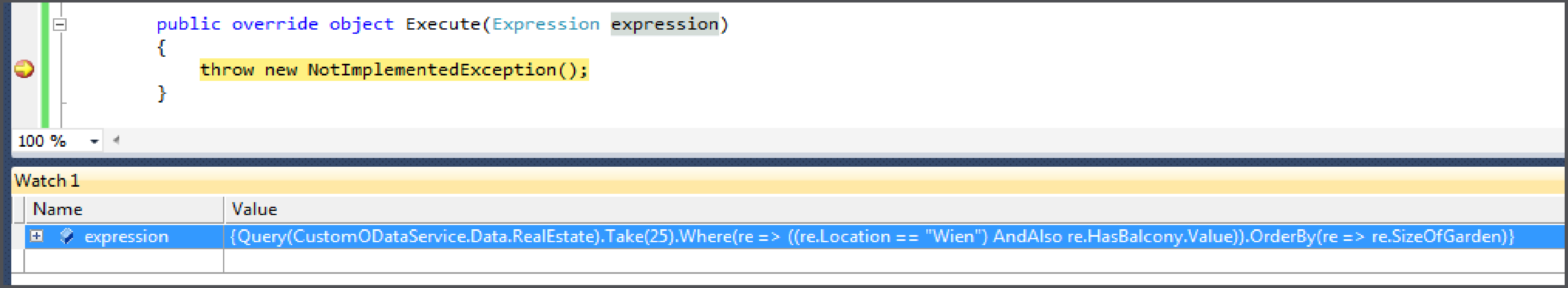
Using Custom IQueryable

```
[TestMethod]
public void TestMethod1()
{
    using (var context = RealEstateEntities.Create())
    {
        var result = context.RealEstate
            .Take(25)
            .Where(re => re.Location == "Wien" && re.HasBalcony.Value)
            .OrderBy(re => re.SizeOfGarden)
            .ToArray();
    }
}

[TestMethod]
public void TestMethod2()
{
    var queryable = CreateQueryableRoot();
    var result = queryable
        .Take(25)
        .Where(re => re.Location == "Wien" && re.HasBalcony.Value)
        .OrderBy(re => re.SizeOfGarden)
        .ToArray();
}
```

Identical LINQ Queries ☺!!

Using Custom IQueryable



```
public override object Execute(Expression expression)
{
    throw new NotImplementedException();
}
```

Name	Value
expression	{Query(CustomODataService.Data.RealEstate).Take(25).Where(re => ((re.Location == "Wien") AndAlso re.HasBalcony.Value)).OrderBy(re => re.SizeOfGarden)}

- Expression tree is received by query provider
- We have to rewrite and run it to all sharding databases

