TRIGENT

Building Scalable Applications Using Microsoft Technologies

Padma Krishnan
Senior Manager

Trigent ■ 2 Willow Street, Suite 201, Southborough, MA 01745 ■ 1-877-387-4436 ■ www.trigent.com

Introduction

CIOs lay great emphasis on application scalability and performance and rightly so. As business grows, the number of transactions and users grow exponentially. Every development team wants to build "scalable" applications. Teams need to have the skills and experience in building scalable applications that can handle several thousand transactions per second. This paper will examine some of the best practices followed by Trigent to develop highly scalable and performing web applications.

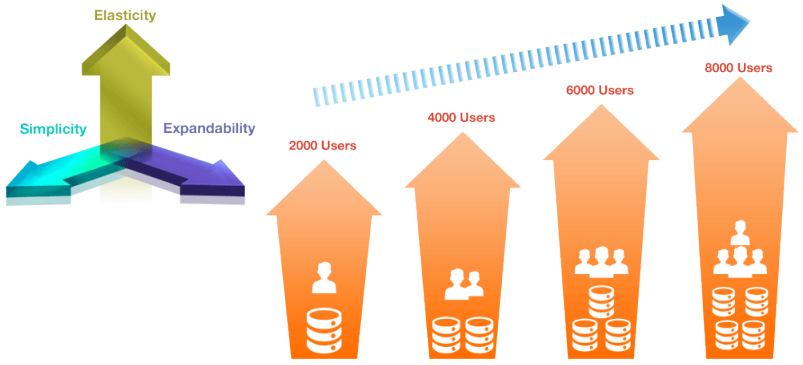
Trigent follows several best practices prescribed by Microsoft's Patterns and Practices group to design scalable applications (http://msdn.microsoft.com/en-us/library/ff921345.aspx). This paper will list some of the most important and practical suggestions followed by our development teams.

Definition of Scalability

Scalability is the capability to increase resources (either processing power or memory) to yield an increase (ideally a linear increase) in service capacity. It is obvious that this does not imply zero cost. But the key difference is that your application should not have to change. Rather, it should be able to handle more and more load commensurate with the amount of resources provided to it.

Scalability cannot be added to an application at the end of its development cycle. It is an integral part of the architecture and design phases. The decisions taken during these phases largely dictate the scalability of the application.

Many people use the terms "performance" and "scalability" interchangeably. These are technically two different characteristics that are closely related and highly desired in multi user applications. Performance is how an application behaves or responds to a single user, under normal conditions. Scalability is how the application behaves or responds to increasing number of users.

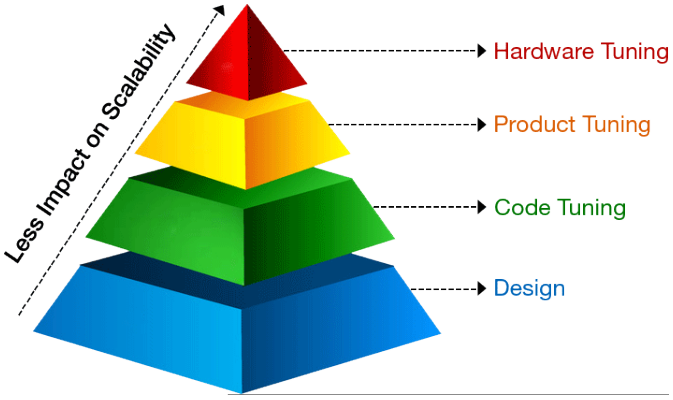


Scalability Types

Two commonly used techniques in scaling are "Scaling Up" and "Scaling Out".

- **Scaling Up:** This is a commonly used term for achieving scalability using better, faster, and more expensive hardware. Scaling up includes adding more memory, adding more or faster processors, or simply run the application on a more powerful, single machine. This method allows for increase in capacity without source code changes.
- **Scaling Out:** Scaling out leverages the economics of using relatively low-cost hardware to distribute the processing load across more than one server. Scaling out is achieved using a collection of machines, essentially functioning as a single unit. By dedicating several machines to a common task, application fault tolerance is also increased. From an administrator's perspective, scaling out presents a greater management challenge, due to the increased number of machines.

This model is effectively used by high volume web applications using a collection of machines, known as a "web farm". In addition to scalability, this method provides redundancy and failover capabilities to the application.



The Scalability Pyramid

In order for this model to work, the application should be written "machine neutral". For instance if the application stores session information within the memory of the web server, or stores files on the server, then it cannot work if the user is moved over to another machine. Therefore web applications must avoid "server affinity", if they are going to be scaled out. As is the case with any complex problem, there are no precise rules for choosing one model over the other and the best answer is - "it depends" on the application.

If the application is used behind the firewall, serving a limited set of users, then scaling up may be a quicker and prudent option. A powerful server can handle several thousands of users but will be a single point of failure. This may very well be the only option if the application has not been designed to scale out in the first place.

If, on the other hand, redundancy and failover capabilities are important and you do not mind the extra licensing and administrative overheads, scaling out may be a better choice. If the application has several logical layers which can be scaled independently of each other, then scaling out is a better choice. For example, the web service layer or business layer can be scaled out separately on more powerful hardware leaving the front end layer running on standard hardware.

Best Practices for Achieving Scalability

The following are some of the best practices consistently followed by our development teams. These practices are divided into four broad categories:

- Application layering techniques: This section recommends best practices on the application structure.
- Design and development techniques: This section provides design and development recommendations that can be applied to all layers.
- Database techniques: This section recommends best practices on the database design and development.
- Other techniques: This section provides some specific alternate solutions and deployment recommendations like Microsoft Azure.

➡ Application Layering Techniques

Logical Separation of Layers

Always logically partition your application between the user interface layer, business logic layer and the data layer. Logical partition does not imply that these layers are actually executing on different physical machines, it makes physical separation possible. If the business logic layer or a portion of it is computationally very intensive, we can wrap that as a service and deploy it on a separate physical machine or several machines. Logical separation of layers is the basic foundation for distributing the functionality of an application across several machines.

Loose Coupling Across Layers

A loosely coupled design based on interfaces is more easily scaled out than tightly-coupled layers with "chatty" interactions. Therefore use the principle of programming to an interface rather than to a concrete implementation. This provides a great deal of flexibility for scalability of different components of your application independently.

➡ Design and Development Techniques

Manage Resources

Contention for resources is the root cause of all scalability problems. A good practice is to acquire resources as late as possible and then release them as soon as possible. The shorter the amount of time that a process is using a resource, the sooner the resource will be available to another process. For example, return database connections to the pool as soon as possible. Also make sure to call Dispose or Close method of the object if available.

A database connection is a scarce resource when several tens of thousands of users are accessing the web application. The application should use SQL Server Authentication mode with pre-defined credentials that have the least privilege needed to perform all the required functionalities. This allows connection pooling and greatly aids scalability as opposed to impersonating users which severely limits connection pooling, and thereby, scalability.

Minimize Round Trips

Minimize round trips to reduce call latency. Design coarse-grained services that allow you to perform a single logical operation by using a single round trip instead of making repeated calls. This is especially relevant when making calls across boundaries like threads, processes or servers.

Multiple SQL queries can be batched using stored procedures.

Multithreading, .NET 4.5's Async Support to Handle Blocking Calls

.NET 4.5 has greatly improved support for handling tasks asynchronously, compared to prior versions. A web server only has a finite number of threads available and under high load conditions, incoming requests will have to wait if the other threads are locked up waiting for database operations or expensive web service calls to complete. By using a combination of "Async" and "Await" keywords, the operations can free up the current thread and serve other requests until those expensive database or web service operations complete. This makes the web servers much more responsive and allows them to handle more load.

The new improvements in parallel programming and tasks enable us to perform several independent tasks in parallel, improving the performance and responsiveness of the application.

Session State Management

To maintain session state across requests in a web application, use "Outproc" mechanism. This allows session information to be maintained out of process, server-side using a cache provider like Velocity. Do not store session state "In Proc", within the memory of that web server. This will be an obvious problem in a load balanced scenario. Session information can be stored in a database or client side using cookies.

If the application is running on Azure, use the Session State Provider for Azure Cache.

Caching

Caching is an important technique for improving performance and scalability. Use caching as appropriate to store common data in memory for fast access. Use a caching framework like Velocity which is a distributed in-memory cache spread across multiple systems. With "Velocity", you can retrieve data using

keys or other identifiers. It supports high availability and a variety of cache configurations without the need to write to a database.

Cache Cow, Memcached and Redis are some other popular open source caching frameworks.

Use StringBuilder

Use StringBuilder to build strings inside loops instead of using concatenation with strings. The StringBuilder class is specially optimized by the .NET compiler making it impossible to duplicate this functionality with equivalent performance. Compute intensive business logic that manipulate a lot of character data in loops achieve a dramatic performance increase by using StringBuilder instead of Strings.

Trust the Garbage Collector

Microsoft's .NET technologies provides a managed environment - meaning that the framework's CLR (Common Language Runtime) takes care of memory management. This is completely different from the nineties era where the programmer had to explicitly allocate and free memory. The Garbage Collector of Microsoft .NET framework has a very sophisticated mechanism for freeing up memory. So, do not invoke the Garbage Collector in your code, unless you are very sure of what you are doing.