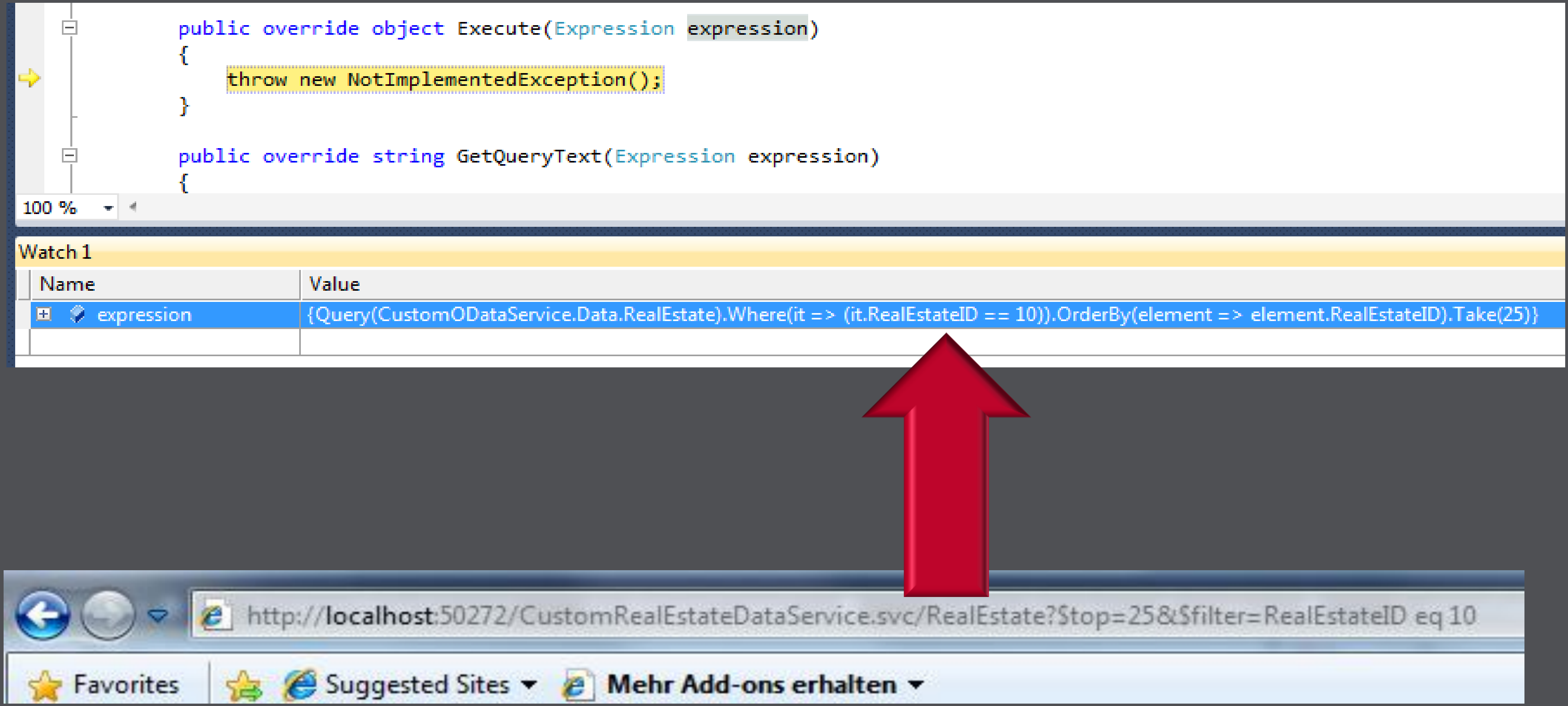
Linking Custom OData & Custom LINQ Provider

```
public class RealEstateContext : CustomDataServiceContext
{
    [...]
     public override IQueryable GetQueryable(ResourceSet set)
     {
         if (set.Name == "RealEstate")
         {
             return CreateQueryable();
         }

         throw new NotSupportedException(
           string.Format("{0} not found", set.Name));
     }
    [...]
}
```

See code
from unit test

Linking Custom OData & Custom LINQ Provider



```
        public override object Execute(Expression expression)
        {
            throw new NotImplementedException();
        }

        public override string GetQueryText(Expression expression)
        {
```

Watch 1

Name	Value
expression	{Query(CustomODataService.Data.RealEstate).Where(it => (it.RealEstateID == 10)).OrderBy(element => element.RealEstateID).Take(25)}

```
http://localhost:50272/CustomRealEstateDataService.svc/RealEstate?$top=25&$filter=RealEstateID eq 10
```

Finishing the custom LINQ provider

IMPLEMENT SHARDING QUERIES

Implement Sharding Queries

- Use .NET‘s visitor pattern to examine and rewrite expression tree
- `VerifyingVistor`
  - Verifies that query is ok (e.g. must contain top-clause, etc.)
  - Stores reference to e.g. order-by clause
- `SwitchQueryable`
  - Replaces the queryable from `Query<T>` to specific sharding database connection (`IQueryable` provided by Entity Framework)

Implement Sharding Queries

- Make sure that the query is ok
  - e.g. must be sorted, must contain top-clause, etc.; business rules defined by the customer in the project mentioned at the beginning of this blog article
- **Parallel** loop over all connections to sharding databases
  - Open entity framework connection to sharding database
  - Replace `Query<T>` in expression tree by connection to sharding database
  - Execute query and return partial result
- Combine partial results by sorting them and applying the top-clause