Scaling Techniques for Web Client Layer

ASP.NET 4.5 offers support for bundling and minification out of the box. These are two techniques that improve individual request load times, improving performance and ultimately aiding in scalability. Bundling and minification help to bundle multiple CSS and JavaScript files into fewer and smaller files. So load times will be faster and the web server can consequently handle more requests.

Other specific scaling techniques for the presentation layer are:

Content Delivery Networks: Content Delivery Network (CDN) providers like Microsoft, Akamai or Google use a large number of servers distributed geographically to serve content faster from a closest point to the user. This speeds up the overall load time of the sites' content and reduces the load on your web servers.

AJAX usage: Use AJAX calls as appropriate to partially load/update the page instead of fetching the page completely from the web server. This reduces the load and processing on the web server thus helping in scalability.

Client side frameworks: Use client side frameworks like jQuery or Kendo UI which can perform several client side operations like paging, sorting and filtering data grids. This provides a great user experience apart from reducing the web server load again helping in scalability.

➡ Database Techniques

Database Fine Tuning

The database can become the weakest link as a web application can always be scaled out by adding additional hardware. But databases cannot be scaled out in a similar fashion though there are specific circumstances where the scale out model can be applied to a database too. For instance, a high volume SaaS (Software as a Service) application can be designed so that each subscriber or set of subscribers is assigned a particular database. This way, the SaaS application can have several different databases for their subscriber base. This model works as a particular subscriber is only accessing data within her database and not aggregating data across the suite of databases. But regular line of business applications do not lend themselves to this model naturally.

Here are some more suggestions for database scalability:

- Frequently check the performance of the application on the database side during the development cycle.
- Ensure non clustered indexes are created for the frequently used slow running queries.
- Query fine tuning to be done to ensure proper joins are maintained instead of Cartesian or cross joins.
- Limit the use of "Order by" and "Distinct".
- Limit the use of sub query - better, replace with other joins appropriately .
- Ensure Select query returns only the required columns and not all the columns.
- Use some of the latest improvements in SQL Server 2012 like Pagination and Column Store Indexes.
- Use database partitioning as appropriate. For example, if you have a very large table used in say a banking environment where the current month of data is being constantly updated and the previous months' data is being constantly reported on, then you can partition this table by month. With partitioning, maintenance operations such as index rebuilds and defragmentation can be performed on the single month of write-only data, while the read-only data is available for online access.
- Use Stored procedures for compute intensive queries. This will always perform faster than an ORM mapper which may not optimize the SQL calls.

➡ Other Techniques

Microsoft Azure

One of the major advantages Windows Azure offers is the ability to use and pay for only what you need, while being able to increase or decrease resources on demand. Web applications can be deployed on

Windows Azure thus providing options to scale. The "SQL Azure" database is based on Microsoft SQL Server technology and can be used as the database. The Azure platform is continuously enhanced by Microsoft. The platform offers several subscription models based on various factors like storage, compute, etc. and is described in detail on Microsoft's site.

Batch Processing, Reporting

In addition to specific techniques for improving scalability, we can also design solutions that simply avoid the problem.

If the application does not require real time processing, design a solution to queue compute or network intensive tasks for deferred execution. These tasks can be executed during non-business hours and the results be made available or emailed to the users. This technique uses the server resources optimally when a large number of users are online thus providing them with a good user experience and performance.

A similar technique can be employed for reporting needs. A separate database of aggregated data for reporting purposes may be an effective solution. This will free up the transactional needs of the application from the compute intensive reporting needs. The reporting database will just be slightly behind the transactional system in terms of data accuracy, but will provide business users all the pertinent information along with a great user experience.

Summary

The above recommendations are standard and proven best practices for building scalable web applications. It is important to factor in these recommendations during the architecture and design phases as these cannot be added discretely to an application at a later point. Using load testing tools, it is also important to measure the performance of an application to obtain its upper limit on capacity so that further improvements in design and code can be undertaken.

References

MSDN articles

- http://msdn.microsoft.com/en-us/library/aa292203(v=vs.71).aspx
- http://msdn.microsoft.com/en-us/library/aa291873(v=vs.71).aspx
- http://msdn.microsoft.com/en-us/library/ff647801.aspx

About Trigent Software Inc.

Trigent is a privately held, professional IT services company and a Microsoft Gold Partner with its U.S. headquarters in the greater Boston area and its Indian headquarters in Bangalore. We provide consulting services in various technologies including Microsoft Solutions. Our operating model is to conduct sales, customer relationships and front-end consulting (e.g., business case, requirements, architecture) onsite with our clients and perform the detail design, development, integration, testing and quality assurance offshore at our world class development and support center in Bangalore. We are a SEI CMM Level 4 company and is ISO 9001:2000 TickIT certified organization.

For sales contact sales@trigent.com or call 508-490-6000.

Microsoft Partner
Gold Application Development
Gold Collaboration and Content
Silver Mobility