Tip: Set minimum threads in thread pool for PLINQ

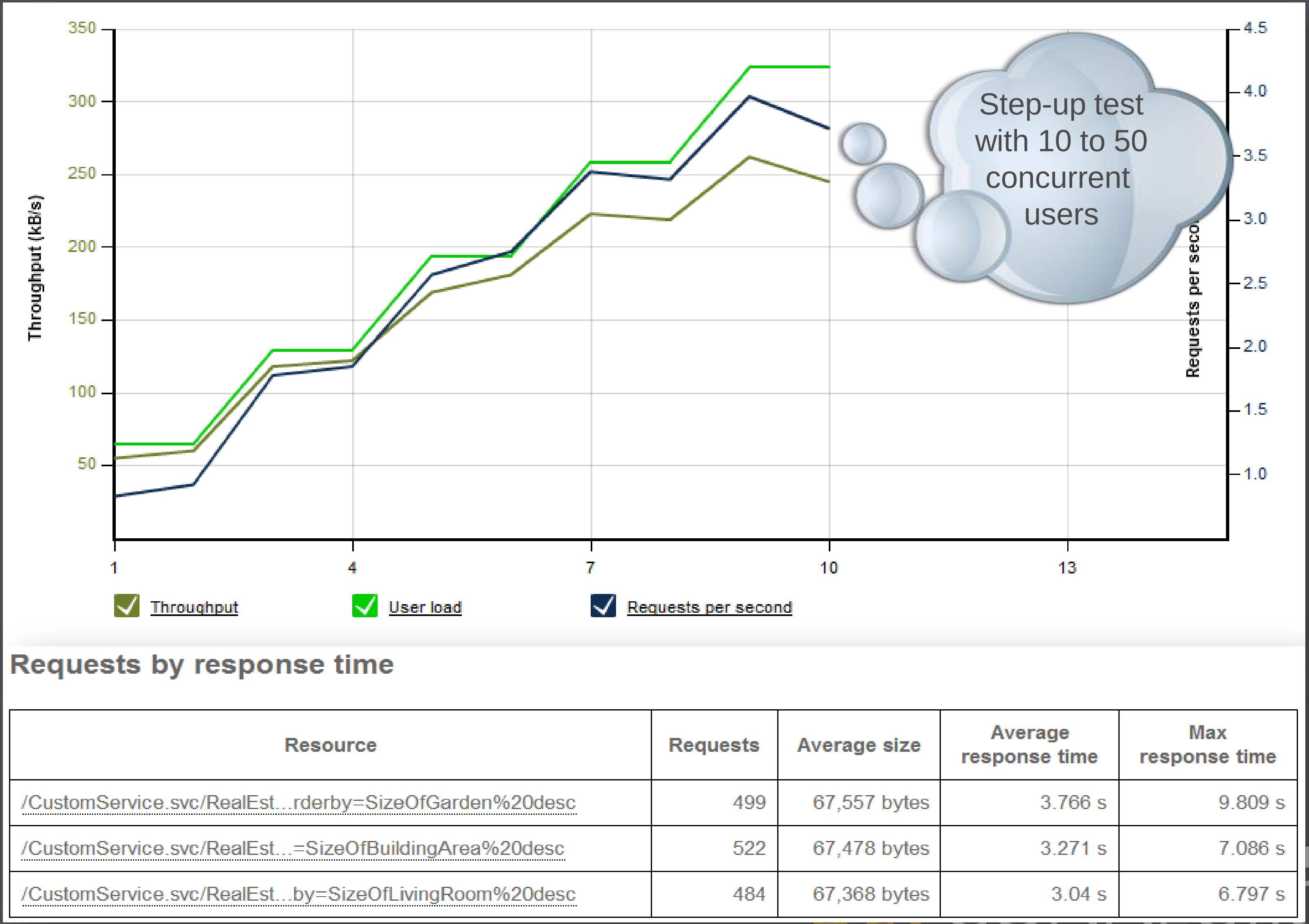
```
static CustomRealEstateDataService()
{
    // Note that this influences whole process;
    // if you need more control over number of threads
    // think about creating your own task scheduler.
    int minThreads, completionPortThreads;
    ThreadPool.GetMinThreads(out minThreads, out completionPortThreads);
    ThreadPool.SetMinThreads(
        Math.Max(minThreads, 11),
        Math.Max(completionPortThreads, 11));
}
```

Was it worth the time?

RESULTS

IT&
Dev CONNECTIONS powered by Microsoft®

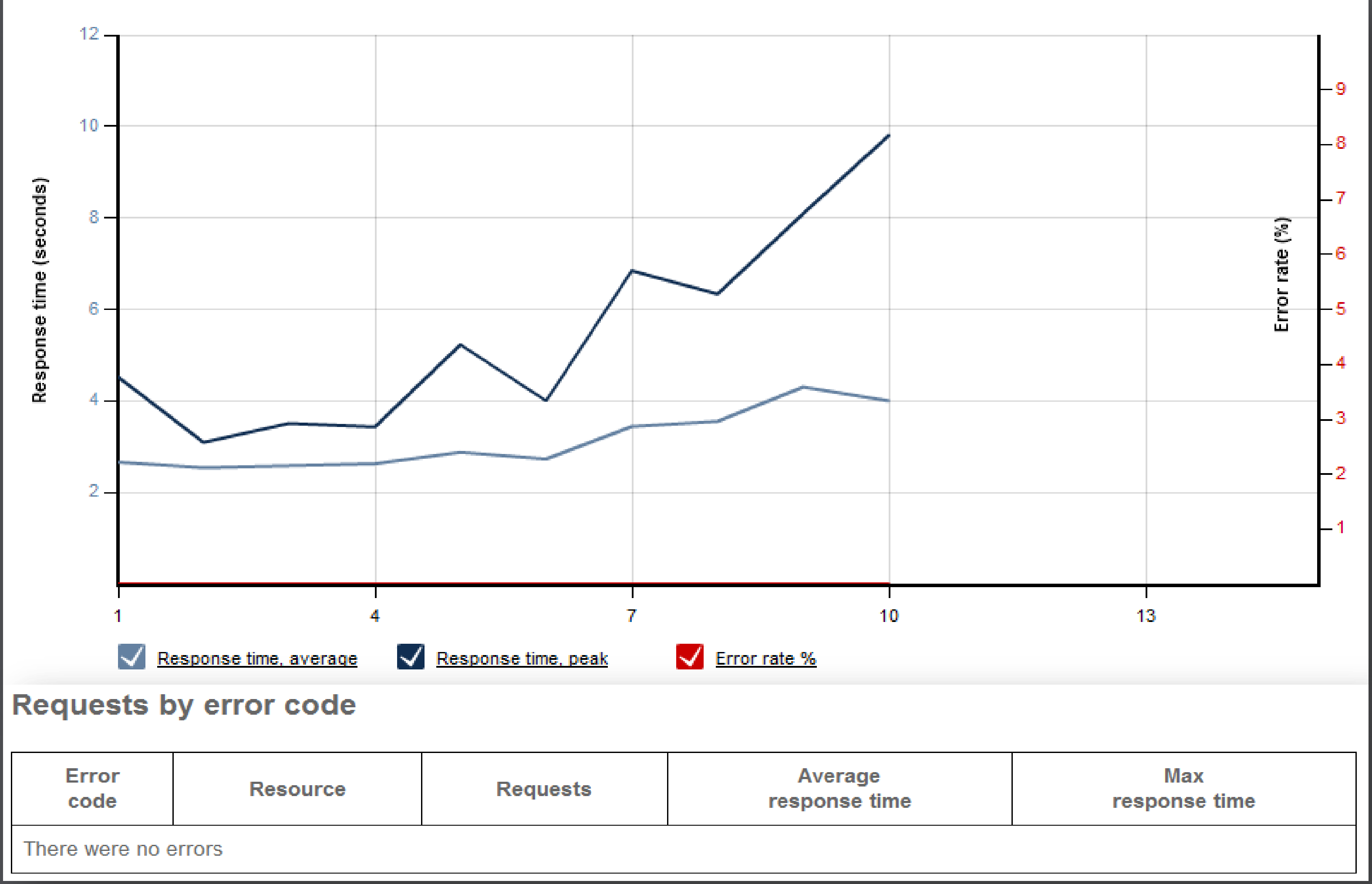
Load Test Results
(Created with LoadStorm.com)



Step-up test with 10 to 50 concurrent users

Requests by response time

Resource	Requests	Average size	Average response time	Max response time
/CustomService.svc/RealEst...rderby=SizeOfGarden%20desc	499	67,557 bytes	3.766 s	9.809 s
/CustomService.svc/RealEst...=SizeOfBuildingArea%20desc	522	67,478 bytes	3.271 s	7.086 s
/CustomService.svc/RealEst...by=SizeOfLivingRoom%20desc	484	67,368 bytes	3.04 s	6.797 s



Requests by error code

Error code	Resource	Requests	Average response time	Max response time
There were no errors				

Your Feedback is Important

Please fill out a session evaluation form.

Thank you!